

ForceBalance Developer API Guide version 1.3

Generated by Doxygen 1.7.6.1



# Contents

<b>1 Project Roadmap</b>	<b>2</b>
1.1 Most Recently Implemented (version 1.3.0):	2
1.2 Current Development Goals, for version 1.3.1:	2
1.3 Longterm Development Ideas	2
<b>2 Todo List</b>	<b>3</b>
<b>3 Namespace Index</b>	<b>4</b>
3.1 Packages	4
<b>4 Hierarchical Index</b>	<b>6</b>
4.1 Class Hierarchy	6
<b>5 Class Index</b>	<b>9</b>
5.1 Class List	9
<b>6 File Index</b>	<b>13</b>
6.1 File List	13
<b>7 Namespace Documentation</b>	<b>14</b>
7.1 forcebalance Namespace Reference	14
7.1.1 Variable Documentation	15
7.2 forcebalance.abinitio Namespace Reference	16
7.2.1 Detailed Description	16
7.2.2 Function Documentation	16
7.2.3 Variable Documentation	16
7.3 forcebalance.abinitio_internal Namespace Reference	16
7.3.1 Detailed Description	17
7.4 forcebalance.amberio Namespace Reference	17
7.4.1 Detailed Description	17
7.4.2 Function Documentation	18
7.4.3 Variable Documentation	18
7.5 forcebalance.binding Namespace Reference	18
7.5.1 Detailed Description	19
7.5.2 Function Documentation	19
7.5.3 Variable Documentation	19
7.6 forcebalance.chemistry Namespace Reference	19
7.6.1 Function Documentation	20
7.6.2 Variable Documentation	20
7.7 forcebalance.contact Namespace Reference	21
7.7.1 Function Documentation	21
7.8 forcebalance.counterpoise Namespace Reference	22
7.8.1 Detailed Description	22
7.8.2 Variable Documentation	22
7.9 forcebalance.custom_io Namespace Reference	23
7.9.1 Detailed Description	23
7.9.2 Variable Documentation	23
7.10 forcebalance.engine Namespace Reference	24
7.10.1 Variable Documentation	24
7.11 forcebalance.finite_difference Namespace Reference	24
7.11.1 Function Documentation	25
7.11.2 Variable Documentation	27
7.12 forcebalance.forcefield Namespace Reference	27

7.12.1	Detailed Description . . . . .	27
7.12.2	Function Documentation . . . . .	28
7.12.3	Variable Documentation . . . . .	29
7.13	forcebalance.gmxio Namespace Reference . . . . .	29
7.13.1	Detailed Description . . . . .	30
7.13.2	Function Documentation . . . . .	31
7.13.3	Variable Documentation . . . . .	32
7.14	forcebalance.interaction Namespace Reference . . . . .	33
7.14.1	Detailed Description . . . . .	34
7.14.2	Variable Documentation . . . . .	34
7.15	forcebalance.leastsq Namespace Reference . . . . .	34
7.15.1	Function Documentation . . . . .	34
7.15.2	Variable Documentation . . . . .	34
7.16	forcebalance.lipid Namespace Reference . . . . .	34
7.16.1	Detailed Description . . . . .	35
7.16.2	Function Documentation . . . . .	35
7.16.3	Variable Documentation . . . . .	35
7.17	forcebalance.liquid Namespace Reference . . . . .	35
7.17.1	Detailed Description . . . . .	35
7.17.2	Function Documentation . . . . .	35
7.17.3	Variable Documentation . . . . .	35
7.18	forcebalance.Mol2 Namespace Reference . . . . .	36
7.18.1	Variable Documentation . . . . .	36
7.19	forcebalance.mol2io Namespace Reference . . . . .	36
7.19.1	Detailed Description . . . . .	36
7.19.2	Variable Documentation . . . . .	36
7.20	forcebalance.molecule Namespace Reference . . . . .	37
7.20.1	Function Documentation . . . . .	38
7.20.2	Variable Documentation . . . . .	42
7.21	forcebalance.moments Namespace Reference . . . . .	44
7.21.1	Detailed Description . . . . .	44
7.21.2	Variable Documentation . . . . .	44
7.22	forcebalance.nifty Namespace Reference . . . . .	44
7.22.1	Detailed Description . . . . .	46
7.22.2	Function Documentation . . . . .	47
7.22.3	Variable Documentation . . . . .	56
7.23	forcebalance.objective Namespace Reference . . . . .	57
7.23.1	Detailed Description . . . . .	57
7.23.2	Variable Documentation . . . . .	57
7.24	forcebalance.openmmio Namespace Reference . . . . .	58
7.24.1	Detailed Description . . . . .	59
7.24.2	Function Documentation . . . . .	59
7.24.3	Variable Documentation . . . . .	61
7.25	forcebalance.optimizer Namespace Reference . . . . .	62
7.25.1	Detailed Description . . . . .	62
7.25.2	Function Documentation . . . . .	62
7.25.3	Variable Documentation . . . . .	63
7.26	forcebalance.output Namespace Reference . . . . .	63
7.27	forcebalance.parser Namespace Reference . . . . .	63
7.27.1	Detailed Description . . . . .	64
7.27.2	Function Documentation . . . . .	65
7.27.3	Variable Documentation . . . . .	66
7.28	forcebalance.psi4io Namespace Reference . . . . .	67
7.28.1	Detailed Description . . . . .	67

7.28.2	Variable Documentation . . . . .	68
7.29	forcebalance.PT Namespace Reference . . . . .	68
7.29.1	Variable Documentation . . . . .	68
7.30	forcebalance.qchemio Namespace Reference . . . . .	68
7.30.1	Detailed Description . . . . .	69
7.30.2	Function Documentation . . . . .	69
7.30.3	Variable Documentation . . . . .	69
7.31	forcebalance.quantity Namespace Reference . . . . .	70
7.31.1	Function Documentation . . . . .	70
7.31.2	Variable Documentation . . . . .	71
7.32	forcebalance.target Namespace Reference . . . . .	71
7.32.1	Variable Documentation . . . . .	71
7.33	forcebalance.thermo Namespace Reference . . . . .	71
7.33.1	Variable Documentation . . . . .	71
7.34	forcebalance.tinkerio Namespace Reference . . . . .	71
7.34.1	Detailed Description . . . . .	72
7.34.2	Function Documentation . . . . .	72
7.34.3	Variable Documentation . . . . .	73
7.35	forcebalance.vibration Namespace Reference . . . . .	74
7.35.1	Detailed Description . . . . .	74
7.35.2	Function Documentation . . . . .	74
7.35.3	Variable Documentation . . . . .	74
<b>8</b>	<b>Class Documentation</b> . . . . .	<b>75</b>
8.1	forcebalance.abinitio.AblInitio Class Reference . . . . .	75
8.1.1	Detailed Description . . . . .	79
8.1.2	Constructor & Destructor Documentation . . . . .	79
8.1.3	Member Function Documentation . . . . .	79
8.1.4	Member Data Documentation . . . . .	92
8.2	forcebalance.amberio.AblInitio_AMBER Class Reference . . . . .	96
8.2.1	Detailed Description . . . . .	101
8.2.2	Constructor & Destructor Documentation . . . . .	102
8.2.3	Member Function Documentation . . . . .	102
8.2.4	Member Data Documentation . . . . .	114
8.3	forcebalance.gmxio.AblInitio_GMX Class Reference . . . . .	118
8.3.1	Detailed Description . . . . .	123
8.3.2	Constructor & Destructor Documentation . . . . .	124
8.3.3	Member Function Documentation . . . . .	124
8.3.4	Member Data Documentation . . . . .	136
8.4	forcebalance.abinitio_internal.AblInitio_Internal Class Reference . . . . .	140
8.4.1	Detailed Description . . . . .	145
8.4.2	Constructor & Destructor Documentation . . . . .	146
8.4.3	Member Function Documentation . . . . .	146
8.4.4	Member Data Documentation . . . . .	158
8.5	forcebalance.openmmio.AblInitio_OpenMM Class Reference . . . . .	162
8.5.1	Detailed Description . . . . .	167
8.5.2	Constructor & Destructor Documentation . . . . .	168
8.5.3	Member Function Documentation . . . . .	168
8.5.4	Member Data Documentation . . . . .	180
8.6	forcebalance.tinkerio.AblInitio_TINKER Class Reference . . . . .	184
8.6.1	Detailed Description . . . . .	189
8.6.2	Constructor & Destructor Documentation . . . . .	190
8.6.3	Member Function Documentation . . . . .	190
8.6.4	Member Data Documentation . . . . .	202

8.7	forcebalance.forcefield.BackedUpDict Class Reference . . . . .	206
8.7.1	Detailed Description . . . . .	207
8.7.2	Constructor & Destructor Documentation . . . . .	207
8.7.3	Member Function Documentation . . . . .	207
8.7.4	Member Data Documentation . . . . .	207
8.8	forcebalance.BaseClass Class Reference . . . . .	207
8.8.1	Detailed Description . . . . .	209
8.8.2	Constructor & Destructor Documentation . . . . .	209
8.8.3	Member Function Documentation . . . . .	209
8.8.4	Member Data Documentation . . . . .	209
8.9	forcebalance.BaseReader Class Reference . . . . .	209
8.9.1	Detailed Description . . . . .	211
8.9.2	Constructor & Destructor Documentation . . . . .	211
8.9.3	Member Function Documentation . . . . .	211
8.9.4	Member Data Documentation . . . . .	212
8.10	forcebalance.binding.BindingEnergy Class Reference . . . . .	212
8.10.1	Detailed Description . . . . .	215
8.10.2	Constructor & Destructor Documentation . . . . .	215
8.10.3	Member Function Documentation . . . . .	215
8.10.4	Member Data Documentation . . . . .	225
8.11	forcebalance.gmxio.BindingEnergy_GMX Class Reference . . . . .	226
8.11.1	Detailed Description . . . . .	230
8.11.2	Constructor & Destructor Documentation . . . . .	230
8.11.3	Member Function Documentation . . . . .	230
8.11.4	Member Data Documentation . . . . .	240
8.12	forcebalance.openmmio.BindingEnergy_OpenMM Class Reference . . . . .	241
8.12.1	Detailed Description . . . . .	245
8.12.2	Constructor & Destructor Documentation . . . . .	245
8.12.3	Member Function Documentation . . . . .	245
8.12.4	Member Data Documentation . . . . .	255
8.13	forcebalance.tinkerio.BindingEnergy_TINKER Class Reference . . . . .	256
8.13.1	Detailed Description . . . . .	260
8.13.2	Constructor & Destructor Documentation . . . . .	260
8.13.3	Member Function Documentation . . . . .	260
8.13.4	Member Data Documentation . . . . .	270
8.14	forcebalance.output.CleanFileHandler Class Reference . . . . .	271
8.14.1	Detailed Description . . . . .	272
8.14.2	Member Function Documentation . . . . .	272
8.15	forcebalance.output.CleanStreamHandler Class Reference . . . . .	273
8.15.1	Detailed Description . . . . .	273
8.15.2	Constructor & Destructor Documentation . . . . .	273
8.15.3	Member Function Documentation . . . . .	274
8.16	forcebalance.counterpoise.Counterpoise Class Reference . . . . .	274
8.16.1	Detailed Description . . . . .	277
8.16.2	Constructor & Destructor Documentation . . . . .	277
8.16.3	Member Function Documentation . . . . .	277
8.16.4	Member Data Documentation . . . . .	289
8.17	forcebalance.engine.Engine Class Reference . . . . .	290
8.17.1	Detailed Description . . . . .	291
8.17.2	Constructor & Destructor Documentation . . . . .	292
8.17.3	Member Function Documentation . . . . .	292
8.17.4	Member Data Documentation . . . . .	292
8.18	forcebalance.forcefield.FF Class Reference . . . . .	293
8.18.1	Detailed Description . . . . .	295

8.18.2	Constructor & Destructor Documentation . . . . .	295
8.18.3	Member Function Documentation . . . . .	296
8.18.4	Member Data Documentation . . . . .	304
8.19	<b>forcebalance.output.ForceBalanceLogger Class Reference</b> . . . . .	306
8.19.1	Detailed Description . . . . .	307
8.19.2	Constructor & Destructor Documentation . . . . .	307
8.19.3	Member Function Documentation . . . . .	307
8.19.4	Member Data Documentation . . . . .	307
8.20	<b>forcebalance.amberio.FrcMod_Reader Class Reference</b> . . . . .	307
8.20.1	Detailed Description . . . . .	309
8.20.2	Constructor & Destructor Documentation . . . . .	309
8.20.3	Member Function Documentation . . . . .	309
8.20.4	Member Data Documentation . . . . .	310
8.21	<b>forcebalance.psi4io.GBS_Reader Class Reference</b> . . . . .	310
8.21.1	Detailed Description . . . . .	312
8.21.2	Constructor & Destructor Documentation . . . . .	312
8.21.3	Member Function Documentation . . . . .	312
8.21.4	Member Data Documentation . . . . .	313
8.22	<b>forcebalance.custom_io.Gen_Reader Class Reference</b> . . . . .	314
8.22.1	Detailed Description . . . . .	315
8.22.2	Constructor & Destructor Documentation . . . . .	315
8.22.3	Member Function Documentation . . . . .	315
8.22.4	Member Data Documentation . . . . .	316
8.23	<b>forcebalance.gmxio.GMX Class Reference</b> . . . . .	316
8.23.1	Detailed Description . . . . .	319
8.23.2	Constructor & Destructor Documentation . . . . .	319
8.23.3	Member Function Documentation . . . . .	319
8.23.4	Member Data Documentation . . . . .	325
8.24	<b>forcebalance.psi4io.Grid_Reader Class Reference</b> . . . . .	327
8.24.1	Detailed Description . . . . .	329
8.24.2	Constructor & Destructor Documentation . . . . .	329
8.24.3	Member Function Documentation . . . . .	329
8.24.4	Member Data Documentation . . . . .	329
8.25	<b>forcebalance.interaction.Interaction Class Reference</b> . . . . .	330
8.25.1	Detailed Description . . . . .	333
8.25.2	Constructor & Destructor Documentation . . . . .	333
8.25.3	Member Function Documentation . . . . .	333
8.25.4	Member Data Documentation . . . . .	343
8.26	<b>forcebalance.gmxio.Interaction_GMX Class Reference</b> . . . . .	345
8.26.1	Detailed Description . . . . .	348
8.26.2	Constructor & Destructor Documentation . . . . .	348
8.26.3	Member Function Documentation . . . . .	349
8.26.4	Member Data Documentation . . . . .	358
8.27	<b>forcebalance.openmmio.Interaction_OpenMM Class Reference</b> . . . . .	360
8.27.1	Detailed Description . . . . .	363
8.27.2	Constructor & Destructor Documentation . . . . .	363
8.27.3	Member Function Documentation . . . . .	364
8.27.4	Member Data Documentation . . . . .	373
8.28	<b>forcebalance.tinkerio.Interaction_TINKER Class Reference</b> . . . . .	375
8.28.1	Detailed Description . . . . .	378
8.28.2	Constructor & Destructor Documentation . . . . .	378
8.28.3	Member Function Documentation . . . . .	379
8.28.4	Member Data Documentation . . . . .	388
8.29	<b>forcebalance.gmxio.ITP_Reader Class Reference</b> . . . . .	390

8.29.1	Detailed Description . . . . .	392
8.29.2	Constructor & Destructor Documentation . . . . .	392
8.29.3	Member Function Documentation . . . . .	392
8.29.4	Member Data Documentation . . . . .	393
8.30	forcebalance.leastsq.LeastSquares Class Reference . . . . .	394
8.30.1	Detailed Description . . . . .	397
8.30.2	Constructor & Destructor Documentation . . . . .	397
8.30.3	Member Function Documentation . . . . .	397
8.30.4	Member Data Documentation . . . . .	407
8.31	forcebalance.nifty.LineChunker Class Reference . . . . .	408
8.31.1	Detailed Description . . . . .	409
8.31.2	Constructor & Destructor Documentation . . . . .	409
8.31.3	Member Function Documentation . . . . .	409
8.31.4	Member Data Documentation . . . . .	410
8.32	forcebalance.lipid.Lipid Class Reference . . . . .	410
8.32.1	Detailed Description . . . . .	414
8.32.2	Constructor & Destructor Documentation . . . . .	414
8.32.3	Member Function Documentation . . . . .	415
8.32.4	Member Data Documentation . . . . .	426
8.33	forcebalance.gmxio.Lipid_GMX Class Reference . . . . .	429
8.33.1	Detailed Description . . . . .	432
8.33.2	Constructor & Destructor Documentation . . . . .	433
8.33.3	Member Function Documentation . . . . .	433
8.33.4	Member Data Documentation . . . . .	444
8.34	forcebalance.liquid.Liquid Class Reference . . . . .	447
8.34.1	Detailed Description . . . . .	449
8.34.2	Constructor & Destructor Documentation . . . . .	450
8.34.3	Member Function Documentation . . . . .	450
8.34.4	Member Data Documentation . . . . .	462
8.35	forcebalance.gmxio.Liquid_GMX Class Reference . . . . .	464
8.35.1	Detailed Description . . . . .	467
8.35.2	Constructor & Destructor Documentation . . . . .	468
8.35.3	Member Function Documentation . . . . .	468
8.35.4	Member Data Documentation . . . . .	481
8.36	forcebalance.openmmio.Liquid_OpenMM Class Reference . . . . .	483
8.36.1	Detailed Description . . . . .	487
8.36.2	Constructor & Destructor Documentation . . . . .	488
8.36.3	Member Function Documentation . . . . .	488
8.36.4	Member Data Documentation . . . . .	500
8.37	forcebalance.tinkerio.Liquid_TINKER Class Reference . . . . .	502
8.37.1	Detailed Description . . . . .	506
8.37.2	Constructor & Destructor Documentation . . . . .	507
8.37.3	Member Function Documentation . . . . .	507
8.37.4	Member Data Documentation . . . . .	520
8.38	forcebalance.output.ModLogger Class Reference . . . . .	522
8.38.1	Detailed Description . . . . .	523
8.38.2	Member Function Documentation . . . . .	523
8.39	forcebalance.Mol2.mol2 Class Reference . . . . .	523
8.39.1	Detailed Description . . . . .	524
8.39.2	Constructor & Destructor Documentation . . . . .	524
8.39.3	Member Function Documentation . . . . .	524
8.39.4	Member Data Documentation . . . . .	527
8.40	forcebalance.Mol2.mol2.atom Class Reference . . . . .	527
8.40.1	Detailed Description . . . . .	528

8.40.2	Constructor & Destructor Documentation . . . . .	528
8.40.3	Member Function Documentation . . . . .	528
8.40.4	Member Data Documentation . . . . .	530
8.41	forcebalance.Mol2.mol2.bond Class Reference . . . . .	530
8.41.1	Detailed Description . . . . .	531
8.41.2	Constructor & Destructor Documentation . . . . .	531
8.41.3	Member Function Documentation . . . . .	531
8.41.4	Member Data Documentation . . . . .	532
8.42	forcebalance.mol2io.Mol2_Reader Class Reference . . . . .	533
8.42.1	Detailed Description . . . . .	534
8.42.2	Constructor & Destructor Documentation . . . . .	534
8.42.3	Member Function Documentation . . . . .	534
8.42.4	Member Data Documentation . . . . .	534
8.43	forcebalance.amberio.Mol2_Reader Class Reference . . . . .	535
8.43.1	Detailed Description . . . . .	537
8.43.2	Constructor & Destructor Documentation . . . . .	537
8.43.3	Member Function Documentation . . . . .	537
8.43.4	Member Data Documentation . . . . .	537
8.44	forcebalance.Mol2.mol2.set Class Reference . . . . .	538
8.44.1	Detailed Description . . . . .	538
8.44.2	Constructor & Destructor Documentation . . . . .	538
8.44.3	Member Function Documentation . . . . .	538
8.44.4	Member Data Documentation . . . . .	538
8.45	forcebalance.molecule.Molecule Class Reference . . . . .	539
8.45.1	Detailed Description . . . . .	542
8.45.2	Constructor & Destructor Documentation . . . . .	543
8.45.3	Member Function Documentation . . . . .	543
8.45.4	Member Data Documentation . . . . .	558
8.46	forcebalance.molecule.MolfileTimestep Class Reference . . . . .	559
8.46.1	Detailed Description . . . . .	560
8.47	forcebalance.moments.Moments Class Reference . . . . .	560
8.47.1	Detailed Description . . . . .	563
8.47.2	Constructor & Destructor Documentation . . . . .	563
8.47.3	Member Function Documentation . . . . .	563
8.47.4	Member Data Documentation . . . . .	573
8.48	forcebalance.gmxio.Moments_GMX Class Reference . . . . .	574
8.48.1	Detailed Description . . . . .	578
8.48.2	Constructor & Destructor Documentation . . . . .	578
8.48.3	Member Function Documentation . . . . .	578
8.48.4	Member Data Documentation . . . . .	588
8.49	forcebalance.openmmio.Moments_OpenMM Class Reference . . . . .	589
8.49.1	Detailed Description . . . . .	593
8.49.2	Constructor & Destructor Documentation . . . . .	593
8.49.3	Member Function Documentation . . . . .	593
8.49.4	Member Data Documentation . . . . .	603
8.50	forcebalance.tinkerio.Moments_TINKER Class Reference . . . . .	604
8.50.1	Detailed Description . . . . .	608
8.50.2	Constructor & Destructor Documentation . . . . .	608
8.50.3	Member Function Documentation . . . . .	608
8.50.4	Member Data Documentation . . . . .	618
8.51	forcebalance.molecule.MyG Class Reference . . . . .	620
8.51.1	Detailed Description . . . . .	621
8.51.2	Constructor & Destructor Documentation . . . . .	621
8.51.3	Member Function Documentation . . . . .	621

8.51.4 Member Data Documentation . . . . .	622
8.52 forcebalance.objective.Objective Class Reference . . . . .	622
8.52.1 Detailed Description . . . . .	624
8.52.2 Constructor & Destructor Documentation . . . . .	624
8.52.3 Member Function Documentation . . . . .	624
8.52.4 Member Data Documentation . . . . .	625
8.53 forcebalance.openmmio.OpenMM Class Reference . . . . .	626
8.53.1 Detailed Description . . . . .	629
8.53.2 Constructor & Destructor Documentation . . . . .	629
8.53.3 Member Function Documentation . . . . .	629
8.53.4 Member Data Documentation . . . . .	635
8.54 forcebalance.openmmio.OpenMM_Reader Class Reference . . . . .	637
8.54.1 Detailed Description . . . . .	638
8.54.2 Constructor & Destructor Documentation . . . . .	638
8.54.3 Member Function Documentation . . . . .	638
8.54.4 Member Data Documentation . . . . .	639
8.55 forcebalance.optimizer.Optimizer Class Reference . . . . .	639
8.55.1 Detailed Description . . . . .	642
8.55.2 Constructor & Destructor Documentation . . . . .	642
8.55.3 Member Function Documentation . . . . .	642
8.55.4 Member Data Documentation . . . . .	651
8.56 forcebalance.objective.Penalty Class Reference . . . . .	653
8.56.1 Detailed Description . . . . .	654
8.56.2 Constructor & Destructor Documentation . . . . .	654
8.56.3 Member Function Documentation . . . . .	654
8.56.4 Member Data Documentation . . . . .	656
8.57 forcebalance.nifty.Pickler_LP Class Reference . . . . .	656
8.57.1 Detailed Description . . . . .	657
8.57.2 Constructor & Destructor Documentation . . . . .	657
8.58 forcebalance.thermo.Point Class Reference . . . . .	658
8.58.1 Detailed Description . . . . .	659
8.58.2 Constructor & Destructor Documentation . . . . .	659
8.58.3 Member Function Documentation . . . . .	659
8.58.4 Member Data Documentation . . . . .	659
8.59 forcebalance.qchemio.QCIn_Reader Class Reference . . . . .	659
8.59.1 Detailed Description . . . . .	661
8.59.2 Constructor & Destructor Documentation . . . . .	661
8.59.3 Member Function Documentation . . . . .	661
8.59.4 Member Data Documentation . . . . .	662
8.60 forcebalance.quantity.Quantity Class Reference . . . . .	662
8.60.1 Detailed Description . . . . .	664
8.60.2 Constructor & Destructor Documentation . . . . .	664
8.60.3 Member Function Documentation . . . . .	664
8.60.4 Member Data Documentation . . . . .	664
8.61 forcebalance.quantity.Quantity_Density Class Reference . . . . .	665
8.61.1 Detailed Description . . . . .	666
8.61.2 Constructor & Destructor Documentation . . . . .	666
8.61.3 Member Function Documentation . . . . .	666
8.61.4 Member Data Documentation . . . . .	666
8.62 forcebalance.quantity.Quantity_H_vap Class Reference . . . . .	667
8.62.1 Detailed Description . . . . .	668
8.62.2 Constructor & Destructor Documentation . . . . .	668
8.62.3 Member Function Documentation . . . . .	668
8.62.4 Member Data Documentation . . . . .	668

8.63	forcebalance.output.RawFileHandler Class Reference . . . . .	669
8.63.1	Detailed Description . . . . .	669
8.63.2	Member Function Documentation . . . . .	669
8.64	forcebalance.output.RawStreamHandler Class Reference . . . . .	670
8.64.1	Detailed Description . . . . .	670
8.64.2	Constructor & Destructor Documentation . . . . .	670
8.64.3	Member Function Documentation . . . . .	671
8.65	forcebalance.psi4io.RDVR3_Psi4 Class Reference . . . . .	671
8.65.1	Detailed Description . . . . .	674
8.65.2	Constructor & Destructor Documentation . . . . .	674
8.65.3	Member Function Documentation . . . . .	674
8.65.4	Member Data Documentation . . . . .	685
8.66	forcebalance.target.RemoteTarget Class Reference . . . . .	687
8.66.1	Detailed Description . . . . .	689
8.66.2	Constructor & Destructor Documentation . . . . .	690
8.66.3	Member Function Documentation . . . . .	690
8.66.4	Member Data Documentation . . . . .	699
8.67	forcebalance.target.Target Class Reference . . . . .	700
8.67.1	Detailed Description . . . . .	704
8.67.2	Constructor & Destructor Documentation . . . . .	704
8.67.3	Member Function Documentation . . . . .	704
8.67.4	Member Data Documentation . . . . .	714
8.68	forcebalance.psi4io.THCDF_Psi4 Class Reference . . . . .	715
8.68.1	Detailed Description . . . . .	718
8.68.2	Constructor & Destructor Documentation . . . . .	718
8.68.3	Member Function Documentation . . . . .	718
8.68.4	Member Data Documentation . . . . .	730
8.69	forcebalance.thermo.Thermo Class Reference . . . . .	732
8.69.1	Detailed Description . . . . .	735
8.69.2	Constructor & Destructor Documentation . . . . .	735
8.69.3	Member Function Documentation . . . . .	735
8.69.4	Member Data Documentation . . . . .	745
8.70	forcebalance.gmxio.Thermo_GMX Class Reference . . . . .	747
8.70.1	Detailed Description . . . . .	751
8.70.2	Constructor & Destructor Documentation . . . . .	751
8.70.3	Member Function Documentation . . . . .	751
8.70.4	Member Data Documentation . . . . .	762
8.71	forcebalance.tinkerio.TINKER Class Reference . . . . .	764
8.71.1	Detailed Description . . . . .	767
8.71.2	Constructor & Destructor Documentation . . . . .	767
8.71.3	Member Function Documentation . . . . .	767
8.71.4	Member Data Documentation . . . . .	772
8.72	forcebalance.tinkerio.Tinker_Reader Class Reference . . . . .	773
8.72.1	Detailed Description . . . . .	775
8.72.2	Constructor & Destructor Documentation . . . . .	775
8.72.3	Member Function Documentation . . . . .	775
8.72.4	Member Data Documentation . . . . .	776
8.73	forcebalance.nifty.Unpickler_LP Class Reference . . . . .	776
8.73.1	Detailed Description . . . . .	777
8.73.2	Constructor & Destructor Documentation . . . . .	777
8.74	forcebalance.vibration.Vibration Class Reference . . . . .	777
8.74.1	Detailed Description . . . . .	780
8.74.2	Constructor & Destructor Documentation . . . . .	781
8.74.3	Member Function Documentation . . . . .	781

8.74.4 Member Data Documentation . . . . .	790
8.75 forcebalance.gmxio.Vibration_GMX Class Reference . . . . .	792
8.75.1 Detailed Description . . . . .	795
8.75.2 Constructor & Destructor Documentation . . . . .	795
8.75.3 Member Function Documentation . . . . .	795
8.75.4 Member Data Documentation . . . . .	805
8.76 forcebalance.tinkerio.Vibration_TINKER Class Reference . . . . .	807
8.76.1 Detailed Description . . . . .	810
8.76.2 Constructor & Destructor Documentation . . . . .	810
8.76.3 Member Function Documentation . . . . .	810
8.76.4 Member Data Documentation . . . . .	820
<b>9 File Documentation</b>	<b>822</b>
9.1 __init__.py File Reference . . . . .	822
9.2 abinitio.py File Reference . . . . .	822
9.3 abinitio_Internal.py File Reference . . . . .	823
9.4 amberio.py File Reference . . . . .	823
9.5 api.dox File Reference . . . . .	823
9.6 binding.py File Reference . . . . .	823
9.7 chemistry.py File Reference . . . . .	824
9.8 contact.py File Reference . . . . .	824
9.9 counterpoise.py File Reference . . . . .	825
9.10 custom_io.py File Reference . . . . .	825
9.11 engine.py File Reference . . . . .	825
9.12 finite_difference.py File Reference . . . . .	826
9.13 forcefield.py File Reference . . . . .	826
9.14 gmxio.py File Reference . . . . .	827
9.15 interaction.py File Reference . . . . .	828
9.16 leastsq.py File Reference . . . . .	828
9.17 lipid.py File Reference . . . . .	829
9.18 liquid.py File Reference . . . . .	829
9.19 Mol2.py File Reference . . . . .	829
9.20 mol2io.py File Reference . . . . .	830
9.21 molecule.py File Reference . . . . .	830
9.22 moments.py File Reference . . . . .	832
9.23 nifty.py File Reference . . . . .	832
9.24 objective.py File Reference . . . . .	834
9.25 openmmio.py File Reference . . . . .	835
9.26 optimizer.py File Reference . . . . .	836
9.27 output.py File Reference . . . . .	837
9.28 parser.py File Reference . . . . .	837
9.29 psi4io.py File Reference . . . . .	838
9.30 PT.py File Reference . . . . .	838
9.31 qchemio.py File Reference . . . . .	838
9.32 quantity.py File Reference . . . . .	839
9.33 target.py File Reference . . . . .	839
9.34 thermo.py File Reference . . . . .	840
9.35 tinkerio.py File Reference . . . . .	840
9.36 vibration.py File Reference . . . . .	841
<b>Index</b>	<b>842</b>

# 1 Project Roadmap

ForceBalance is a work in progress and is continually being improved and expanded! Here are some current and future project development ideas.

Some notes on updating the version:

The formalism for the version number goes something like this:

v1.2.1[ab][1-9]

where:

- 1.2.1 stands for major release (may break compatibility), medium-sized release (important features), minor release (improvements)
- a or b, if present, stands for alpha or beta release, with numbers standing for the n-th alpha or beta release

To manually specify a release, create a tag and push it to the remote repository: git tag -a v1.2.1 -m "version 1.2.1" git push --tags

The version number should then be automatically generated by "git describe" which is run by setup.py at installation.

Finally, remember to update the version number in the documentation generation scripts!

## 1.1 Most Recently Implemented (version 1.3.0):

- Engine class is a unified interface to MD simulation codes.
- Added Gromacs, OpenMM, and TINKER engines.
- Thermo target; simple support for general thermodynamic properties. (Erik)
- Lipid target; lipid bilayer properties. (Keri)
- ForceBalance –continue option continues an aborted run and loads as much data as possible from the latest iteration.
- Parameter filtering allows targets to skip over parameters that are known to be irrelevant, for efficiency of finite difference derivatives.
- (Optimizer / Liquid / Lipid) Increase simulation length as we get closer to convergence.
- (Gromacs) Now supports binding energies, interaction energies, multipole moments and vibrational frequencies.
- (OpenMM) Now supports binding energies, interaction energies, and multipole moments.
- ([nifty.py](#)) exec\_() reads from stdout and stderr asynchronously, allowing us to split the streams and tail -f the output at the same time.

## 1.2 Current Development Goals, for version 1.3.1:

- More comprehensive tutorial to walk users through the initial process of setting up targets and preparing for a successful ForceBalance run

## 1.3 Longterm Development Ideas

- Visualization of running calculations

## 2 Todo List

### Member [forcebalance.abinitio.AblInitio.\\_\\_init\\_\\_](#)

Obtain the number of true atoms (or the particle -> atom mapping) from the force field.

### Member [forcebalance.abinitio.AblInitio.get\\_energy\\_force](#)

Parallelization over snapshots is not implemented yet

### Member [forcebalance.abinitio.AblInitio.read\\_reference\\_data](#)

Add an option for picking any slice out of qdata.txt, helpful for cross-validation

Closer integration of reference data with program - leave behind the qdata.txt format? (For now, I like the readability of qdata.txt)

The WHAM Boltzmann weights are generated by external scripts (wanalyze.py and make-wham-data.sh) and passed in; perhaps these scripts can be added to the ForceBalance distribution or integrated more tightly.

Closer integration of reference data with program - leave behind the qdata.txt format? (For now, I like the readability of qdata.txt)

The WHAM Boltzmann weights are generated by external scripts (wanalyze.py and make-wham-data.sh) and passed in; perhaps these scripts can be added to the ForceBalance distribution or integrated more tightly.

Closer integration of reference data with program - leave behind the qdata.txt format? (For now, I like the readability of qdata.txt)

The WHAM Boltzmann weights are generated by external scripts (wanalyze.py and make-wham-data.sh) and passed in; perhaps these scripts can be added to the ForceBalance distribution or integrated more tightly.

Closer integration of reference data with program - leave behind the qdata.txt format? (For now, I like the readability of qdata.txt)

The WHAM Boltzmann weights are generated by external scripts (wanalyze.py and make-wham-data.sh) and passed in; perhaps these scripts can be added to the ForceBalance distribution or integrated more tightly.

Closer integration of reference data with program - leave behind the qdata.txt format? (For now, I like the readability of qdata.txt)

The WHAM Boltzmann weights are generated by external scripts (wanalyze.py and make-wham-data.sh) and passed in; perhaps these scripts can be added to the ForceBalance distribution or integrated more tightly.

Closer integration of reference data with program - leave behind the qdata.txt format? (For now, I like the readability of qdata.txt)

The WHAM Boltzmann weights are generated by external scripts (wanalyze.py and make-wham-data.sh) and passed in; perhaps these scripts can be added to the ForceBalance distribution or integrated more tightly.

### Member [forcebalance.counterpoise.Counterpoise.loadxyz](#)

I should probably put this into a more general library for reading coordinates.

### Member [forcebalance.forcefield.FF.mktransmat](#)

Only project out changes in total charge of a molecule, and perhaps generalize to fragments of molecules or other types of parameters.

The AMOEBA selection of charge depends not only on the atom type, but what that atom is bonded to.

### Member [forcebalance.forcefield.FF.rsmake](#)

Pass in rsfactors through the input file

### Namespace [forcebalance.gmxio](#)

Even more stuff from [forcefield.py](#) needs to go into here.

### Class [forcebalance.gmxio.ITP\\_Reader](#)

Note that I can also create the opposite virtual site position by changing the atom labeling, woo!

**Member [forcebalance.openmmio.OpenMM.Reader.build\\_pid](#)**

Add a link here

**Member [forcebalance.optimizer.Optimizer.GeneticAlgorithm](#)**

Massive parallelization hasn't been implemented yet

**Member [forcebalance.optimizer.Optimizer.Scan\\_Values](#)**

Maybe a multidimensional grid can be done.

Parameters

in	<i>MathPhys</i>	Switch to use mathematical (True) or physical (False) parameters.
----	-----------------	---

**Member [forcebalance.tinkerio.Tinker\\_Reader.feed](#)**

Put the rescaling factors for TINKER parameters in here. Currently we're using the initial value to determine the rescaling factor which is not very good.

**Member [forcebalance::gmxio.pdict](#)**

This needs to become more flexible because the parameter isn't always in the same field. Still need to figure out how to do this.

How about making the PDIHS less ugly?

**Member [forcebalance::nifty.floatornan](#)**

I could use suggestions for making this better.

**Member [forcebalance::parser.parse\\_inputs](#)**

Implement internal coordinates.

Implement sampling correction.

Implement charge groups.

## 3 Namespace Index

### 3.1 Packages

Here are the packages with brief descriptions (if available):

<b><a href="#">forcebalance</a></b>	<b>14</b>
<b><a href="#">forcebalance.abinitio</a></b>	
Ab-initio fitting module (energies, forces, resp)	<b>16</b>
<b><a href="#">forcebalance.abinitio_internal</a></b>	
Internal implementation of energy matching (for TIP3P water only)	<b>16</b>
<b><a href="#">forcebalance.amberio</a></b>	
AMBER force field input/output	<b>17</b>
<b><a href="#">forcebalance.binding</a></b>	
Binding energy fitting module	<b>18</b>
<b><a href="#">forcebalance.chemistry</a></b>	<b>19</b>
<b><a href="#">forcebalance.contact</a></b>	<b>21</b>
<b><a href="#">forcebalance.counterpoise</a></b>	
Match an empirical potential to the counterpoise correction for basis set superposition error (BS-SE)	<b>22</b>

<b>forcebalance.custom_io</b>	
Custom force field parser	23
<b>forcebalance.engine</b>	24
<b>forcebalance.finite_difference</b>	24
<b>forcebalance.forcefield</b>	
Force field module	27
<b>forcebalance.gmxio</b>	
GROMACS input/output	29
<b>forcebalance.interaction</b>	
Interaction energy fitting module	33
<b>forcebalance.leastsq</b>	34
<b>forcebalance.lipid</b>	
Matching of lipid bulk properties	34
<b>forcebalance.liquid</b>	
Matching of liquid bulk properties	35
<b>forcebalance.Mol2</b>	36
<b>forcebalance.mol2io</b>	
Mol2 I/O	36
<b>forcebalance.molecule</b>	37
<b>forcebalance.moments</b>	
Multipole moment fitting module	44
<b>forcebalance.nifty</b>	
Nifty functions, intended to be imported by any module within ForceBalance	44
<b>forcebalance.objective</b>	
ForceBalance objective function	57
<b>forcebalance.openmmio</b>	
OpenMM input/output	58
<b>forcebalance.optimizer</b>	
Optimization algorithms	62
<b>forcebalance.output</b>	63
<b>forcebalance.parser</b>	
Input file parser for ForceBalance jobs	63
<b>forcebalance.psi4io</b>	
PSI4 force field input/output	67
<b>forcebalance.PT</b>	68

<b>forcebalance.qchemio</b>	
Q-Chem input file parser	68
<b>forcebalance.quantity</b>	70
<b>forcebalance.target</b>	71
<b>forcebalance.thermo</b>	71
<b>forcebalance.tinkerio</b>	
TINKER input/output	71
<b>forcebalance.vibration</b>	
Vibrational mode fitting module	74

## 4 Hierarchical Index

### 4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

dict	
<b>forcebalance.forcefield.BackedUpDict</b>	206
FileHandler	
<b>forcebalance.output.CleanFileHandler</b>	271
Graph	
<b>forcebalance.molecule.MyG</b>	620
Logger	
<b>forcebalance.output.ForceBalanceLogger</b>	306
<b>forcebalance.output.ModLogger</b>	522
<b>forcebalance.Mol2.mol2</b>	523
<b>forcebalance.Mol2.mol2_atom</b>	527
<b>forcebalance.Mol2.mol2_bond</b>	530
<b>forcebalance.Mol2.mol2_set</b>	538
object	
<b>forcebalance.BaseClass</b>	207
<b>forcebalance.engine.Engine</b>	290
<b>forcebalance.gmxio.GMX</b>	316
<b>forcebalance.openmmio.OpenMM</b>	626
<b>forcebalance.tinkerio.TINKER</b>	764
<b>forcebalance.forcefield.FF</b>	293

forcebalance.objective.Objective	622
forcebalance.optimizer.Optimizer	639
forcebalance.target.Target	700
forcebalance.abinitio.ABInitio	75
forcebalance.abinitio_internal.ABInitio_Internal	140
forcebalance.amberio.ABInitio_AMBER	96
forcebalance.gmxio.ABInitio_GMX	118
forcebalance.openmmio.ABInitio_OpenMM	162
forcebalance.tinkerio.ABInitio_TINKER	184
forcebalance.binding.BindingEnergy	212
forcebalance.gmxio.BindingEnergy_GMX	226
forcebalance.openmmio.BindingEnergy_OpenMM	241
forcebalance.tinkerio.BindingEnergy_TINKER	256
forcebalance.counterpoise.Counterpoise	274
forcebalance.interaction.Interaction	330
forcebalance.gmxio.Interaction_GMX	345
forcebalance.openmmio.Interaction_OpenMM	360
forcebalance.tinkerio.Interaction_TINKER	375
forcebalance.leastsq.LeastSquares	394
forcebalance.psi4io.THCDF_Psi4	715
forcebalance.lipid.Lipid	410
forcebalance.gmxio.Lipid_GMX	429
forcebalance.liquid.Liquid	447
forcebalance.gmxio.Liquid_GMX	464
forcebalance.openmmio.Liquid_OpenMM	483
forcebalance.tinkerio.Liquid_TINKER	502
forcebalance.moments.Moments	560
forcebalance.gmxio.Moments_GMX	574
forcebalance.openmmio.Moments_OpenMM	589
forcebalance.tinkerio.Moments_TINKER	604

<b>forcebalance.psi4io.RDVR3_Psi4</b>	<b>671</b>
<b>forcebalance.target.RemoteTarget</b>	<b>687</b>
<b>forcebalance.thermo.Thermo</b>	<b>732</b>
<b>forcebalance.gmxio.Thermo_GMX</b>	<b>747</b>
<b>forcebalance.vibration.Vibration</b>	<b>777</b>
<b>forcebalance.gmxio.Vibration_GMX</b>	<b>792</b>
<b>forcebalance.tinkerio.Vibration_TINKER</b>	<b>807</b>
<b>forcebalance.BaseReader</b>	<b>209</b>
<b>forcebalance.amberio.FrcMod_Reader</b>	<b>307</b>
<b>forcebalance.amberio.Mol2_Reader</b>	<b>535</b>
<b>forcebalance.custom_io.Gen_Reader</b>	<b>314</b>
<b>forcebalance.gmxio.ITP_Reader</b>	<b>390</b>
<b>forcebalance.mol2io.Mol2_Reader</b>	<b>533</b>
<b>forcebalance.openmmio.OpenMM_Reader</b>	<b>637</b>
<b>forcebalance.psi4io.GBS_Reader</b>	<b>310</b>
<b>forcebalance.psi4io.Grid_Reader</b>	<b>327</b>
<b>forcebalance.qchemio.QCIn_Reader</b>	<b>659</b>
<b>forcebalance.tinkerio.Tinker_Reader</b>	<b>773</b>
<b>forcebalance.molecule.Molecule</b>	<b>539</b>
<b>forcebalance.nifty.LineChunker</b>	<b>408</b>
<b>forcebalance.quantity.Quantity</b>	<b>662</b>
<b>forcebalance.quantity.Quantity_Density</b>	<b>665</b>
<b>forcebalance.quantity.Quantity_H_vap</b>	<b>667</b>
<b>forcebalance.thermo.Point</b>	<b>658</b>
<b>forcebalance.objective.Penalty</b>	<b>653</b>
Pickler	
<b>forcebalance.nifty.Pickler_LP</b>	<b>656</b>
StreamHandler	
<b>forcebalance.output.CleanStreamHandler</b>	<b>273</b>
<b>forcebalance.output.RawStreamHandler</b>	<b>670</b>
Structure	

<b>forcebalance.molecule.MolfileTimestep</b>	<b>559</b>
Unpickler	
<b>forcebalance.nifty.Unpickler_LP</b>	<b>776</b>

## 5 Class Index

### 5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<b>forcebalance.abinitio.AbInitio</b> Subclass of Target for fitting force fields to ab initio data	75
<b>forcebalance.amberio.AbInitio_AMBER</b> Subclass of Target for force and energy matching using AMBER	96
<b>forcebalance.gmxio.AbInitio_GMX</b> Subclass of AbInitio for force and energy matching using GROMACS	118
<b>forcebalance.abinitio_internal.AbInitio_Internal</b> Subclass of Target for force and energy matching using an internal implementation	140
<b>forcebalance.openmmio.AbInitio_OpenMM</b> Force and energy matching using OpenMM	162
<b>forcebalance.tinkerio.AbInitio_TINKER</b> Subclass of Target for force and energy matching using TINKER	184
<b>forcebalance.forcefield.BackedUpDict</b>	206
<b>forcebalance.BaseClass</b> Provides some nifty functions that are common to all ForceBalance classes	207
<b>forcebalance.BaseReader</b> The 'reader' class	209
<b>forcebalance.binding.BindingEnergy</b> Improved subclass of Target for fitting force fields to binding energies	212
<b>forcebalance.gmxio.BindingEnergy_GMX</b> Binding energy matching using Gromacs	226
<b>forcebalance.openmmio.BindingEnergy_OpenMM</b> Binding energy matching using OpenMM	241
<b>forcebalance.tinkerio.BindingEnergy_TINKER</b> Binding energy matching using TINKER	256
<b>forcebalance.output.CleanFileHandler</b> File handler that does not write terminal escape codes and carriage returns to files	271
<b>forcebalance.output.CleanStreamHandler</b> Similar to <b>RawStreamHandler</b> except it does not write terminal escape codes	273

<b>forcebalance.counterpoise.Counterpoise</b>	Target subclass for matching the counterpoise correction	274
<b>forcebalance.engine.Engine</b>	Base class for all engines	290
<b>forcebalance.forcefield.FF</b>	Force field class	293
<b>forcebalance.output.ForceBalanceLogger</b>	This logger starts out with a default handler that writes to stdout addHandler removes this default the first time another handler is added	306
<b>forcebalance.amberio.FrcMod_Reader</b>	Finite state machine for parsing FrcMod force field file	307
<b>forcebalance.psi4io.GBS_Reader</b>	Interaction type -> Parameter Dictionary	310
<b>forcebalance.custom_io.Gen_Reader</b>	Finite state machine for parsing custom GROMACS force field files	314
<b>forcebalance.gmxio.GMX</b>	Derived from Engine object for carrying out general purpose GROMACS calculations	316
<b>forcebalance.psi4io.Grid_Reader</b>	Finite state machine for parsing DVR grid files	327
<b>forcebalance.interaction.Interaction</b>	Subclass of Target for fitting force fields to interaction energies	330
<b>forcebalance.gmxio.Interaction_GMX</b>	Interaction energy matching using GROMACS	345
<b>forcebalance.openmmio.Interaction_OpenMM</b>	Interaction matching using OpenMM	360
<b>forcebalance.tinkerio.Interaction_TINKER</b>	Subclass of Target for interaction matching using TINKER	375
<b>forcebalance.gmxio.ITP_Reader</b>	Finite state machine for parsing GROMACS force field files	390
<b>forcebalance.leastsq.LeastSquares</b>	Subclass of Target for general least squares fitting	394
<b>forcebalance.nifty.LineChunker</b>		408
<b>forcebalance.lipid.Lipid</b>	Subclass of Target for lipid property matching	410
<b>forcebalance.gmxio.Lipid_GMX</b>		429
<b>forcebalance.liquid.Liquid</b>	Subclass of Target for liquid property matching	447
<b>forcebalance.gmxio.Liquid_GMX</b>		464

<b>forcebalance.openmmio.Liquid_OpenMM</b>		
Condensed phase property matching using <a href="#">OpenMM</a>		483
<b>forcebalance.tinkerio.Liquid_TINKER</b>		
Condensed phase property matching using <a href="#">TINKER</a>		502
<b>forcebalance.output.ModLogger</b>		522
<b>forcebalance.Mol2.mol2</b>		
This is to manage one <a href="#">mol2</a> series of lines on the form:		523
<b>forcebalance.Mol2.mol2_atom</b>		
This is to manage <a href="#">mol2</a> atomic lines on the form: 1 C1 5.4790 42.2880 49.5910 C.ar 1 <1> 0.0424		527
<b>forcebalance.Mol2.mol2_bond</b>		
This is to manage <a href="#">mol2</a> bond lines on the form: 1 1 2 ar		530
<b>forcebalance.mol2io.Mol2_Reader</b>		
Finite state machine for parsing <a href="#">Mol2</a> force field file		533
<b>forcebalance.amberio.Mol2_Reader</b>		
Finite state machine for parsing <a href="#">Mol2</a> force field file		535
<b>forcebalance.Mol2.mol2_set</b>		538
<b>forcebalance.molecule.Molecule</b>		
Lee-Ping's general file format conversion class		539
<b>forcebalance.molecule.MolfileTimestep</b>		
Wrapper for the timestep C structure used in molfile plugins		559
<b>forcebalance.moments.Moments</b>		
Subclass of Target for fitting force fields to multipole moments (from experiment or theory)		560
<b>forcebalance.gmxio.Moments_GMX</b>		
Multipole moment matching using GROMACS		574
<b>forcebalance.openmmio.Moments_OpenMM</b>		
Multipole moment matching using <a href="#">OpenMM</a>		589
<b>forcebalance.tinkerio.Moments_TINKER</b>		
Subclass of Target for multipole moment matching using <a href="#">TINKER</a>		604
<b>forcebalance.molecule.MyG</b>		620
<b>forcebalance.objective.Objective</b>		
Objective function		622
<b>forcebalance.openmmio.OpenMM</b>		
Derived from Engine object for carrying out general purpose <a href="#">OpenMM</a> calculations		626
<b>forcebalance.openmmio.OpenMM_Reader</b>		
Class for parsing <a href="#">OpenMM</a> force field files		637
<b>forcebalance.optimizer.Optimizer</b>		
Optimizer class		639

<b>forcebalance.objective.Penalty</b>	<b>Penalty</b> functions for regularizing the force field optimizer	653
<b>forcebalance.nifty.Pickler_LP</b>	A subclass of the python Pickler that implements pickling of <code>ElementTree</code> types	656
<b>forcebalance.thermo.Point</b>		658
<b>forcebalance.qchemio.QCIn.Reader</b>	Finite state machine for parsing Q-Chem input files	659
<b>forcebalance.quantity.Quantity</b>	Base class for thermodynamical quantity used for fitting	662
<b>forcebalance.quantity.Quantity_Density</b>		665
<b>forcebalance.quantity.Quantity_H_vap</b>		667
<b>forcebalance.output.RawFileHandler</b>	Exactly like <code>output.FileHandler</code> except it does no extra formatting before sending logging messages to the file	669
<b>forcebalance.output.RawStreamHandler</b>	Exactly like <code>output.StreamHandler</code> except it does no extra formatting before sending logging messages to the stream	670
<b>forcebalance.psi4io.RDVR3_Psi4</b>	Subclass of Target for R-DVR3 grid fitting	671
<b>forcebalance.target.RemoteTarget</b>		687
<b>forcebalance.target.Target</b>	Base class for all fitting targets	700
<b>forcebalance.psi4io.THCDF_Psi4</b>		715
<b>forcebalance.thermo.Thermo</b>	A target for fitting general experimental data sets	732
<b>forcebalance.gmxio.Thermo_GMX</b>	Thermodynamical property matching using GROMACS	747
<b>forcebalance.tinkerio.TINKER</b>	Engine for carrying out general purpose <code>TINKER</code> calculations	764
<b>forcebalance.tinkerio.Tinker_Reader</b>	Finite state machine for parsing <code>TINKER</code> force field files	773
<b>forcebalance.nifty.Unpickler_LP</b>	A subclass of the python Unpickler that implements unpickling of <code>ElementTree</code> types	776
<b>forcebalance.vibration.Vibration</b>	Subclass of Target for fitting force fields to vibrational spectra (from experiment or theory)	777
<b>forcebalance.gmxio.Vibration_GMX</b>	Vibrational frequency matching using GROMACS	792

## 6 File Index

### 6.1 File List

Here is a list of all files with brief descriptions:

<code>__init__.py</code>	822
<code>abinitio.py</code>	822
<code>abinitio_internal.py</code>	823
<code>amberio.py</code>	823
<code>binding.py</code>	823
<code>chemistry.py</code>	824
<code>contact.py</code>	824
<code>counterpoise.py</code>	825
<code>custom_io.py</code>	825
<code>engine.py</code>	825
<code>finite_difference.py</code>	826
<code>forcefield.py</code>	826
<code>gmxio.py</code>	827
<code>interaction.py</code>	828
<code>leastsq.py</code>	828
<code>lipid.py</code>	829
<code>liquid.py</code>	829
<code>Mol2.py</code>	829
<code>mol2io.py</code>	830
<code>molecule.py</code>	830
<code>moments.py</code>	832
<code>nifty.py</code>	832
<code>objective.py</code>	834
<code>openmmio.py</code>	835

<a href="#">optimizer.py</a>	836
<a href="#">output.py</a>	837
<a href="#">parser.py</a>	837
<a href="#">psi4io.py</a>	838
<a href="#">PT.py</a>	838
<a href="#">qchemio.py</a>	838
<a href="#">quantity.py</a>	839
<a href="#">target.py</a>	839
<a href="#">thermo.py</a>	840
<a href="#">tinkerio.py</a>	840
<a href="#">vibration.py</a>	841

## 7 Namespace Documentation

### 7.1 forcebalance Namespace Reference

#### Namespaces

- [abinitio](#)  
*Ab-initio fitting module (energies, forces, resp).*
- [abinitio\\_internal](#)  
*Internal implementation of energy matching (for TIP3P water only)*
- [amberio](#)  
*AMBER force field input/output.*
- [binding](#)  
*Binding energy fitting module.*
- [chemistry](#)
- [contact](#)
- [counterpoise](#)  
*Match an empirical potential to the counterpoise correction for basis set superposition error (BSSE).*
- [custom\\_io](#)  
*Custom force field parser.*
- [engine](#)
- [finite\\_difference](#)
- [forcefield](#)  
*Force field module.*
- [gmxio](#)  
*GROMACS input/output.*
- [interaction](#)  
*Interaction energy fitting module.*
- [leastsq](#)
- [lipid](#)

- **liquid**  
*Matching of lipid bulk properties.*
- **Mol2**  
*Matching of liquid bulk properties.*
- **mol2io**  
*Mol2 I/O.*
- **molecule**
- **moments**  
*Multipole moment fitting module.*
- **nifty**  
*Nifty functions, intended to be imported by any module within ForceBalance.*
- **objective**  
*ForceBalance objective function.*
- **openmmio**  
*OpenMM input/output.*
- **optimizer**  
*Optimization algorithms.*
- **output**
- **parser**  
*Input file parser for ForceBalance jobs.*
- **psi4io**  
*PSI4 force field input/output.*
- **PT**
- **qchemio**  
*Q-Chem input file parser.*
- **quantity**
- **target**
- **thermo**
- **tinkerio**  
*TINKER input/output.*
- **vibration**  
*Vibrational mode fitting module.*

## Classes

- class **BaseClass**  
*Provides some nifty functions that are common to all ForceBalance classes.*
- class **BaseReader**  
*The 'reader' class.*

## Variables

- tuple **\_\_version\_\_** = pkg\_resources.get\_distribution("forcebalance")

### 7.1.1 Variable Documentation

**string forcebalance.\_\_version\_\_ = pkg\_resources.get\_distribution("forcebalance")** Definition at line 17 of file \_\_init\_\_.py.

## 7.2 forcebalance.abinitio Namespace Reference

Ab-initio fitting module (energies, forces, resp).

### Classes

- class [AbInitio](#)

*Subclass of Target for fitting force fields to ab initio data.*

### Functions

- def [weighted\\_variance](#)

*A more generalized version of build\_objective which is callable for derivatives, but the covariance is not there anymore.*

- def [weighted\\_variance2](#)

*A bit of a hack, since we have to subtract out two mean quantities to get Hessian elements.*

- def [build\\_objective](#)

*This function builds an objective function (number) from the complicated polytensor and covariance matrices.*

### Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)

#### 7.2.1 Detailed Description

Ab-initio fitting module (energies, forces, resp).

Author

Lee-Ping Wang

Date

05/2012

#### 7.2.2 Function Documentation

**def forcebalance.abinitio.build\_objective ( SPiXi, WCiW, Z, Q0, M0, NCP1, subtract\_mean = True )**

This function builds an objective function (number) from the complicated polytensor and covariance matrices.

Definition at line 1208 of file abinitio.py.

**def forcebalance.abinitio.weighted\_variance ( SPiXi, WCiW, Z, L, R, NCP1, subtract\_mean = True )** A more generalized version of build\_objective which is callable for derivatives, but the covariance is not there anymore.

Definition at line 1178 of file abinitio.py.

**def forcebalance.abinitio.weighted\_variance2 ( SPiXi, WCiW, Z, L, R, L2, R2, NCP1, subtract\_mean = True )** A bit of a hack, since we have to subtract out two mean quantities to get Hessian elements.

Definition at line 1192 of file abinitio.py.

#### 7.2.3 Variable Documentation

**tuple forcebalance.abinitio.logger = getLogger(\_\_name\_\_)** Definition at line 24 of file abinitio.py.

## 7.3 forcebalance.abinitio\_internal Namespace Reference

Internal implementation of energy matching (for TIP3P water only)

## Classes

- class [AbInitio\\_Internal](#)

*Subclass of Target for force and energy matching using an internal implementation.*

### 7.3.1 Detailed Description

Internal implementation of energy matching (for TIP3P water only)

Author

Lee-Ping Wang

Date

04/2012

## 7.4 forcebalance.amberio Namespace Reference

AMBER force field input/output.

## Classes

- class [Mol2\\_Reader](#)  
*Finite state machine for parsing Mol2 force field file.*
- class [FrcMod\\_Reader](#)  
*Finite state machine for parsing FrcMod force field file.*
- class [AbInitio\\_AMBER](#)  
*Subclass of Target for force and energy matching using AMBER.*

## Functions

- def [is\\_mol2\\_atom](#)

## Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)
- dictionary [mol2\\_pdct](#) = {'COUL': {'Atom': [1], 8: ''}}
- dictionary [frcmod\\_pdct](#)

### 7.4.1 Detailed Description

AMBER force field input/output. This serves as a good template for writing future force matching I/O modules for other programs because it's so simple.

Author

Lee-Ping Wang

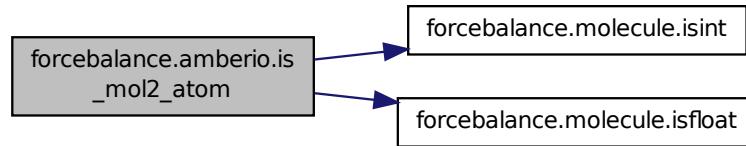
Date

01/2012

## 7.4.2 Function Documentation

**def forcebalance.amberio.is\_mol2\_atom ( *line* )** Definition at line 35 of file amberio.py.

Here is the call graph for this function:



## 7.4.3 Variable Documentation

**dictionary forcebalance.amberio.frcmod\_pdct Initial value:**

```
1 = {'BONDS': {'Atom':[0], 1:'K', 2:'B'},  
2      'ANGLES': {'Atom':[0], 1:'K', 2:'B'},  
3      'PDIHS1': {'Atom':[0], 2:'K', 3:'B'},  
4      'PDIHS2': {'Atom':[0], 2:'K', 3:'B'},  
5      'PDIHS3': {'Atom':[0], 2:'K', 3:'B'},  
6      'PDIHS4': {'Atom':[0], 2:'K', 3:'B'},  
7      'PDIHS5': {'Atom':[0], 2:'K', 3:'B'},  
8      'PDIHS6': {'Atom':[0], 2:'K', 3:'B'},  
9      'IDIHS': {'Atom':[0], 1:'K', 3:'B'},  
10     'VDW': {'Atom':[0], 1:'S', 2:'T'}  
11 }
```

Definition at line 23 of file amberio.py.

**tuple forcebalance.amberio.logger = getLogger(*\_name\_*)** Definition at line 19 of file amberio.py.

**dictionary forcebalance.amberio.mol2\_pdct = {'COUL': {'Atom': [1], 8: ''}}** Definition at line 21 of file amberio.py.

## 7.5 forcebalance.binding Namespace Reference

Binding energy fitting module.

### Classes

- class [BindingEnergy](#)

*Improved subclass of Target for fitting force fields to binding energies.*

### Functions

- def [parse\\_interactions](#)

*Parse through the interactions input file.*

### Variables

- tuple [logger = getLogger\(\*\\_name\\_\*\)](#)

### 7.5.1 Detailed Description

Binding energy fitting module.

Author

Lee-Ping Wang

Date

05/2012

### 7.5.2 Function Documentation

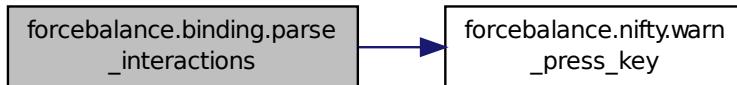
**def forcebalance.binding.parse.interactions ( *input\_file* )** Parse through the interactions input file.

Parameters

in	<i>input_file</i>	The name of the input file.
----	-------------------	-----------------------------

Definition at line 30 of file binding.py.

Here is the call graph for this function:



### 7.5.3 Variable Documentation

**tuple forcebalance.binding.logger = getLogger(\_\_name\_\_)** Definition at line 22 of file binding.py.

## 7.6 forcebalance.chemistry Namespace Reference

### Functions

- def [LookupByMass](#)
- def [BondStrengthByLength](#)

### Variables

- tuple [BondEnergies](#) = defaultdict(lambda:defaultdict(dict))
- list [Radii](#)

*Covalent radii from Cordero et al.*
- tuple [PeriodicTable](#)
- list [Elements](#)
- list [BondChars](#) = ['-','=','3']
- string [data\\_from\\_web](#)
- tuple [line](#) = line.expandtabs()
- tuple [BE](#) = float(line.split()[1])
- tuple [L](#) = float(line.split()[2])
- tuple [atoms](#) = re.split('[-=3]', line.split()[0])

- list **A** = atoms[0]
- list **B** = atoms[1]
- tuple **bo** = BondChars.index(re.findall('[-=3]', line.split()[0])[0])

### 7.6.1 Function Documentation

**def forcebalance.chemistry.BondStrengthByLength ( A, B, length, artol = 0.33, bias = 0.0 )** Definition at line 164 of file chemistry.py.

**def forcebalance.chemistry.LookupByMass ( mass )** Definition at line 155 of file chemistry.py.

### 7.6.2 Variable Documentation

**list forcebalance.chemistry.A = atoms[0]** Definition at line 149 of file chemistry.py.

**tuple forcebalance.chemistry.atoms = re.split('[-=3]', line.split()[0])** Definition at line 148 of file chemistry.py.

**list forcebalance.chemistry.B = atoms[1]** Definition at line 150 of file chemistry.py.

**tuple forcebalance.chemistry.BE = float(line.split()[1])** Definition at line 146 of file chemistry.py.

**tuple forcebalance.chemistry.bo = BondChars.index(re.findall('[-=3]', line.split()[0])[0])** Definition at line 151 of file chemistry.py.

**list forcebalance.chemistry.BondChars = [-,=,3]** Definition at line 49 of file chemistry.py.

**tuple forcebalance.chemistry.BondEnergies = defaultdict(lambda:defaultdict(dict))** Definition at line 7 of file chemistry.py.

**string forcebalance.chemistry.data\_from\_web** Definition at line 51 of file chemistry.py.

**list forcebalance.chemistry.Elements Initial value:**

```
1 = ["None",'H','He',
2           'Li','Be','B','C','N','O','F','Ne',
3           'Na','Mg','Al','Si','P','S','Cl','Ar',
4           'K','Ca','Sc','Ti','V','Cr','Mn','Fe','Co','Ni','Cu','Zn','Ga','Ge','As','Se','Br','Kr',
5           'Rb','Sr','Y','Zr','Nb','Mo','Tc','Ru','Rh','Pd','Ag','Cd','In','Sn','Sb','Te','I','Xe',
6           'Cs','Ba','La','Ce','Pr','Nd','Pm','Sm','Eu','Gd','Tb','Dy','Ho','Er','Tm','Yb',
7           'Lu','Hf','Ta','W','Re','Os','Ir','Pt','Au','Hg','Tl','Pb','Bi','Po','At','Rn',
8           'Fr','Ra','Ac','Th','Pa','U','Np','Pu','Am','Cm','Bk','Cf','Es','Fm','Md','No','Lr','Rf','Db',
     Sg,'Bh','Hs','Mt']
```

Definition at line 40 of file chemistry.py.

**tuple forcebalance.chemistry.L = float(line.split()[2])** Definition at line 147 of file chemistry.py.

**tuple forcebalance.chemistry.line = line.expandtabs()** Definition at line 145 of file chemistry.py.

### tuple forcebalance.chemistry.PeriodicTable Initial value:

```
1 = OrderedDict([( ('H', 1.0079), ('He', 4.0026),
2                 ('Li', 6.941), ('Be', 9.0122), ('B', 10.811), ('C', 12.0107), ('N', 14.0067), ('O', 15.99
3                 94), ('F', 18.9984), ('Ne', 20.1797),
4                 ('Na', 22.9897), ('Mg', 24.305), ('Al', 26.9815), ('Si', 28.0855), ('P', 30.9738), ('S',
5                 32.065), ('Cl', 35.453), ('Ar', 39.948),
6                 ('K', 39.0983), ('Ca', 40.078), ('Sc', 44.9559), ('Ti', 47.867), ('V', 50.9415), ('Cr', 5
7                 1.9961), ('Mn', 54.938), ('Fe', 55.845), ('Co', 58.9332),
8                 ('Ni', 58.6934), ('Cu', 63.546), ('Zn', 65.39), ('Ga', 69.723), ('Ge', 72.64), ('As', 74.
9                 9216), ('Se', 78.96), ('Br', 79.904), ('Kr', 83.8),
10                ('Rb', 85.4678), ('Sr', 87.62), ('Y', 88.9059), ('Zr', 91.224), ('Nb', 92.9064), ('Mo', 9
11                5.94), ('Tc', 98), ('Ru', 101.07), ('Rh', 102.9055),
12                ('Pd', 106.42), ('Ag', 107.8682), ('Cd', 112.411), ('In', 114.818), ('Sn', 118.71), ('Sb
13                ', 121.76), ('Te', 127.6), ('I', 126.9045), ('Xe', 131.293),
14                ('Cs', 132.9055), ('Ba', 137.327), ('La', 138.9055), ('Ce', 140.116), ('Pr', 140.9077),
('Nd', 144.24), ('Pm', 145), ('Sm', 150.36),
('Eu', 151.964), ('Gd', 157.25), ('Tb', 158.9253), ('Dy', 162.5), ('Ho', 164.9303), ('Er
167.259), ('Tm', 168.9342), ('Yb', 173.04),
('Lu', 174.967), ('Hf', 178.49), ('Ta', 180.9479), ('W', 183.84), ('Re', 186.207), ('Os
190.23), ('Ir', 192.217), ('Pt', 195.078),
('Au', 196.9665), ('Hg', 200.59), ('Tl', 204.3833), ('Pb', 207.2), ('Bi', 208.9804), ('Po
209), ('At', 210), ('Rn', 222),
('Fr', 223), ('Ra', 226), ('Ac', 227), ('Th', 232.0381), ('Pa', 231.0359), ('U
238.0289), ('Np', 237), ('Pu', 244),
('Am', 243), ('Cm', 247), ('Bk', 247), ('Cf', 251), ('Es', 252), ('Fm', 257), ('Md', 258),
('No', 259),
('Lr', 262), ('Rf', 261), ('Db', 262), ('Sg', 266), ('Bh', 264), ('Hs', 277), ('Mt', 268)])
```

Definition at line 25 of file chemistry.py.

### list forcebalance.chemistry.Radii Initial value:

```
1 = [0.31, 0.28, # H and He
2     1.28, 0.96, 0.84, 0.76, 0.71, 0.66, 0.57, 0.58, # First row elements
3     1.66, 1.41, 1.21, 1.11, 1.07, 1.05, 1.02, 1.06, # Second row elements
4     2.03, 1.76, 1.70, 1.60, 1.53, 1.39, 1.61, 1.52, 1.50,
5     1.24, 1.32, 1.22, 1.20, 1.19, 1.20, 1.20, 1.16, # Third row elements, K through Kr
6     2.20, 1.95, 1.90, 1.75, 1.64, 1.54, 1.47, 1.46, 1.42,
7     1.39, 1.45, 1.44, 1.42, 1.39, 1.39, 1.38, 1.39, 1.40, # Fourth row elements, Rb through Xe
8     2.44, 2.15, 2.07, 2.04, 2.03, 2.01, 1.99, 1.98,
9     1.98, 1.96, 1.94, 1.92, 1.92, 1.89, 1.90, 1.87, # Fifth row elements, s and f blocks
10    1.87, 1.75, 1.70, 1.62, 1.51, 1.44, 1.41, 1.36,
11    1.36, 1.32, 1.45, 1.46, 1.48, 1.40, 1.50, 1.50, # Fifth row elements, d and p blocks
12    2.60, 2.21, 2.15, 2.06, 2.00, 1.96, 1.90, 1.87, 1.80, 1.69]
```

Covalent radii from Cordero et al.

'Covalent radii revisited' Dalton Transactions 2008, 2832-2838.

Definition at line 10 of file chemistry.py.

## 7.7 forcebalance.contact Namespace Reference

### Functions

- def `atom_distances`

For each frame in xyzlist, compute the (euclidean) distance between pairs of atoms whos indices are given in contacts.

- def `residue_distances`

For each frame in xyzlist, and for each pair of residues in the array contact, compute the distance between the closest pair of atoms such that one of them belongs to each residue.

### 7.7.1 Function Documentation

**def forcebalance.contact.atom\_distances ( xyzlist, atom\_contacts, box = None )** For each frame in xyzlist, compute the (euclidean) distance between pairs of atoms whos indices are given in contacts.

xyzlist should be a traj\_length x num\_atoms x num\_dims array of type float32

contacts should be a num\_contacts x 2 array where each row gives the indices of 2 atoms whos distance you care to monitor.

box should be a 3-element array containing the a, b, and c lattice lengths.

Returns: traj\_length x num\_contacts array of euclidean distances

Note: For nice wrappers around this, see the prepare\_trajectory method of various metrics in metrics.py

Definition at line 29 of file contact.py.

**def forcebalance.contact.residue\_distances ( xyzlist, residue\_membership, residue\_contacts )** For each frame in xyzlist, and for each pair of residues in the array contact, compute the distance between the closest pair of atoms such that one of them belongs to each residue.

xyzlist should be a traj\_length x num\_atoms x num\_dims array of type float32

residue\_membership should be a list of lists where residue\_membership[i] gives the list of atomindices that belong to residue i. residue\_membership should NOT be a numpy 2D array unless you really mean that all of the residues have the same number of atoms

residue\_contacts should be a 2D numpy array of shape num\_contacts x 2 where each row gives the indices of the two RESIDUES who you are interested in monitoring for a contact.

Returns: a 2D array of traj\_length x num\_contacts where out[i,j] contains the distance between the pair of atoms, one from residue\_membership[residue\_contacts[j,0]] and one from residue\_membership[residue\_contacts[j,1]] that are closest.

Definition at line 99 of file contact.py.

## 7.8 forcebalance.counterpoise Namespace Reference

Match an empirical potential to the counterpoise correction for basis set superposition error (BSSE).

### Classes

- class [Counterpoise](#)

*Target subclass for matching the counterpoise correction.*

### Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)

#### 7.8.1 Detailed Description

Match an empirical potential to the counterpoise correction for basis set superposition error (BSSE). Here we test two different functional forms: a three-parameter Gaussian repulsive potential and a four-parameter Gaussian which goes smoothly to an exponential. The latter can be written in two different ways - one which gives us control over the exponential, the switching distance and the Gaussian decay constant, and another which gives us control over the Gaussian and the switching distance. They are called 'CPGAUSS', 'CPEXPG', and 'CPGEXP'. I think the third option is the best although our early tests have indicated that none of the force fields perform particularly well for the water dimer.

This subclass of Target implements the 'get' method.

Author

Lee-Ping Wang

Date

12/2011

#### 7.8.2 Variable Documentation

**tuple forcebalance.counterpoise.logger = getLogger(\_\_name\_\_)** Definition at line 30 of file counterpoise.py.

## 7.9 forcebalance.custom.io Namespace Reference

Custom force field parser.

### Classes

- class [Gen\\_Reader](#)

*Finite state machine for parsing custom GROMACS force field files.*

### Variables

- list [cptypes](#) = [None, 'CPGAUSS', 'CPEXPG', 'CPGEXP']

*Types of counterpoise correction.*

- list [ndtypes](#) = [None]

*Types of NDDO correction.*

- dictionary [fdict](#)

*Section -> Interaction type dictionary.*

- dictionary [pdict](#)

*Interaction type -> Parameter Dictionary.*

### 7.9.1 Detailed Description

Custom force field parser. We take advantage of the sections in GROMACS and the 'interaction type' concept, but these interactions are not supported in GROMACS; rather, they are computed within our program.

Author

Lee-Ping Wang

Date

12/2011

### 7.9.2 Variable Documentation

**list forcebalance.custom.io.cptypes = [None, 'CPGAUSS', 'CPEXPG', 'CPGEXP']** Types of counterpoise correction.

Definition at line 16 of file custom.io.py.

**dictionary forcebalance.custom.io.fdict Initial value:**

```
1 = {
2     'counterpoise' : cptypes }
```

Section -> Interaction type dictionary.

Definition at line 21 of file custom.io.py.

**list forcebalance.custom.io.ndtypes = [None]** Types of NDDO correction.

Definition at line 18 of file custom.io.py.

**dictionary forcebalance.custom.io.pdict Initial value:**

```
1 = {'CPGAUSS':{3:'A', 4:'B', 5:'C'},
2      'CPGEXP':{3:'A', 4:'B', 5:'G', 6:'X'},
3      'CPEXPG':{3:'A1', 4:'B', 5:'X0', 6:'A2'}
4 }
```

Interaction type -> Parameter Dictionary.

Definition at line 25 of file custom.io.py.

## 7.10 forcebalance.engine Namespace Reference

### Classes

- class [Engine](#)

*Base class for all engines.*

### Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)

#### 7.10.1 Variable Documentation

**tuple forcebalance.engine.logger = getLogger(\_\_name\_\_)** Definition at line 17 of file engine.py.

## 7.11 forcebalance.finite\_difference Namespace Reference

### Functions

- def [f1d2p](#)

*A two-point finite difference stencil.*

- def [f1d5p](#)

*A highly accurate five-point finite difference stencil for computing derivatives of a function.*

- def [f1d7p](#)

*A highly accurate seven-point finite difference stencil for computing derivatives of a function.*

- def [f12d7p](#)

- def [f12d3p](#)

*A three-point finite difference stencil.*

- def [f2var](#)

*A finite difference stencil for a function of two variables.*

- def [in\\_fd](#)

*Invoking this function from anywhere will tell us whether we're being called by a finite-difference function.*

- def [in\\_fd\\_srch](#)

*Invoking this function from anywhere will tell us whether we're being called by a finite-difference function.*

- def [fdwrap](#)

*A function wrapper for finite difference designed for differentiating 'get'-type functions.*

- def [fdwrap\\_G](#)

*A driver to fdwrap for gradients (see documentation for fdwrap) Inputs: tgt = The Target containing the objective function that we want to differentiate mvals0 = The 'central' values of the mathematical parameters - i.e.*

- def [fdwrap\\_H](#)

*A driver to fdwrap for Hessians (see documentation for fdwrap) Inputs: tgt = The Target containing the objective function that we want to differentiate mvals0 = The 'central' values of the mathematical parameters - i.e.*

### Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)

### 7.11.1 Function Documentation

**def forcebalance.finite\_difference.f12d3p ( f, h, f0 = None )** A three-point finite difference stencil.

This function does either two computations or three, depending on whether the 'center' value is supplied. This is done in order to avoid recomputing the center value many times.

The first derivative is evaluated using central difference. One advantage of using central difference (as opposed to forward difference) is that we get zero at the bottom of a parabola.

Using this formula we also get an approximate second derivative, which can then be inserted into the diagonal of the Hessian. This is very useful for optimizations like BFGS where the diagonal determines how far we step in the parameter space.

How to use: use fdwrap or something similar to generate a one-variable function from the (usually) much more complicated function that we wish to differentiate. Then pass it to this function.

Inputs: f = The one-variable function f(x) that we're differentiating h = The finite difference step size, usually a small number

Outputs: fp = The finite difference derivative of the function f(x) around x=0.

Definition at line 109 of file finite\_difference.py.

**def forcebalance.finite\_difference.f12d7p ( f, h )** Definition at line 75 of file finite\_difference.py.

**def forcebalance.finite\_difference.f1d2p ( f, h, f0 = None )** A two-point finite difference stencil.

This function does either two computations or one, depending on whether the 'center' value is supplied. This is done in order to avoid recomputing the center value many times when we repeat this function for each index of the gradient.

How to use: use fdwrap or something similar to generate a one-variable function from the (usually) much more complicated function that we wish to differentiate. Then pass it to this function.

Inputs: f = The one-variable function f(x) that we're differentiating h = The finite difference step size, usually a small number

Outputs: fp = The finite difference derivative of the function f(x) around x=0.

Definition at line 29 of file finite\_difference.py.

**def forcebalance.finite\_difference.f1d5p ( f, h )** A highly accurate five-point finite difference stencil for computing derivatives of a function.

It works on both scalar and vector functions (i.e. functions that return arrays). Since the function does four computations, it's costly but recommended if we really need an accurate reference value.

The function is evaluated at points -2h, -h, +h and +2h and these values are combined to make the derivative according to: <http://www.holoborodko.com/pavel/numerical-methods/numerical-derivative/central-difference-formulas/>

How to use: use fdwrap or something similar to generate a one-variable function from the (usually) much more complicated function that we wish to differentiate. Then pass it to this function.

Inputs: f = The one-variable function f(x) that we're differentiating h = The finite difference step size, usually a small number

Outputs: fp = The finite difference derivative of the function f(x) around x=0.

Definition at line 60 of file finite\_difference.py.

**def forcebalance.finite\_difference.f1d7p ( f, h )** A highly accurate seven-point finite difference stencil for computing derivatives of a function.

Definition at line 70 of file finite\_difference.py.

**def forcebalance.finite\_difference.f2var ( f, h )** A finite difference stencil for a function of two variables.

Definition at line 120 of file finite\_difference.py.

```
def forcebalance.finite.difference.fdwrap ( func, mvals0, pidx, key = None, kwargs )
```

A function wrapper for finite difference designed for differentiating 'get'-type functions.

Since our finite difference stencils take single-variable functions and differentiate them around zero, and our objective function is quite a complicated function, we need a wrapper to serve as a middleman. The alternative would be to copy the finite difference formula to wherever we're taking the derivative, and that is prone to mistakes.

Inputs: func = Either get\_X or get\_G; these functions return dictionaries. ['X'] = 1.23, ['G'] = [0.12, 3, 45, ...] mvals0 = The 'central' values of the mathematical parameters - i.e. the wrapped function's origin is here. pidx = The index of the parameter that we're differentiating key = either 'G' or 'X', the value we wish to take out of the dictionary kwargs = Anything else we want to pass to the objective function (for instance, Project.Objective takes Order as an argument)

Outputs: func1 = Wrapped version of func, which takes a single float argument.

Definition at line 160 of file finite\_difference.py.

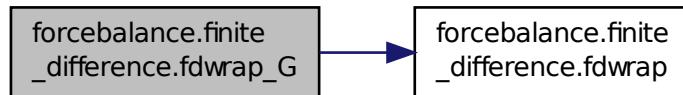
```
def forcebalance.finite.difference.fdwrap_G ( tgt, mvals0, pidx )
```

A driver to fdwrap for gradients (see documentation for fdwrap) Inputs: tgt = The Target containing the objective function that we want to differentiate mvals0 = The 'central' values of the mathematical parameters - i.e.

the wrapped function's origin is here. pidx = The index of the parameter that we're differentiating

Definition at line 179 of file finite\_difference.py.

Here is the call graph for this function:



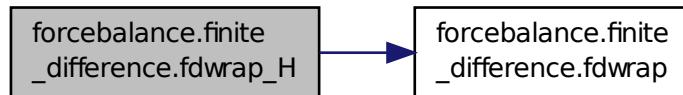
```
def forcebalance.finite.difference.fdwrap_H ( tgt, mvals0, pidx )
```

A driver to fdwrap for Hessians (see documentation for fdwrap) Inputs: tgt = The Target containing the objective function that we want to differentiate mvals0 = The 'central' values of the mathematical parameters - i.e.

the wrapped function's origin is here. pidx = The index of the parameter that we're differentiating

Definition at line 190 of file finite\_difference.py.

Here is the call graph for this function:



```
def forcebalance.finite.difference.in_fd ( )
```

Invoking this function from anywhere will tell us whether we're being called by a finite-difference function.

This is mainly useful for deciding when to update the 'qualitative indicators' and when not to.  
Definition at line 127 of file finite\_difference.py.

**def forcebalance.finite\_difference.in\_fd\_srch ( )** Invoking this function from anywhere will tell us whether we're being called by a finite-difference function.

This is mainly useful for deciding when to update the 'qualitative indicators' and when not to.  
Definition at line 134 of file finite\_difference.py.

### 7.11.2 Variable Documentation

**tuple forcebalance.finite\_difference.logger = getLogger(\_\_name\_\_)** Definition at line 7 of file finite\_difference.py.

## 7.12 forcebalance.forcefield Namespace Reference

Force field module.

### Classes

- class [BackedUpDict](#)
- class [FF](#)

*Force field class.*

### Functions

- def [determine\\_fftype](#)  
*Determine the type of a force field file.*
- def [rs\\_override](#)  
*This function takes in a dictionary (rsfactors) and a string (termtype).*

### Variables

- tuple [logger = getLogger\(\\_\\_name\\_\\_\)](#)
- dictionary [FF\\_Extensions](#)
- dictionary [FF\\_IOModules](#)

### 7.12.1 Detailed Description

Force field module. In ForceBalance a 'force field' is built from a set of files containing physical parameters. These files can be anything that enter into any computation - our original program was quite dependent on the GROMACS force field format, but this program is set up to allow very general input formats.

We introduce several important concepts:

1) Adjustable parameters are allocated into a vector.

To cast the force field optimization as a math problem, we treat all of the parameters on equal footing and write them as indices in a parameter vector.

2) A mapping from interaction type to parameter number.

Each element in the parameter vector corresponds to one or more interaction types. Whenever we change the parameter vector and recompute the objective function, this amounts to changing the physical parameters in the simulations, so we print out new force field files for external programs. In addition, when these programs are computing the objective function we are often in low-level subroutines that compute terms in the energy and force. If we need an analytic derivative of the objective function, then these subroutines need to know which index of the parameter vector needs to be modified.

This is done by way of a hash table: for example, when we are computing a Coulomb interaction between atom 4 and atom 5, we can build the words 'COUL4' and 'COUL5' and look it up in the parameter map; this gives us two

numbers (say, 10 and 11) corresponding to the eleventh and twelfth element of the parameter vector. Then we can compute the derivatives of the energy w.r.t. these parameters (in this case, COUL5/rij and COUL4/rij) and increment these values in the objective function gradient.

In custom-implemented force fields (see counterpoisematch.py) the hash table can also be used to look up parameter values for computation of interactions. This is probably not the fastest way to do things, however.

### 3) Distinction between physical and mathematical parameters.

The optimization algorithm works in a space that is related to, but not exactly the same as the physical parameter space. The reasons for why we do this are:

a) Each parameter has its own physical units. On the one hand it's not right to treat different physical units all on the same footing, so nondimensionalization is desirable. To make matters worse, the force field parameters can be small as 1e-8 or as large as 1e+6 depending on the parameter type. This means the elements of the objective function gradient / Hessian have elements that differ from each other in size by 10+ orders of magnitude, leading to mathematical instabilities in the optimizer.

b) The parameter space can be constrained, most notably for atomic partial charges where we don't want to change the overall charge on a molecule. Thus we wish to project out certain movements in the mathematical parameters such that they don't change the physical parameters.

c) We wish to regularize our optimization so as to avoid changing our parameters in very insensitive directions (linear dependencies). However, the sensitivity of the objective function to changes in the force field depends on the physical units!

For all of these reasons, we introduce a 'transformation matrix' which maps mathematical parameters onto physical parameters. The diagonal elements in this matrix are rescaling factors; they take the mathematical parameter and magnify it by this constant factor. The off-diagonal elements correspond to rotations and other linear transformations, and currently I just use them to project out the 'increase the net charge' direction in the physical parameter space.

Note that with regularization, these rescaling factors are equivalent to the widths of prior distributions in a maximum likelihood framework. Because there is such a correspondence between rescaling factors and choosing a prior, they need to be chosen carefully. This is work in progress. Another possibility is to sample the width of the priors from a noninformative distribution – the hyperprior (we can choose the Jeffreys prior or something). This is work in progress.

Right now only GROMACS parameters are supported, but this class is extensible, we need more modules!

Author

Lee-Ping Wang

Date

04/2012

## 7.12.2 Function Documentation

```
def forcebalance.forcefield.determine_fftype( ffname, verbose = False )
```

Determine the type of a force field file.

It is possible to specify the file type explicitly in the input file using the syntax 'force\_field.ext:type'. Otherwise this function will try to determine the force field type by extension.

Definition at line 146 of file forcefield.py.

```
def forcebalance.forcefield.rs_override( rsfactors, termtype, Temperature = 298.15 )
```

This function takes in a dictionary (rsfactors) and a string (termtype).

If termtype matches any of the strings below, rsfactors[termtype] is assigned to one of the numbers below.

This is LPW's attempt to simplify the rescaling factors.

Parameters

---

<code>out</code>	<code>rsfactors</code>	The computed rescaling factor.
<code>in</code>	<code>termtyp</code>	The interaction type (corresponding to a physical unit)
<code>in</code>	<code>Temperature</code>	The temperature for computing the kT energy scale

Definition at line 1212 of file forcefield.py.

### 7.12.3 Variable Documentation

**dictionary `forcebalance.forcefield.FF_Extensions` Initial value:**

```

1 = {"itp" : "gmx",
2          "top" : "gmx",
3          "in" : "qchem",
4          "prm" : "tinker",
5          "gen" : "custom",
6          "xml" : "openmm",
7          "frcmod" : "frcmod",
8          "mol2" : "mol2",
9          "gbs" : "gbs",
10         "grid" : "grid"
11        }

```

Definition at line 117 of file forcefield.py.

**dictionary `forcebalance.forcefield.FF_IOModules` Initial value:**

```

1 = {"gmx": gmxio.ITP_Reader ,
2      "qchem": qchemio.QCIn.Reader ,
3      "tinker": tinkerio.Tinker.Reader ,
4      "custom": customio.Gen.Reader ,
5      "openmm": openmmio.OpenMM.Reader,
6      "frcmod": amberio.FrcMod.Reader,
7      "mol2": amberio.Mol2.Reader,
8      "gbs": psi4io.GBS.Reader,
9      "grid": psi4io.Grid.Reader
10     }

```

Definition at line 130 of file forcefield.py.

**tuple `forcebalance.forcefield.logger = getLogger(__name__)`** Definition at line 115 of file forcefield.py.

## 7.13 forcebalance.gmxio Namespace Reference

GROMACS input/output.

### Classes

- class [ITP\\_Reader](#)  
*Finite state machine for parsing GROMACS force field files.*
- class [GMX](#)  
*Derived from Engine object for carrying out general purpose GROMACS calculations.*
- class [Liquid\\_GMX](#)
- class [Lipid\\_GMX](#)
- class [AbInitio\\_GMX](#)  
*Subclass of AbInitio for force and energy matching using GROMACS.*
- class [BindingEnergy\\_GMX](#)  
*Binding energy matching using Gromacs.*
- class [Interaction\\_GMX](#)  
*Interaction energy matching using GROMACS.*
- class [Moments\\_GMX](#)

*Multipole moment matching using GROMACS.*

- class `Vibration_GMX`

*Vibrational frequency matching using GROMACS.*

- class `Thermo_GMX`

*Thermodynamical property matching using GROMACS.*

## Functions

- def `write_mdp`

*Create or edit a Gromacs MDP file.*

- def `write_ndx`

*Create or edit a Gromacs ndx file.*

- def `parse_atomtype_line`

*Parses the 'atomtype' line.*

- def `rm_gmx_baks`

## Variables

- tuple `logger` = getLogger(\_\_name\_\_)
- list `nftypes` = [None, 'VDW', 'VDW\_BHAM']

*VdW interaction function types.*

- list `pftypes` = [None, 'VPAIR', 'VPAIR\_BHAM']

*Pairwise interaction function types.*

- list `bftypes` = [None, 'BONDS', 'G96BONDS', 'MORSE']

*Bonded interaction function types.*

- list `aftypes`

*Angle interaction function types.*

- list `dftypes` = [None, 'PDIHS', 'IDIHS', 'RBDIHS', 'PIMPDIHS', 'FOURDIHS', None, None, 'TABDIHS', 'PDIHMLS']

*Dihedral interaction function types.*

- dictionary `fdict`

*Section -> Interaction type dictionary.*

- dictionary `pdict`

*Interaction type -> Parameter Dictionary.*

### 7.13.1 Detailed Description

GROMACS input/output.

**Todo** Even more stuff from `forcefield.py` needs to go into here.

Author

Lee-Ping Wang

Date

12/2011

### 7.13.2 Function Documentation

```
def forcebalance.gmxio.parse_atomtype_line( line ) Parses the 'atomtype' line.  
Parses lines like this:  
opls_135 CT 6 12.0107 0.0000 A 3.5000e-01 2.7614e-01  
C 12.0107 0.0000 A 3.7500e-01 4.3932e-01  
Na 11 22.9897 0.0000 A 6.068128070229e+03 2.662662556402e+01 0.0000e+00 ;  
PRM 5 6
```

Look at all the variety!

#### Parameters

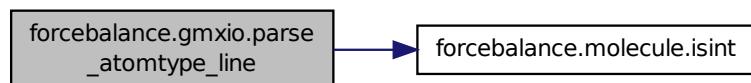
in	<i>line</i>	Input line.
----	-------------	-------------

#### Returns

answer Dictionary containing:  
atom type  
bonded atom type (if any)  
atomic number (if any)  
atomic mass  
charge  
particle type  
force field parameters  
number of optional fields

Definition at line 234 of file gmxio.py.

Here is the call graph for this function:



```
def forcebalance.gmxio.rm_gmx_baks( dir ) Definition at line 481 of file gmxio.py.
```

```
def forcebalance.gmxio.write_mdp( fout, options, fin = None, defaults = {}, verbose = False ) Create or edit a Gromacs MDP file.
```

The MDP file contains GROMACS run parameters.

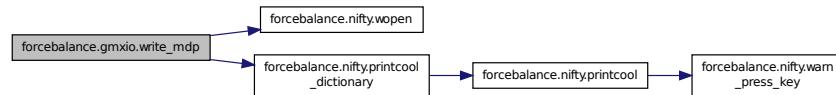
#### Parameters

in	<i>fout</i>	Output file name, can be the same as input file name.
in	<i>options</i>	Dictionary containing mdp options. Existing options are replaced, new options are added at the end.

in	<i>fin</i>	Input file name.
in	<i>defaults</i>	Default options to add to the mdp only if they don't already exist.
in	<i>verbose</i>	Print out all modifications to the file.

Definition at line 46 of file gmxio.py.

Here is the call graph for this function:

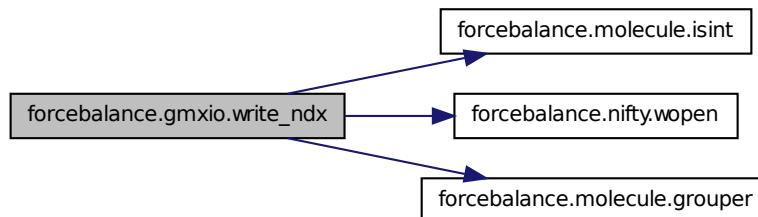


**def forcebalance.gmxio.write\_ndx ( *fout*, *grps*, *fin = None* )** Create or edit a Gromacs ndx file.  
Parameters

in	<i>fout</i>	Output file name, can be the same as input file name.
in	<i>grps</i>	Dictionary containing key : atom selections.
in	<i>fin</i>	Input file name.

Definition at line 116 of file gmxio.py.

Here is the call graph for this function:



### 7.13.3 Variable Documentation

**list forcebalance.gmxio.aftypes Initial value:**

```
1 = [None, 'ANGLES', 'G96ANGLES', 'CROSS_BOND_BOND',
2      'CROSS_BOND_ANGLE', 'UREY_BRADLEY', 'QANGLES']
```

Angle interaction function types.

Definition at line 144 of file gmxio.py.

**list forcebalance.gmxio.bftypes = [None, 'BONDS', 'G96BONDS', 'MORSE']** Bonded interaction function types.  
Definition at line 142 of file gmxio.py.

**list forcebalance.gmxio.dftypes = [None, 'PDIHS', 'IDIHS', 'RBDIHS', 'PIMPDIHS', 'FOURDIHS', None, None, 'T-ABDIHS', 'PDIHMULS']** Dihedral interaction function types.

Definition at line 147 of file gmxio.py.

**dictionary forcebalance.gmxio.fdict Initial value:**

```
1 = {
2     'atomtypes'      : nftypes,
3     'nonbond_params': pftypes,
4     'bonds'          : bftypes,
5     'bondtypes'      : bftypes,
6     'angles'         : aftytypes,
7     'angletypes'     : aftytypes,
8     'dihedrals'      : dftytypes,
9     'dihedraltypes'  : dftytypes,
10    'virtual_sites2': ['NONE','VSITE2'],
11    'virtual_sites3': ['NONE','VSITE3','VSITE3FD','VSITE3FAD','VSITE3OUT'],
12    'virtual_sites4': ['NONE','VSITE4FD','VSITE4FDN']
13 }
```

Section -> Interaction type dictionary.

Based on the section you're in and the integer given on the current line, this looks up the 'interaction type' - for example, within bonded interactions there are four interaction types: harmonic, G96, Morse, and quartic interactions.

Definition at line 155 of file gmxio.py.

**tuple forcebalance.gmxio.logger = getLogger(\_\_name\_\_)** Definition at line 35 of file gmxio.py.

**list forcebalance.gmxio.nftypes = [None, 'VDW', 'VDW\_BHAM']** VdW interaction function types.

Definition at line 138 of file gmxio.py.

**dictionary forcebalance.gmxio.pdict** Interaction type -> Parameter Dictionary.

A list of supported GROMACS interaction types in force matching. The keys in this dictionary (e.g. 'BONDS';'ANGLES') are values in the interaction type dictionary. As the program loops through the force field file, it first looks up the interaction types in 'fdict' and then goes here to do the parameter lookup by field.

**Todo** This needs to become more flexible because the parameter isn't always in the same field. Still need to figure out how to do this.

How about making the PDIHS less ugly?

Definition at line 178 of file gmxio.py.

**list forcebalance.gmxio.pftypes = [None, 'VPAIR', 'VPAIR\_BHAM']** Pairwise interaction function types.

Definition at line 140 of file gmxio.py.

## 7.14 forcebalance.interaction Namespace Reference

[Interaction](#) energy fitting module.

### Classes

- class [Interaction](#)

*Subclass of Target for fitting force fields to interaction energies.*

### Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)

### 7.14.1 Detailed Description

[Interaction](#) energy fitting module.

Author

Lee-Ping Wang

Date

05/2012

### 7.14.2 Variable Documentation

**tuple forcebalance.interaction.logger = getLogger(\_\_name\_\_)** Definition at line 21 of file interaction.py.

## 7.15 forcebalance.leastsq Namespace Reference

### Classes

- class [LeastSquares](#)

*Subclass of Target for general least squares fitting.*

### Functions

- def [CheckBasis](#)
- def [LastMvals](#)

### Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)
- [CHECK\\_BASIS](#) = False
- [LAST\\_MVALS](#) = None

### 7.15.1 Function Documentation

**def forcebalance.leastsq.CheckBasis ( )** Definition at line 24 of file leastsq.py.

**def forcebalance.leastsq.LastMvals ( )** Definition at line 29 of file leastsq.py.

### 7.15.2 Variable Documentation

**forcebalance.leastsq.CHECK\_BASIS = False** Definition at line 23 of file leastsq.py.

**forcebalance.leastsq.LAST\_MVALS = None** Definition at line 28 of file leastsq.py.

**tuple forcebalance.leastsq.logger = getLogger(\_\_name\_\_)** Definition at line 21 of file leastsq.py.

## 7.16 forcebalance.lipid Namespace Reference

Matching of lipid bulk properties.

### Classes

- class [Lipid](#)

*Subclass of Target for lipid property matching.*

## Functions

- def [weight\\_info](#)

## Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)

### 7.16.1 Detailed Description

Matching of lipid bulk properties. Under development.

author Lee-Ping Wang

Date

04/2012

### 7.16.2 Function Documentation

**def forcebalance.lipid.weight\_info ( W, PT, N\_k, verbose = True )** Definition at line 32 of file [lipid.py](#).

### 7.16.3 Variable Documentation

**tuple forcebalance.lipid.logger = getLogger(\_\_name\_\_)** Definition at line 30 of file [lipid.py](#).

## 7.17 forcebalance.liquid Namespace Reference

Matching of liquid bulk properties.

## Classes

- class [Liquid](#)

*Subclass of Target for liquid property matching.*

## Functions

- def [weight\\_info](#)

## Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)

### 7.17.1 Detailed Description

Matching of liquid bulk properties. Under development.

author Lee-Ping Wang

Date

04/2012

### 7.17.2 Function Documentation

**def forcebalance.liquid.weight\_info ( W, PT, N\_k, verbose = True )** Definition at line 32 of file [liquid.py](#).

### 7.17.3 Variable Documentation

**tuple forcebalance.liquid.logger = getLogger(\_\_name\_\_)** Definition at line 30 of file [liquid.py](#).

## 7.18 forcebalance.Mol2 Namespace Reference

### Classes

- class `mol2_atom`  
*This is to manage mol2 atomic lines on the form: 1 C1 5.4790 42.2880 49.5910 C.ar 1 <1> 0.0424.*
- class `mol2_bond`  
*This is to manage mol2 bond lines on the form: 1 1 2 ar.*
- class `mol2`  
*This is to manage one mol2 series of lines on the form:*
- class `mol2_set`

### Variables

- tuple `data = mol2_set(sys.argv[1], subset=["RNase.xray.inh8.1QHC"])`

#### 7.18.1 Variable Documentation

`tuple forcebalance.Mol2.data = mol2_set(sys.argv[1], subset=["RNase.xray.inh8.1QHC"])` Definition at line 651 of file Mol2.py.

## 7.19 forcebalance.mol2io Namespace Reference

### Mol2 I/O.

### Classes

- class `Mol2_Reader`  
*Finite state machine for parsing Mol2 force field file.*

### Variables

- dictionary `mol2_pdct = {'COUL': {'Atom': [1], 6: ''}}`

#### 7.19.1 Detailed Description

Mol2 I/O. This serves as a good template for writing future force matching I/O modules for other programs because it's so simple.

Author

Lee-Ping Wang

Date

05/2012

#### 7.19.2 Variable Documentation

`dictionary forcebalance.mol2io.mol2_pdct = {'COUL': {'Atom': [1], 6: ''}}` Definition at line 18 of file mol2io.py.

## 7.20 forcebalance.molecule Namespace Reference

### Classes

- class [MyG](#)
- class [MolfileTimestep](#)

*Wrapper for the timestep C structure used in molfile plugins.*

- class [Molecule](#)

*Lee-Ping's general file format conversion class.*

### Functions

- def [getElement](#)

- def [elem\\_from\\_atomname](#)

*Given an atom name, attempt to get the element in most cases.*

- def [nodematch](#)

- def [isint](#)

*ONLY matches integers! If you have a decimal point? None shall pass!*

- def [isfloat](#)

*Matches ANY number; it can be a decimal, scientific notation, integer, or what have you.*

- def [CubicLattice](#)

*This function takes in three lattice lengths and three lattice angles, and tries to return a complete box specification.*

- def [BuildLatticeFromLengthsAngles](#)

*This function takes in three lattice lengths and three lattice angles, and tries to return a complete box specification.*

- def [BuildLatticeFromVectors](#)

*This function takes in three lattice vectors and tries to return a complete box specification.*

- def [format\\_xyz\\_coord](#)

*Print a line consisting of (element, x, y, z) in accordance with .xyz file format.*

- def [format\\_gro\\_coord](#)

*Print a line in accordance with .gro file format, with six decimal points of precision.*

- def [format\\_xyzgen\\_coord](#)

*Print a line consisting of (element, p, q, r, s, t, ...) where (p, q, r) are arbitrary atom-wise data (this might happen, for instance, with atomic charges)*

- def [format\\_gro\\_box](#)

*Print a line corresponding to the box vector in accordance with .gro file format.*

- def [is\\_gro\\_coord](#)

*Determines whether a line contains GROMACS data or not.*

- def [is\\_charmm\\_coord](#)

*Determines whether a line contains CHARMM data or not.*

- def [is\\_gro\\_box](#)

*Determines whether a line contains a GROMACS box vector or not.*

- def [add\\_strip\\_to\\_mat](#)

- def [pvec](#)

- def [grouper](#)

*Groups a big long iterable into groups of ten or what have you.*

- def [even\\_list](#)

*Creates a list of number sequences divided as evenly as possible.*

- def [both](#)

- def [diff](#)

- def `either`
- def `EulerMatrix`  
*Constructs an Euler matrix from three Euler angles.*
- def `ComputeOverlap`  
*Computes an 'overlap' between two molecules based on some fictitious density.*
- def `AlignToDensity`  
*Computes a "overlap density" from two frames.*
- def `AlignToMoments`  
*Pre-aligns molecules to 'moment of inertia'.*
- def `get_rotate_translate`
- def `cartesian_product2`  
*Form a Cartesian product of two NumPy arrays.*
- def `main`

## Variables

- tuple `FrameVariableNames`
- tuple `AtomVariableNames` = set(['elem', 'partial\_charge', 'atomname', 'atomtype', 'tinkersuf', 'resid', 'resname', 'qcsuf', 'qm\_ghost', 'chain', 'altloc', 'icode'])
- tuple `MetaVariableNames` = set(['fnm', 'ftype', 'qcrems', 'qctemplate', 'qcerr', 'charge', 'mult', 'bonds'])
- tuple `QuantumVariableNames` = set(['qcrems', 'qctemplate', 'charge', 'mult', 'qcsuf', 'qm\_ghost'])
- `AllVariableNames` = `QuantumVariableNames|AtomVariableNames|MetaVariableNames|FrameVariableNames`
- list `Radii`
- list `Elements`
- tuple `PeriodicTable`
- float `bohrang` = 0.529177249  
*One bohr equals this many angstroms.*
- tuple `splitter` = re.compile(r'(\s+|\S+)')
- tuple `Box` = namedtuple('Box',[‘a’,‘b’,‘c’,‘alpha’,‘beta’,‘gamma’,‘A’,‘B’,‘C’,‘V’])
- int `radian` = 180
- int `have_contact` = 0

### 7.20.1 Function Documentation

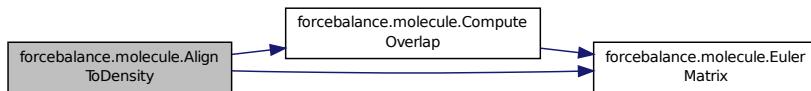
`def forcebalance.molecule.add_strip_to_mat ( mat, strip )` Definition at line 466 of file molecule.py.

`def forcebalance.molecule.AlignToDensity ( elem, xyz1, xyz2, binary = False )` Computes a "overlap density" from two frames.

This function can be called by AlignToMoments to get rid of inversion problems

Definition at line 572 of file molecule.py.

Here is the call graph for this function:



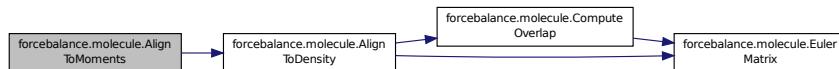
```
def forcebalance.molecule.AlignToMoments( elem, xyz1, xyz2 = None )
```

Pre-aligns molecules to 'moment of inertia'.

If xyz2 is passed in, it will assume that xyz1 is already aligned to the moment of inertia, and it simply does 180-degree rotations to make sure nothing is inverted.

Definition at line 584 of file molecule.py.

Here is the call graph for this function:



```
def forcebalance.molecule.both( A, B, key )
```

Definition at line 504 of file molecule.py.

```
def forcebalance.molecule.BuildLatticeFromLengthsAngles( a, b, c, alpha, beta, gamma )
```

This function takes in three lattice lengths and three lattice angles, and tries to return a complete box specification.

Definition at line 282 of file molecule.py.

```
def forcebalance.molecule.BuildLatticeFromVectors( v1, v2, v3 )
```

This function takes in three lattice vectors and tries to return a complete box specification.

Definition at line 297 of file molecule.py.

```
def forcebalance.molecule.cartesian_product2( arrays )
```

Form a Cartesian product of two NumPy arrays.

Definition at line 645 of file molecule.py.

```
def forcebalance.molecule.ComputeOverlap( theta, elem, xyz1, xyz2 )
```

Computes an 'overlap' between two molecules based on some fictitious density.

Good for fine-tuning alignment but gets stuck in local minima.

Definition at line 555 of file molecule.py.

Here is the call graph for this function:



```
def forcebalance.molecule.CubicLattice( a )
```

This function takes in three lattice lengths and three lattice angles, and tries to return a complete box specification.

Definition at line 262 of file molecule.py.

```
def forcebalance.molecule.diff( A, B, key )
```

Definition at line 507 of file molecule.py.

```
def forcebalance.molecule.either( A, B, key )
```

Definition at line 518 of file molecule.py.

**def forcebalance.molecule.elem\_from\_atomname ( *atomname* )** Given an atom name, attempt to get the element in most cases.

Definition at line 194 of file molecule.py.

**def forcebalance.molecule.EulerMatrix ( *T1*, *T2*, *T3* )** Constructs an Euler matrix from three Euler angles.

Definition at line 527 of file molecule.py.

**def forcebalance.molecule.even\_list ( *totlen*, *splitsize* )** Creates a list of number sequences divided as evenly as possible.

Definition at line 486 of file molecule.py.

**def forcebalance.molecule.format\_gro\_box ( *box* )** Print a line corresponding to the box vector in accordance with .gro file format.

Parameters

in	<i>box</i>	Box NamedTuple
----	------------	----------------

Definition at line 417 of file molecule.py.

**def forcebalance.molecule.format\_gro\_coord ( *resid*, *resname*, *aname*, *seqno*, *xyz* )** Print a line in accordance with .gro file format, with six decimal points of precision.

Nine decimal points of precision are necessary to get forces below 1e-3 kJ/mol/nm.

Parameters

in	<i>resid</i>	The number of the residue that the atom belongs to
in	<i>resname</i>	The name of the residue that the atom belongs to
in	<i>aname</i>	The name of the atom
in	<i>seqno</i>	The sequential number of the atom
in	<i>xyz</i>	A 3-element array containing x, y, z coordinates of that atom

Definition at line 396 of file molecule.py.

**def forcebalance.molecule.format\_xyz\_coord ( *element*, *xyz*, *tinker = False* )** Print a line consisting of (element, x, y, z) in accordance with .xyz file format.

Parameters

in	<i>element</i>	A chemical element of a single atom
in	<i>xyz</i>	A 3-element array containing x, y, z coordinates of that atom

Definition at line 378 of file molecule.py.

**def forcebalance.molecule.format\_xyzgen\_coord ( *element*, *xyzgen* )** Print a line consisting of (element, p, q, r, s, t, ...) where (p, q, r) are arbitrary atom-wise data (this might happen, for instance, with atomic charges)

Parameters

in	<i>element</i>	A chemical element of a single atom
in	<i>xyzgen</i>	A N-element array containing data for that atom

Definition at line 408 of file molecule.py.

**def forcebalance.molecule.get\_rotate\_translate ( *matrix1*, *matrix2* )** Definition at line 607 of file molecule.py.

**def forcebalance.molecule.getElement ( *mass* )** Definition at line 189 of file molecule.py.

**def forcebalance.molecule.grouper ( *n*, *iterable* )** Groups a big long iterable into groups of ten or what have you.

Definition at line 480 of file molecule.py.

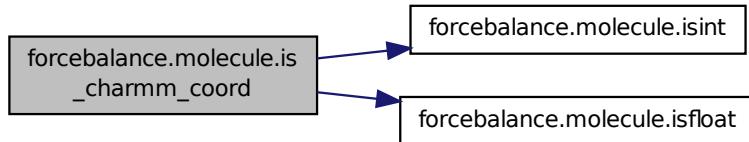
**def forcebalance.molecule.is\_charmm\_coord ( *line* )** Determines whether a line contains CHARMM data or not.

Parameters

in	<i>line</i>	The line to be tested
----	-------------	-----------------------

Definition at line 444 of file molecule.py.

Here is the call graph for this function:



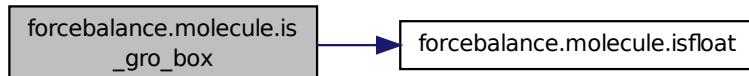
**def forcebalance.molecule.is\_gro\_box ( *line* )** Determines whether a line contains a GROMACS box vector or not.

Parameters

in	<i>line</i>	The line to be tested
----	-------------	-----------------------

Definition at line 457 of file molecule.py.

Here is the call graph for this function:



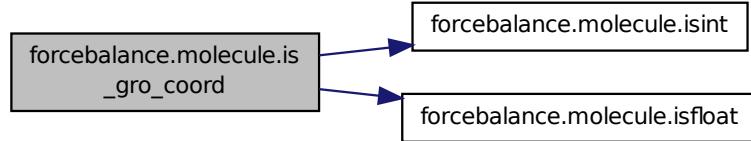
**def forcebalance.molecule.is\_gro\_coord ( *line* )** Determines whether a line contains GROMACS data or not.

Parameters

in	<i>line</i>	The line to be tested
----	-------------	-----------------------

Definition at line 429 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.isfloat ( word )** Matches ANY number; it can be a decimal, scientific notation, integer, or what have you.

Definition at line 251 of file molecule.py.

**def forcebalance.molecule.isint ( word )** ONLY matches integers! If you have a decimal point? None shall pass!

Definition at line 246 of file molecule.py.

**def forcebalance.molecule.main ( )** Definition at line 3009 of file molecule.py.

**def forcebalance.molecule.nodematch ( node1, node2 )** Definition at line 240 of file molecule.py.

**def forcebalance.molecule.pvec ( vec )** Definition at line 475 of file molecule.py.

## 7.20.2 Variable Documentation

**forcebalance.molecule.AllVariableNames = QuantumVariableNames|AtomVariableNames|MetaVariableNames|FrameVariableNames** Definition at line 137 of file molecule.py.

**tuple forcebalance.molecule.AtomVariableNames = set(['elem', 'partial\_charge', 'atomname', 'atomtype', 'tinkersuf', 'resid', 'resname', 'qcsuf', 'qm\_ghost', 'chain', 'altloc', 'icode'])** Definition at line 120 of file molecule.py.

**float forcebalance.molecule.bohrang = 0.529177249** One bohr equals this many angstroms.

Definition at line 238 of file molecule.py.

**tuple forcebalance.molecule.Box = namedtuple('Box',[‘a’,‘b’,‘c’,‘alpha’,‘beta’,‘gamma’,‘A’,‘B’,‘C’,‘V’])** Definition at line 258 of file molecule.py.

**list forcebalance.molecule.Elements Initial value:**

```
1 = ["None",'H','He',
2   'Li','Be','B','C','N','O','F','Ne',
3   'Na','Mg','Al','Si','P','S','Cl','Ar',
4   'K','Ca','Sc','Ti','V','Cr','Mn','Fe','Co','Ni','Cu','Zn','Ga','Ge','As','Se','Br','Kr',
5   'Rb','Sr','Y','Zr','Nb','Mo','Tc','Ru','Rh','Pd','Ag','Cd','In','Sn','Sb','Te','I','Xe',
6   'Cs','Ba','La','Ce','Pr','Nd','Pm','Sm','Eu','Gd','Tb','Dy','Ho','Er','Tm','Yb',
7   'Lu','Hf','Ta','W','Re','Os','Ir','Pt','Au','Hg','Tl','Pb','Bi','Po','At','Rn',
8   'Fr','Ra','Ac','Th','Pa','U','Np','Pu','Am','Cm','Bk','Cf','Es','Fm','Md','No','Lr','Rf','Db',
  Sg,'Bh','Hs','Mt']
```

Definition at line 164 of file molecule.py.

**tuple forcebalance.molecule.FrameVariableNames Initial value:**

```
1 = set(['xyzs', 'comms', 'boxes', 'qm_hessians', 'qm_forces', 'qm_energies', 'qm_interaction',
2       'qm_espxyzs', 'qm_espvals', 'qm_extchgs', 'qm_mulliken_charges', 'qm_mulliken_spins'])
```

Definition at line 106 of file molecule.py.

**int forcebalance.molecule.have\_contact = 0** Definition at line 321 of file molecule.py.

**tuple forcebalance.molecule.MetaVariableNames = set(['fnm', 'ftype', 'qcrems', 'qctemplate', 'qcerr', 'charge', 'mult', 'bonds'])** Definition at line 133 of file molecule.py.

**tuple forcebalance.molecule.PeriodicTable Initial value:**

```
1 = OrderedDict([(('H', 1.0079), ('He', 4.0026),
2                 ('Li', 6.941), ('Be', 9.0122), ('B', 10.811), ('C', 12.0107), ('N', 14.00
3                 67), ('O', 15.9994), ('F', 18.9984), ('Ne', 20.1797),
4                 ('Na', 22.9897), ('Mg', 24.305), ('Al', 26.9815), ('Si', 28.0855), ('P',
5                 30.9738), ('S', 32.065), ('Cl', 35.453), ('Ar', 39.948),
6                 ('K', 39.0983), ('Ca', 40.078), ('Sc', 44.9559), ('Ti', 47.867), ('V',
7                 50.9415), ('Cr', 51.9961), ('Mn', 54.938), ('Fe', 55.845), ('Co', 58.9332),
8                 ('Ni', 58.6934), ('Cu', 63.546), ('Zn', 65.39), ('Ga', 69.723), ('Ge',
9                 72.64), ('As', 74.9216), ('Se', 78.96), ('Br', 79.904), ('Kr', 83.8),
10                ('Rb', 85.4678), ('Sr', 87.62), ('Y', 88.9059), ('Zr', 91.224), ('Nb',
11                92.9064), ('Mo', 95.94), ('Tc', 98), ('Ru', 101.07), ('Rh', 102.9055),
12                ('Pd', 106.42), ('Ag', 107.8682), ('Cd', 112.411), ('In', 114.818), ('Sn',
13                118.71), ('Sb', 121.76), ('Te', 127.6), ('I', 126.9045), ('Xe', 131.293),
14                ('Cs', 132.9055), ('Ba', 137.327), ('La', 138.9055), ('Ce', 140.116), ('Pr',
15                140.9077), ('Nd', 144.24), ('Pm', 145), ('Sm', 150.36),
16                ('Eu', 151.964), ('Gd', 157.25), ('Tb', 158.9253), ('Dy', 162.5), ('Ho',
17                164.9303), ('Er', 167.259), ('Tm', 168.9342), ('Yb', 173.04),
18                ('Lu', 174.967), ('Hf', 178.49), ('Ta', 180.9479), ('W', 183.84), ('Re',
19                186.207), ('Os', 190.23), ('Ir', 192.217), ('Pt', 195.078),
20                ('Au', 196.9665), ('Hg', 200.59), ('Tl', 204.3833), ('Pb', 207.2), ('Bi',
21                208.9804), ('Po', 209), ('At', 210), ('Rn', 222),
22                ('Fr', 223), ('Ra', 226), ('Ac', 227), ('Th', 232.0381), ('Pa', 231.0359),
23                ('U', 238.0289), ('Np', 237), ('Pu', 244),
24                ('Am', 243), ('Cm', 247), ('Bk', 247), ('Cf', 251), ('Es', 252), ('Fm',
25                257), ('Md', 258), ('No', 259),
26                ('Lr', 262), ('Rf', 261), ('Db', 262), ('Sg', 266), ('Bh', 264), ('Hs',
27                277), ('Mt', 268)])
```

Definition at line 174 of file molecule.py.

**tuple forcebalance.molecule.QuantumVariableNames = set(['qcrems', 'qctemplate', 'charge', 'mult', 'qcsuf', 'qm\_ghost'])** Definition at line 135 of file molecule.py.

**int forcebalance.molecule.radian = 180** Definition at line 259 of file molecule.py.

**list forcebalance.molecule.Radii Initial value:**

```
1 = [0.31, 0.28, # H and He
2     1.28, 0.96, 0.84, 0.76, 0.71, 0.66, 0.57, 0.58, # First row elements
3     1.66, 1.41, 1.21, 1.11, 1.07, 1.05, 1.02, 1.06, # Second row elements
4     2.03, 1.76, 1.70, 1.60, 1.53, 1.39, 1.61, 1.52, 1.50,
5     1.24, 1.32, 1.22, 1.20, 1.19, 1.20, 1.20, 1.16, # Third row elements, K through Kr
6     2.20, 1.95, 1.90, 1.75, 1.64, 1.54, 1.47, 1.46, 1.42,
7     1.39, 1.45, 1.44, 1.42, 1.39, 1.39, 1.38, 1.39, 1.40, # Fourth row elements, Rb through Xe
8     2.44, 2.15, 2.07, 2.04, 2.03, 2.01, 1.99, 1.98,
9     1.98, 1.96, 1.94, 1.92, 1.92, 1.89, 1.90, 1.87, # Fifth row elements, s and f blocks
10    1.87, 1.75, 1.70, 1.62, 1.51, 1.44, 1.41, 1.36,
11    1.36, 1.32, 1.45, 1.46, 1.48, 1.40, 1.50, 1.50, # Fifth row elements, d and p blocks
12    2.60, 2.21, 2.15, 2.06, 2.00, 1.96, 1.90, 1.87, 1.80, 1.69]
```

Definition at line 150 of file molecule.py.

**tuple forcebalance.molecule.splitter = re.compile(r'(\s+|\S+)')** Definition at line 255 of file molecule.py.

## 7.21 forcebalance.moments Namespace Reference

Multipole moment fitting module.

### Classes

- class [Moments](#)

*Subclass of Target for fitting force fields to multipole moments (from experiment or theory).*

### Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)

#### 7.21.1 Detailed Description

Multipole moment fitting module.

Author

Lee-Ping Wang

Date

09/2012

#### 7.21.2 Variable Documentation

[tuple forcebalance.moments.logger = getLogger\(\\_\\_name\\_\\_\)](#) Definition at line 22 of file moments.py.

## 7.22 forcebalance.nifty Namespace Reference

Nifty functions, intended to be imported by any module within ForceBalance.

### Classes

- class [Pickler\\_LP](#)  
*A subclass of the python Pickler that implements pickling of \_ElementTree types.*
- class [Unpickler\\_LP](#)  
*A subclass of the python Unpickler that implements unpickling of \_ElementTree types.*
- class [LineChunker](#)

### Functions

- def [pvec1d](#)  
*Printout of a 1-D vector.*
- def [astr](#)  
*Write an array to a string so we can use it to key a dictionary.*
- def [pmat2d](#)  
*Printout of a 2-D matrix.*
- def [grouper](#)
- def [encode](#)
- def [segments](#)
- def [commadash](#)
- def [uncommadash](#)

- def `extract_int`  
*Get the representative integer value from an array.*
- def `printcool`  
*Cool-looking printout for slick formatting of output.*
- def `printcool_dictionary`  
*See documentation for printcool; this is a nice way to print out keys/values in a dictionary.*
- def `isint`  
*ONLY matches integers! If you have a decimal point? None shall pass!*
- def `isfloat`  
*Matches ANY number; it can be a decimal, scientific notation, what have you CAUTION - this will also match an integer.*
- def `isdecimal`  
*Matches things with a decimal only; see isint and isfloat.*
- def `floatornan`  
*Returns a big number if we encounter NaN.*
- def `col`  
*Given any list, array, or matrix, return a 1-column matrix.*
- def `row`  
*Given any list, array, or matrix, return a 1-row matrix.*
- def `flat`  
*Given any list, array, or matrix, return a single-index array.*
- def `monotonic`
- def `orthogonalize`  
*Given two vectors vec1 and vec2, project out the component of vec1 that is along the vec2-direction.*
- def `invert_svd`  
*Invert a matrix using singular value decomposition.*
- def `get_least_squares`
- def `statisticalinefficiency`  
*Compute the (cross) statistical inefficiency of (two) timeseries.*
- def `multiD_statisticalinefficiency`
- def `lp_dump`  
*Use this instead of pickle.dump for pickling anything that contains \_ElementTree types.*
- def `lp_load`  
*Use this instead of pickle.load for unpickling anything that contains \_ElementTree types.*
- def `getWorkQueue`
- def `getWQIds`
- def `createWorkQueue`
- def `destroyWorkQueue`
- def `queue_up`  
*Submit a job to the Work Queue.*
- def `queue_up_src_dest`  
*Submit a job to the Work Queue.*
- def `wq_wait1`  
*This function waits ten seconds to see if a task in the Work Queue has finished.*
- def `wq_wait`  
*This function waits until the work queue is completely empty.*
- def `click`  
*Stopwatch function for timing.*

- def `bak`
- def `onefile`
- def `Golnto`
- def `allsplit`
- def `Leave`
- def `MissingFileInspection`
- def `wopen`

*If trying to write to a symbolic link, remove it first.*
- def `LinkFile`
- def `CopyFile`
- def `link_dir_contents`
- def `remove_if_exists`

*Remove the file if it exists (doesn't return an error).*
- def `which`
- def `warn_press_key`
- def `warn_once`

*Prints a warning but will only do so once in a given run.*
- def `concurrent_map`

*Similar to the builtin function map().*

## Variables

- tuple `logger` = getLogger(\_\_name\_\_)
- float `kb` = 0.0083144100163

*Boltzmann constant.*
- float `eqcgmx` = 2625.5002

*Q-Chem to GMX unit conversion for energy.*
- float `fqcgmx` = -49621.9

*Q-Chem to GMX unit conversion for force.*
- float `bohrang` = 0.529177249

*One bohr equals this many angstroms.*
- string `XMLFILE` = 'x'

*Pickle uses 'flags' to pickle and unpickle different variable types.*
- `WORK_QUEUE` = None
- tuple `WQIDS` = defaultdict(list)
- list `specific_lst`
- tuple `specific_dct` = dict(list(itertools.chain(\*[[j,i[1]] for j in i[0]] for i in `specific_lst`))))

### 7.22.1 Detailed Description

Nifty functions, intended to be imported by any module within ForceBalance. Table of Contents:

- I/O formatting
- Math: Variable manipulation, linear algebra, least squares polynomial fitting
- Pickle: Expand Python's own pickle to accommodate writing XML etree objects
- Commands for submitting things to the Work Queue
- Various file and process management functions
- Development stuff (not commonly used)

Named after the mighty Sniffy Handy Nifty (King Sniffy)

Author

Lee-Ping Wang

Date

12/2011

## 7.22.2 Function Documentation

**def forcebalance.nifty.allsplit ( Dir )** Definition at line 852 of file nifty.py.

**def forcebalance.nifty.astr ( vec1d, precision = 4 )** Write an array to a string so we can use it to key a dictionary.  
Definition at line 60 of file nifty.py.

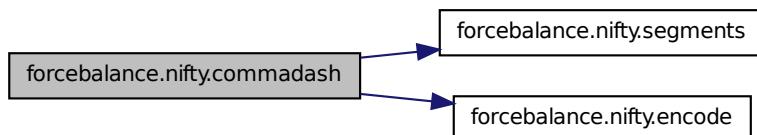
**def forcebalance.nifty.bak ( path, dest = None )** Definition at line 803 of file nifty.py.

**def forcebalance.nifty.click ( )** Stopwatch function for timing.  
Definition at line 796 of file nifty.py.

**def forcebalance.nifty.col ( vec )** Given any list, array, or matrix, return a 1-column matrix.  
Input: vec = The input vector that is to be made into a column  
Output: A column matrix  
Definition at line 302 of file nifty.py.

**def forcebalance.nifty.commadash ( I )** Definition at line 91 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.concurrent\_map ( func, data )** Similar to the builtin function map().

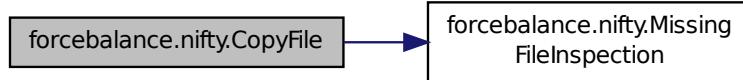
But spawn a thread for each argument and apply `func` concurrently.

Note: unlike `map()`, we cannot take an iterable argument. `data` should be an indexable sequence.

Definition at line 1127 of file nifty.py.

**def forcebalance.nifty.CopyFile ( src, dest )** Definition at line 910 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.createWorkQueue ( wq\_port, debug = True )** Definition at line 643 of file nifty.py.

**def forcebalance.nifty.destroyWorkQueue ( )** Definition at line 652 of file nifty.py.

**def forcebalance.nifty.encode ( I )** Definition at line 82 of file nifty.py.

**def forcebalance.nifty.extract\_int ( arr, avgthre, limthre, label = "value", verbose = True )** Get the representative integer value from an array.

Sanity check: Make sure the value does not go through big excursions. The integer value is the rounded value of the average. thresh = A threshold to make sure we're not dealing with fluctuations that are too large.

Definition at line 140 of file nifty.py.

**def forcebalance.nifty.flat ( vec )** Given any list, array, or matrix, return a single-index array.

Parameters

in	vec	The data to be flattened
----	-----	--------------------------

Returns

answer The flattened data

Definition at line 321 of file nifty.py.

**def forcebalance.nifty.floatornan ( word )** Returns a big number if we encounter NaN.

Parameters

in	word	The string to be converted
----	------	----------------------------

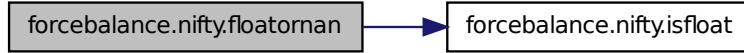
Returns

answer The string converted to a float; if not a float, return 1e10

**Todo** I could use suggestions for making this better.

Definition at line 284 of file nifty.py.

Here is the call graph for this function:



```
def forcebalance.nifty.get_least_squares( x, y, w=None, thresh=1e-12 )
```

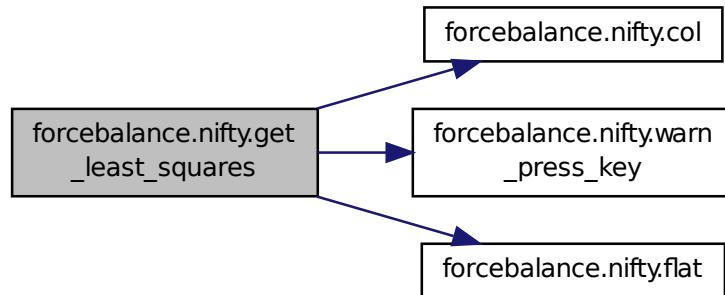
```
1 -- -- |  
2 | 1 (x0) (x0)^2 (x0)^3 |  
3 | 1 (x1) (x1)^2 (x1)^3 |  
4 | 1 (x2) (x2)^2 (x2)^3 |  
5 | 1 (x3) (x3)^2 (x3)^3 |  
6 | 1 (x4) (x4)^2 (x4)^3 |  
7 | -- |  
8 | -- |
```

#### Parameters

in	X	(2-D array) An array of X-values (see above)
in	Y	(array) An array of Y-values (only used in getting the least squares coefficients)
in	w	(array) An array of weights, hopefully normalized to one.
out	Beta	The least-squares coefficients
out	Hat	The hat matrix that takes linear combinations of data y-values to give fitted y-values (weights)
out	yfit	The fitted y-values
out	MPPI	The Moore-Penrose pseudoinverse (multiply by Y to get least-squares coefficients, multiply by dY/dk to get derivatives of least-squares coefficients)

Definition at line 410 of file nifty.py.

Here is the call graph for this function:



```
def forcebalance.nifty.getWorkQueue( ) Definition at line 635 of file nifty.py.
```

```

def forcebalance.nifty.getWQIds ( ) Definition at line 639 of file nifty.py.

def forcebalance.nifty.GoInto ( Dir ) Definition at line 844 of file nifty.py.

def forcebalance.nifty.grouper ( iterable, n ) Definition at line 75 of file nifty.py.

def forcebalance.nifty.invert_svd ( X, thresh = 1e-12 ) Invert a matrix using singular value decomposition.
Parameters

```

in	<i>X</i>	The matrix to be inverted
in	<i>thresh</i>	The SVD threshold; eigenvalues below this are not inverted but set to zero

Returns

*Xt* The inverted matrix

Definition at line 369 of file nifty.py.

```

def forcebalance.nifty.isdecimal ( word ) Matches things with a decimal only; see isint and isfloat.
Parameters

```

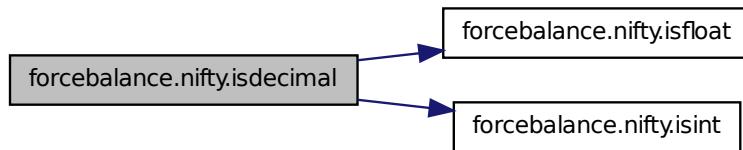
in	<i>word</i>	String (for instance, '123', '153.0', '2.', '-354')
----	-------------	---

Returns

*answer* Boolean which specifies whether the string is a number with a decimal point

Definition at line 274 of file nifty.py.

Here is the call graph for this function:



```

def forcebalance.nifty.isfloat ( word ) Matches ANY number; it can be a decimal, scientific notation, what have you
CAUTION - this will also match an integer.
Parameters

```

in	<i>word</i>	String (for instance, '123', '153.0', '2.', '-354')
----	-------------	---

Returns

*answer* Boolean which specifies whether the string is any number

Definition at line 264 of file nifty.py.

```

def forcebalance.nifty.isint ( word ) ONLY matches integers! If you have a decimal point? None shall pass!

```

#### Parameters

in	word	String (for instance, '123', '153.0', '2.', '-354')
----	------	---

#### Returns

answer Boolean which specifies whether the string is an integer (only +/- sign followed by digits)

Definition at line 253 of file nifty.py.

**def forcebalance.nifty.Leave ( Dir )** Definition at line 858 of file nifty.py.

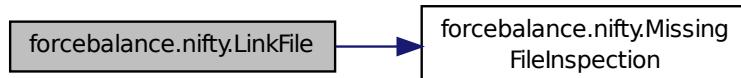
Here is the call graph for this function:



**def forcebalance.nifty.link\_dir\_contents ( abssrcdir, absdestdir )** Definition at line 920 of file nifty.py.

**def forcebalance.nifty.LinkFile ( src, dest, nosrcok = False )** Definition at line 897 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.lp\_dump ( obj, file, protocol = None )** Use this instead of pickle.dump for pickling anything that contains \_ElementTree types.

Definition at line 613 of file nifty.py.

**def forcebalance.nifty.lp\_load ( file )** Use this instead of pickle.load for unpickling anything that contains \_ElementTree types.

Definition at line 618 of file nifty.py.

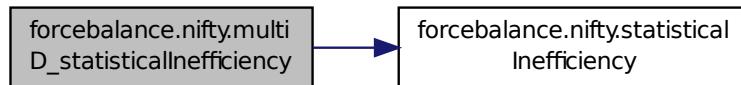
**def forcebalance.nifty.MissingFileInspection ( fnm )** Definition at line 879 of file nifty.py.

**def forcebalance.nifty.monotonic ( arr, start, end )** Definition at line 324 of file nifty.py.

```
def forcebalance.nifty.multiD_statisticalInefficiency ( A_n, B_n = None, fast = False, mintime = 3, warn = True )
```

Definition at line 539 of file nifty.py.

Here is the call graph for this function:



```
def forcebalance.nifty.onefile ( ext, arg = None )
```

Definition at line 826 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.orthogonalize ( vec1, vec2 )** Given two vectors vec1 and vec2, project out the component of vec1 that is along the vec2-direction.

Parameters

in	vec1	The projectee (i.e. output is some modified version of vec1)
in	vec2	The projector (component subtracted out from vec1 is parallel to this)

Returns

answer A copy of vec1 but with the vec2-component projected out.

Definition at line 356 of file nifty.py.

```
def forcebalance.nifty.pmat2d ( mat2d, precision = 1, format = "e", loglevel = INFO )
```

Printout of a 2-D matrix.

Parameters

in	mat2d	a 2-D matrix
----	-------	--------------

Definition at line 68 of file nifty.py.

```
def forcebalance.nifty.printcool ( text, sym = "#", bold = False, color = 2, ansi = None, bottom = '-' , minwidth = 50, center = True, sym2 = "=" )
```

Cool-looking printout for slick formatting of output.

#### Parameters

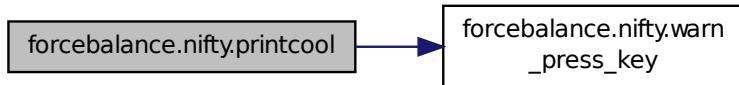
in	<i>text</i>	The string that the printout is based upon. This function will print out the string, ANSI-colored and enclosed in the symbol for example: ##### I am cool #####
in	<i>sym</i>	The surrounding symbol
in	<i>bold</i>	Whether to use bold print
in	<i>color</i>	The ANSI color: 1 red 2 green 3 yellow 4 blue 5 magenta 6 cyan 7 white
in	<i>bottom</i>	The symbol for the bottom bar
in	<i>minwidth</i>	The minimum width for the box, if the text is very short then we insert the appropriate number of padding spaces

#### Returns

*bar* The bottom bar is returned for the user to print later, e.g. to mark off a 'section'

Definition at line 184 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.printcool\_dictionary( *Dict*, *title* = "General options", *bold* = False, *color* = 2, *keywidth* = 25, *topwidth* = 50, *center* = True, *leftpad* = 0 )** See documentation for printcool; this is a nice way to print out keys/values in a dictionary.

The keys in the dictionary are sorted before printing out.

#### Parameters

in	<i>dict</i>	The dictionary to be printed
in	<i>title</i>	The title of the printout

Definition at line 229 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.pvec1d ( *vec1d*, *precision* = 1, *format* = "e", *loglevel* = INFO )** Printout of a 1-D vector.

Parameters

in	<i>vec1d</i>	a 1-D vector
----	--------------	--------------

Definition at line 52 of file nifty.py.

**def forcebalance.nifty.queue.up ( *wq*, *command*, *input\_files*, *output\_files*, *tag* = None, *tgt* = None, *verbose* = True )** Submit a job to the Work Queue.

Parameters

in	<i>wq</i>	(Work Queue Object)
in	<i>command</i>	(string) The command to run on the remote worker.
in	<i>input_files</i>	(list of files) A list of locations of the input files.
in	<i>output_files</i>	(list of files) A list of locations of the output files.

Definition at line 667 of file nifty.py.

**def forcebalance.nifty.queue.up\_src\_dest ( *wq*, *command*, *input\_files*, *output\_files*, *tag* = None, *tgt* = None, *verbose* = True )** Submit a job to the Work Queue.

This function is a bit fancier in that we can explicitly specify where the input files come from, and where the output files go to.

Parameters

in	<i>wq</i>	(Work Queue Object)
in	<i>command</i>	(string) The command to run on the remote worker.
in	<i>input_files</i>	(list of 2-tuples) A list of local and remote locations of the input files.
in	<i>output_files</i>	(list of 2-tuples) A list of local and remote locations of the output files.

Definition at line 700 of file nifty.py.

**def forcebalance.nifty.remove\_if\_exists ( *fnm* )** Remove the file if it exists (doesn't return an error).

Definition at line 931 of file nifty.py.

**def forcebalance.nifty.row ( *vec* )** Given any list, array, or matrix, return a 1-row matrix.

Parameters

in	<i>vec</i>	The input vector that is to be made into a row
----	------------	--

Returns

answer A row matrix

Definition at line 312 of file nifty.py.

**def forcebalance.nifty.segments ( e )** Definition at line 85 of file nifty.py.

**def forcebalance.nifty.statisticalinefficiency ( A\_n, B\_n = None, fast = False, mintime = 3, warn = True )** Compute the (cross) statistical inefficiency of (two) timeseries.

Notes The same timeseries can be used for both A\_n and B\_n to get the autocorrelation statistical inefficiency. The fast method described in Ref [1] is used to compute g.

References [1] J. D. Chodera, W. C. Swope, J. W. Pitera, C. Seok, and K. A. Dill. Use of the weighted histogram analysis method for the analysis of simulated and parallel tempering simulations. JCTC 3(1):26-41, 2007.

Examples

Compute statistical inefficiency of timeseries data with known correlation time.

```
import timeseries A_n = timeseries.generateCorrelatedTimeseries(N=100000,
tau=5.0) g = statisticalInefficiency(A_n, fast=True)
```

@param[in] A\_n (required, numpy array) - A\_n[n] is nth value of timeseries A. Length is deduced from vector.

@param[in] B\_n (optional, numpy array) - B\_n[n] is nth value of timeseries B. Length is deduced from vector. If supplied, the cross-correlation of timeseries A and B will be estimated instead of the autocorrelation of timeseries A.

@param[in] fast (optional, boolean) - if True, will use faster (but less accurate) method to estimate correlation time, described in Ref. [1] (default: False)

@param[in] mintime (optional, int) - minimum amount of correlation function to compute (default: 3) The algorithm terminates after computing the correlation time out to mintime when the correlation function first goes negative. Note that this time may need to be increased if there is a strong initial negative peak in the correlation function.

@return g The estimated statistical inefficiency (equal to 1 + 2 tau, where tau is the correlation time). We enforce g >= 1.0.

Definition at line 486 of file nifty.py.

**def forcebalance.nifty.uncommadash ( s )** Definition at line 101 of file nifty.py.

**def forcebalance.nifty.warn\_once ( warning, warnhash = None )** Prints a warning but will only do so once in a given run.

Definition at line 1103 of file nifty.py.

**def forcebalance.nifty.warn\_press\_key ( warning, timeout = 10 )** Definition at line 1091 of file nifty.py.

**def forcebalance.nifty.which ( fnm )** Definition at line 935 of file nifty.py.

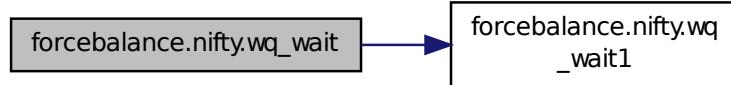
**def forcebalance.nifty.wopen ( dest )** If trying to write to a symbolic link, remove it first.

Definition at line 891 of file nifty.py.

**def forcebalance.nifty.wq\_wait ( wq, wait\_time = 10, wait\_intvl = 10, print\_time = 60, verbose = False )** This function waits until the work queue is completely empty.

Definition at line 787 of file nifty.py.

Here is the call graph for this function:



```
def forcebalance.nifty.wq_wait1 ( wq, wait_time = 10, wait_intvl = 1, print_time = 60, verbose = False )
```

This function waits ten seconds to see if a task in the Work Queue has finished.

Definition at line 722 of file nifty.py.

### 7.22.3 Variable Documentation

**float forcebalance.nifty.bohrang = 0.529177249** One bohr equals this many angstroms.

Definition at line 42 of file nifty.py.

**float forcebalance.nifty.eqcgmx = 2625.5002** Q-Chem to GMX unit conversion for energy.

Definition at line 38 of file nifty.py.

**float forcebalance.nifty.fqcgmx = -49621.9** Q-Chem to GMX unit conversion for force.

Definition at line 40 of file nifty.py.

**float forcebalance.nifty.kb = 0.0083144100163** Boltzmann constant.

Definition at line 36 of file nifty.py.

**tuple forcebalance.nifty.logger = getLogger(\_\_name\_\_)** Definition at line 31 of file nifty.py.

**tuple forcebalance.nifty.specific\_dct = dict(list(itertools.chain(\*[[[j,i[1]] for j in i[0]] for i in specific\_lst])))** Definition at line 877 of file nifty.py.

#### list forcebalance.nifty.specific\_lst Initial value:

```
1 = [[(['mdrun','grompp','trjconv','g-energy','g-traj'], "Make sure to install GROMACS and add it to your path  
2     (or set the gmxpath option"),  
3         ['force.mdin', 'stage.leap'], "This file is needed for setting up AMBER force matching  
targets"),  
4             ['conf.pdb', 'mono.pdb'], "This file is needed for setting up OpenMM condensed phase  
property targets"),  
5                 ['liquid.xyz', 'liquid.key', 'mono.xyz', 'mono.key'], "This file is needed for setting up  
OpenMM condensed phase property targets"),  
6                     ['dynamic', 'analyze', 'minimize', 'testgrad', 'vibrate', 'optimize', 'polarize', '  
superpose'], "Make sure to install TINKER and add it to your path (or set the tinkerpath option")],  
7                         ['truncuda.sh', 'npt.py', 'npt.tinker.py'], "This file belongs in the ForceBalance source  
directory, not sure why it is missing"),  
8                             ['input.xyz'], "This file is needed for TINKER molecular property targets"),  
9                               ('.*key$', '.*xyz$'), "I am guessing this file is probably needed by TINKER"),  
10                                 ('.*gro$', '.*top$', '.*itp$', '.*mdp$', '.*ndx$'), "I am guessing this file is probably  
needed by GROMACS")  
10 ]
```

Definition at line 865 of file nifty.py.

**forcebalance.nifty.WORK\_QUEUE = None** Definition at line 630 of file nifty.py.

**tuple forcebalance.nifty.WQIDS = defaultdict(list)** Definition at line 633 of file nifty.py.

**string forcebalance.nifty.XMLFILE = 'x'** Pickle uses 'flags' to pickle and unpickle different variable types.

Here we use the letter 'x' to signify that the variable type is an XML file.

Definition at line 559 of file nifty.py.

## 7.23 forcebalance.objective Namespace Reference

ForceBalance objective function.

### Classes

- class **Objective**  
*Objective function.*
- class **Penalty**  
*Penalty functions for regularizing the force field optimizer.*

### Variables

- tuple **logger** = getLogger(\_\_name\_\_)
- dictionary **Implemented\_Tests**  
*The table of implemented Targets.*
- list **Letters** = ['X','G','H']  
*This is the canonical lettering that corresponds to : objective function, gradient, Hessian.*

#### 7.23.1 Detailed Description

ForceBalance objective function.

#### 7.23.2 Variable Documentation

**dictionary forcebalance.objectiveImplemented\_Tests Initial value:**

```
1 = {
2   'ABINITIO_GMX':AbInitio_GMX,
3   'ABINITIO_TINKER':AbInitio_TINKER,
4   'ABINITIO_OPENMM':AbInitio_OpenMM,
5   'ABINITIO_AMBER':AbInitio_AMBER,
6   'ABINITIO_INTERNAL':AbInitio_Internal,
7   'VIBRATION_TINKER':Vibration_TINKER,
8   'VIBRATION_GMX':Vibration_GMX,
9   'THERMO_GMX':Thermo_GMX,
10  'LIQUID_OPENMM':Liquid_OpenMM,
11  'LIQUID_TINKER':Liquid_TINKER,
12  'LIQUID_GMX':Liquid_GMX,
13  'LIPID_GMX':Lipid_GMX,
14  'COUNTERPOISE':Counterpoise,
15  'THCDF_PSI4':THCDF_Psi4,
16  'RDVR3_PSI4':RDVR3_Psi4,
17  'INTERACTION_GMX':Interaction_GMX,
18  'INTERACTION_TINKER':Interaction_TINKER,
19  'INTERACTION_OPENMM':Interaction_OpenMM,
20  'BINDINGENERGY_TINKER':BindingEnergy_TINKER,
21  'BINDINGENERGY_GMX':BindingEnergy_GMX,
22  'BINDINGENERGY_OPENMM':BindingEnergy_OpenMM,
23  'MOMENTS_TINKER':Moments_TINKER,
24  'MOMENTS_GMX':Moments_GMX,
25  'MOMENTS_OPENMM':Moments_OpenMM,
26  'REMOTE_TARGET':RemoteTarget,
27 }
```

The table of implemented Targets.  
Definition at line 68 of file objective.py.

**list forcebalance.objective.Letters = ['X','G','H']** This is the canonical lettering that corresponds to : objective function, gradient, Hessian.

Definition at line 97 of file objective.py.

**tuple forcebalance.objective.logger = getLogger(\_\_name\_\_)** Definition at line 17 of file objective.py.

## 7.24 forcebalance.openmmio Namespace Reference

OpenMM input/output.

### Classes

- class [OpenMM\\_Reader](#)  
*Class for parsing OpenMM force field files.*
- class [OpenMM](#)  
*Derived from Engine object for carrying out general purpose OpenMM calculations.*
- class [Liquid\\_OpenMM](#)  
*Condensed phase property matching using OpenMM.*
- class [AbInitio\\_OpenMM](#)  
*Force and energy matching using OpenMM.*
- class [BindingEnergy\\_OpenMM](#)  
*Binding energy matching using OpenMM.*
- class [Interaction\\_OpenMM](#)  
*Interaction matching using OpenMM.*
- class [Moments\\_OpenMM](#)  
*Multipole moment matching using OpenMM.*

### Functions

- def [energy\\_components](#)
- def [get\\_multipoles](#)  
*Return the current multipole moments in Debye and Buckingham units.*
- def [get\\_dipole](#)  
*Return the current dipole moment in Debye.*
- def [PrepareVirtualSites](#)  
*Prepare a list of function wrappers and vsite parameters from the system.*
- def [ResetVirtualSites\\_fast](#)  
*Given a set of OpenMM-compatible positions and a System object, compute the correct virtual site positions according to the System.*
- def [ResetVirtualSites](#)  
*Given a set of OpenMM-compatible positions and a System object, compute the correct virtual site positions according to the System.*
- def [GetVirtualSiteParameters](#)  
*Return an array of all virtual site parameters in the system.*
- def [CopyAmoebaBondParameters](#)
- def [CopyAmoebaOutOfPlaneBendParameters](#)
- def [CopyAmoebaAngleParameters](#)

- def `CopyAmoebaInPlaneAngleParameters`
- def `CopyAmoebaVdwParameters`
- def `CopyAmoebaMultipoleParameters`
- def `CopyHarmonicBondParameters`
- def `CopyHarmonicAngleParameters`
- def `CopyPeriodicTorsionParameters`
- def `CopyNonbondedParameters`
- def `do_nothing`
- def `CopySystemParameters`

*Copy parameters from one system (i.e.*
- def `UpdateSimulationParameters`
- def `SetAmoebaVirtualExclusions`
- def `MTSVVVRIntegrator`

*Create a multiple timestep velocity verlet with velocity randomization (VVVR) integrator.*

## Variables

- tuple `logger` = getLogger(`_name_`)
  - dictionary `suffix_dict`
  - string `pdict` = "XML\_Override"
- pdict is a useless variable if the force field is XML.*

### 7.24.1 Detailed Description

[OpenMM](#) input/output.

Author

Lee-Ping Wang

Date

04/2012

### 7.24.2 Function Documentation

**def forcebalance.openmmio.CopyAmoebaAngleParameters ( *src*, *dest* )** Definition at line 227 of file openmmio.py.

**def forcebalance.openmmio.CopyAmoebaBondParameters ( *src*, *dest* )** Definition at line 213 of file openmmio.py.

**def forcebalance.openmmio.CopyAmoebaInPlaneAngleParameters ( *src*, *dest* )** Definition at line 236 of file openmmio.py.

**def forcebalance.openmmio.CopyAmoebaMultipoleParameters ( *src*, *dest* )** Definition at line 249 of file openmmio.py.

**def forcebalance.openmmio.CopyAmoebaOutOfPlaneBendParameters ( *src*, *dest* )** Definition at line 219 of file openmmio.py.

**def forcebalance.openmmio.CopyAmoebaVdwParameters ( *src*, *dest* )** Definition at line 245 of file openmmio.py.

```
def forcebalance.openmmio.CopyHarmonicAngleParameters ( src, dest ) Definition at line 257 of file openmmio.py.
```

```
def forcebalance.openmmio.CopyHarmonicBondParameters ( src, dest ) Definition at line 253 of file openmmio.py.
```

```
def forcebalance.openmmio.CopyNonbondedParameters ( src, dest ) Definition at line 265 of file openmmio.py.
```

```
def forcebalance.openmmio.CopyPeriodicTorsionParameters ( src, dest ) Definition at line 261 of file openmmio.py.
```

```
def forcebalance.openmmio.CopySystemParameters ( src, dest ) Copy parameters from one system (i.e. that which is created by a new force field) sto another system (i.e. the one stored inside the Target object). DANGER: These need to be implemented manually!!!
Definition at line 279 of file openmmio.py.
```

```
def forcebalance.openmmio.do_nothing ( src, dest ) Definition at line 272 of file openmmio.py.
```

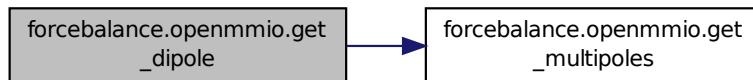
```
def forcebalance.openmmio.energy_components ( Sim, verbose = False ) Definition at line 38 of file openmmio.py.
```

```
def forcebalance.openmmio.get_dipole ( simulation, q = None, mass = None, positions = None ) Return the current dipole moment in Debye.
```

Note that this quantity is meaningless if the system carries a net charge.

Definition at line 107 of file openmmio.py.

Here is the call graph for this function:



```
def forcebalance.openmmio.get_multipoles ( simulation, q = None, mass = None, positions = None, rmcom = True ) Return the current multipole moments in Debye and Buckingham units.
```

Definition at line 48 of file openmmio.py.

```
def forcebalance.openmmio.GetVirtualSiteParameters ( system ) Return an array of all virtual site parameters in the system.
```

Used for comparison purposes.

Definition at line 194 of file openmmio.py.

```
def forcebalance.openmmio.MTSVVVRIntegrator( temperature, collision_rate, timestep, system, ninnersteps = 4 ) Create a multiple timestep velocity verlet with velocity randomization (VVVR) integrator.
```

#### ARGUMENTS

temperature (Quantity compatible with kelvin) - the temperature  
collision\_rate (Quantity compatible with 1/picoseconds) - the collision rate  
timestep (Quantity compatible with femtoseconds) - the integration timestep  
system (simtk.openmm.System) - system whose forces will be partitioned  
ninnersteps (int) - number of inner timesteps (default: 4)

#### RETURNS

integrator (openmm.CustomIntegrator) - a VVVR integrator

#### NOTES

This integrator is equivalent to a Langevin integrator in the velocity Verlet discretization with a timestep correction to ensure that the field-free diffusion constant is timestep invariant. The inner velocity Verlet discretization is transformed into a multiple timestep algorithm.

#### REFERENCES

VVVR Langevin integrator:

- <http://arxiv.org/abs/1301.3800>
- <http://arxiv.org/abs/1107.2967> (to appear in PRX 2013)

#### TODO

Move initialization of 'sigma' to setting the per-particle variables.

Definition at line 357 of file openmmio.py.

```
def forcebalance.openmmio.PrepareVirtualSites( system ) Prepare a list of function wrappers and vsite parameters from the system.
```

Definition at line 112 of file openmmio.py.

```
def forcebalance.openmmio.ResetVirtualSites( positions, system ) Given a set of OpenMM-compatible positions and a System object, compute the correct virtual site positions according to the System.
```

Definition at line 167 of file openmmio.py.

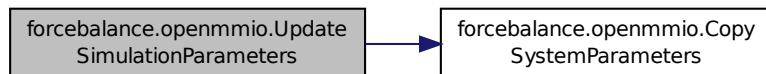
```
def forcebalance.openmmio.ResetVirtualSites.fast( positions, vsinfo ) Given a set of OpenMM-compatible positions and a System object, compute the correct virtual site positions according to the System.
```

Definition at line 152 of file openmmio.py.

```
def forcebalance.openmmio.SetAmoebaVirtualExclusions( system ) Definition at line 304 of file openmmio.py.
```

```
def forcebalance.openmmio.UpdateSimulationParameters( src_system, dest_simulation ) Definition at line 298 of file openmmio.py.
```

Here is the call graph for this function:



### 7.24.3 Variable Documentation

```
tuple forcebalance.openmmio.logger = getLogger(__name__) Definition at line 29 of file openmmio.py.
```

```
string forcebalance.openmmio.pdict = "XML_Override" pdict is a useless variable if the force field is XML.  
Definition at line 436 of file openmmio.py.
```

#### dictionary forcebalance.openmmio.suffix\_dict Initial value:

```
1 = { "HarmonicBondForce" : {"Bond" : ["class1", "class2"]},  
2     "HarmonicAngleForce" : {"Angle" : ["class1", "class2", "class3"],},  
3     "PeriodicTorsionForce" : {"Proper" : ["class1", "class2", "class3", "class4"],},  
4     "NonbondedForce" : {"Atom" : ["type"]},  
5     "AmoebaBondForce" : {"Bond" : ["class1", "class2"]},  
6     "AmoebaAngleForce" : {"Angle" : ["class1", "class2", "class3"]},  
7     "AmoebaStretchBendForce" : {"StretchBend" : ["class1", "class2", "class3"]},  
8     "AmoebaVdwForce" : {"Vdw" : ["class"]},  
9     "AmoebaMultipoleForce" : {"Multipole" : ["type", "kz", "kx"], "Polarize" : ["type"]},  
10    "AmoebaUreyBradleyForce" : {"UreyBradley" : ["class1", "class2", "class3"]},  
11    "Residues.Residue" : {"VirtualSite" : ["index"]}  
12 }
```

Definition at line 422 of file openmmio.py.

## 7.25 forcebalance.optimizer Namespace Reference

Optimization algorithms.

### Classes

- class [Optimizer](#)  
*Optimizer class.*

### Functions

- def [Counter](#)
- def [First](#)
- def [GoodStep](#)

### Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)
- int [ITERATION](#) = 0
- int [GOODSTEP](#) = 0
- int [ITERINIT](#) = 0

### 7.25.1 Detailed Description

Optimization algorithms. My current implementation is to have a single optimizer class with several methods contained inside.

Author

Lee-Ping Wang

Date

12/2011

### 7.25.2 Function Documentation

**def forcebalance.optimizer.Counter( )** Definition at line 32 of file optimizer.py.

```
def forcebalance.optimizer.First( ) Definition at line 36 of file optimizer.py.  
  
def forcebalance.optimizer.GoodStep( ) Definition at line 40 of file optimizer.py.
```

### 7.25.3 Variable Documentation

```
int forcebalance.optimizer.GOODSTEP = 0 Definition at line 27 of file optimizer.py.
```

```
int forcebalance.optimizer.ITERATION = 0 Definition at line 25 of file optimizer.py.
```

```
int forcebalance.optimizer.ITERINIT = 0 Definition at line 30 of file optimizer.py.
```

```
tuple forcebalance.optimizer.logger = getLogger(__name__) Definition at line 22 of file optimizer.py.
```

## 7.26 forcebalance.output Namespace Reference

### Classes

- class [ForceBalanceLogger](#)  
*This logger starts out with a default handler that writes to stdout addHandler removes this default the first time another handler is added.*
- class [RawStreamHandler](#)  
*Exactly like output.StreamHandler except it does no extra formatting before sending logging messages to the stream.*
- class [RawFileHandler](#)  
*Exactly like output.FileHandler except it does no extra formatting before sending logging messages to the file.*
- class [CleanStreamHandler](#)  
*Similar to [RawStreamHandler](#) except it does not write terminal escape codes.*
- class [CleanFileHandler](#)  
*File handler that does not write terminal escape codes and carriage returns to files.*
- class [ModLogger](#)

## 7.27 forcebalance.parser Namespace Reference

Input file parser for ForceBalance jobs.

### Functions

- def [read\\_mvals](#)
- def [read\\_pvals](#)
- def [read\\_priors](#)
- def [read\\_internals](#)
- def [printsection](#)  
*Print out a section of the input file in a parser-compliant and readable format.*
- def [parse\\_inputs](#)  
*Parse through the input file and read all user-supplied options.*

## Variables

- tuple `logger` = getLogger(`_name_`)
- dictionary `gen_opts_types`

*Default general options.*
- dictionary `tgt_opts_types`

*Default fitting target options.*
- tuple `all_opts_names` = list(itertools.chain(\*[i.keys() for i in gen\_opts\_types.values()]))
- list `iocc` = []

*Check for uniqueness of option names.*
- dictionary `gen_opts_defaults` = {}

*Default general options - basically a collapsed version of gen\_opts\_types.*
- dictionary `subdict` = {}
- dictionary `tgt_opts_defaults` = {}

*Default target options - basically a collapsed version of tgt\_opts\_types.*
- dictionary `bkwd`

*Option maps for maintaining backward compatibility.*
- list `mainsections` = ["SIMULATION", "TARGET", "OPTIONS", "END", "NONE"]

*Listing of sections in the input file.*
- dictionary `ParsTab`

*ParsTab that refers to subsection parsers.*

### 7.27.1 Detailed Description

Input file parser for ForceBalance jobs. Additionally, the location for all default options.

Although I will do my best to write good documentation, for many programs the input parser becomes the most up-to-date source for documentation. So this is a great place to write lots of comments for those who implement new functionality.

There are two types of sections for options - GENERAL and TARGET. Since there can be many fitting targets within a single job (i.e. we may wish to fit water trimers and hexamers, which constitutes two fitting targets) the input is organized into sections, like so:

```
$options
gen_option_1 Big
gen_option_2 Mao
$target
tgt_option_1 Sniffy
tgt_option_2 Schmao
$target
tgt_option_1 Nifty
tgt_option_2 Jiffy
$end
```

In this case, two sets of target options are generated in addition to the general option.

(Note: "Target" used to be called "Simulation". Backwards compatibility is maintained.)

Each option is meant to be parsed as a certain variable type.

- String option values are read in directly; note that only the first two words in the line are processed
- Some strings are capitalized when they are read in; this is mainly for function tables like OptTab and TgtTab
- List option types will pick up all of the words on the line and use them as values, plus if the option occurs more than once it will aggregate all of the values.
- Integer and float option types are read in a pretty straightforward way

- Boolean option types are always set to true, unless the second word is '0', 'no', or 'false' (not case sensitive)
- Section option types are meant to treat more elaborate inputs, such as the user pasting in output parameters from a previous job as input, or a specification of internal coordinate system. I imagine that for every section type I would have to write my own parser. Maybe a ParsTab of parsing functions would work. :)

To add a new option, simply add it to the dictionaries below and give it a default value if desired. If you add an entirely new type, make sure to implement the interpretation of that type in the parse\_inputs function.

Author

Lee-Ping Wang

Date

11/2012

## 7.27.2 Function Documentation

**def forcebalance.parser.parse\_inputs ( *input\_file = None* )** Parse through the input file and read all user-supplied options.

This is usually the first thing that happens when an executable script is called. Our parser first loads the default options, and then updates these options as it encounters keywords.

Each keyword corresponds to a variable type; each variable type (e.g. string, integer, float, boolean) is treated differently. For more elaborate inputs, there is a 'section' variable type.

There is only one set of general options, but multiple sets of target options. Each target has its own section delimited by the \$target keyword, and we build a list of target options.

Parameters

in	<i>input_file</i>	The name of the input file.
----	-------------------	-----------------------------

Returns

options General options.

tgt\_opts List of fitting target options.

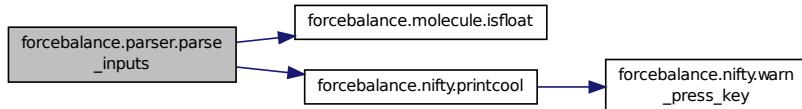
**Todo** Implement internal coordinates.

Implement sampling correction.

Implement charge groups.

Definition at line 438 of file parser.py.

Here is the call graph for this function:



**def forcebalance.parser.printsection ( *heading, optdict, typedict* )** Print out a section of the input file in a parser-compliant and readable format.

At the time of writing of this function, it's mainly intended to be called by MakeInputFile.py. The heading is printed first (it is something like \$options or \$target). Then it loops through the variable types (strings, allcaps, etc...) and the keys in each variable type. The one-line description of each key is printed out as a comment, and then the key itself is printed out along with the value provided in optdict. If optdict is None, then the default value is printed out instead.

## Parameters

in	<i>heading</i>	Heading, either \$options or \$target
in	<i>optdict</i>	Options dictionary or None.
in	<i>typedict</i>	Option type dictionary, either gen_opts_types or tgt_opts_types specified in this file.

## Returns

Answer List of strings for the section that we are printing out.

Definition at line 341 of file parser.py.

**def forcebalance.parser.read\_internals ( *fobj* )** Definition at line 315 of file parser.py.

**def forcebalance.parser.read\_mvals ( *fobj* )** Definition at line 290 of file parser.py.

**def forcebalance.parser.read\_priors ( *fobj* )** Definition at line 306 of file parser.py.

**def forcebalance.parser.read\_pvals ( *fobj* )** Definition at line 298 of file parser.py.

## 7.27.3 Variable Documentation

**tuple forcebalance.parser.all\_opts\_names = list(itertools.chain(\*[i.keys() for i in gen\_opts\_types.values()]))** Definition at line 243 of file parser.py.

**dictionary forcebalance.parser.bkwd Initial value:**

```
1 = {"simtype" : "type",
2     "masterfile" : "inter.txt",
3     "openmm.cuda.precision" : "openmm.precision",
4     "mdrun.threads" : "md.threads",
5     "mts.vvvr" : "mts.integrator",
6     "amoeba.polarization" : "amoeba.pol",
7     "liquid.prod.steps" : "liquid.md.steps",
8     "gas.prod.steps" : "gas.md.steps",
9     "liquid.equ.steps" : "liquid.eq.steps",
10    "gas.equ.steps" : "gas.eq.steps",
11    "lipid.prod.steps" : "lipid.md.steps",
12    "lipid.equ.steps" : "lipid.eq.steps",
13 }
```

Option maps for maintaining backward compatibility.

Definition at line 273 of file parser.py.

**dictionary forcebalance.parser.gen\_opts\_defaults = {}** Default general options - basically a collapsed veresion of gen\_opts\_types.

Definition at line 257 of file parser.py.

**dictionary forcebalance.parser.gen\_opts\_types** Default general options.

Note that the documentation is included in part of the key; this will aid in automatic doc-extraction. :) In the 5-tuple we have: Default value, priority (larger number means printed first), short docstring, description of scope, list of filter strings for pulling out pertinent targets (MakeInputFile.py)

Definition at line 65 of file parser.py.

**list forcebalance.parser.iocc = []** Check for uniqueness of option names.

Definition at line 246 of file parser.py.

**tuple forcebalance.parser.logger = getLogger(\_\_name\_\_)** Definition at line 60 of file parser.py.

**list forcebalance.parser.mainsections = ["SIMULATION","TARGET","OPTIONS","END","NONE"]** Listing of sections in the input file.

Definition at line 288 of file parser.py.

**dictionary forcebalance.parser.ParsTab Initial value:**

```
1 = {"read_mvals" : read_mvals,
2     "read_pvals" : read_pvals,
3     "priors"      : read_priors,
4     "internal"    : read_internals
5 }
```

ParsTab that refers to subsection parsers.

Definition at line 319 of file parser.py.

**dictionary forcebalance.parser.subdict = {}** Definition at line 259 of file parser.py.

**dictionary forcebalance.parser.tgt\_opts\_defaults = {}** Default target options - basically a collapsed version of tgt\_opts\_types.

Definition at line 265 of file parser.py.

**dictionary forcebalance.parser.tgt\_opts\_types** Default fitting target options.

Definition at line 135 of file parser.py.

## 7.28 forcebalance.psi4io Namespace Reference

PSI4 force field input/output.

### Classes

- class [GBS\\_Reader](#)  
*Interaction type -> Parameter Dictionary.*
- class [THCDF\\_Psi4](#)
- class [Grid\\_Reader](#)  
*Finite state machine for parsing DVR grid files.*
- class [RDVR3\\_Psi4](#)  
*Subclass of Target for R-DVR3 grid fitting.*

### Variables

- tuple [logger = getLogger\(\\_\\_name\\_\\_\)](#)

#### 7.28.1 Detailed Description

PSI4 force field input/output. This serves as a good template for writing future force matching I/O modules for other programs because it's so simple.

Author

Lee-Ping Wang

Date

01/2012

## 7.28.2 Variable Documentation

tuple `forcebalance.psi4io.logger = getLogger(__name__)` Definition at line 24 of file `psi4io.py`.

## 7.29 forcebalance.PT Namespace Reference

### Variables

- dictionary `PeriodicTable`
- list `Elements`

### 7.29.1 Variable Documentation

list `forcebalance.PT.Elements` Initial value:

```
1 = ["None",'H','He',
2      'Li','Be','B','C','N','O','F','Ne',
3      'Na','Mg','Al','Si','P','S','Cl','Ar',
4      'K','Ca','Sc','Ti','V','Cr','Mn','Fe','Co','Ni','Cu','Zn','Ga','Ge','As','Se','Br','Kr',
5      'Rb','Sr','Y','Zr','Nb','Mo','Tc','Ru','Rh','Pd','Ag','Cd','In','Sn','Sb','Te','I','Xe',
6      'Cs','Ba','La','Ce','Pr','Nd','Pm','Sm','Eu','Gd','Tb','Dy','Ho','Er','Tm','Yb',
7      'Lu','Hf','Ta','W','Re','Os','Ir','Pt','Au','Hg','Tl','Pb','Bi','Po','At','Rn',
8      'Fr','Ra','Ac','Th','Pa','U','Np','Pu','Am','Cm','Bk','Cf','Es','Fm','Md','No','Lr','Rf','Db',
     Sg,'Bh','Hs','Mt']
```

Definition at line 18 of file `PT.py`.

dictionary `forcebalance.PT.PeriodicTable` Initial value:

```
1 = {'H' : 1.0079, 'He' : 4.0026,
2      'Li' : 6.941, 'Be' : 9.0122, 'B' : 10.811, 'C' : 12.0107, 'N' : 14.0067, 'O' : 15.9994, 'F'
     : 18.9984, 'Ne' : 20.1797,
3      'Na' : 22.9897, 'Mg' : 24.305, 'Al' : 26.9815, 'Si' : 28.0855, 'P' : 30.9738, 'S' : 32.065
     , 'Cl' : 35.453, 'Ar' : 39.948,
4      'K' : 39.0983, 'Ca' : 40.078, 'Sc' : 44.9559, 'Ti' : 47.867, 'V' : 50.9415, 'Cr' : 51.9961
     , 'Mn' : 54.938, 'Fe' : 55.845, 'Co' : 58.9332,
5      'Ni' : 58.6934, 'Cu' : 63.546, 'Zn' : 65.39, 'Ga' : 69.723, 'Ge' : 72.64, 'As' : 74.9216,
     'Se' : 78.96, 'Br' : 79.904, 'Kr' : 83.8,
6      'Rb' : 85.4678, 'Sr' : 87.62, 'Y' : 88.9059, 'Zr' : 91.224, 'Nb' : 92.9064, 'Mo' : 95.94,
     'Tc' : 98, 'Ru' : 101.07, 'Rh' : 102.9055,
7      'Pd' : 106.42, 'Ag' : 107.8682, 'Cd' : 112.411, 'In' : 114.818, 'Sn' : 118.71, 'Sb' : 121.
    76, 'Te' : 127.6, 'I' : 126.9045, 'Xe' : 131.293,
8      'Cs' : 132.9055, 'Ba' : 137.327, 'La' : 138.9055, 'Ce' : 140.116, 'Pr' : 140.9077, 'Nd' :
    144.24, 'Pm' : 145, 'Sm' : 150.36,
9      'Eu' : 151.964, 'Gd' : 157.25, 'Tb' : 158.9253, 'Dy' : 162.5, 'Ho' : 164.9303, 'Er' : 167.
    259, 'Tm' : 168.9342, 'Yb' : 173.04,
10     'Lu' : 174.967, 'Hf' : 178.49, 'Ta' : 180.9479, 'W' : 183.84, 'Re' : 186.207, 'Os' : 190.2
    3, 'Ir' : 192.217, 'Pt' : 195.078,
11     'Au' : 196.9665, 'Hg' : 200.59, 'Tl' : 204.3833, 'Pb' : 207.2, 'Bi' : 208.9804, 'Po' : 209
     , 'At' : 210, 'Rn' : 222,
12     'Fr' : 223, 'Ra' : 226, 'Ac' : 227, 'Th' : 232.0381, 'Pa' : 231.0359, 'U' : 238.0289, 'Np'
     : 237, 'Pu' : 244,
13     'Am' : 243, 'Cm' : 247, 'Bk' : 247, 'Cf' : 251, 'Es' : 252, 'Fm' : 257, 'Md' : 258, 'No' :
    259,
     'Lr' : 262, 'Rf' : 261, 'Db' : 262, 'Sg' : 266, 'Bh' : 264, 'Hs' : 277, 'Mt' : 268}
```

Definition at line 3 of file `PT.py`.

## 7.30 forcebalance.qchemio Namespace Reference

Q-Chem input file parser.

### Classes

- class `QCIn_Reader`

*Finite state machine for parsing Q-Chem input files.*

## Functions

- def `QChem_Dielectric_Energy`

## Variables

- tuple `logger` = getLogger(`__name__`)
- list `ndtypes` = [None]  
*Types of counterpoise correction cptypes = [None, 'BASS', 'BASSP'] Types of NDDO correction.*
- dictionary `pdict`  
*Section -> Interaction type dictionary.*

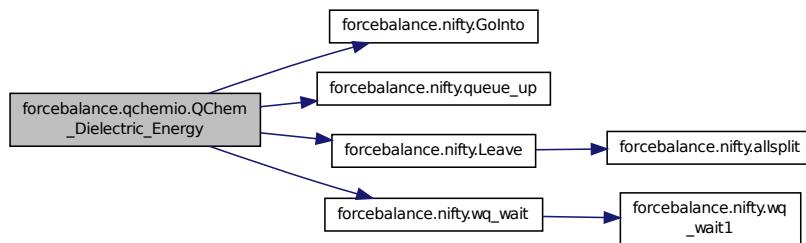
### 7.30.1 Detailed Description

Q-Chem input file parser.

### 7.30.2 Function Documentation

**def forcebalance.qchemio.QChem\_Dielectric\_Energy ( `fnm`, `wq` )** Definition at line 74 of file qchemio.py.

Here is the call graph for this function:



### 7.30.3 Variable Documentation

**tuple forcebalance.qchemio.logger = getLogger(`__name__`)** Definition at line 11 of file qchemio.py.

**list forcebalance.qchemio.ndtypes = [None]** Types of counterpoise correction cptypes = [None, 'BASS', 'BASSP']  
Types of NDDO correction.

Definition at line 16 of file qchemio.py.

**dictionary forcebalance.qchemio.pdict Initial value:**

```
1 = {'BASS': {0:'A', 1:'C'},  
2      'BASSP' :{0:'A', 1:'B', 2:'C'}  
3      }
```

Section -> Interaction type dictionary.

`fdict = { 'basis' : bastypes }` Interaction type -> Parameter Dictionary.

Definition at line 23 of file qchemio.py.

## 7.31 forcebalance.quantity Namespace Reference

### Classes

- class [Quantity](#)  
*Base class for thermodynamical quantity used for fitting.*
- class [Quantity\\_Density](#)
- class [Quantity\\_H\\_vap](#)

### Functions

- def [mean\\_stderr](#)  
*Return mean and standard deviation of a time series ts.*
- def [energy\\_derivatives](#)  
*Compute the first derivatives of a set of snapshot energies with respect to the force field parameters.*

### Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)

#### 7.31.1 Function Documentation

**def forcebalance.quantity.energy\_derivatives ( engine, FF, mvals, h, pgrad, length, AGrad = True )**

Compute the first derivatives of a set of snapshot energies with respect to the force field parameters.

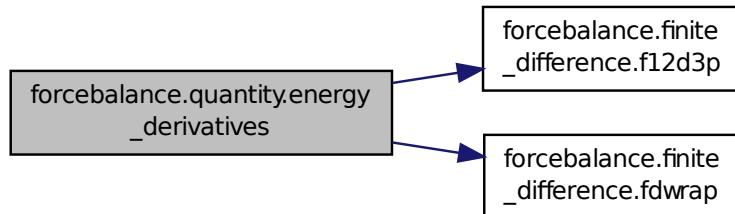
The function calls the finite difference subroutine on the energy\_driver subroutine also in this script.

**Parameters** engine : Engine Use this Engine (GMX,TINKER,OPENMM etc.) object to get the energy snapshots. FF : FF Force field object. mvals : list Mathematical parameter values. h : float Finite difference step size. length : int Number of snapshots (length of energy trajectory). AGrad : Boolean Switch that turns derivatives on or off; if off, return all zeros.

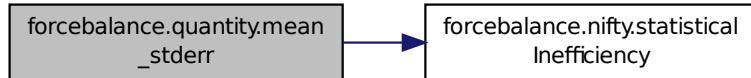
**Returns** G : np.array Derivative of the energy in a FF.np x length array.

Definition at line 50 of file quantity.py.

Here is the call graph for this function:



```
def forcebalance.quantity.mean_stderr( ts )  Return mean and standard deviation of a time series ts.  
Definition at line 17 of file quantity.py.  
Here is the call graph for this function:
```



### 7.31.2 Variable Documentation

tuple `forcebalance.quantity.logger = getLogger(__name__)` Definition at line 12 of file quantity.py.

## 7.32 forcebalance.target Namespace Reference

### Classes

- class [Target](#)  
*Base class for all fitting targets.*
- class [RemoteTarget](#)

### Variables

- tuple `logger = getLogger(__name__)`

### 7.32.1 Variable Documentation

tuple `forcebalance.target.logger = getLogger(__name__)` Definition at line 17 of file target.py.

## 7.33 forcebalance.thermo Namespace Reference

### Classes

- class [Thermo](#)  
*A target for fitting general experimental data sets.*
- class [Point](#)

### Variables

- tuple `logger = getLogger(__name__)`

### 7.33.1 Variable Documentation

tuple `forcebalance.thermo.logger = getLogger(__name__)` Definition at line 15 of file thermo.py.

## 7.34 forcebalance.tinkerio Namespace Reference

[TINKER](#) input/output.

## Classes

- class `Tinker_Reader`  
*Finite state machine for parsing `TINKER` force field files.*
- class `TINKER`  
*Engine for carrying out general purpose `TINKER` calculations.*
- class `Liquid_TINKER`  
*Condensed phase property matching using `TINKER`.*
- class `AbInitio_TINKER`  
*Subclass of Target for force and energy matching using `TINKER`.*
- class `BindingEnergy_TINKER`  
*Binding energy matching using `TINKER`.*
- class `Interaction_TINKER`  
*Subclass of Target for interaction matching using `TINKER`.*
- class `Moments_TINKER`  
*Subclass of Target for multipole moment matching using `TINKER`.*
- class `Vibration_TINKER`  
*Vibrational frequency matching using `TINKER`.*

## Functions

- def `write_key`  
*Create or edit a `TINKER` .key file.*

## Variables

- list `allp`
- list `eckeys`
- tuple `logger = getLogger(__name__)`
- dictionary `pdict`

### 7.34.1 Detailed Description

`TINKER` input/output. This serves as a good template for writing future force matching I/O modules for other programs because it's so simple.

Author

Lee-Ping Wang

Date

01/2012

### 7.34.2 Function Documentation

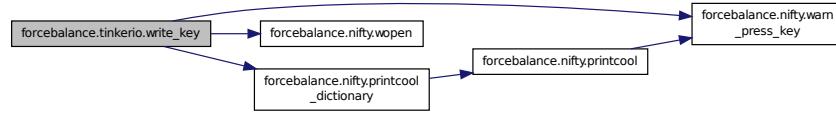
```
def forcebalance.tinkerio.write_key( fout, options, fin = None, defaults = {}, verbose = False, prmfnm = None, chk = [] ) Create or edit a TINKER .key file.
```

## Parameters

in	<i>fout</i>	Output file name, can be the same as input file name.
in	<i>options</i>	Dictionary containing .key options. Existing options are replaced, new options are added at the end. Passing None causes options to be deleted. To pass an option without an argument, use ".
in	<i>fin</i>	Input file name.
in	<i>defaults</i>	Default options to add to the mdp only if they don't already exist.
in	<i>verbose</i>	Print out all modifications to the file.
in	<i>prmfnm</i>	TINKER parameter file name.
in	<i>chk</i>	Crash if the key file does NOT have these options by the end.

Definition at line 179 of file tinkerio.py.

Here is the call graph for this function:



### 7.34.3 Variable Documentation

**list forcebalance.tinkerio.allp Initial value:**

```

1 = ['atom', 'vdw', 'vdw14', 'vdwpr', 'hbond', 'bond', 'bond5', 'bond4',
2      'bond3', 'electneg', 'angle', 'angle5', 'angle4', 'angle3', 'anglef',
3      'strbnd', 'ureybrad', 'angang', 'opbend', 'opdist', 'improper', 'imptors',
4      'torsion', 'torsion5', 'torsion4', 'pitors', 'strtors', 'tortors', 'charge',
5      'dipole', 'dipole5', 'dipole4', 'dipole3', 'multipole', 'polarize', 'piatom',
6      'pibond', 'pibond5', 'pibond4', 'metal', 'biotype', 'mmffvdw', 'mmffbond',
7      'mmffbonder', 'mmffangle', 'mmffstrbnd', 'mmffopbend', 'mmfftorsion', 'mmffbci',
8      'mmffpbc1', 'mmffequiv', 'mmffdefstbn', 'mmffcovrad', 'mmffprop', 'mmffarom']
  
```

Definition at line 35 of file tinkerio.py.

**list forcebalance.tinkerio.eckey Initial value:**

```

1 = ['Angle-Angle', 'Angle Bending', 'Atomic Multipoles', 'Bond Stretching', 'Charge-Charge',
2      'Charge-Dipole', 'Dipole-Dipole', 'Extra Energy Terms', 'Geometric Restraints', 'Implicit
     Solvation',
3      'Improper Dihedral', 'Improper Torsion', 'Metal Ligand Field', 'Out-of-Plane Bend', 'Out-of-Plane
     Distance',
4      'Pi-Orbital Torsion', 'Polarization', 'Reaction Field', 'Stretch-Bend', 'Stretch-Torsion',
5      'Torsional Angle', 'Torsion-Torsion', 'Urey-Bradley', 'Van der Waals']
  
```

Definition at line 45 of file tinkerio.py.

**tuple forcebalance.tinkerio.logger = getLogger(\_\_name\_\_)** Definition at line 52 of file tinkerio.py.

**dictionary forcebalance.tinkerio.pdict Initial value:**

```

1 = {'VDW' : {'Atom':[1], 2:'S',3:'T',4:'D'}, # Van der Waals distance, well depth, distance from
     bonded neighbor?
2     'BOND' : {'Atom':[1,2], 3:'K',4:'B'}, # Bond force constant and equilibrium distance
     (Angstrom)
3     'ANGLE' : {'Atom':[1,2,3], 4:'K',5:'B'}, # Angle force constant and equilibrium angle
4     'STRBND' : {'Atom':[1,2,3], 4:'K1',5:'K2'}, # Two stretch-bend force constants (usually
     same)
5     'OPBEND' : {'Atom':[1,2,3,4], 5:'K'}, # Out-of-plane bending force constant
     'UREYBRAD' : {'Atom':[1,2,3], 4:'K',5:'B'}, # Urey-Bradley force constant and equilibrium
  
```

```

7      distance (Angstrom)
8          'TORSION'    : ({'Atom':[1,2,3,4], 5:'1K', 6:'1B',
9                           8:'2K', 9:'2B', 11:'3K', 12:'3B'}), # Torsional force constants and equilibrium
10         phi-angles
11         'PITORS'     : {'Atom':[1,2], 3:'K'},           # Pi-torsion force constants (usually 6.85 ..)
12             # Note torsion-torsion (CMAP) not implemented at this time.
13         'MCHARGE'    : {'Atom':[1,2,3], 4:''},           # Atomic charge
14         'DIPOLE'     : {0:'X',1:'Y',2:'Z'},            # Dipole moment in local frame
15         'QUADX'      : {0:'X'},                         # Quadrupole moment, X component
16         'QUADY'      : {0:'X',1:'Y'},                  # Quadrupole moment, Y component
17         'QUADZ'      : {0:'X',1:'Y',2:'Z'},            # Quadrupole moment, Z component
18         'POLARIZE'   : {'Atom':[1], 2:'A',3:'T'},        # Atomic dipole polarizability
19         'BOND-CUBIC' : {'Atom':[], 0:''},              # Below are global parameters.
20             # Ignored for now: stretch/bend coupling, out-of-plane
21             # torsional parameters, pi-torsion, torsion-torsion
22         bending,
23     }

```

Definition at line 54 of file tinkerio.py.

## 7.35 forcebalance.vibration Namespace Reference

Vibrational mode fitting module.

### Classes

- class [Vibration](#)

*Subclass of Target for fitting force fields to vibrational spectra (from experiment or theory).*

### Functions

- def [count\\_assignment](#)

### Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)

#### 7.35.1 Detailed Description

Vibrational mode fitting module.

Author

Lee-Ping Wang

Date

08/2012

#### 7.35.2 Function Documentation

**def forcebalance.vibration.count\_assignment ( *assignment*, *verbose* = *True* )** Definition at line 25 of file vibration.py.

#### 7.35.3 Variable Documentation

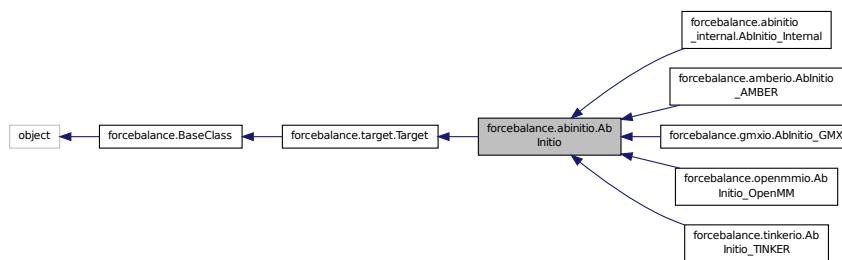
**tuple forcebalance.vibration.logger = getLogger(\_\_name\_\_)** Definition at line 23 of file vibration.py.

## 8 Class Documentation

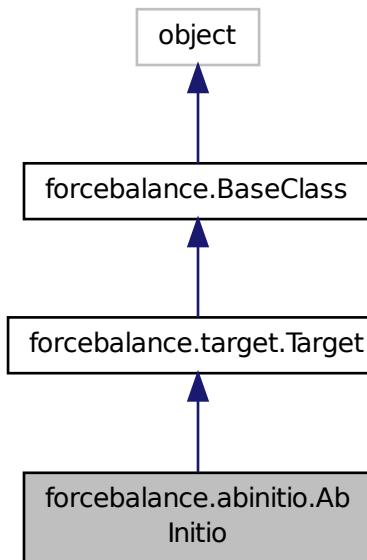
### 8.1 forcebalance.abinitio.AbInitio Class Reference

Subclass of Target for fitting force fields to ab initio data.

Inheritance diagram for forcebalance.abinitio.AbInitio:



Collaboration diagram for forcebalance.abinitio.AbInitio:



#### Public Member Functions

- def `_init_`  
*Initialization; define a few core concepts.*
- def `build_invdist`
- def `compute_netforce_torque`

- def `read_reference_data`  
*Read the reference ab initio data from a file such as qdata.txt.*
- def `indicate`
- def `energy_all`
- def `energy_force_all`
- def `energy_force_transform`
- def `energy_one`
- def `energy_force_one`
- def `energy_force_transform_one`
- def `get_energy_force`  
*LPW 06-30-2013.*
- def `get_resp`  
*Electrostatic potential fitting.*
- def `get`
- def `get_X`  
*Computes the objective function contribution without any parametric derivatives.*
- def `read_0grads`  
*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_0grads`  
*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`  
*Computes the objective function contribution and its gradient.*
- def `get_H`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- **boltz\_wts**  
*Initialize the base class.*
- **qmboltz\_wts**  
*QM Boltzmann weights.*
- **eqm**  
*Reference (QM) energies.*
- **emd0**  
*Energies of the sampling simulation.*
- **fqm**  
*Reference (QM) forces.*
- **espxyz**  
*ESP grid points.*
- **espval**  
*ESP values.*
- **qfnm**  
*The qdata.txt file that contains the QM energies and forces.*
- **qmatoms**  
*The number of atoms in the QM calculation (Irrelevant if not fitting forces)*
- **e\_err**  
*Qualitative Indicator: average energy error (in kJ/mol)*
- **e\_err\_pct**
- **f\_err**  
*Qualitative Indicator: average force error (fractional)*
- **f\_err\_pct**
- **esp\_err**  
*Qualitative Indicator: "relative RMS" for electrostatic potential.*
- **nf\_err**
- **nf\_err\_pct**
- **tq\_err\_pct**
- **use\_nft**  
*Whether to compute net forces and torques, or not.*
- **ns**  
*Read in the trajectory file.*
- **mol**
- **nparticles**  
*The number of (atoms + drude particles + virtual sites)*
- **engine**  
*Build keyword dictionaries to pass to engine.*
- **AtomLists**  
*Lists of atoms to do net force/torque fitting and excluding virtual sites.*
- **AtomMask**
- **new\_vsites**  
*Read in the reference data.*
- **save\_vmvalls**  
*Save the mvalls from the last time we updated the vsites.*
- **force\_map**

- `nnf`
- `ntq`
- `force`
- `w_force`
- `nesp`
- `fitatoms`
- `whamboltz`
- `nftqm`
- `fref`
- `w_energy`
- `w_netforce`
- `w_torque`
- `e_ref`
- `f_ref`
- `nf_ref`
- `tq_ref`
- `tq_err`
- `w_resp`
- `invdists`
- `respterm`
- `objective`
- `rd`

*Root directory of the whole project.*
- `pgrad`

*Iteration where we turn on zero-gradient skipping.*
- `tempbase`

*Relative directory of target.*
- `tempdir`
- `rundir`

*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- `FF`

*Need the forcefield (here for now)*
- `xct`

*Counts how often the objective function was computed.*
- `gct`

*Counts how often the gradient was computed.*
- `hct`

*Counts how often the Hessian was computed.*
- `read_indicate`

*Whether to read indicate.log from file when restarting an aborted run.*
- `write_indicate`

*Whether to write indicate.log at every iteration (true for all but remote.)*
- `read_objective`

*Whether to read objective.p from file when restarting an aborted run.*
- `write_objective`

*Whether to write objective.p at every iteration (true for all but remote.)*
- `verbose_options`
- `PrintOptionDict`

### 8.1.1 Detailed Description

Subclass of Target for fitting force fields to ab initio data.

Currently Gromacs-X2, Gromacs, Tinker, OpenMM and AMBER are supported.

We introduce the following concepts:

- The number of snapshots
- The reference energies and forces (eqm, fqm) and the file they belong in (qdata.txt)
- The sampling simulation energies (emd0)
- The WHAM Boltzmann weights (these are computed externally and passed in)
- The QM Boltzmann weights (computed internally using the difference between eqm and emd0)

There are also these little details:

- Switches for whether to turn on certain Boltzmann weights (they stack)
- Temperature for the QM Boltzmann weights
- Whether to fit a subset of atoms

This subclass contains the 'get' method for building the objective function from any simulation software (a driver to run the program and read output is still required). The 'get' method can be overridden by subclasses like AbInitio\_GMX. Definition at line 47 of file abinitio.py.

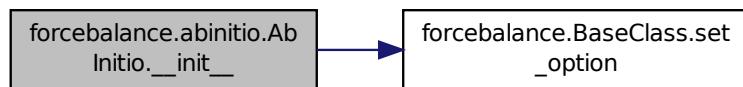
### 8.1.2 Constructor & Destructor Documentation

```
def forcebalance.abinitio.AbInitio.__init__( self, options, tgt_opts, forcefield ) Initialization; define a few core concepts.
```

**Todo** Obtain the number of true atoms (or the particle -> atom mapping) from the force field.

Definition at line 57 of file abinitio.py.

Here is the call graph for this function:



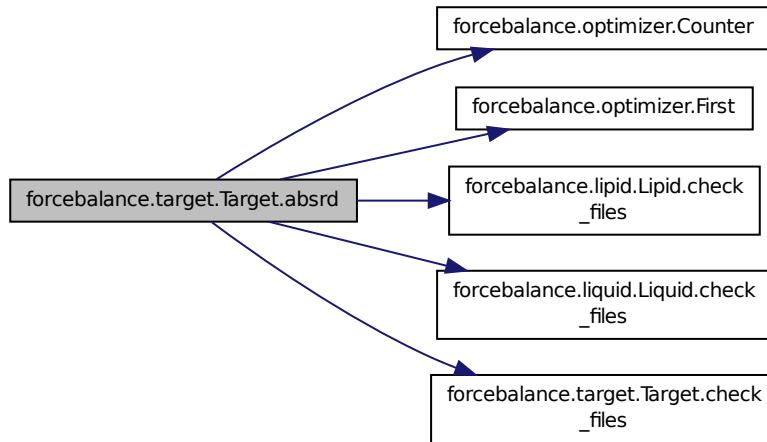
### 8.1.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory  
specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.AblInitio.build_invdist ( self, mvals ) Definition at line 171 of file abinitio.py.
```

```
def forcebalance.target.Target.check_files ( self, there ) [inherited] Check this directory for the presence  
of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.abinitio.AblInitio.compute_netforce_torque ( self, xyz, force, QM = False ) Definition at  
line 200 of file abinitio.py.
```

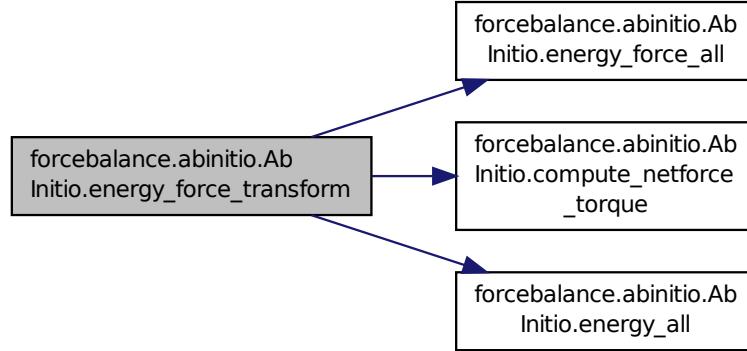
```
def forcebalance.abinitio.AblInitio.energy_all ( self ) Definition at line 472 of file abinitio.py.
```

```
def forcebalance.abinitio.AblInitio.energy_force_all ( self ) Definition at line 478 of file abinitio.py.
```

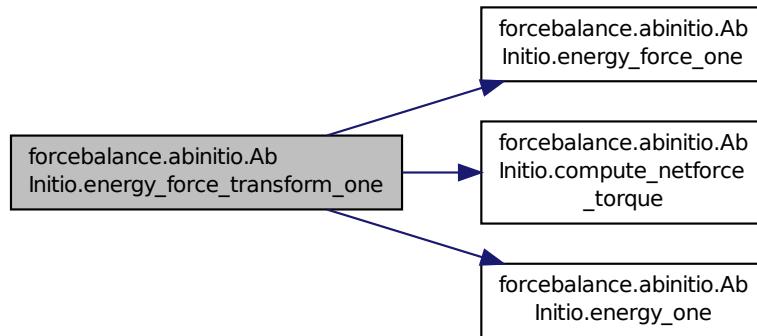
```
def forcebalance.abinitio.AblInitio.energy_force_one ( self, i ) Definition at line 507 of file abinitio.py.
```

```
def forcebalance.abinitio.AblInitio.energy_force_transform ( self ) Definition at line 484 of file abinitio.py.
```

Here is the call graph for this function:



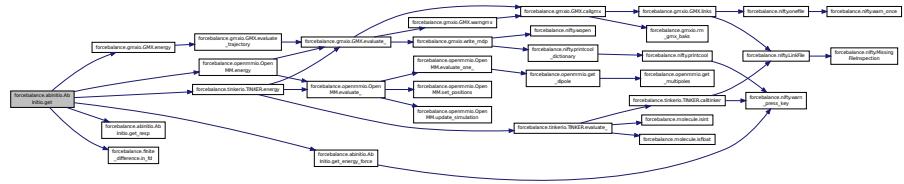
**def forcebalance.abinitio.AbInitio.energy\_force\_transform\_one ( self, i )** Definition at line 513 of file abinitio.py.  
Here is the call graph for this function:



**def forcebalance.abinitio.AbInitio.energy\_one ( self, i )** Definition at line 501 of file abinitio.py.

**def forcebalance.abinitio.AbInitio.get ( self, mvals, AGrad = False, AHess = False )** Definition at line 1152 of file abinitio.py.

Here is the call graph for this function:



**def forcebalance.abinitio.ABInitio.get\_energy\_force ( self, mvals, AGrad = False, AHess = False )** LPW  
06-30-2013.

This subroutine builds the objective function (and optionally its derivatives) from a general simulation software. This is in contrast to using GROMACS-X2, which computes the objective function and prints it out; then 'get' only needs to call GROMACS and read it in.

This subroutine interfaces with simulation software 'drivers'. The driver is only expected to give the energy and forces.

Now this subroutine may sound trivial since the objective function is simply a least-squares quantity  $(M-Q)^2$  - but there are a number of nontrivial considerations. I will list them here.

0) Polytensor formulation: Because there may exist covariance between different components of the force (or covariance between the energy and the force), we build the objective function by taking outer products of vectors that have the form [E F\_1x F\_1y F\_1z F\_2x F\_2y ... ], and then we trace it with the inverse of the covariance matrix to get the objective function.

This version implements both the polytensor formulation and the standard formulation.

1) Boltzmann weights and normalization: Each snapshot has its own Boltzmann weight, which may or may not be normalized. This subroutine does the normalization automatically.

2) Subtracting out the mean energy gap: The zero-point energy difference between reference data and simulation is meaningless. This subroutine subtracts it out.

3) Hybrid ensembles: This program builds a combined objective function from both MM and QM ensembles, which is rigorously better than using a single ensemble

Note that this subroutine does not do **EVERYTHING** that GROMACS-X2 can do, which includes:

Note that this subroutine does not do EVERYTHING that CHIMESSE XL can do, which includes:  
1) Internal coordinate systems 2) 'Sampling correction' (deprecated, since it doesn't seem to work) 3) Analytic derivatives

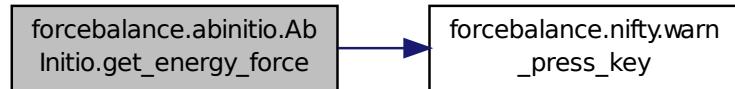
In the previous code (ForTune) this subroutine used analytic first derivatives of the energy and force to build the derivatives of the objective function. Here I will take a simplified approach, because building the derivatives are cumbersome. For now we will return the objective function ONLY. A two-point central difference should give us the first and diagonal second derivative anyhow.

**Todo** Parallelization over snapshots is not implemented yet.

```
@param[in] mvals Mathematical parameter values  
@param[in] AGrad Switch to turn on analytic gradient  
@param[in] AHess Switch to turn on analytic Hessian  
@return Answer Contribution to the objective function
```

Definition at line 588 of file abinitio.py.

Here is the call graph for this function:



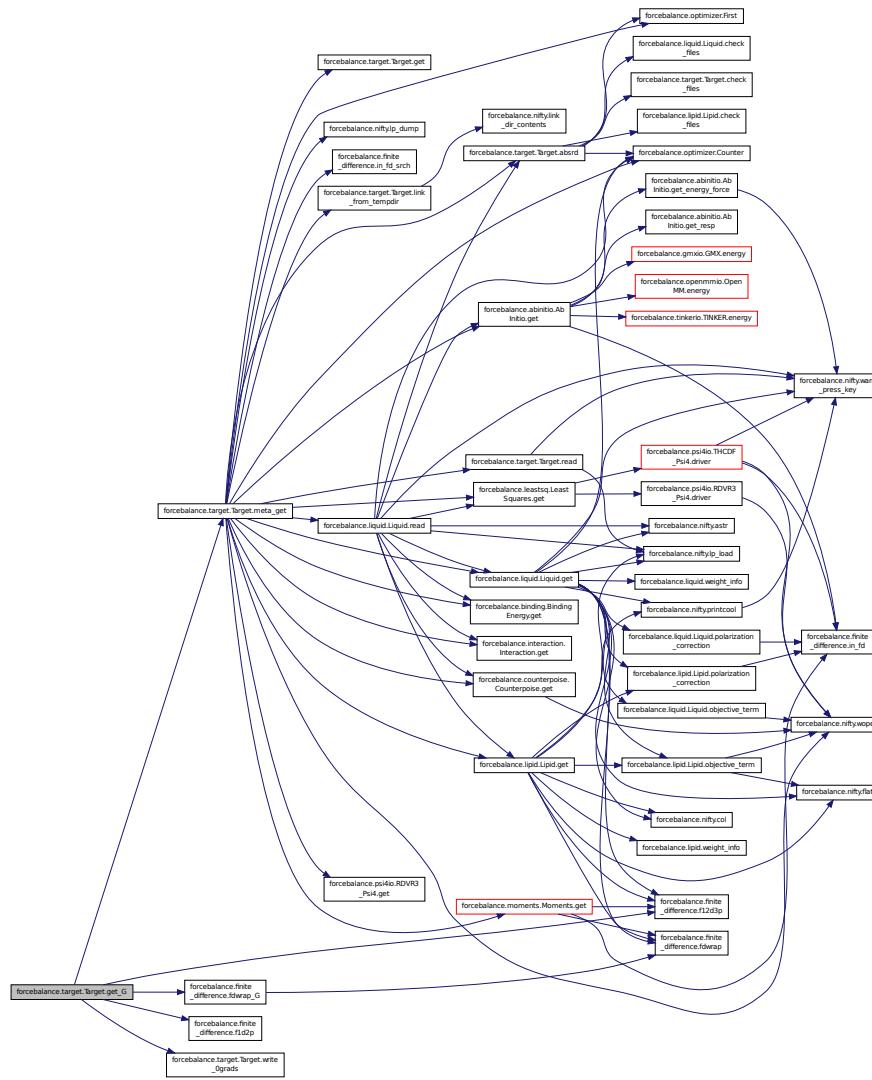
```
def forcebalance.target.Target.get_G( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



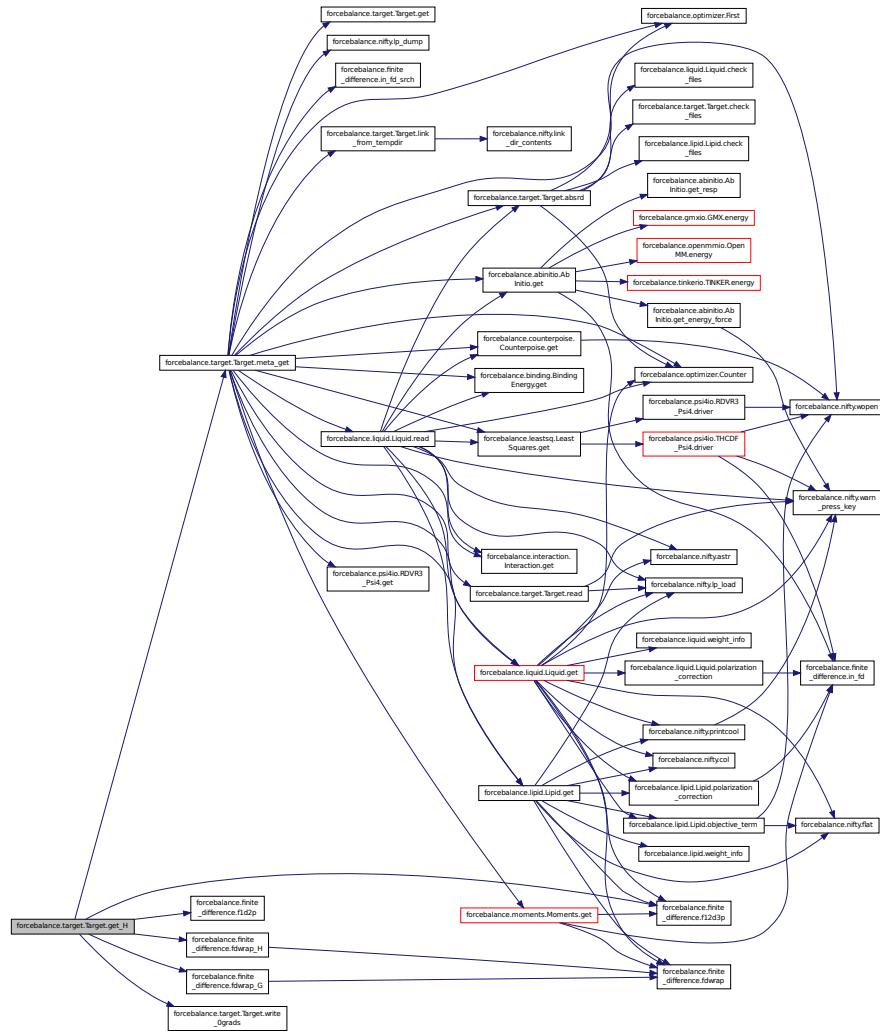
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.ABInitio.get_resp ( self, mvals, AGrad = False, AHess = False ) Electrostatic potential fitting.
```

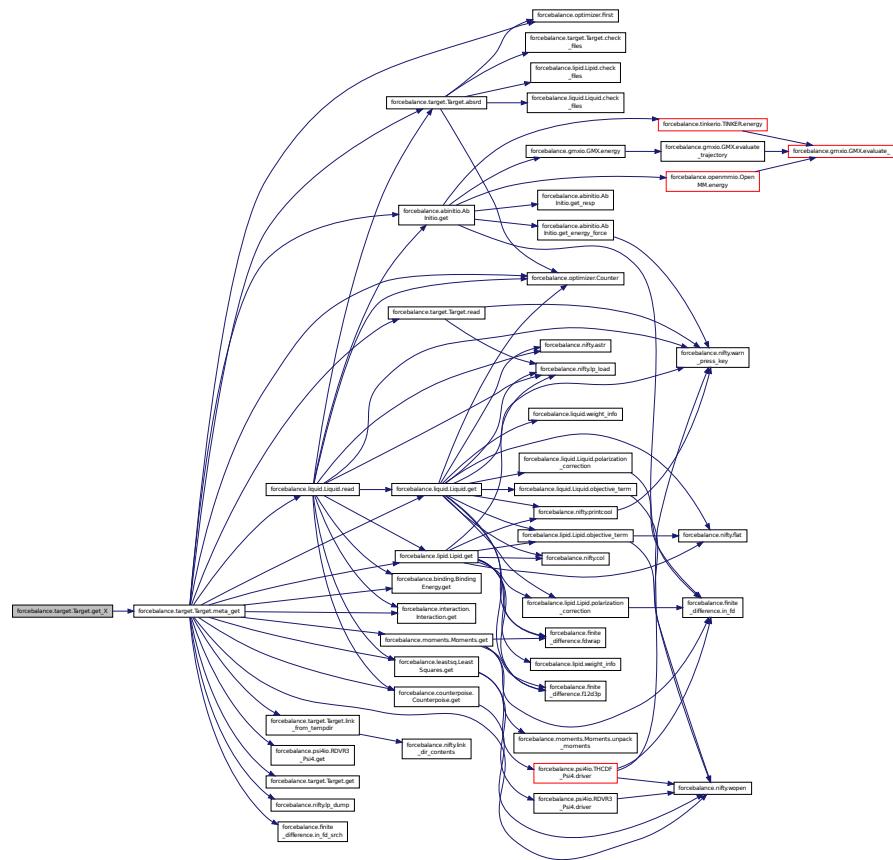
Implements the RESP objective function. (In Python so obviously not optimized.) This function takes the mathematical parameter values and returns the charges on the ATOMS (fancy mapping going on)

Definition at line 1053 of file abinitio.py.

**def forcebalance.target.Target.get\_X( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

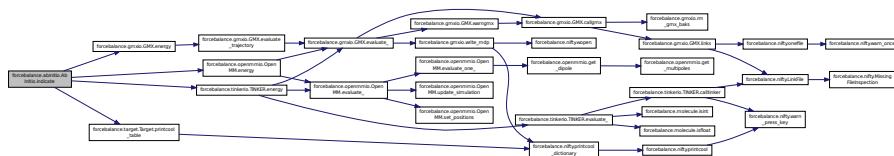
Definition at line 184 of file target.py.

Here is the call graph for this function:



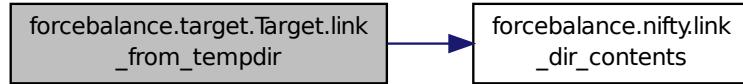
**def forcebalance.abinitio.AblInitio.optimize ( self )** Definition at line 448 of file abinitio.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

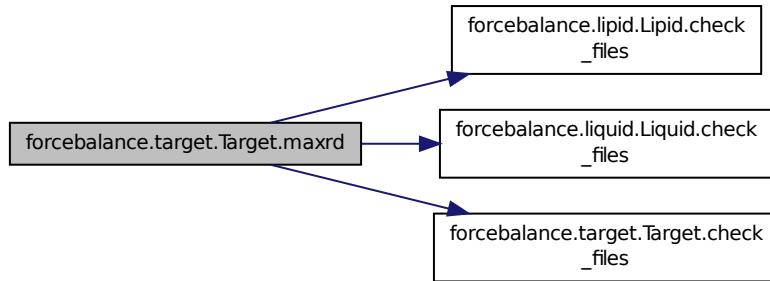
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

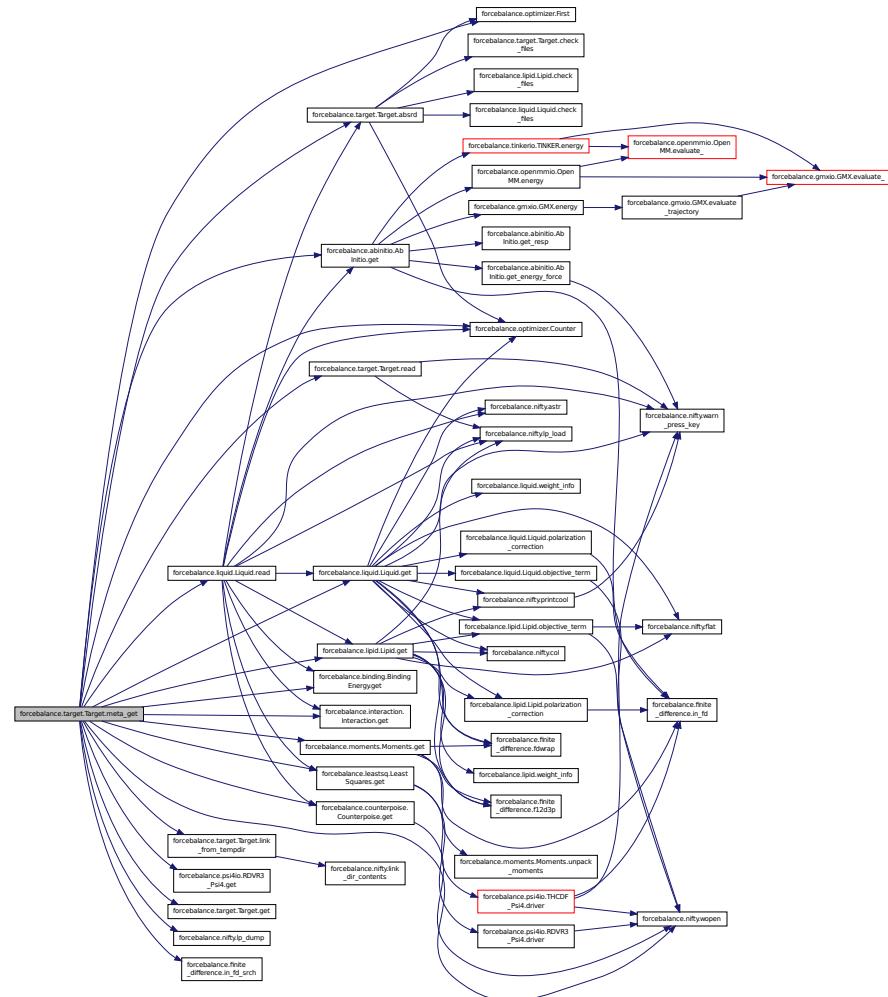


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

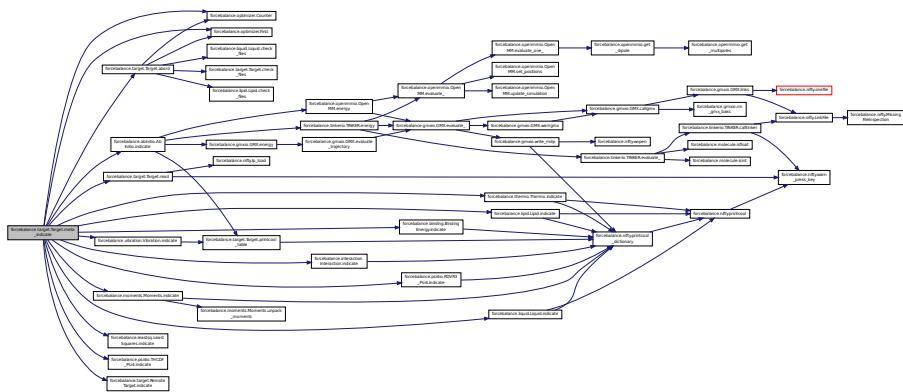
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

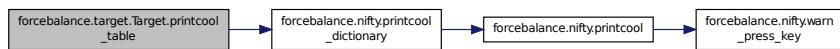
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

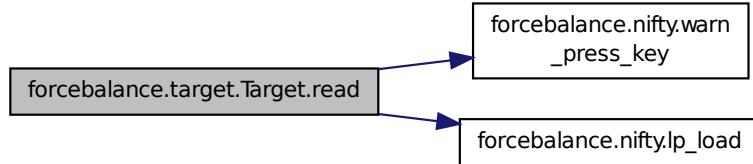
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.read_0grads ( self ) [inherited] Read a file from the target directory containing names of parameters that don't contribute to the gradient.
```

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

```
def forcebalance.abinitio.AblInitio.read_reference_data ( self ) Read the reference ab initio data from a file such as qdata.txt.
```

**Todo** Add an option for picking any slice out of qdata.txt, helpful for cross-validation

**Todo** Closer integration of reference data with program - leave behind the qdata.txt format? (For now, I like the readability of qdata.txt)

After reading in the information from qdata.txt, it is converted into the GROMACS energy units (kind of an arbitrary choice); forces (kind of a misnomer in qdata.txt) are multiplied by -1 to convert gradients to forces.

We also subtract out the mean energies of all energy arrays because energy/force matching does not account for zero-point energy differences between MM and QM (i.e. energy of electrons in core orbitals).

The configurations in force/energy matching typically come from a the thermodynamic ensemble of the MM force field at some temperature (by running MD, for example), and for many reasons it is helpful to introduce non-Boltzmann weights in front of these configurations. There are two options: WHAM Boltzmann weights (for combining the weights of several simulations together) and QM Boltzmann weights (for converting MM weights into QM weights). Note that the two sets of weights 'stack'; i.e. they can be used at the same time.

A 'hybrid' ensemble is possible where we use 50% MM and 50% QM weights. Please read more in LPW and Troy Van Voorhis, JCP Vol. 133, Pg. 231101 (2010), doi:10.1063/1.3519043.

**Todo** The WHAM Boltzmann weights are generated by external scripts (wanalyze.py and make-wham-data.sh) and passed in; perhaps these scripts can be added to the ForceBalance distribution or integrated more tightly.

Finally, note that using non-Boltzmann weights degrades the statistical information content of the snapshots. This problem will generally become worse if the ensemble to which we're reweighting is dramatically different from the one we're sampling from. We end up with a set of Boltzmann weights like [1e-9, 1e-9, 1.0, 1e-9, 1e-9 ... ] and this is essentially just one snapshot. I believe Troy is working on something to cure this problem.

Here, we have a measure for the information content of our snapshots, which comes easily from the definition of information entropy:

```
S = -1*Sum_i(P_i*log(P_i))
InfoContent = exp(-S)
```

With uniform weights, InfoContent is equal to the number of snapshots; with horrible weights, InfoContent is closer to one.

Definition at line 329 of file abinitio.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

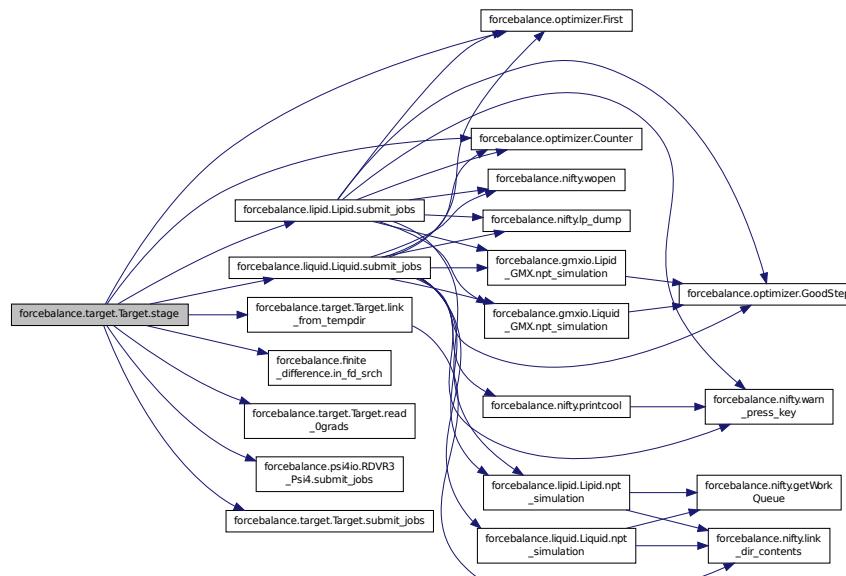
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

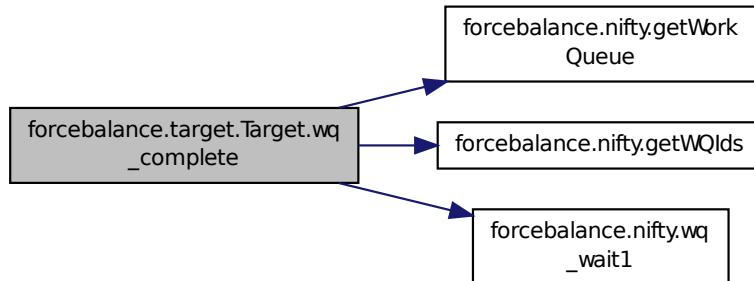


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.write_Ograds ( self, Ans ) [inherited] Write a file to the target directory containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

#### 8.1.4 Member Data Documentation

**forcebalance.abinitio.ABInitio.AtomLists** Lists of atoms to do net force/torque fitting and excluding virtual sites.

Definition at line 160 of file abinitio.py.

**forcebalance.abinitio.ABInitio.AtomMask** Definition at line 161 of file abinitio.py.

**forcebalance.abinitio.ABInitio.boltz\_wts** Initialize the base class.

Number of snapshots Whether to use WHAM Boltzmann weights Whether to use the Sampling Correction Whether to match Absolute Energies (make sure you know what you're doing) Whether to use the Covariance Matrix Whether to use QM Boltzmann weights The temperature for QM Boltzmann weights Number of atoms that we are fitting Whether to fit Energies. Whether to fit Forces. Whether to fit Electrostatic Potential. Weights for the three components. Option for how much data to write to disk. Whether to do energy and force calculations for the whole trajectory, or to do one calculation per snapshot. OpenMM-only option - whether to run the energies and forces internally. Whether we have virtual sites (set at the global option level) Attenuate the weights as a function of energy What is the energy denominator? (Valid for 'attenuate') Set upper cutoff energy WHAM Boltzmann weights

Definition at line 116 of file abinitio.py.

**forcebalance.abinitio.ABInitio.e\_err** Qualitative Indicator: average energy error (in kJ/mol)

Definition at line 134 of file abinitio.py.

**forcebalance.abinitio.ABInitio.e\_err\_pct** Definition at line 135 of file abinitio.py.

**forcebalance.abinitio.AbInitio.e.ref** Definition at line 1030 of file abinitio.py.

**forcebalance.abinitio.AbInitio.emd0** Energies of the sampling simulation.  
Definition at line 122 of file abinitio.py.

**forcebalance.abinitio.AbInitio.engine** Build keyword dictionaries to pass to engine.  
Create engine object.  
Definition at line 158 of file abinitio.py.

**forcebalance.abinitio.AbInitio.eqm** Reference (QM) energies.  
Definition at line 120 of file abinitio.py.

**forcebalance.abinitio.AbInitio.esp\_err** Qualitative Indicator: "relative RMS" for electrostatic potential.  
Definition at line 140 of file abinitio.py.

**forcebalance.abinitio.AbInitio.espval** ESP values.  
Definition at line 128 of file abinitio.py.

**forcebalance.abinitio.AbInitio.espxyz** ESP grid points.  
Definition at line 126 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err** Qualitative Indicator: average force error (fractional)  
Definition at line 137 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err\_pct** Definition at line 138 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_ref** Definition at line 1034 of file abinitio.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)  
Definition at line 160 of file target.py.

**forcebalance.abinitio.AbInitio.fitatoms** Definition at line 366 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force** Definition at line 361 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force\_map** Definition at line 209 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fqm** Reference (QM) forces.  
Definition at line 124 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fref** Definition at line 442 of file abinitio.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.  
Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.abinitio.AbInitio.invdists** Definition at line 1061 of file abinitio.py.

**forcebalance.abinitio.ABInitio.mol** Definition at line 148 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nesp** Definition at line 363 of file abinitio.py.

**forcebalance.abinitio.ABInitio.new\_vsites** Read in the reference data.

The below two options are related to whether we want to rebuild virtual site positions. Rebuild the distance matrix if virtual site positions have changed

Definition at line 166 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_err** Definition at line 141 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_err\_pct** Definition at line 142 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_ref** Definition at line 1038 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nftqm** Definition at line 438 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nnf** Definition at line 267 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nparticles** The number of (atoms + drude particles + virtual sites)

Definition at line 153 of file abinitio.py.

**forcebalance.abinitio.ABInitio.ns** Read in the trajectory file.

Definition at line 147 of file abinitio.py.

**forcebalance.abinitio.ABInitio.ntq** Definition at line 268 of file abinitio.py.

**forcebalance.abinitio.ABInitio.objective** Definition at line 1172 of file abinitio.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.qfnm** The qdata.txt file that contains the QM energies and forces.

Definition at line 130 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmatoms** The number of atoms in the QM calculation (Irrelevant if not fitting forces)

Definition at line 132 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmboltz\_wts** QM Boltzmann weights.

Definition at line 118 of file abinitio.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.abinitio.ABInitio.respterm** Definition at line 1140 of file abinitio.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.abinitio.ABInitio.save\_vmvales** Save the mvales from the last time we updated the vsites.

Definition at line 168 of file abinitio.py.

**forcebalance.target.Target.tempbase [inherited]** Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir [inherited]** Definition at line 155 of file target.py.

**forcebalance.abinitio.ABInitio.tq\_err** Definition at line 1042 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_err\_pct** Definition at line 143 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_ref** Definition at line 1041 of file abinitio.py.

**forcebalance.abinitio.ABInitio.use\_nft** Whether to compute net forces and torques, or not.

Definition at line 145 of file abinitio.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.w\_energy** Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_force** Definition at line 362 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_netforce** Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_resp** Definition at line 1054 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_torque** Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz** Definition at line 382 of file abinitio.py.

**forcebalance.target.Target.write\_indicate** [**inherited**] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

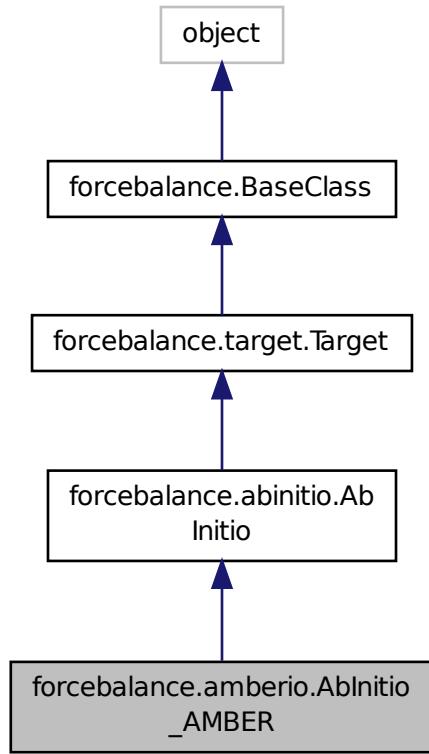
The documentation for this class was generated from the following file:

- [abinitio.py](#)

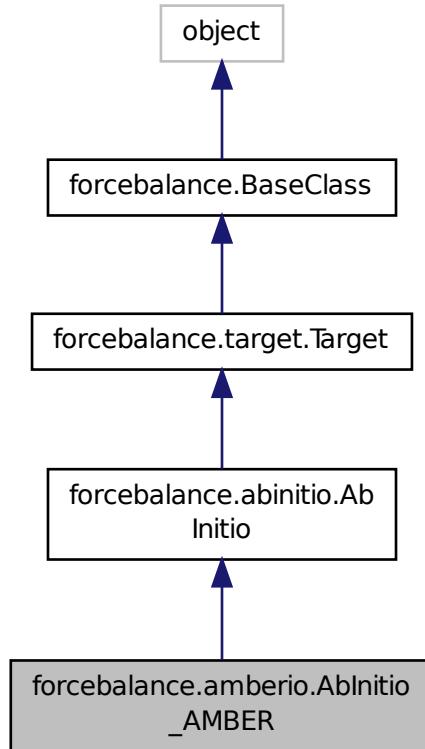
## 8.2 forcebalance.amberio.ABInitio\_AMBER Class Reference

Subclass of Target for force and energy matching using AMBER.

Inheritance diagram for forcebalance.amberio.AbInitio\_AMBER:



Collaboration diagram for forcebalance.amberio.AbInitio\_AMBER:



### Public Member Functions

- def `_init_`
- def `prepare_temp_directory`
- def `energy_force_driver_all_external_`
- def `energy_force_driver_all`
- def `build_invdist`
- def `compute_netforce_torque`
- def `read_reference_data`

*Read the reference ab initio data from a file such as qdata.txt.*

- def `indicate`
- def `energy_all`
- def `energy_force_all`
- def `energy_force_transform`
- def `energy_one`
- def `energy_force_one`
- def `energy_force_transform_one`
- def `get_energy_force`

*LPW 06-30-2013.*

- def `get_resp`  
*Electrostatic potential fitting.*
- def `get`
- def `get_X`  
*Computes the objective function contribution without any parametric derivatives.*
- def `read_0grads`  
*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_0grads`  
*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`  
*Computes the objective function contribution and its gradient.*
- def `get_H`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `coords`  
*Name of the trajectory, we need this BEFORE initializing the SuperClass.*
- `all_at_once`  
*all\_at\_once is not implemented.*
- `boltz_wts`  
*Initialize the base class.*
- `qmboltz_wts`  
*QM Boltzmann weights.*

- **eqm**  
*Reference (QM) energies.*
- **emd0**  
*Energies of the sampling simulation.*
- **fqm**  
*Reference (QM) forces.*
- **espxyz**  
*ESP grid points.*
- **espval**  
*ESP values.*
- **qfnm**  
*The qdata.txt file that contains the QM energies and forces.*
- **qatoms**  
*The number of atoms in the QM calculation (Irrelevant if not fitting forces)*
- **e\_err**  
*Qualitative Indicator: average energy error (in kJ/mol)*
- **e\_err\_pct**
- **f\_err**  
*Qualitative Indicator: average force error (fractional)*
- **f\_err\_pct**
- **esp\_err**  
*Qualitative Indicator: "relative RMS" for electrostatic potential.*
- **nf\_err**
- **nf\_err\_pct**
- **tq\_err\_pct**
- **use\_nft**  
*Whether to compute net forces and torques, or not.*
- **ns**  
*Read in the trajectory file.*
- **mol**
- **nparticles**  
*The number of (atoms + drude particles + virtual sites)*
- **engine**  
*Build keyword dictionaries to pass to engine.*
- **AtomLists**  
*Lists of atoms to do net force/torque fitting and excluding virtual sites.*
- **AtomMask**
- **new\_vsites**  
*Read in the reference data.*
- **save\_vmvalls**  
*Save the mvalls from the last time we updated the vsites.*
- **force\_map**
- **nnf**
- **ntq**
- **force**
- **w\_force**
- **nesp**
- **fitatoms**

- `whamboltz`
- `nftqm`
- `fref`
- `w_energy`
- `w_netforce`
- `w_torque`
- `e_ref`
- `f_ref`
- `nf_ref`
- `tq_ref`
- `tq_err`
- `w_resp`
- `invdists`
- `respterm`
- `objective`
- `rd`

*Root directory of the whole project.*
- `pgrad`

*Iteration where we turn on zero-gradient skipping.*
- `tempbase`

*Relative directory of target.*
- `tempdir`
- `rundir`

*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- `FF`

*Need the forcefield (here for now)*
- `xct`

*Counts how often the objective function was computed.*
- `gct`

*Counts how often the gradient was computed.*
- `hct`

*Counts how often the Hessian was computed.*
- `read_indicate`

*Whether to read indicate.log from file when restarting an aborted run.*
- `write_indicate`

*Whether to write indicate.log at every iteration (true for all but remote.)*
- `read_objective`

*Whether to read objective.p from file when restarting an aborted run.*
- `write_objective`

*Whether to write objective.p at every iteration (true for all but remote.)*
- `verbose_options`
- `PrintOptionDict`

### 8.2.1 Detailed Description

Subclass of Target for force and energy matching using AMBER.

Implements the prepare and energy\_force\_driver methods. The get method is in the base class.

Definition at line 171 of file amberio.py.

## 8.2.2 Constructor & Destructor Documentation

```
def forcebalance.amberio.AbInitio_AMBER.__init__( self, options, tgt_opts, forcefield ) Definition at line 174  
of file amberio.py.
```

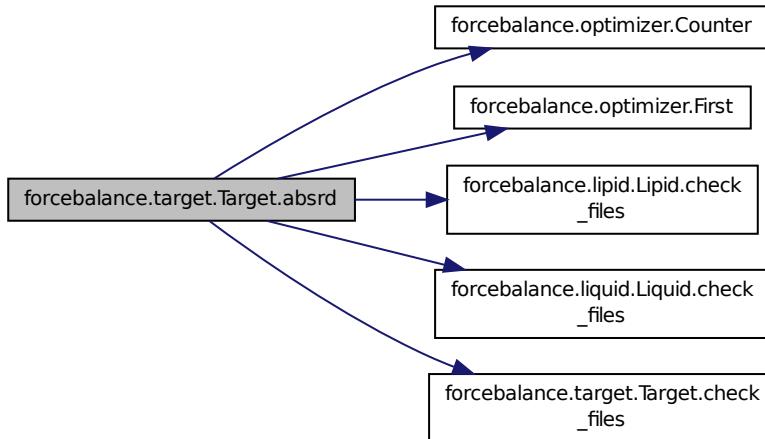
## 8.2.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd( self, inum = None ) [inherited] Supply the correct directory  
specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.AbInitio.build_invdist( self, mvals ) [inherited] Definition at line 171 of file  
abinitio.py.
```

```
def forcebalance.target.Target.check_files( self, there ) [inherited] Check this directory for the presence  
of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

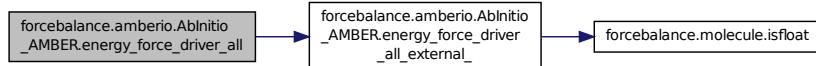
```
def forcebalance.abinitio.AbInitio.compute_netforce_torque( self, xyz, force, QM = False )  
[inherited] Definition at line 200 of file abinitio.py.
```

```
def forcebalance.abinitio.AbInitio.energy_all( self ) [inherited] Definition at line 472 of file abinitio.py.
```

```
def forcebalance.abinitio.AbInitio.energy_force_all( self ) [inherited] Definition at line 478 of file abinitio.py.
```

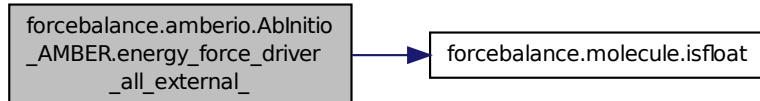
```
def forcebalance.amberio.ABInitio_AMBER.energy_force_driver_all ( self ) Definition at line 228 of file amberio.py.
```

Here is the call graph for this function:



```
def forcebalance.amberio.ABInitio_AMBER.energy_force_driver_all_external ( self ) Definition at line 190 of file amberio.py.
```

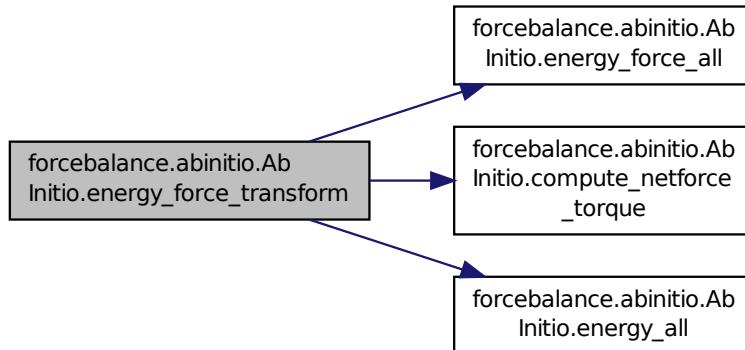
Here is the call graph for this function:



```
def forcebalance.abinitio.ABInitio.energy_force_one ( self, i ) [inherited] Definition at line 507 of file abinitio.py.
```

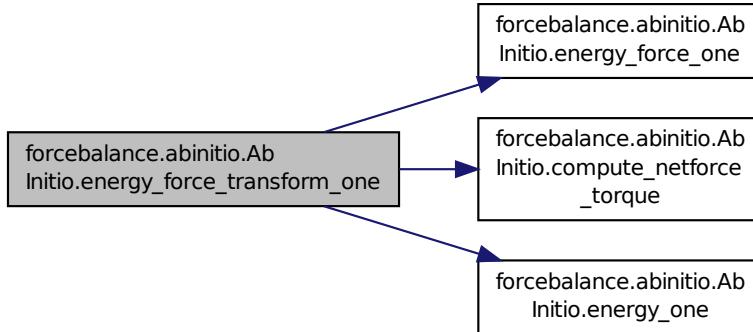
```
def forcebalance.abinitio.ABInitio.energy_force_transform ( self ) [inherited] Definition at line 484 of file abinitio.py.
```

Here is the call graph for this function:



**def forcebalance.abinitio.ABInitio.energy\_force.transform.one ( self, i ) [inherited]** Definition at line 513 of file abinitio.py.

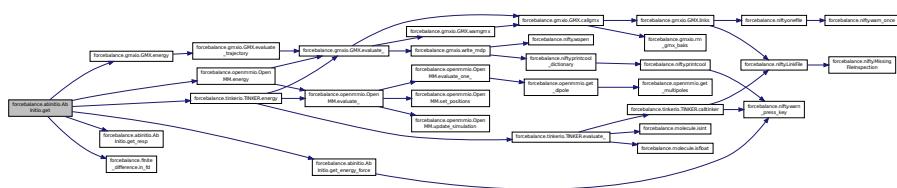
Here is the call graph for this function:



**def forcebalance.abinitio.ABInitio.energy\_one( self, i ) [inherited]** Definition at line 501 of file abinitio.py.

**def forcebalance.abinitio.ABInitio.get( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Definition at line 1152 of file abinitio.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.ABInitio.get_energy_force ( self, mvals, AGrad = False, AHess = False )  
[inherited] LPW 06-30-2013.
```

This subroutine builds the objective function (and optionally its derivatives) from a general simulation software. This is in contrast to using GROMACS-X2, which computes the objective function and prints it out; then 'get' only needs to call GROMACS and read it in.

This subroutine interfaces with simulation software 'drivers'. The driver is only expected to give the energy and forces.

Now this subroutine may sound trivial since the objective function is simply a least-squares quantity  $(M-Q)^2$  - but there are a number of nontrivial considerations. I will list them here.

0) Polytensor formulation: Because there may exist covariance between different components of the force (or covariance between the energy and the force), we build the objective function by taking outer products of vectors that have the form [E F\_1x F\_1y F\_1z F\_2x F\_2y ... ], and then we trace it with the inverse of the covariance matrix to get the objective function.

This version implements both the polytensor formulation and the standard formulation.

1) Boltzmann weights and normalization: Each snapshot has its own Boltzmann weight, which may or may not be normalized. This subroutine does the normalization automatically.

2) Subtracting out the mean energy gap: The zero-point energy difference between reference data and simulation is meaningless. This subroutine subtracts it out.

3) Hybrid ensembles: This program builds a combined objective function from both MM and QM ensembles, which is rigorously better than using a single ensemble.

Note that this subroutine does not do EVERYTHING that GROMACS-X2 can do, which includes:

1) Internal coordinate systems 2) 'Sampling correction' (deprecated, since it doesn't seem to work) 3) Analytic derivatives

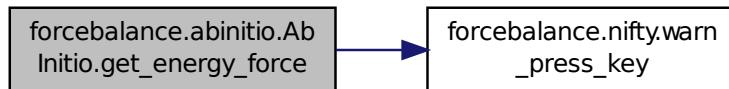
In the previous code (ForTune) this subroutine used analytic first derivatives of the energy and force to build the derivatives of the objective function. Here I will take a simplified approach, because building the derivatives are cumbersome. For now we will return the objective function ONLY. A two-point central difference should give us the first and diagonal second derivative anyhow.

**Todo** Parallelization over snapshots is not implemented yet

```
@param[in] mvals Mathematical parameter values
@param[in] AGrad Switch to turn on analytic gradient
@param[in] AHess Switch to turn on analytic Hessian
@return Answer Contribution to the objective function
```

Definition at line 588 of file abinitio.py.

Here is the call graph for this function:



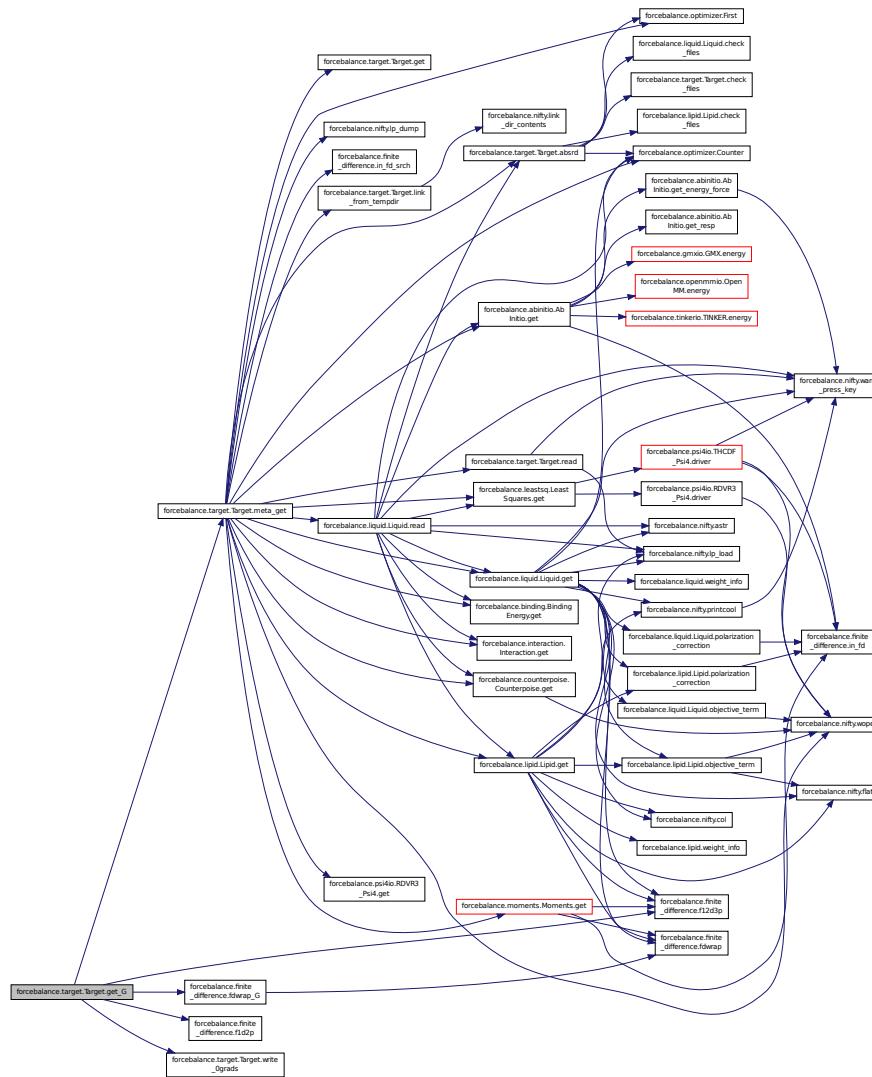
**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



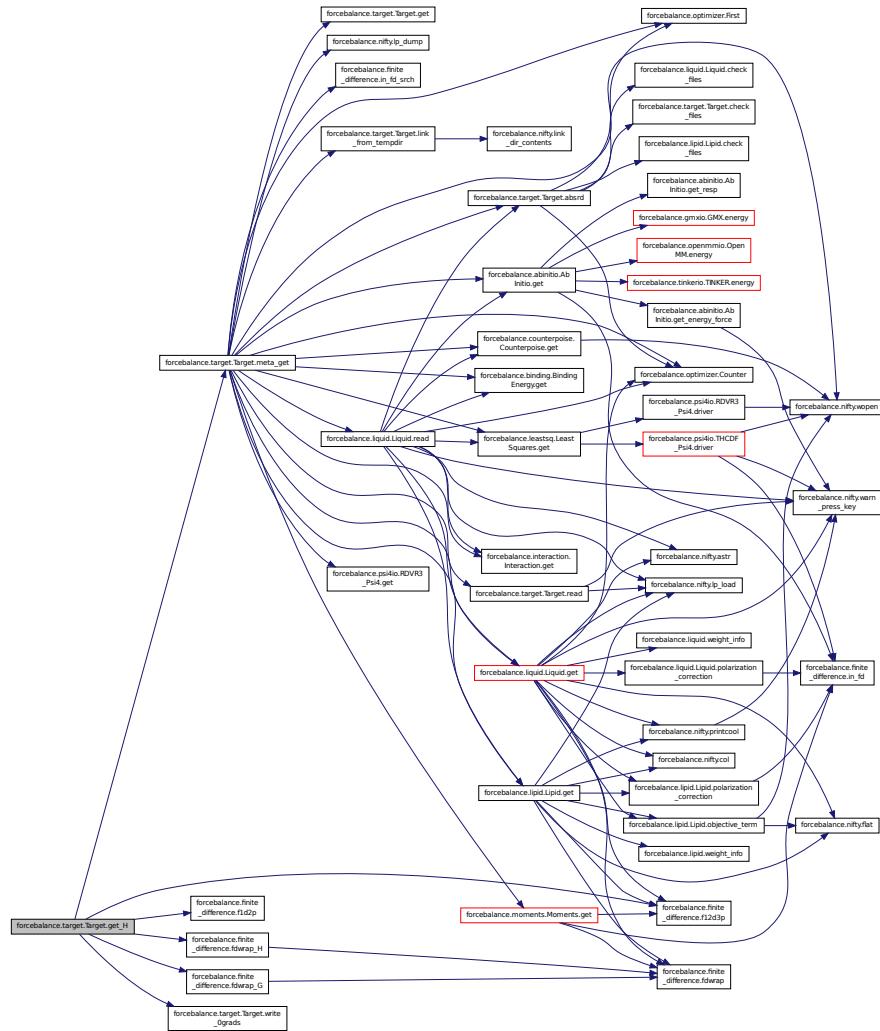
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.AbInitio.get_resp ( self, mvals, AGrad = False, AHess = False )  
[inherited] Electrostatic potential fitting.
```

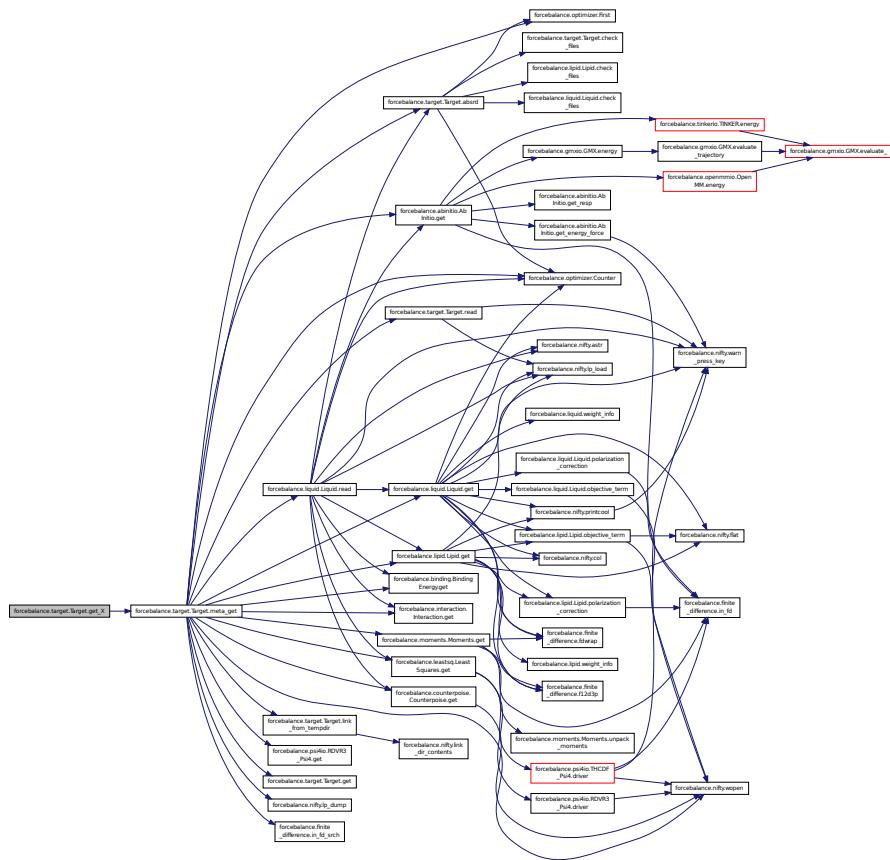
Implements the RESP objective function. (In Python so obviously not optimized.) This function takes the mathematical parameter values and returns the charges on the ATOMS (fancy mapping going on)

Definition at line 1053 of file abinitio.py.

**def forcebalance.target.Target.get\_X( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

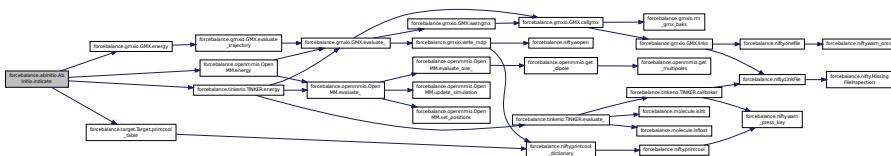
Definition at line 184 of file target.py.

Here is the call graph for this function:



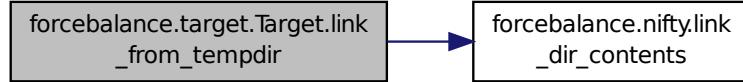
**def forcebalance.abinitio.ABInitio.indicate ( self ) [inherited]** Definition at line 448 of file abinitio.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.link_from_tempdir( self, absdestdir ) [inherited] Definition at line 315 of file target.py.
```

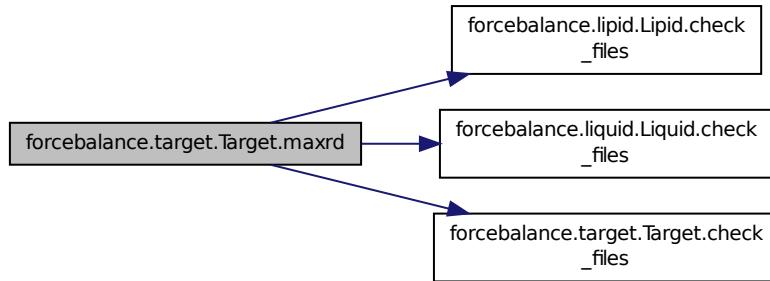
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

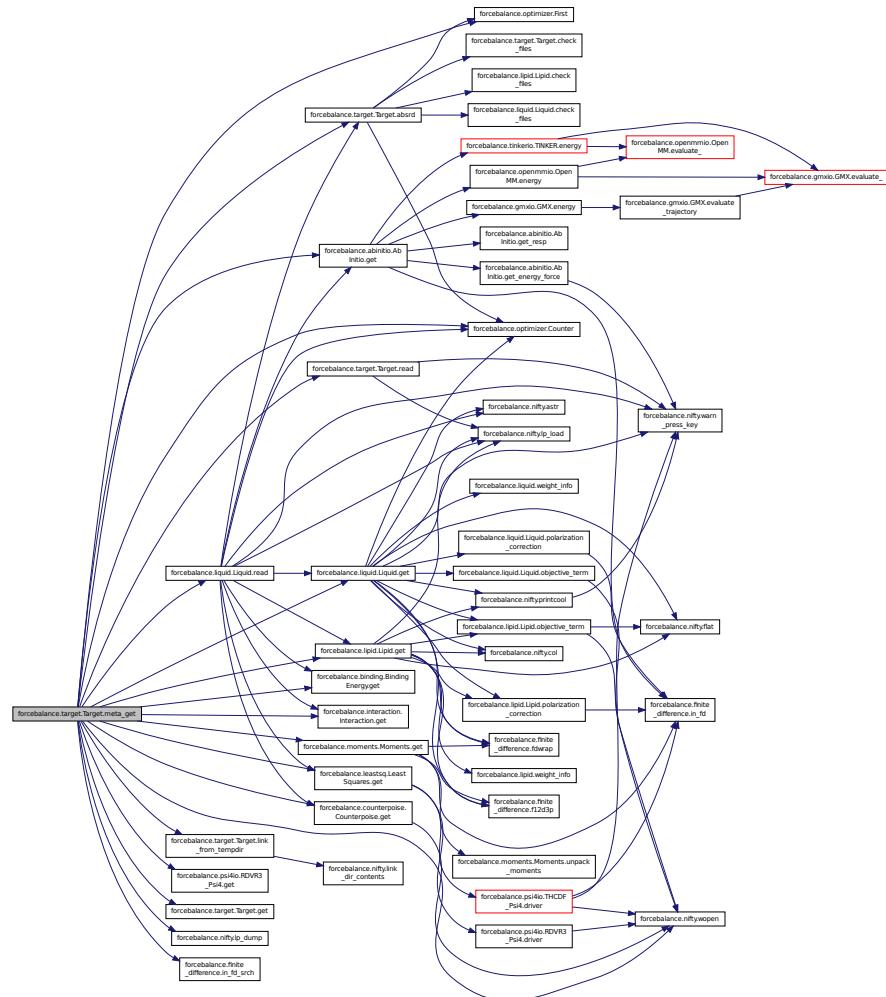


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

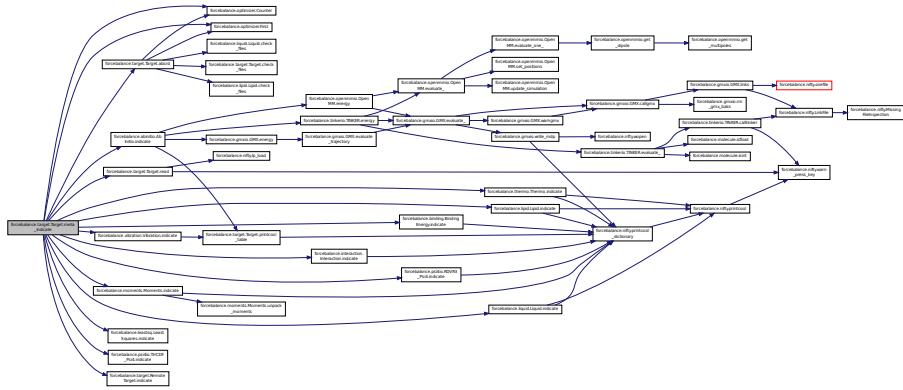
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



**def forcebalance.amberio.ABInitio\_AMBER.prepare\_temp\_directory ( self, options, tgt\_opts )** Definition at line 181 of file amberio.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

Here is the call graph for this function:

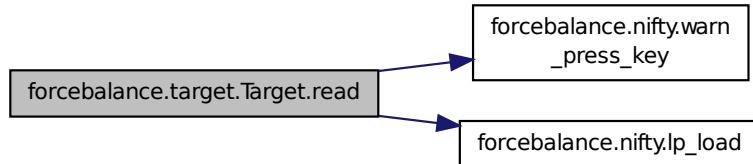


```
def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]
```

Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.read_0grads ( self ) [inherited]
```

Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

```
def forcebalance.abinitio.AbInitio.read_reference_data ( self ) [inherited]
```

Read the reference ab initio data from a file such as qdata.txt.

**Todo** Add an option for picking any slice out of qdata.txt, helpful for cross-validation

**Todo** Closer integration of reference data with program - leave behind the qdata.txt format? (For now, I like the readability of qdata.txt)

After reading in the information from qdata.txt, it is converted into the GROMACS energy units (kind of an arbitrary choice); forces (kind of a misnomer in qdata.txt) are multiplied by -1 to convert gradients to forces.

We also subtract out the mean energies of all energy arrays because energy/force matching does not account for zero-point energy differences between MM and QM (i.e. energy of electrons in core orbitals).

The configurations in force/energy matching typically come from a the thermodynamic ensemble of the MM force field at some temperature (by running MD, for example), and for many reasons it is helpful to introduce non-Boltzmann weights in front of these configurations. There are two options: WHAM Boltzmann weights (for combining the weights of several simulations together) and QM Boltzmann weights (for converting MM weights into QM weights). Note that the two sets of weights 'stack'; i.e. they can be used at the same time.

A 'hybrid' ensemble is possible where we use 50% MM and 50% QM weights. Please read more in LPW and Troy Van Voorhis, JCP Vol. 133, Pg. 231101 (2010), doi:10.1063/1.3519043.

**Todo** The WHAM Boltzmann weights are generated by external scripts (wanalyze.py and make-wham-data.sh) and passed in; perhaps these scripts can be added to the ForceBalance distribution or integrated more tightly.

Finally, note that using non-Boltzmann weights degrades the statistical information content of the snapshots. This problem will generally become worse if the ensemble to which we're reweighting is dramatically different from the one we're sampling from. We end up with a set of Boltzmann weights like [1e-9, 1e-9, 1.0, 1e-9, 1e-9 ... ] and this is essentially just one snapshot. I believe Troy is working on something to cure this problem.

Here, we have a measure for the information content of our snapshots, which comes easily from the definition of information entropy:

```
S = -1*Sum_i(P_i*log(P_i))
InfoContent = exp(-S)
```

With uniform weights, InfoContent is equal to the number of snapshots; with horrible weights, InfoContent is closer to one.

Definition at line 329 of file abinitio.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

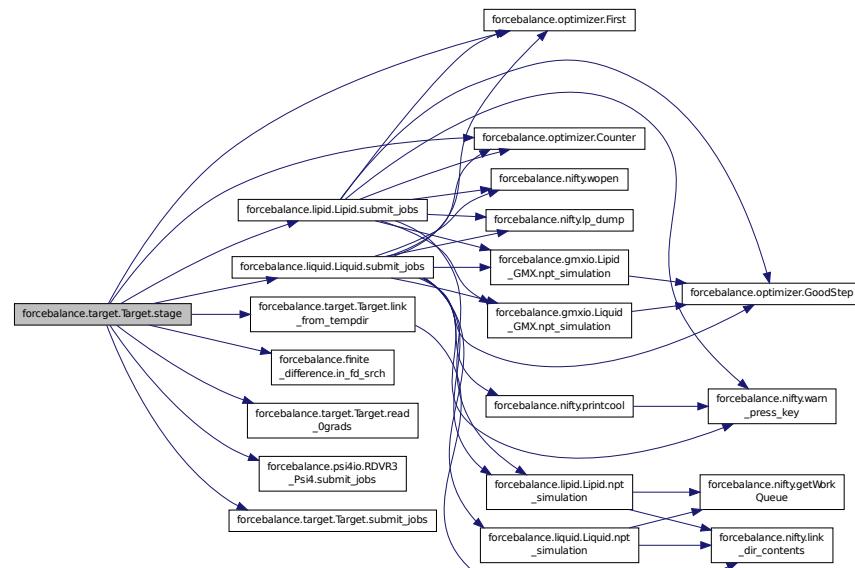
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

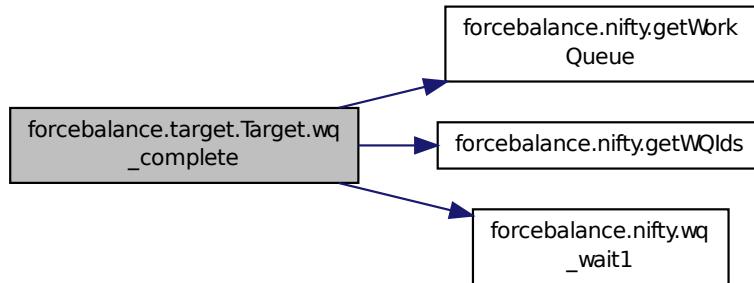


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work  
Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.write_0grads ( self, Ans ) [inherited] Write a file to the target directory  
containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

#### 8.2.4 Member Data Documentation

```
forcebalance.amberio.AbInitio_AMBER.all_at_once all_at_once is not implemented.
```

Definition at line 179 of file amberio.py.

```
forcebalance.abinitio.AbInitio.AtomLists [inherited] Lists of atoms to do net force/torque fitting and excluding virtual sites.
```

Definition at line 160 of file abinitio.py.

```
forcebalance.abinitio.AbInitio.AtomMask [inherited] Definition at line 161 of file abinitio.py.
```

```
forcebalance.abinitio.AbInitio.boltz_wts [inherited] Initialize the base class.
```

Number of snapshots Whether to use WHAM Boltzmann weights Whether to use the Sampling Correction Whether to match Absolute Energies (make sure you know what you're doing) Whether to use the Covariance Matrix Whether to use QM Boltzmann weights The temperature for QM Boltzmann weights Number of atoms that we are fitting Whether to fit Energies. Whether to fit Forces. Whether to fit Electrostatic Potential. Weights for the three components. Option for how much data to write to disk. Whether to do energy and force calculations for the whole trajectory, or to do one calculation per snapshot. OpenMM-only option - whether to run the energies and forces internally. Whether we have virtual sites (set at the global option level) Attenuate the weights as a function of energy What is the energy denominator? (Valid for 'attenuate') Set upper cutoff energy WHAM Boltzmann weights

Definition at line 116 of file abinitio.py.

**forcebalance.amberio.ABInitio\_AMBER.coords** Name of the trajectory, we need this BEFORE initializing the Super-Class.

Definition at line 176 of file amberio.py.

**forcebalance.abinitio.ABInitio.e\_err [inherited]** Qualitative Indicator: average energy error (in kJ/mol)  
Definition at line 134 of file abinitio.py.

**forcebalance.abinitio.ABInitio.e\_err\_pct [inherited]** Definition at line 135 of file abinitio.py.

**forcebalance.abinitio.ABInitio.e\_ref [inherited]** Definition at line 1030 of file abinitio.py.

**forcebalance.abinitio.ABInitio.emd0 [inherited]** Energies of the sampling simulation.  
Definition at line 122 of file abinitio.py.

**forcebalance.abinitio.ABInitio.engine [inherited]** Build keyword dictionaries to pass to engine.  
Create engine object.  
Definition at line 158 of file abinitio.py.

**forcebalance.abinitio.ABInitio.eqm [inherited]** Reference (QM) energies.  
Definition at line 120 of file abinitio.py.

**forcebalance.abinitio.ABInitio.esp\_err [inherited]** Qualitative Indicator: "relative RMS" for electrostatic potential.  
Definition at line 140 of file abinitio.py.

**forcebalance.abinitio.ABInitio.espval [inherited]** ESP values.  
Definition at line 128 of file abinitio.py.

**forcebalance.abinitio.ABInitio.espxyz [inherited]** ESP grid points.  
Definition at line 126 of file abinitio.py.

**forcebalance.abinitio.ABInitio.f\_err [inherited]** Qualitative Indicator: average force error (fractional)  
Definition at line 137 of file abinitio.py.

**forcebalance.abinitio.ABInitio.f\_err\_pct [inherited]** Definition at line 138 of file abinitio.py.

**forcebalance.abinitio.ABInitio.f\_ref [inherited]** Definition at line 1034 of file abinitio.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)  
Definition at line 160 of file target.py.

**forcebalance.abinitio.ABInitio.fitatoms [inherited]** Definition at line 366 of file abinitio.py.

**forcebalance.abinitio.ABInitio.force [inherited]** Definition at line 361 of file abinitio.py.

**forcebalance.abinitio.ABInitio.force\_map [inherited]** Definition at line 209 of file abinitio.py.

**forcebalance.abinitio.ABInitio.fqm [inherited]** Reference (QM) forces.  
Definition at line 124 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fref** [inherited] Definition at line 442 of file abinitio.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.  
Definition at line 164 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.abinitio.AbInitio.invdists** [inherited] Definition at line 1061 of file abinitio.py.

**forcebalance.abinitio.AbInitio.mol** [inherited] Definition at line 148 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nesp** [inherited] Definition at line 363 of file abinitio.py.

**forcebalance.abinitio.AbInitio.new\_vsites** [inherited] Read in the reference data.

The below two options are related to whether we want to rebuild virtual site positions. Rebuild the distance matrix if virtual site positions have changed  
Definition at line 166 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_err** [inherited] Definition at line 141 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_err\_pct** [inherited] Definition at line 142 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_ref** [inherited] Definition at line 1038 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nftqm** [inherited] Definition at line 438 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nnf** [inherited] Definition at line 267 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nparticles** [inherited] The number of (atoms + drude particles + virtual sites)  
Definition at line 153 of file abinitio.py.

**forcebalance.abinitio.AbInitio.ns** [inherited] Read in the trajectory file.  
Definition at line 147 of file abinitio.py.

**forcebalance.abinitio.AbInitio.ntq** [inherited] Definition at line 268 of file abinitio.py.

**forcebalance.abinitio.AbInitio.objective** [inherited] Definition at line 1172 of file abinitio.py.

**forcebalance.target.Target.pgrad** [inherited] Iteration where we turn on zero-gradient skipping.  
Dictionary of whether to call the derivatives.  
Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.abinitio.AbInitio.qfnm** [inherited] The qdata.txt file that contains the QM energies and forces.  
Definition at line 130 of file abinitio.py.

**forcebalance.abinitio.AblInitio.qmatoms** [inherited] The number of atoms in the QM calculation (Irrelevant if not fitting forces)

Definition at line 132 of file abinitio.py.

**forcebalance.abinitio.AblInitio.qmboltz\_wts** [inherited] QM Boltzmann weights.

Definition at line 118 of file abinitio.py.

**forcebalance.target.Target.rd** [inherited] Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate** [inherited] Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [inherited] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.abinitio.AblInitio.respterm** [inherited] Definition at line 1140 of file abinitio.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.abinitio.AblInitio.save\_vmvabs** [inherited] Save the mvabs from the last time we updated the vsites.

Definition at line 168 of file abinitio.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target\_name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.abinitio.AblInitio.tq\_err** [inherited] Definition at line 1042 of file abinitio.py.

**forcebalance.abinitio.AblInitio.tq\_err\_pct** [inherited] Definition at line 143 of file abinitio.py.

**forcebalance.abinitio.AblInitio.tq\_ref** [inherited] Definition at line 1041 of file abinitio.py.

**forcebalance.abinitio.ABInitio.use\_nft [inherited]** Whether to compute net forces and torques, or not.  
Definition at line 145 of file abinitio.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.w\_energy [inherited]** Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_force [inherited]** Definition at line 362 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_netforce [inherited]** Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_resp [inherited]** Definition at line 1054 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_torque [inherited]** Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz [inherited]** Definition at line 382 of file abinitio.py.

**forcebalance.target.Target.write\_indicate [inherited]** Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective [inherited]** Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 162 of file target.py.

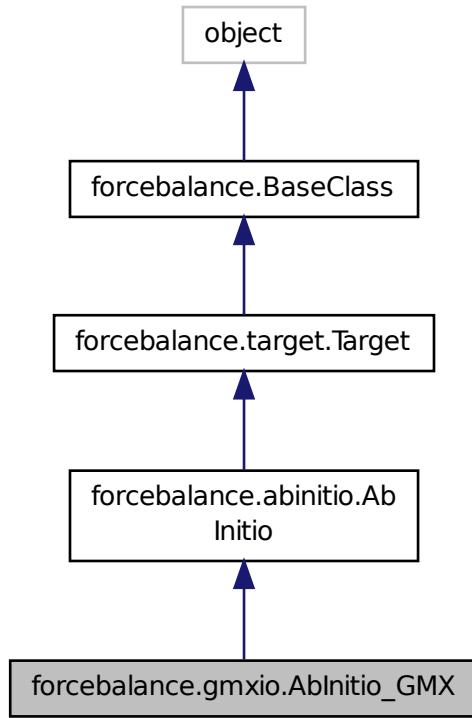
The documentation for this class was generated from the following file:

- [amberio.py](#)

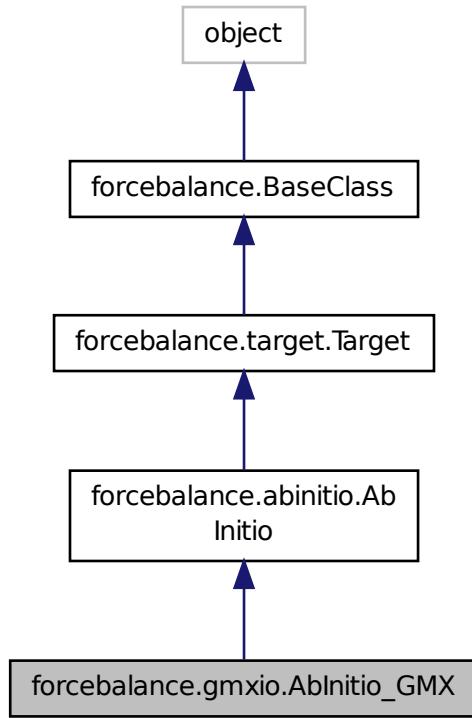
### 8.3 forcebalance.gmxio.ABInitio\_GMX Class Reference

Subclass of ABInitio for force and energy matching using GROMACS.

Inheritance diagram for forcebalance.gmxio.AbInitio\_GMX:



Collaboration diagram for forcebalance.gmxio.AbInitio\_GMX:



### Public Member Functions

- def `_init_`
- def `build_invdist`
- def `compute_netforce_torque`
- def `read_reference_data`

*Read the reference ab initio data from a file such as qdata.txt.*

- def `indicate`
- def `energy_all`
- def `energy_force_all`
- def `energy_force_transform`
- def `energy_one`
- def `energy_force_one`
- def `energy_force_transform_one`
- def `get_energy_force`

*LPW 06-30-2013.*

- def `get_resp`

*Electrostatic potential fitting.*

- def `get`

- def `get_X`  
*Computes the objective function contribution without any parametric derivatives.*
- def `read_0grads`  
*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_0grads`  
*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`  
*Computes the objective function contribution and its gradient.*
- def `get_H`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `engine_`  
*Default file names for coordinates, top and mdp files.*
- `boltz_wts`  
*Initialize the base class.*
- `qmboltz_wts`  
*QM Boltzmann weights.*
- `eqm`  
*Reference (QM) energies.*
- `emd0`  
*Energies of the sampling simulation.*
- `fqm`

*Reference (QM) forces.*

- [espxyz](#)  
*ESP grid points.*
- [espval](#)  
*ESP values.*
- [qfnm](#)  
*The qdata.txt file that contains the QM energies and forces.*
- [qatoms](#)  
*The number of atoms in the QM calculation (Irrelevant if not fitting forces)*
- [e\\_err](#)  
*Qualitative Indicator: average energy error (in kJ/mol)*
- [e\\_err\\_pct](#)
- [f\\_err](#)  
*Qualitative Indicator: average force error (fractional)*
- [f\\_err\\_pct](#)
- [esp\\_err](#)  
*Qualitative Indicator: "relative RMS" for electrostatic potential.*
- [nf\\_err](#)
- [nf\\_err\\_pct](#)
- [tq\\_err\\_pct](#)
- [use\\_nft](#)  
*Whether to compute net forces and torques, or not.*
- [ns](#)  
*Read in the trajectory file.*
- [mol](#)
- [nparticles](#)  
*The number of (atoms + drude particles + virtual sites)*
- [engine](#)  
*Build keyword dictionaries to pass to engine.*
- [AtomLists](#)  
*Lists of atoms to do net force/torque fitting and excluding virtual sites.*
- [AtomMask](#)
- [new\\_vsites](#)  
*Read in the reference data.*
- [save\\_vmvalls](#)  
*Save the mvalls from the last time we updated the vsites.*
- [force\\_map](#)
- [nnf](#)
- [ntq](#)
- [force](#)
- [w\\_force](#)
- [nesp](#)
- [fitatoms](#)
- [whamboltz](#)
- [nftqm](#)
- [fref](#)
- [w\\_energy](#)
- [w\\_netforce](#)

- `w_torque`
- `e_ref`
- `f_ref`
- `nf_ref`
- `tq_ref`
- `tq_err`
- `w_resp`
- `invdists`
- `respterm`
- `objective`
- `rd`

*Root directory of the whole project.*
- `pgrad`

*Iteration where we turn on zero-gradient skipping.*
- `tempbase`

*Relative directory of target.*
- `tempdir`
- `rundir`

*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- `FF`

*Need the forcefield (here for now)*
- `xct`

*Counts how often the objective function was computed.*
- `gct`

*Counts how often the gradient was computed.*
- `hct`

*Counts how often the Hessian was computed.*
- `read_indicate`

*Whether to read indicate.log from file when restarting an aborted run.*
- `write_indicate`

*Whether to write indicate.log at every iteration (true for all but remote.)*
- `read_objective`

*Whether to read objective.p from file when restarting an aborted run.*
- `write_objective`

*Whether to write objective.p at every iteration (true for all but remote.)*
- `verbose_options`
- `PrintOptionDict`

### 8.3.1 Detailed Description

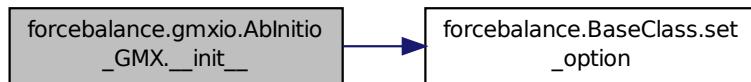
Subclass of AbInitio for force and energy matching using GROMACS.

Definition at line 1449 of file gmxio.py.

### 8.3.2 Constructor & Destructor Documentation

```
def forcebalance.gmxio.AbInitio_GMX.__init__ ( self, options, tgt_opts, forcefield ) Definition at line 1450 of file gmxio.py.
```

Here is the call graph for this function:



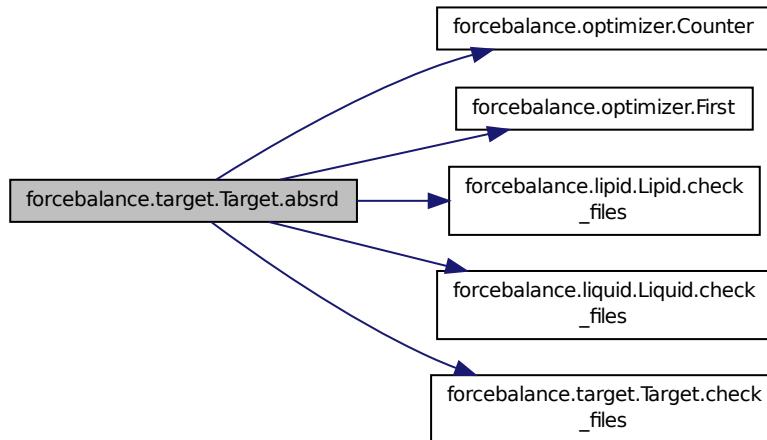
### 8.3.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.AbInitio.build_invdist ( self, mvals ) [inherited] Definition at line 171 of file abinitio.py.
```

```
def forcebalance.target.Target.check_files ( self, there ) [inherited] Check this directory for the presence of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.abinitio.AbInitio.compute_netforce_torque ( self, xyz, force, QM = False )  
[inherited] Definition at line 200 of file abinitio.py.
```

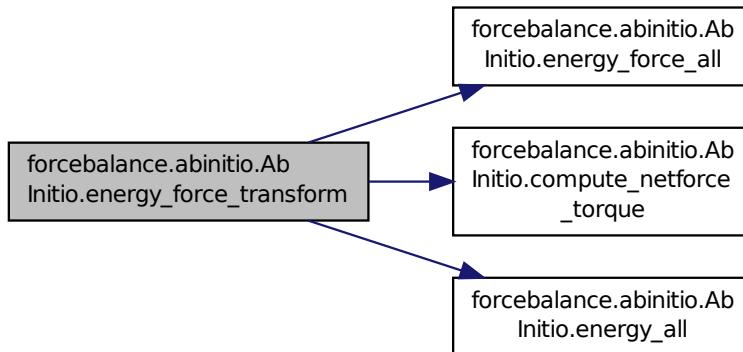
```
def forcebalance.abinitio.AbInitio.energy_all ( self ) [inherited] Definition at line 472 of file abinitio.py.
```

```
def forcebalance.abinitio.AbInitio.energy_force_all ( self ) [inherited] Definition at line 478 of file abinitio.py.
```

```
def forcebalance.abinitio.AbInitio.energy_force_one ( self, i ) [inherited] Definition at line 507 of file abinitio.py.
```

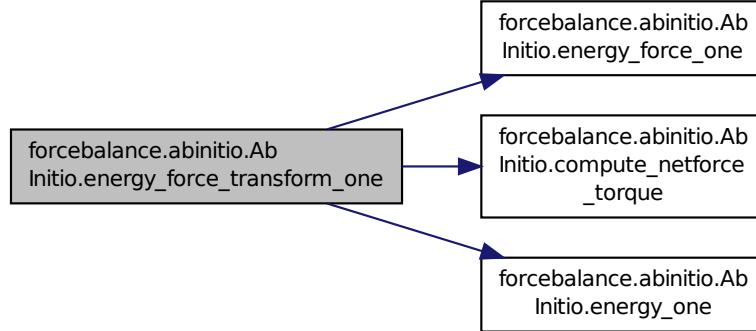
```
def forcebalance.abinitio.AbInitio.energy_force_transform ( self ) [inherited] Definition at line 484 of file abinitio.py.
```

Here is the call graph for this function:



```
def forcebalance.abinitio.AbInitio.energy_force_transform_one ( self, i ) [inherited] Definition at line 513 of file abinitio.py.
```

Here is the call graph for this function:

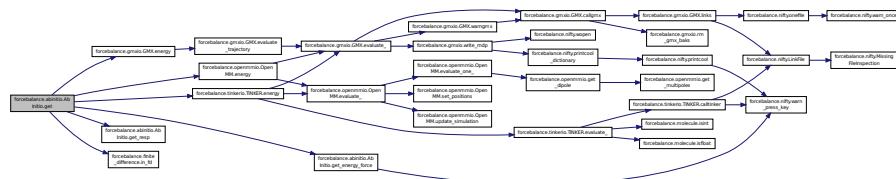


**def forcebalance.abinitio.AbInitio.energy\_one ( self, i ) [inherited]** Definition at line 501 of file abinitio.py.

**def forcebalance.abinitio.AbInitio.get ( self, mvals, AGrad = False, AHess = False ) [inherited]**

Definition at line 1152 of file abinitio.py.

Here is the call graph for this function:



**def forcebalance.abinitio.AbInitio.get\_energy\_force ( self, mvals, AGrad = False, AHess = False ) [inherited]** LPW 06-30-2013.

This subroutine builds the objective function (and optionally its derivatives) from a general simulation software. This is in contrast to using GROMACS-X2, which computes the objective function and prints it out; then 'get' only needs to call GROMACS and read it in.

This subroutine interfaces with simulation software 'drivers'. The driver is only expected to give the energy and forces.

Now this subroutine may sound trivial since the objective function is simply a least-squares quantity  $(M-Q)^2$  - but there are a number of nontrivial considerations. I will list them here.

0) Polytensor formulation: Because there may exist covariance between different components of the force (or covariance between the energy and the force), we build the objective function by taking outer products of vectors that have the form  $[E \ F_{1x} \ F_{1y} \ F_{1z} \ F_{2x} \ F_{2y} \dots]$ , and then we trace it with the inverse of the covariance matrix to get the objective function.

This version implements both the polytensor formulation and the standard formulation.

1) Boltzmann weights and normalization: Each snapshot has its own Boltzmann weight, which may or may not be normalized. This subroutine does the normalization automatically.

2) Subtracting out the mean energy gap: The zero-point energy difference between reference data and simulation is meaningless. This subroutine subtracts it out.

3) Hybrid ensembles: This program builds a combined objective function from both MM and QM ensembles, which is rigorously better than using a single ensemble.

Note that this subroutine does not do EVERYTHING that GROMACS-X2 can do, which includes:

1) Internal coordinate systems 2) 'Sampling correction' (deprecated, since it doesn't seem to work) 3) Analytic derivatives

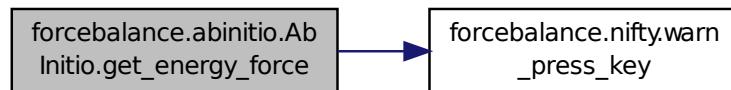
In the previous code (ForTune) this subroutine used analytic first derivatives of the energy and force to build the derivatives of the objective function. Here I will take a simplified approach, because building the derivatives are cumbersome. For now we will return the objective function ONLY. A two-point central difference should give us the first and diagonal second derivative anyhow.

**Todo** Parallelization over snapshots is not implemented yet

```
@param[in] mvals Mathematical parameter values
@param[in] AGrad Switch to turn on analytic gradient
@param[in] AHess Switch to turn on analytic Hessian
@return Answer Contribution to the objective function
```

Definition at line 588 of file abinitio.py.

Here is the call graph for this function:



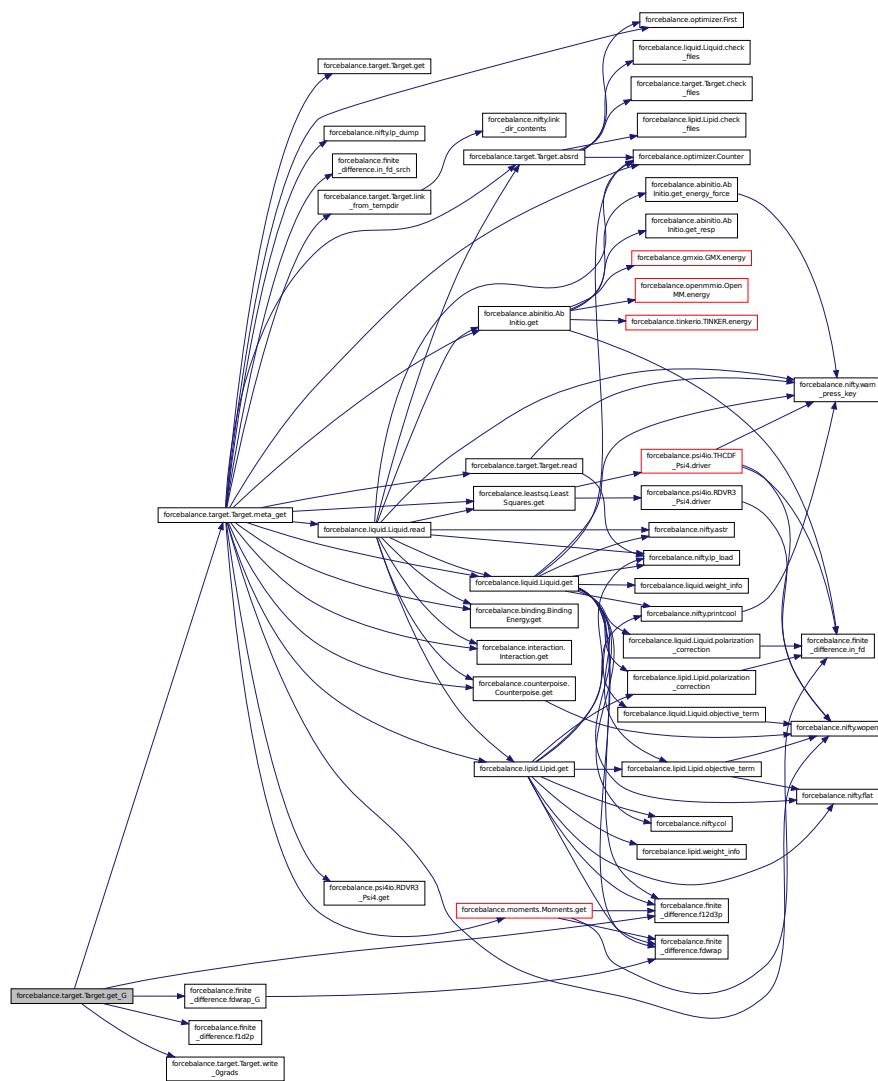
**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



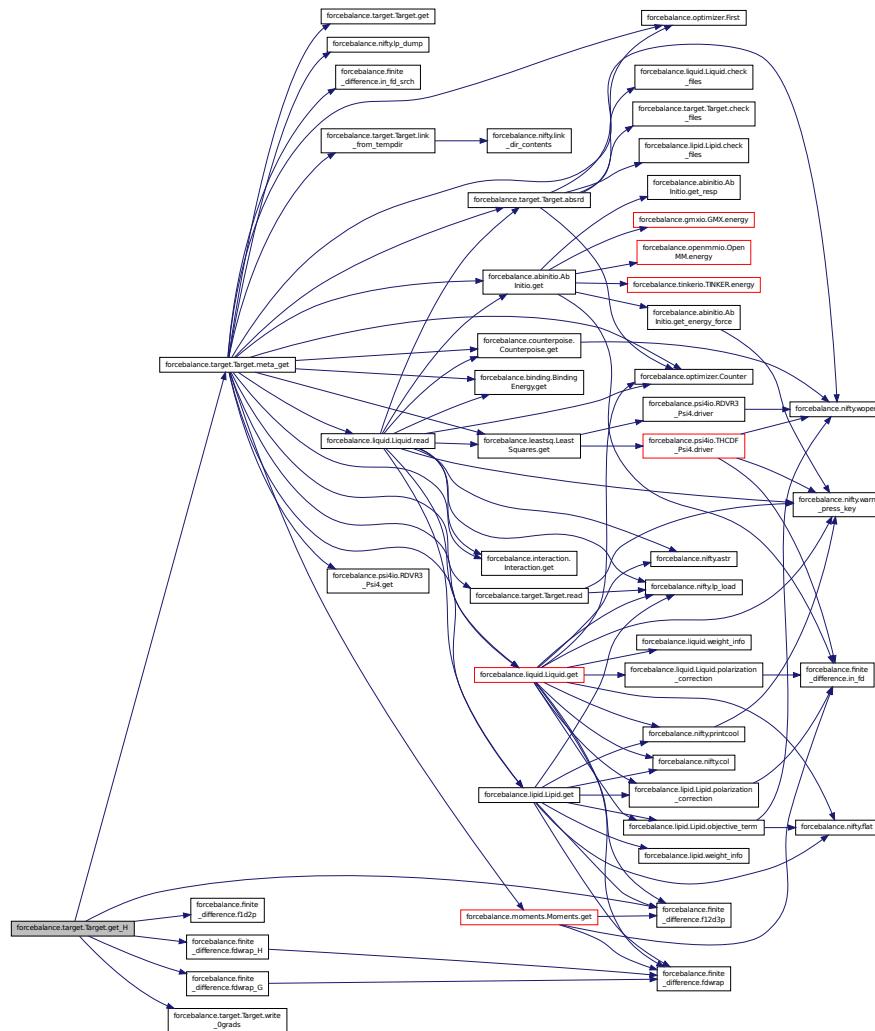
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.AbInitio.get_resp ( self, mvals, AGrad = False, AHess = False )
[inherited] Electrostatic potential fitting.
```

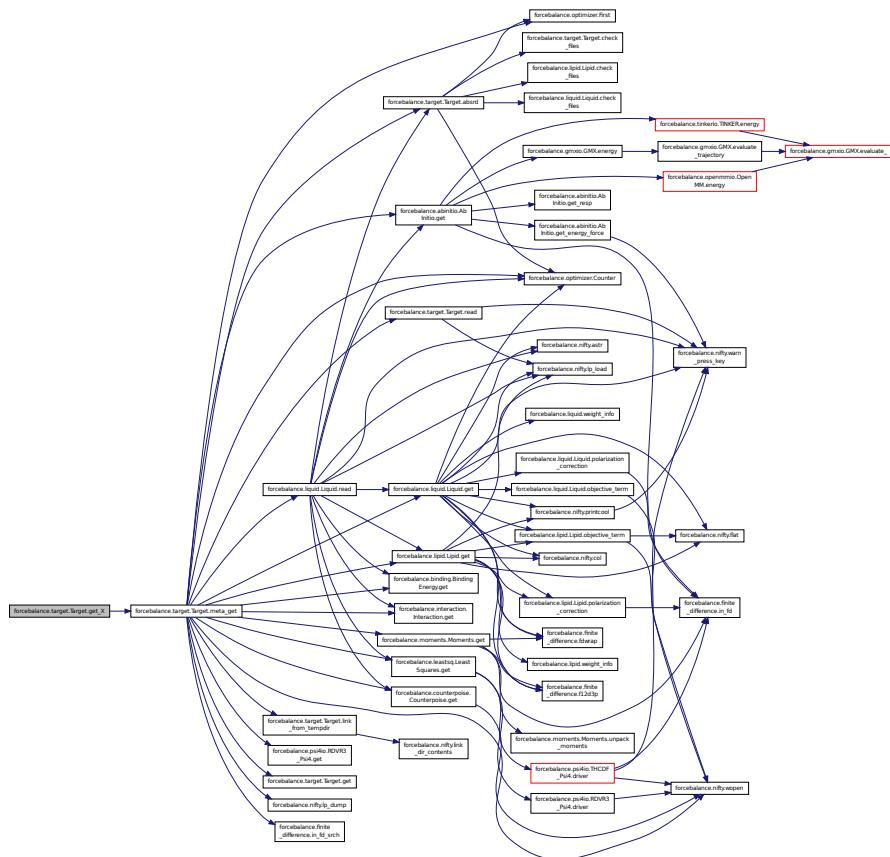
Implements the RESP objective function. (In Python so obviously not optimized.) This function takes the mathematical parameter values and returns the charges on the ATOMS (fancy mapping going on)

Definition at line 1053 of file abinitio.py.

```
def forcebalance.target.Target.get_X ( self, mvals = None ) [inherited] Computes the objective function contribution without any parametric derivatives.
```

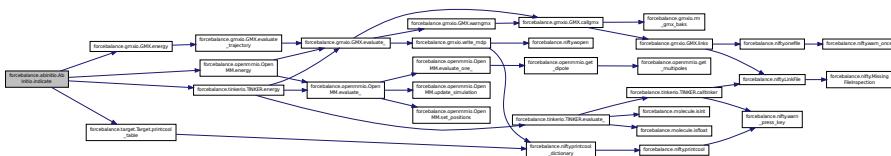
Definition at line 184 of file target.py.

Here is the call graph for this function:



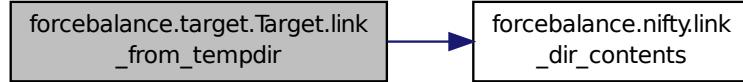
**def forcebalance.abinitio.ABInitio.indicate ( self ) [inherited]** Definition at line 448 of file abinitio.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.link_from_tempdir( self, absdestdir ) [inherited] Definition at line 315 of file target.py.
```

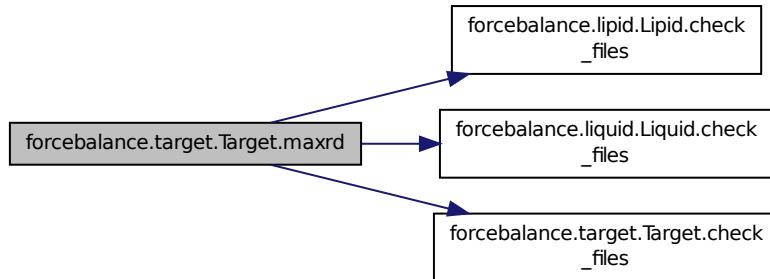
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

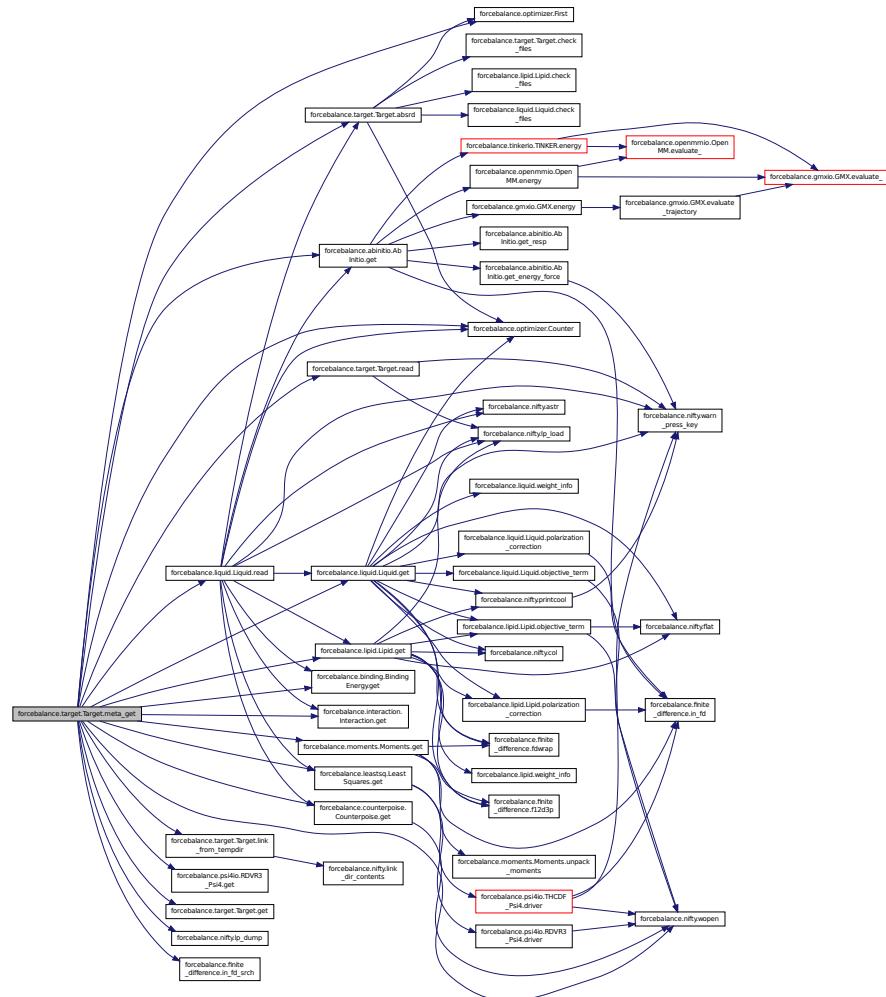


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

Here is the call graph for this function:

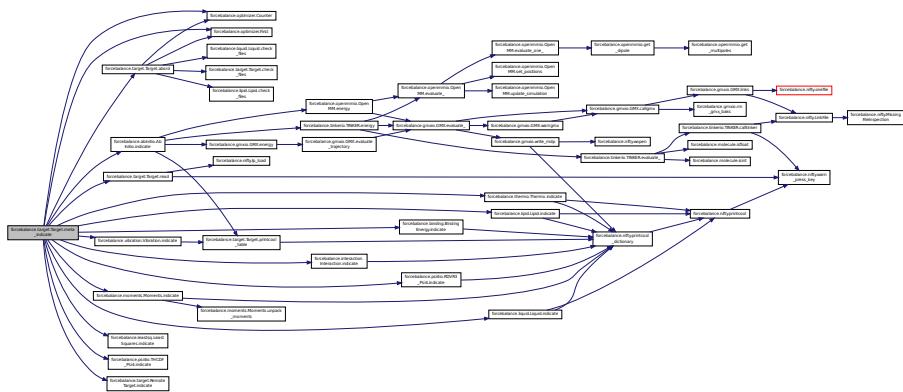


**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.

Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

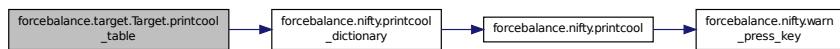
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

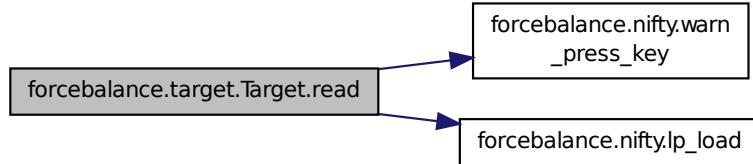
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.abinitio.AbInitio.read\_reference\_data ( self ) [inherited]** Read the reference ab initio data from a file such as qdata.txt.

**Todo** Add an option for picking any slice out of qdata.txt, helpful for cross-validation

**Todo** Closer integration of reference data with program - leave behind the qdata.txt format? (For now, I like the readability of qdata.txt)

After reading in the information from qdata.txt, it is converted into the GROMACS energy units (kind of an arbitrary choice); forces (kind of a misnomer in qdata.txt) are multiplied by -1 to convert gradients to forces.

We also subtract out the mean energies of all energy arrays because energy/force matching does not account for zero-point energy differences between MM and QM (i.e. energy of electrons in core orbitals).

The configurations in force/energy matching typically come from a the thermodynamic ensemble of the MM force field at some temperature (by running MD, for example), and for many reasons it is helpful to introduce non-Boltzmann weights in front of these configurations. There are two options: WHAM Boltzmann weights (for combining the weights of several simulations together) and QM Boltzmann weights (for converting MM weights into QM weights). Note that the two sets of weights 'stack'; i.e. they can be used at the same time.

A 'hybrid' ensemble is possible where we use 50% MM and 50% QM weights. Please read more in LPW and Troy Van Voorhis, JCP Vol. 133, Pg. 231101 (2010), doi:10.1063/1.3519043.

**Todo** The WHAM Boltzmann weights are generated by external scripts (wanalyze.py and make-wham-data.sh) and passed in; perhaps these scripts can be added to the ForceBalance distribution or integrated more tightly.

Finally, note that using non-Boltzmann weights degrades the statistical information content of the snapshots. This problem will generally become worse if the ensemble to which we're reweighting is dramatically different from the one we're sampling from. We end up with a set of Boltzmann weights like [1e-9, 1e-9, 1.0, 1e-9, 1e-9 ... ] and this is essentially just one snapshot. I believe Troy is working on something to cure this problem.

Here, we have a measure for the information content of our snapshots, which comes easily from the definition of information entropy:

```
S = -1*Sum_i(P_i*log(P_i))
InfoContent = exp(-S)
```

With uniform weights, InfoContent is equal to the number of snapshots; with horrible weights, InfoContent is closer to one.

Definition at line 329 of file abinitio.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

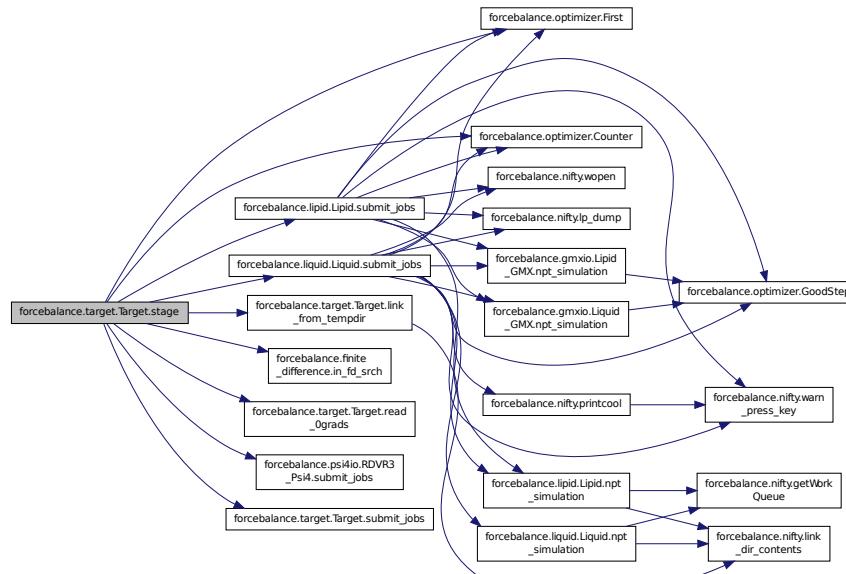
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

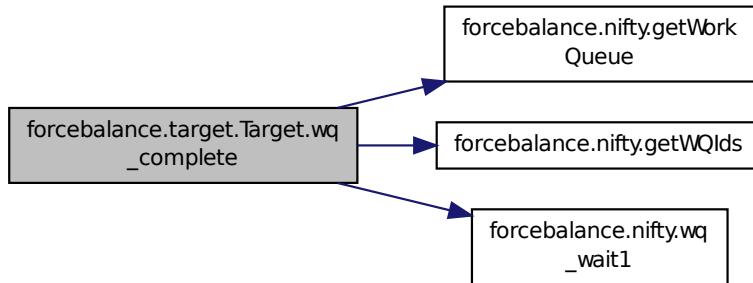


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work  
Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.write_Ograds ( self, Ans ) [inherited] Write a file to the target directory  
containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

### 8.3.4 Member Data Documentation

```
forcebalance.abinitio.ABInitio.AtomLists [inherited] Lists of atoms to do net force/torque fitting and excluding virtual sites.
```

Definition at line 160 of file abinitio.py.

```
forcebalance.abinitio.ABInitio.AtomMask [inherited] Definition at line 161 of file abinitio.py.
```

```
forcebalance.abinitio.ABInitio.boltz_wts [inherited] Initialize the base class.
```

Number of snapshots Whether to use WHAM Boltzmann weights Whether to use the Sampling Correction Whether to match Absolute Energies (make sure you know what you're doing) Whether to use the Covariance Matrix Whether to use QM Boltzmann weights The temperature for QM Boltzmann weights Number of atoms that we are fitting Whether to fit Energies. Whether to fit Forces. Whether to fit Electrostatic Potential. Weights for the three components. Option for how much data to write to disk. Whether to do energy and force calculations for the whole trajectory, or to do one calculation per snapshot. OpenMM-only option - whether to run the energies and forces internally. Whether we have virtual sites (set at the global option level) Attenuate the weights as a function of energy What is the energy denominator? (Valid for 'attenuate') Set upper cutoff energy WHAM Boltzmann weights

Definition at line 116 of file abinitio.py.

```
forcebalance.abinitio.ABInitio.e_err [inherited] Qualitative Indicator: average energy error (in kJ/mol)
```

Definition at line 134 of file abinitio.py.

```
forcebalance.abinitio.ABInitio.e_err_pct [inherited] Definition at line 135 of file abinitio.py.
```

**forcebalance.abinitio.AbInitio.e.ref** [inherited] Definition at line 1030 of file abinitio.py.

**forcebalance.abinitio.AbInitio.emd0** [inherited] Energies of the sampling simulation.  
Definition at line 122 of file abinitio.py.

**forcebalance.abinitio.AbInitio.engine** [inherited] Build keyword dictionaries to pass to engine.  
Create engine object.  
Definition at line 158 of file abinitio.py.

**forcebalance.gmxio.AbInitio\_GMX.engine** Default file names for coordinates, top and mdp files.  
Definition at line 1455 of file gmxio.py.

**forcebalance.abinitio.AbInitio.eqm** [inherited] Reference (QM) energies.  
Definition at line 120 of file abinitio.py.

**forcebalance.abinitio.AbInitio.esp\_err** [inherited] Qualitative Indicator: "relative RMS" for electrostatic potential.  
Definition at line 140 of file abinitio.py.

**forcebalance.abinitio.AbInitio.espval** [inherited] ESP values.  
Definition at line 128 of file abinitio.py.

**forcebalance.abinitio.AbInitio.espxyz** [inherited] ESP grid points.  
Definition at line 126 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err** [inherited] Qualitative Indicator: average force error (fractional)  
Definition at line 137 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err\_pct** [inherited] Definition at line 138 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_ref** [inherited] Definition at line 1034 of file abinitio.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)  
Definition at line 160 of file target.py.

**forcebalance.abinitio.AbInitio.fitatoms** [inherited] Definition at line 366 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force** [inherited] Definition at line 361 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force\_map** [inherited] Definition at line 209 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fqm** [inherited] Reference (QM) forces.  
Definition at line 124 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fref** [inherited] Definition at line 442 of file abinitio.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.  
Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.abinitio.ABInitio.invdist [inherited]** Definition at line 1061 of file abinitio.py.

**forcebalance.abinitio.ABInitio.mol [inherited]** Definition at line 148 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nesp [inherited]** Definition at line 363 of file abinitio.py.

**forcebalance.abinitio.ABInitio.new\_vsites [inherited]** Read in the reference data.

The below two options are related to whether we want to rebuild virtual site positions. Rebuild the distance matrix if virtual site positions have changed

Definition at line 166 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_err [inherited]** Definition at line 141 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_err\_pct [inherited]** Definition at line 142 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_ref [inherited]** Definition at line 1038 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nftqm [inherited]** Definition at line 438 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nnf [inherited]** Definition at line 267 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nparticles [inherited]** The number of (atoms + drude particles + virtual sites)  
Definition at line 153 of file abinitio.py.

**forcebalance.abinitio.ABInitio.ns [inherited]** Read in the trajectory file.

Definition at line 147 of file abinitio.py.

**forcebalance.abinitio.ABInitio.ntq [inherited]** Definition at line 268 of file abinitio.py.

**forcebalance.abinitio.ABInitio.objective [inherited]** Definition at line 1172 of file abinitio.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.qfnm [inherited]** The qdata.txt file that contains the QM energies and forces.  
Definition at line 130 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmatoms [inherited]** The number of atoms in the QM calculation (Irrelevant if not fitting forces)

Definition at line 132 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmboltz\_wts [inherited]** QM Boltzmann weights.

Definition at line 118 of file abinitio.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.abinitio.ABInitio.respterm [inherited]** Definition at line 1140 of file abinitio.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.abinitio.ABInitio.save\_vmvales [inherited]** Save the mvales from the last time we updated the vsites.

Definition at line 168 of file abinitio.py.

**forcebalance.target.Target.tempbase [inherited]** Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir [inherited]** Definition at line 155 of file target.py.

**forcebalance.abinitio.ABInitio.tq\_err [inherited]** Definition at line 1042 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_err\_pct [inherited]** Definition at line 143 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_ref [inherited]** Definition at line 1041 of file abinitio.py.

**forcebalance.abinitio.ABInitio.use\_nft [inherited]** Whether to compute net forces and torques, or not.

Definition at line 145 of file abinitio.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.w\_energy [inherited]** Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_force** [inherited] Definition at line 362 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_netforce** [inherited] Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_resp** [inherited] Definition at line 1054 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_torque** [inherited] Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz** [inherited] Definition at line 382 of file abinitio.py.

**forcebalance.target.Target.write\_indicate** [inherited] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

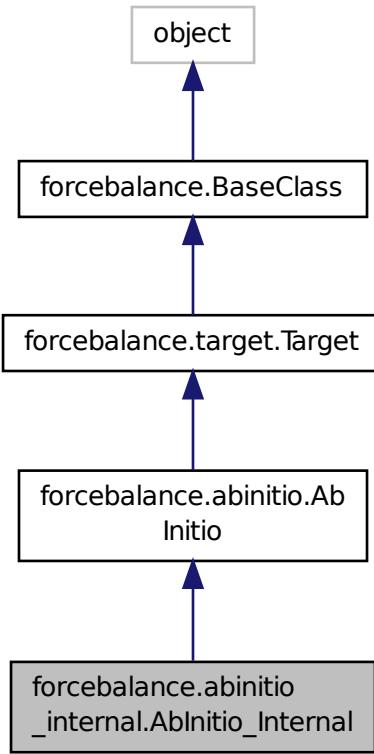
The documentation for this class was generated from the following file:

- [gmxio.py](#)

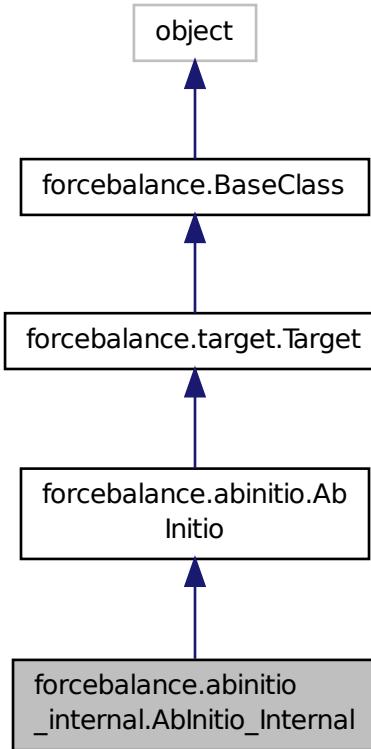
## 8.4 forcebalance.abinitio\_internal.ABInitio\_Internal Class Reference

Subclass of Target for force and energy matching using an internal implementation.

Inheritance diagram for forcebalance.abinitio\_internal.AbInitio\_Internal:



Collaboration diagram for forcebalance.abinitio\_internal.AbInitio\_Internal:



### Public Member Functions

- def `_init_`
- def `energy_force_driver_all`

*Here we actually compute the interactions and return the energies and forces.*

- def `build_invdist`
- def `compute_netforce_torque`
- def `read_reference_data`

*Read the reference ab initio data from a file such as qdata.txt.*

- def `indicate`
- def `energy_all`
- def `energy_force_all`
- def `energy_force_transform`
- def `energy_one`
- def `energy_force_one`
- def `energy_force_transform_one`
- def `get_energy_force`

*LPW 06-30-2013.*

- def `get_resp`  
*Electrostatic potential fitting.*
- def `get`
- def `get_X`  
*Computes the objective function contribution without any parametric derivatives.*
- def `read_0grads`  
*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_0grads`  
*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`  
*Computes the objective function contribution and its gradient.*
- def `get_H`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `coords`  
*Name of the trajectory, we need this BEFORE initializing the SuperClass.*
- `boltz_wts`  
*Initialize the base class.*
- `qmboltz_wts`  
*QM Boltzmann weights.*
- `eqm`  
*Reference (QM) energies.*

- **emd0**  
*Energies of the sampling simulation.*
- **fqm**  
*Reference (QM) forces.*
- **espxyz**  
*ESP grid points.*
- **espval**  
*ESP values.*
- **qfnm**  
*The qdata.txt file that contains the QM energies and forces.*
- **qatoms**  
*The number of atoms in the QM calculation (Irrelevant if not fitting forces)*
- **e\_err**  
*Qualitative Indicator: average energy error (in kJ/mol)*
- **e\_err\_pct**
- **f\_err**  
*Qualitative Indicator: average force error (fractional)*
- **f\_err\_pct**
- **esp\_err**  
*Qualitative Indicator: "relative RMS" for electrostatic potential.*
- **nf\_err**
- **nf\_err\_pct**
- **tq\_err\_pct**
- **use\_nft**  
*Whether to compute net forces and torques, or not.*
- **ns**  
*Read in the trajectory file.*
- **mol**
- **nparticles**  
*The number of (atoms + drude particles + virtual sites)*
- **engine**  
*Build keyword dictionaries to pass to engine.*
- **AtomLists**  
*Lists of atoms to do net force/torque fitting and excluding virtual sites.*
- **AtomMask**
- **new\_vsites**  
*Read in the reference data.*
- **save\_vmvalls**  
*Save the mvalls from the last time we updated the vsites.*
- **force\_map**
- **nnf**
- **ntq**
- **force**
- **w\_force**
- **nesp**
- **fitatoms**
- **whamboltz**
- **nftqm**

- `fref`
- `w_energy`
- `w_netforce`
- `w_torque`
- `e_ref`
- `f_ref`
- `nf_ref`
- `tq_ref`
- `tq_err`
- `w_resp`
- `invdists`
- `respterm`
- `objective`
- `rd`

*Root directory of the whole project.*
- `pgrad`

*Iteration where we turn on zero-gradient skipping.*
- `tempbase`

*Relative directory of target.*
- `tempdir`
- `rundir`

*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- `FF`

*Need the forcefield (here for now)*
- `xct`

*Counts how often the objective function was computed.*
- `gct`

*Counts how often the gradient was computed.*
- `hct`

*Counts how often the Hessian was computed.*
- `read_indicate`

*Whether to read indicate.log from file when restarting an aborted run.*
- `write_indicate`

*Whether to write indicate.log at every iteration (true for all but remote.)*
- `read_objective`

*Whether to read objective.p from file when restarting an aborted run.*
- `write_objective`

*Whether to write objective.p at every iteration (true for all but remote.)*
- `verbose_options`
- `PrintOptionDict`

#### 8.4.1 Detailed Description

Subclass of Target for force and energy matching using an internal implementation.

Implements the prepare and energy\_force\_driver methods. The get method is in the superclass.

The purpose of this class is to provide an extremely simple test case that does not require the user to install any external software. It only runs with one of the included sample test calculations (internal\_tip3p), and the objective function is energy matching.

## Warning

This class is only intended to work with a very specific test case (internal.tip3p). This is because the topology and ordering of the atoms is hard-coded (12 water molecules with 3 atoms each).

This class does energy matching only (no forces)

Definition at line 37 of file abinitio\_internal.py.

### 8.4.2 Constructor & Destructor Documentation

**def forcebalance.abinitio\_internal.AbInitio\_Internal.\_\_init\_\_ ( self, options, tgt\_opts, forcefield )** Definition at line 40 of file abinitio\_internal.py.

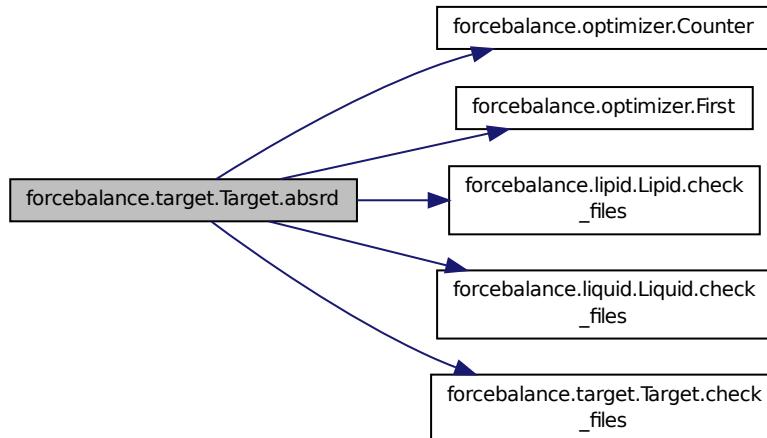
### 8.4.3 Member Function Documentation

**def forcebalance.BaseClass.\_\_setattr\_\_ ( self, key, value ) [inherited]** Definition at line 28 of file \_\_init\_\_.py.

**def forcebalance.target.Target.absrd ( self, inum = None ) [inherited]** Supply the correct directory specified by user's "read" option.

Definition at line 393 of file target.py.

Here is the call graph for this function:



**def forcebalance.abinitio.AbInitio.build\_invdist ( self, mvals ) [inherited]** Definition at line 171 of file abinitio.py.

**def forcebalance.target.Target.check\_files ( self, there ) [inherited]** Check this directory for the presence of readable files when the 'read' option is set.

Definition at line 364 of file target.py.

**def forcebalance.abinitio.AbInitio.compute\_netforce\_torque ( self, xyz, force, QM = False ) [inherited]** Definition at line 200 of file abinitio.py.

```
def forcebalance.abinitio.AbInitio.energy_all( self ) [inherited] Definition at line 472 of file abinitio.py.
```

```
def forcebalance.abinitio.AbInitio.energy_force_all( self ) [inherited] Definition at line 478 of file abinitio.py.
```

**def forcebalance.abinitio\_internal.AbInitio\_Internal.energy\_force\_driver\_all ( self )** Here we actually compute the interactions and return the energies and forces.

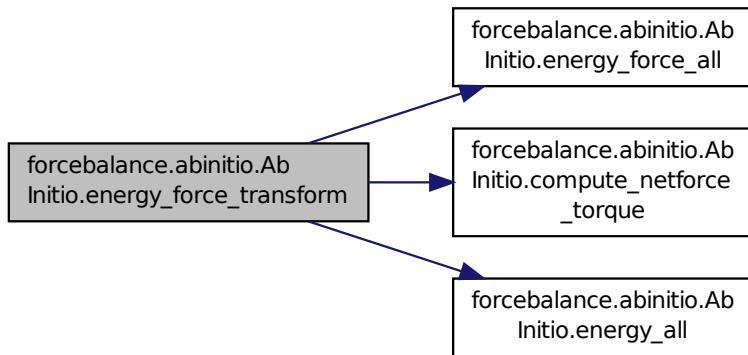
I verified this to give the same answer as GROMACS.

Definition at line 50 of file abinitio\_internal.py.

```
def forcebalance.abinitio.AbInitio.energy_force_one ( self, i ) [inherited] Definition at line 507 of file abinitio.py.
```

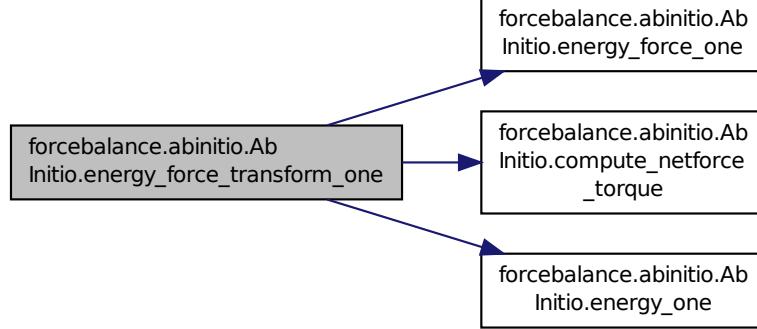
```
def forcebalance.abinitio.AbInitio.energy_force_transform ( self ) [inherited] Definition at line 484 of file abinitio.py.
```

Here is the call graph for this function:



```
def forcebalance.abinitio.AbInitio.energy_force_transform_one ( self, i ) [inherited] Definition at line 513 of file abinitio.py.
```

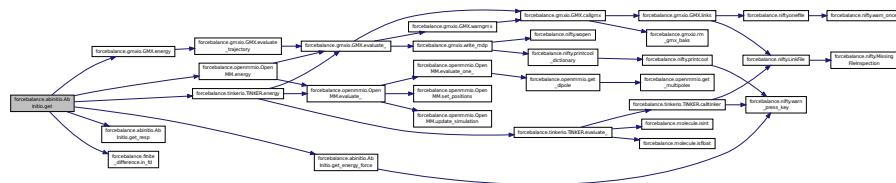
Here is the call graph for this function:



**def forcebalance.abinitio.AblInitio.energy\_one( self, i ) [inherited]** Definition at line 501 of file abinitio.py.

```
def forcebalance.abinitio.ABInitio.get( self, mvals, AGrad = False, AHess = False ) [inherited]
Definition at line 1152 of file abinitio.py.
```

Here is the call graph for this function:



```
def forcebalance.abinitio.ABInitio.get_energy_force ( self, mvals, AGrad = False, AHess = False )  
[inherited] LPW 06-30-2013.
```

This subroutine builds the objective function (and optionally its derivatives) from a general simulation software. This is in contrast to using GROMACS-X2, which computes the objective function and prints it out; then 'get' only needs to call GROMACS and read it in.

This subroutine interfaces with simulation software 'drivers'. The driver is only expected to give the energy and forces.

Now this subroutine may sound trivial since the objective function is simply a least-squares quantity  $(M-Q)^2$  - but there are a number of nontrivial considerations. I will list them here.

0) Polytensor formulation: Because there may exist covariance between different components of the force (or covariance between the energy and the force), we build the objective function by taking outer products of vectors that have the form [E F\_1x F\_1y F\_1z F\_2x F\_2y ... ], and then we trace it with the inverse of the covariance matrix to get the objective function.

This version implements both the polytensor formulation and the standard formulation.

1) Boltzmann weights and normalization: Each snapshot has its own Boltzmann weight, which may or may not be normalized. This subroutine does the normalization automatically.

2) Subtracting out the mean energy gap: The zero-point energy difference between reference data and simulation is meaningless. This subroutine subtracts it out.

3) Hybrid ensembles: This program builds a combined objective function from both MM and QM ensembles, which is rigorously better than using a single ensemble.

Note that this subroutine does not do EVERYTHING that GROMACS-X2 can do, which includes:

1) Internal coordinate systems 2) 'Sampling correction' (deprecated, since it doesn't seem to work) 3) Analytic derivatives

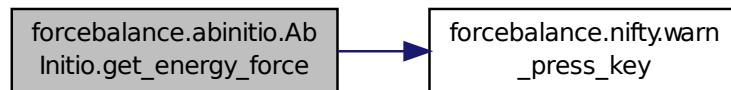
In the previous code (ForTune) this subroutine used analytic first derivatives of the energy and force to build the derivatives of the objective function. Here I will take a simplified approach, because building the derivatives are cumbersome. For now we will return the objective function ONLY. A two-point central difference should give us the first and diagonal second derivative anyhow.

**Todo** Parallelization over snapshots is not implemented yet

```
@param[in] mvals Mathematical parameter values
@param[in] AGrad Switch to turn on analytic gradient
@param[in] AHess Switch to turn on analytic Hessian
@return Answer Contribution to the objective function
```

Definition at line 588 of file abinitio.py.

Here is the call graph for this function:



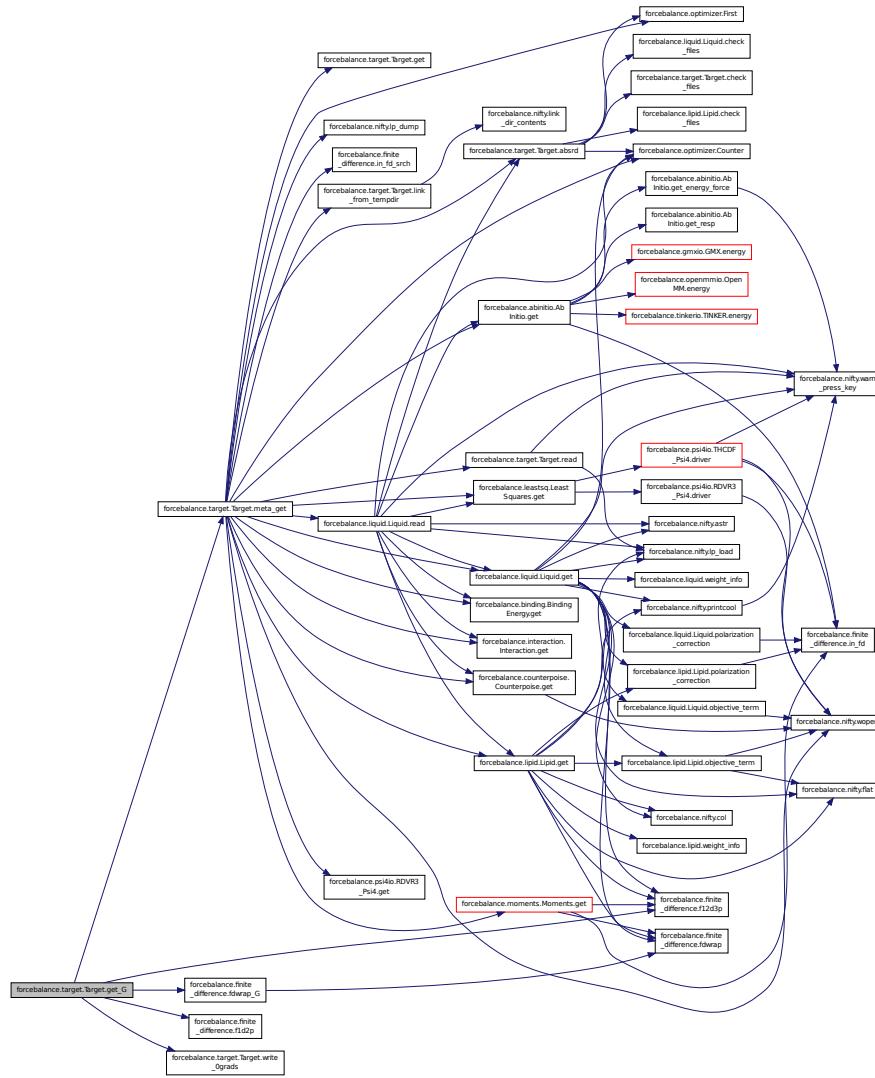
**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



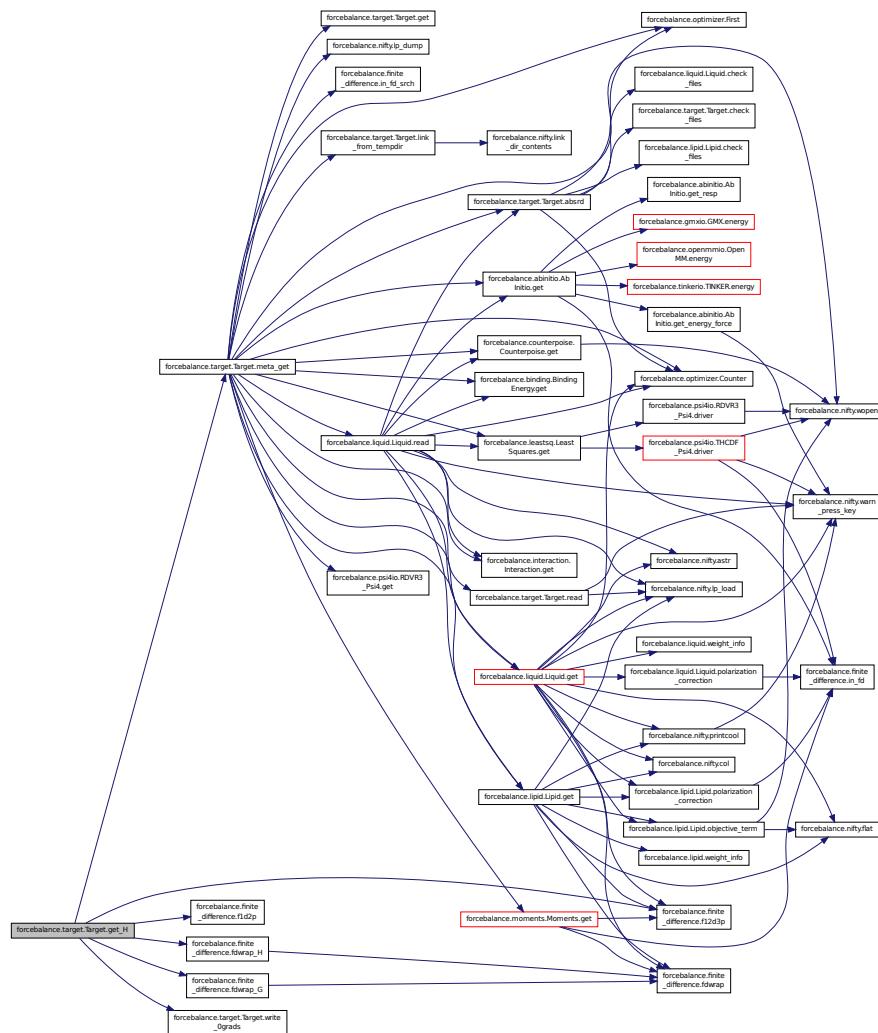
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.AbInitio.get_resp ( self, mvals, AGrad = False, AHess = False )  
[inherited] Electrostatic potential fitting.
```

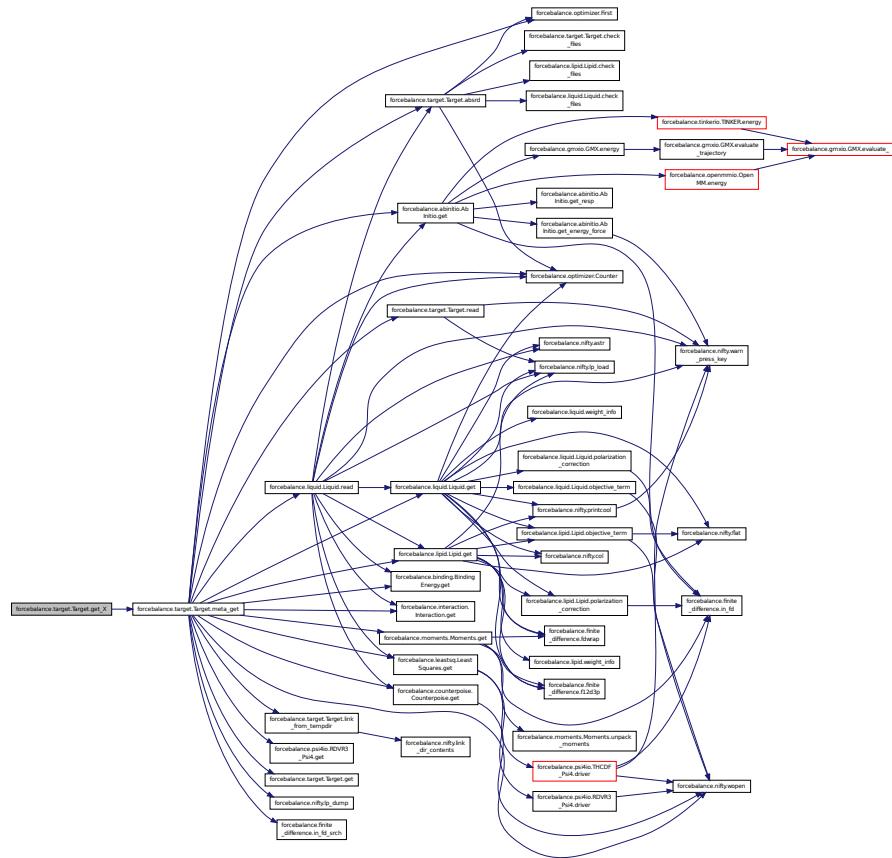
Implements the RESP objective function. (In Python so obviously not optimized.) This function takes the mathematical parameter values and returns the charges on the ATOMS (fancy mapping going on)

Definition at line 1053 of file abinitio.py.

**def forcebalance.target.Target.get\_X( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

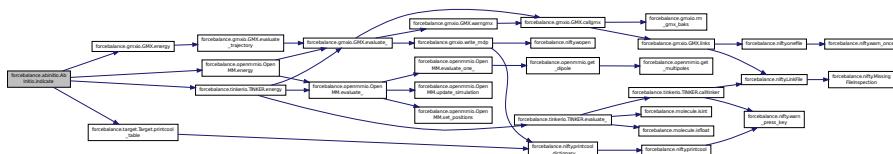
Definition at line 184 of file target.py.

Here is the call graph for this function:



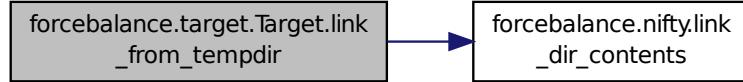
**def forcebalance.abinitio.AblInitio.indicate ( self ) [inherited]** Definition at line 448 of file abinitio.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

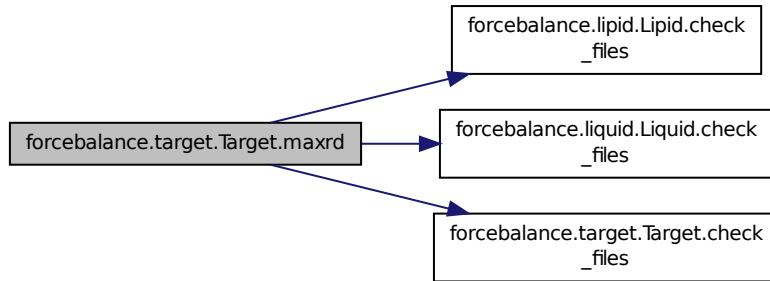
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

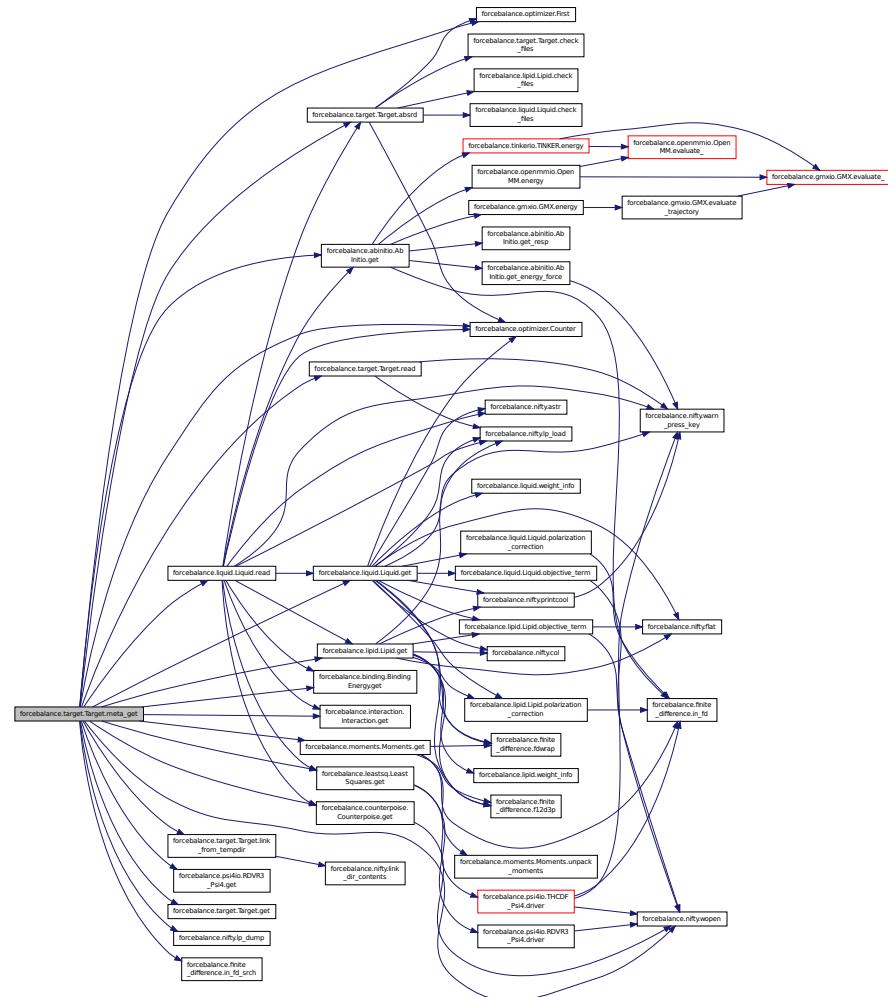


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

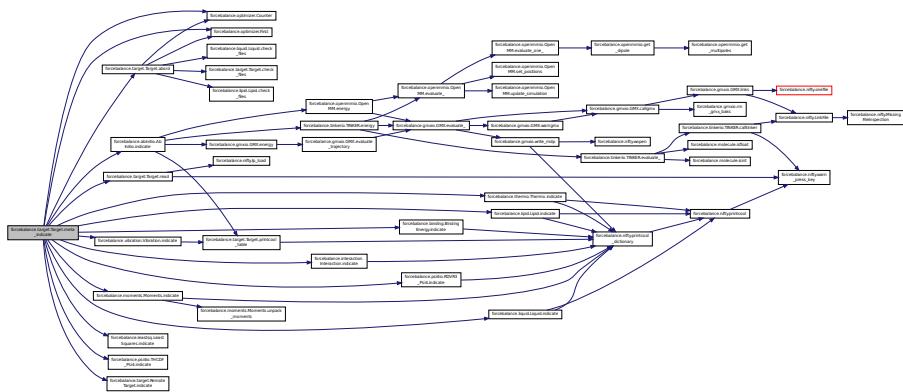
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

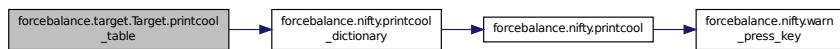
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

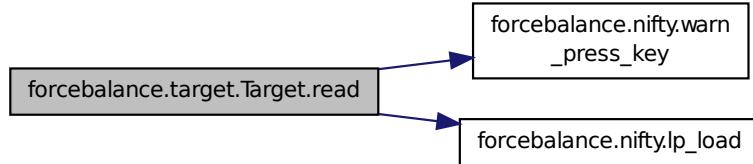
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.abinitio.AblInitio.read\_reference\_data ( self ) [inherited]** Read the reference ab initio data from a file such as qdata.txt.

**Todo** Add an option for picking any slice out of qdata.txt, helpful for cross-validation

**Todo** Closer integration of reference data with program - leave behind the qdata.txt format? (For now, I like the readability of qdata.txt)

After reading in the information from qdata.txt, it is converted into the GROMACS energy units (kind of an arbitrary choice); forces (kind of a misnomer in qdata.txt) are multiplied by -1 to convert gradients to forces.

We also subtract out the mean energies of all energy arrays because energy/force matching does not account for zero-point energy differences between MM and QM (i.e. energy of electrons in core orbitals).

The configurations in force/energy matching typically come from a the thermodynamic ensemble of the MM force field at some temperature (by running MD, for example), and for many reasons it is helpful to introduce non-Boltzmann weights in front of these configurations. There are two options: WHAM Boltzmann weights (for combining the weights of several simulations together) and QM Boltzmann weights (for converting MM weights into QM weights). Note that the two sets of weights 'stack'; i.e. they can be used at the same time.

A 'hybrid' ensemble is possible where we use 50% MM and 50% QM weights. Please read more in LPW and Troy Van Voorhis, JCP Vol. 133, Pg. 231101 (2010), doi:10.1063/1.3519043.

**Todo** The WHAM Boltzmann weights are generated by external scripts (wanalyze.py and make-wham-data.sh) and passed in; perhaps these scripts can be added to the ForceBalance distribution or integrated more tightly.

Finally, note that using non-Boltzmann weights degrades the statistical information content of the snapshots. This problem will generally become worse if the ensemble to which we're reweighting is dramatically different from the one we're sampling from. We end up with a set of Boltzmann weights like [1e-9, 1e-9, 1.0, 1e-9, 1e-9 ... ] and this is essentially just one snapshot. I believe Troy is working on something to cure this problem.

Here, we have a measure for the information content of our snapshots, which comes easily from the definition of information entropy:

```
S = -1*Sum_i(P_i*log(P_i))
InfoContent = exp(-S)
```

With uniform weights, InfoContent is equal to the number of snapshots; with horrible weights, InfoContent is closer to one.

Definition at line 329 of file abinitio.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

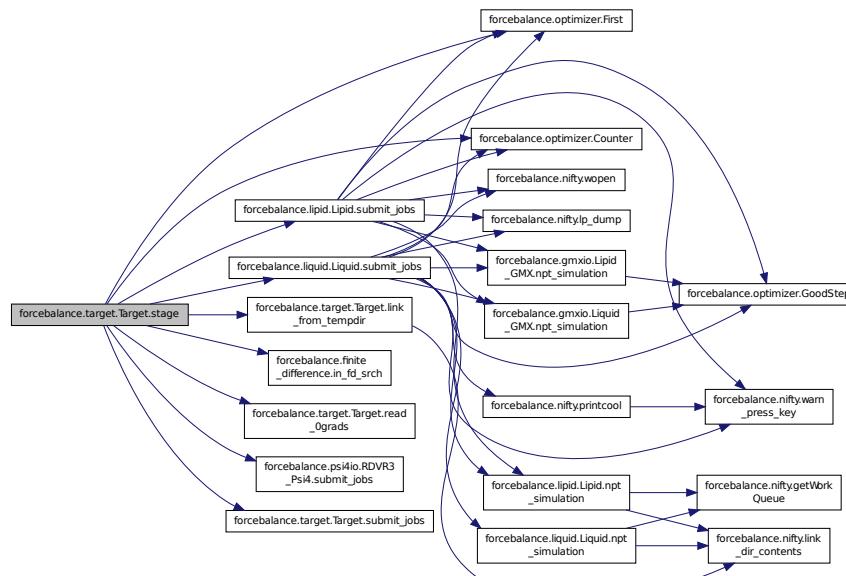
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

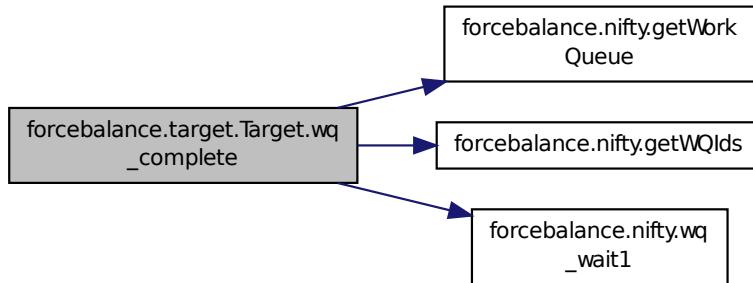


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work  
Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.write_Ograds ( self, Ans ) [inherited] Write a file to the target directory  
containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

#### 8.4.4 Member Data Documentation

```
forcebalance.abinitio.AblInitio.AtomLists [inherited] Lists of atoms to do net force/torque fitting and excluding virtual sites.
```

Definition at line 160 of file abinitio.py.

```
forcebalance.abinitio.AblInitio.AtomMask [inherited] Definition at line 161 of file abinitio.py.
```

```
forcebalance.abinitio.AblInitio.boltz_wts [inherited] Initialize the base class.
```

Number of snapshots Whether to use WHAM Boltzmann weights Whether to use the Sampling Correction Whether to match Absolute Energies (make sure you know what you're doing) Whether to use the Covariance Matrix Whether to use QM Boltzmann weights The temperature for QM Boltzmann weights Number of atoms that we are fitting Whether to fit Energies. Whether to fit Forces. Whether to fit Electrostatic Potential. Weights for the three components. Option for how much data to write to disk. Whether to do energy and force calculations for the whole trajectory, or to do one calculation per snapshot. OpenMM-only option - whether to run the energies and forces internally. Whether we have virtual sites (set at the global option level) Attenuate the weights as a function of energy What is the energy denominator? (Valid for 'attenuate') Set upper cutoff energy WHAM Boltzmann weights

Definition at line 116 of file abinitio.py.

```
forcebalance.abinitio_internal.AblInitio_Internal.coords Name of the trajectory, we need this BEFORE initializing  
the SuperClass.
```

Definition at line 42 of file abinitio\_internal.py.

**forcebalance.abinitio.AbInitio.e\_err [inherited]** Qualitative Indicator: average energy error (in kJ/mol)  
Definition at line 134 of file abinitio.py.

**forcebalance.abinitio.AbInitio.e\_err\_pct [inherited]** Definition at line 135 of file abinitio.py.

**forcebalance.abinitio.AbInitio.e\_ref [inherited]** Definition at line 1030 of file abinitio.py.

**forcebalance.abinitio.AbInitio.emd0 [inherited]** Energies of the sampling simulation.  
Definition at line 122 of file abinitio.py.

**forcebalance.abinitio.AbInitio.engine [inherited]** Build keyword dictionaries to pass to engine.  
Create engine object.  
Definition at line 158 of file abinitio.py.

**forcebalance.abinitio.AbInitio.eqm [inherited]** Reference (QM) energies.  
Definition at line 120 of file abinitio.py.

**forcebalance.abinitio.AbInitio.esp\_err [inherited]** Qualitative Indicator: "relative RMS" for electrostatic potential.  
Definition at line 140 of file abinitio.py.

**forcebalance.abinitio.AbInitio.espval [inherited]** ESP values.  
Definition at line 128 of file abinitio.py.

**forcebalance.abinitio.AbInitio.espxyz [inherited]** ESP grid points.  
Definition at line 126 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err [inherited]** Qualitative Indicator: average force error (fractional)  
Definition at line 137 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err\_pct [inherited]** Definition at line 138 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_ref [inherited]** Definition at line 1034 of file abinitio.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)  
Definition at line 160 of file target.py.

**forcebalance.abinitio.AbInitio.fitatoms [inherited]** Definition at line 366 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force [inherited]** Definition at line 361 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force\_map [inherited]** Definition at line 209 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fqm [inherited]** Reference (QM) forces.  
Definition at line 124 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fref [inherited]** Definition at line 442 of file abinitio.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.  
Definition at line 164 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.abinitio.ABInitio.invdists** [inherited] Definition at line 1061 of file abinitio.py.

**forcebalance.abinitio.ABInitio.mol** [inherited] Definition at line 148 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nesp** [inherited] Definition at line 363 of file abinitio.py.

**forcebalance.abinitio.ABInitio.new\_vsites** [inherited] Read in the reference data.

The below two options are related to whether we want to rebuild virtual site positions. Rebuild the distance matrix if virtual site positions have changed

Definition at line 166 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_err** [inherited] Definition at line 141 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_err\_pct** [inherited] Definition at line 142 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_ref** [inherited] Definition at line 1038 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nftqm** [inherited] Definition at line 438 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nnf** [inherited] Definition at line 267 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nparticles** [inherited] The number of (atoms + drude particles + virtual sites)  
Definition at line 153 of file abinitio.py.

**forcebalance.abinitio.ABInitio.ns** [inherited] Read in the trajectory file.  
Definition at line 147 of file abinitio.py.

**forcebalance.abinitio.ABInitio.ntq** [inherited] Definition at line 268 of file abinitio.py.

**forcebalance.abinitio.ABInitio.objective** [inherited] Definition at line 1172 of file abinitio.py.

**forcebalance.target.Target.pgrad** [inherited] Iteration where we turn on zero-gradient skipping.  
Dictionary of whether to call the derivatives.  
Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.qfnm** [inherited] The qdata.txt file that contains the QM energies and forces.  
Definition at line 130 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmatoms** [inherited] The number of atoms in the QM calculation (Irrelevant if not fitting forces)  
Definition at line 132 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmboltz\_wts** [inherited] QM Boltzmann weights.

Definition at line 118 of file abinitio.py.

**forcebalance.target.Target.rd** [inherited] Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate** [inherited] Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [inherited] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.abinitio.ABInitio.respterm** [inherited] Definition at line 1140 of file abinitio.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number. The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.abinitio.ABInitio.save\_mvvals** [inherited] Save the mvvals from the last time we updated the vsites.

Definition at line 168 of file abinitio.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target\_name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.abinitio.ABInitio.tq\_err** [inherited] Definition at line 1042 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_err\_pct** [inherited] Definition at line 143 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_ref** [inherited] Definition at line 1041 of file abinitio.py.

**forcebalance.abinitio.ABInitio.use\_nft** [inherited] Whether to compute net forces and torques, or not.

Definition at line 145 of file abinitio.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.w.energy** [inherited] Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w.force** [inherited] Definition at line 362 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w.netforce** [inherited] Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w.resp** [inherited] Definition at line 1054 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w.torque** [inherited] Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz** [inherited] Definition at line 382 of file abinitio.py.

**forcebalance.target.Target.write\_indicate** [inherited] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

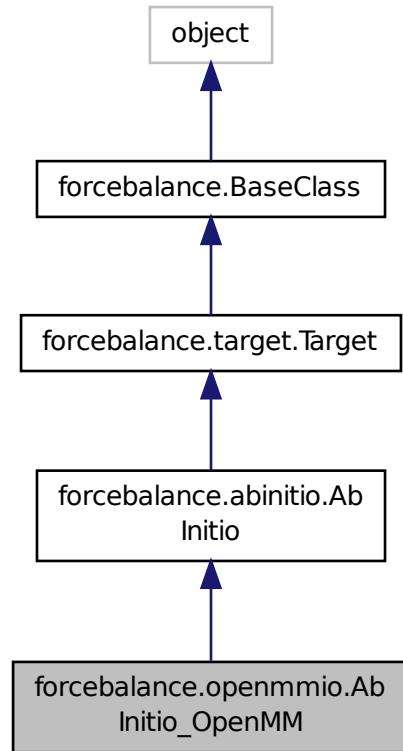
The documentation for this class was generated from the following file:

- [abinitio\\_internal.py](#)

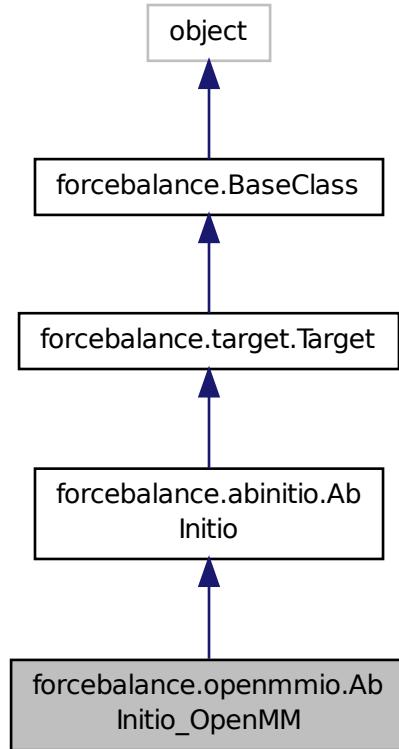
## 8.5 forcebalance.openmmio.ABInitio\_OpenMM Class Reference

Force and energy matching using [OpenMM](#).

Inheritance diagram for forcebalance.openmmio.AbInitio\_OpenMM:



Collaboration diagram for forcebalance.openmmio.AbInitio\_OpenMM:



#### Public Member Functions

- def `_init_`
- def `build_invdist`
- def `compute_netforce_torque`
- def `read_reference_data`  
*Read the reference ab initio data from a file such as qdata.txt.*
- def `indicate`
- def `energy_all`
- def `energy_force_all`
- def `energy_force_transform`
- def `energy_one`
- def `energy_force_one`
- def `energy_force_transform_one`
- def `get_energy_force`  
*LPW 06-30-2013.*
- def `get_resp`  
*Electrostatic potential fitting.*

- def `get`  
 Computes the objective function contribution without any parametric derivatives.
- def `get_X`  
 Read a file from the target directory containing names of parameters that don't contribute to the gradient.
- def `read_0grads`  
 Write a file to the target directory containing names of parameters that don't contribute to the gradient.
- def `write_0grads`  
 Write a file to the target directory containing names of parameters that don't contribute to the gradient.
- def `get_G`  
 Computes the objective function contribution and its gradient.
- def `get_H`  
 Computes the objective function contribution and its gradient / Hessian.
- def `link_from_tempdir`
- def `refresh_temp_directory`  
 Back up the temporary directory if desired, delete it and then create a new one.
- def `check_files`  
 Check this directory for the presence of readable files when the 'read' option is set.
- def `read`  
 Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.
- def `absrd`  
 Supply the correct directory specified by user's "read" option.
- def `maxrd`  
 Supply the latest existing temp-directory containing valid data.
- def `meta_indicate`  
 Wrap around the indicate function, so it can print to screen and also to a file.
- def `meta_get`  
 Wrapper around the get function.
- def `submit_jobs`
- def `stage`  
 Stages the directory for the target, and then launches Work Queue processes if any.
- def `wq_complete`  
 This method determines whether the Work Queue tasks for the current target have completed.
- def `printcool_table`  
 Print target information in an organized table format.
- def `__setattr__`
- def `set_option`

## Public Attributes

- `engine_`  
 Default file names for coordinates and key file.
- `boltz_wts`  
 Initialize the base class.
- `qmboltz_wts`  
 QM Boltzmann weights.
- `eqm`  
 Reference (QM) energies.
- `emd0`  
 Energies of the sampling simulation.

- **fqm**  
*Reference (QM) forces.*
- **espxyz**  
*ESP grid points.*
- **espval**  
*ESP values.*
- **qfnm**  
*The qdata.txt file that contains the QM energies and forces.*
- **qatoms**  
*The number of atoms in the QM calculation (Irrelevant if not fitting forces)*
- **e\_err**  
*Qualitative Indicator: average energy error (in kJ/mol)*
- **e\_err\_pct**
- **f\_err**  
*Qualitative Indicator: average force error (fractional)*
- **f\_err\_pct**
- **esp\_err**  
*Qualitative Indicator: "relative RMS" for electrostatic potential.*
- **nf\_err**
- **nf\_err\_pct**
- **tq\_err\_pct**
- **use\_nft**  
*Whether to compute net forces and torques, or not.*
- **ns**  
*Read in the trajectory file.*
- **mol**
- **nparticles**  
*The number of (atoms + drude particles + virtual sites)*
- **engine**  
*Build keyword dictionaries to pass to engine.*
- **AtomLists**  
*Lists of atoms to do net force/torque fitting and excluding virtual sites.*
- **AtomMask**
- **new\_vsites**  
*Read in the reference data.*
- **save\_vmvabs**  
*Save the mvabs from the last time we updated the vsites.*
- **force\_map**
- **nnf**
- **ntq**
- **force**
- **w\_force**
- **nesp**
- **fitatoms**
- **whamboltz**
- **nftqm**
- **fref**
- **w\_energy**

- `w_netforce`
- `w_torque`
- `e_ref`
- `f_ref`
- `nf_ref`
- `tq_ref`
- `tq_err`
- `w_resp`
- `invdists`
- `respterm`
- `objective`
- `rd`

*Root directory of the whole project.*
- `pgrad`

*Iteration where we turn on zero-gradient skipping.*
- `tempbase`

*Relative directory of target.*
- `tempdir`
- `rundir`

*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- `FF`

*Need the forcefield (here for now)*
- `xct`

*Counts how often the objective function was computed.*
- `gct`

*Counts how often the gradient was computed.*
- `hct`

*Counts how often the Hessian was computed.*
- `read_indicate`

*Whether to read indicate.log from file when restarting an aborted run.*
- `write_indicate`

*Whether to write indicate.log at every iteration (true for all but remote.)*
- `read_objective`

*Whether to read objective.p from file when restarting an aborted run.*
- `write_objective`

*Whether to write objective.p at every iteration (true for all but remote.)*
- `verbose_options`
- `PrintOptionDict`

### 8.5.1 Detailed Description

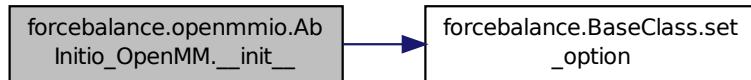
Force and energy matching using [OpenMM](#).

Definition at line 1166 of file openmmio.py.

### 8.5.2 Constructor & Destructor Documentation

```
def forcebalance.openmmio.ABInitio_OpenMM.__init__( self, options, tgt_opts, forcefield ) Definition at line 1167 of file openmmio.py.
```

Here is the call graph for this function:



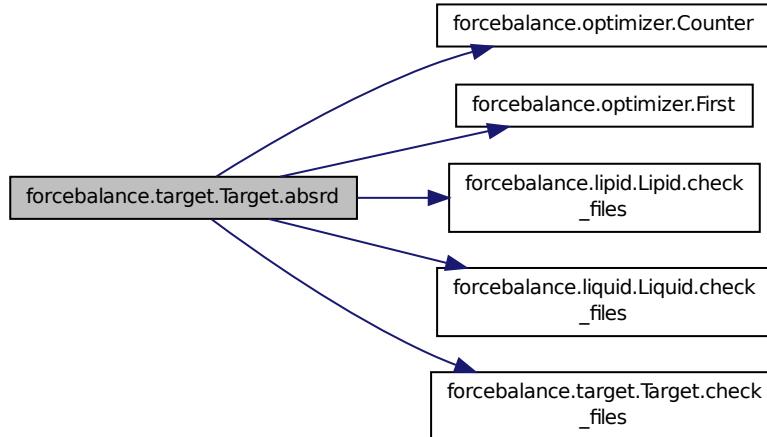
### 8.5.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.ABInitio.build_invdist( self, mvals ) [inherited] Definition at line 171 of file abinitio.py.
```

```
def forcebalance.target.Target.check_files( self, there ) [inherited] Check this directory for the presence of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.abinitio.AbInitio.compute_netforce_torque ( self, xyz, force, QM = False )  
[inherited] Definition at line 200 of file abinitio.py.
```

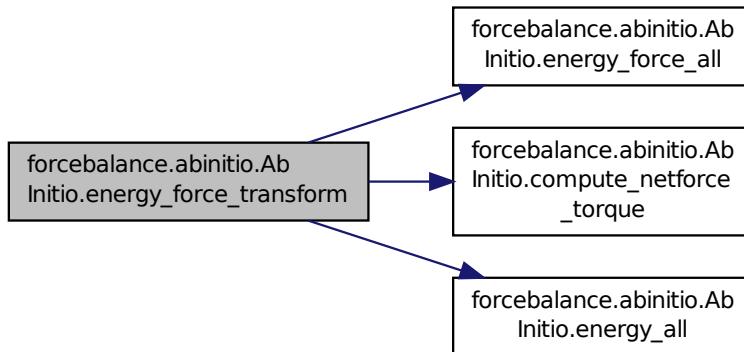
```
def forcebalance.abinitio.AbInitio.energy_all ( self ) [inherited] Definition at line 472 of file abinitio.py.
```

```
def forcebalance.abinitio.AbInitio.energy_force_all ( self ) [inherited] Definition at line 478 of file abinitio.py.
```

```
def forcebalance.abinitio.AbInitio.energy_force_one ( self, i ) [inherited] Definition at line 507 of file abinitio.py.
```

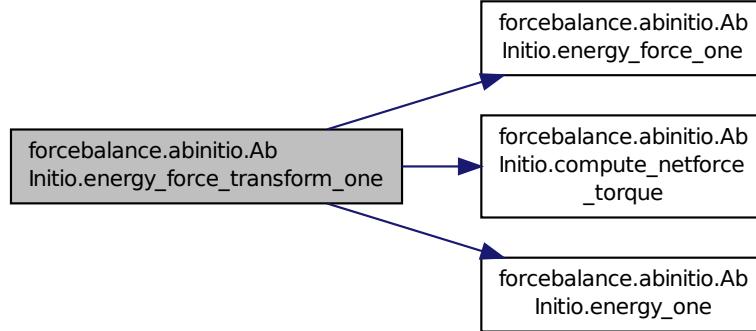
```
def forcebalance.abinitio.AbInitio.energy_force_transform ( self ) [inherited] Definition at line 484 of file abinitio.py.
```

Here is the call graph for this function:



```
def forcebalance.abinitio.AbInitio.energy_force_transform_one ( self, i ) [inherited] Definition at line 513 of file abinitio.py.
```

Here is the call graph for this function:

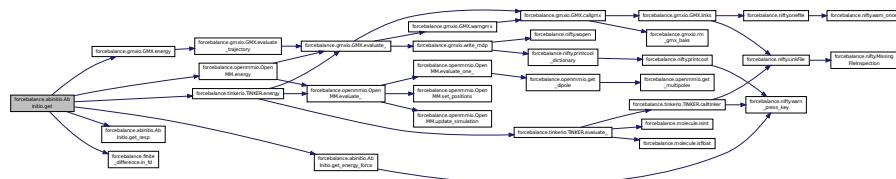


**def forcebalance.abinitio.AbInitio.energy\_one ( self, i ) [inherited]** Definition at line 501 of file abinitio.py.

**def forcebalance.abinitio.AbInitio.get ( self, mvals, AGrad = False, AHess = False ) [inherited]**

Definition at line 1152 of file abinitio.py.

Here is the call graph for this function:



**def forcebalance.abinitio.AbInitio.get\_energy\_force ( self, mvals, AGrad = False, AHess = False ) [inherited]** LPW 06-30-2013.

This subroutine builds the objective function (and optionally its derivatives) from a general simulation software. This is in contrast to using GROMACS-X2, which computes the objective function and prints it out; then 'get' only needs to call GROMACS and read it in.

This subroutine interfaces with simulation software 'drivers'. The driver is only expected to give the energy and forces.

Now this subroutine may sound trivial since the objective function is simply a least-squares quantity  $(M-Q)^2$  - but there are a number of nontrivial considerations. I will list them here.

0) Polytensor formulation: Because there may exist covariance between different components of the force (or covariance between the energy and the force), we build the objective function by taking outer products of vectors that have the form  $[E \ F_{1x} \ F_{1y} \ F_{1z} \ F_{2x} \ F_{2y} \dots]$ , and then we trace it with the inverse of the covariance matrix to get the objective function.

This version implements both the polytensor formulation and the standard formulation.

1) Boltzmann weights and normalization: Each snapshot has its own Boltzmann weight, which may or may not be normalized. This subroutine does the normalization automatically.

2) Subtracting out the mean energy gap: The zero-point energy difference between reference data and simulation is meaningless. This subroutine subtracts it out.

3) Hybrid ensembles: This program builds a combined objective function from both MM and QM ensembles, which is rigorously better than using a single ensemble.

Note that this subroutine does not do EVERYTHING that GROMACS-X2 can do, which includes:

1) Internal coordinate systems 2) 'Sampling correction' (deprecated, since it doesn't seem to work) 3) Analytic derivatives

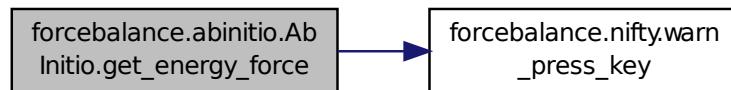
In the previous code (ForTune) this subroutine used analytic first derivatives of the energy and force to build the derivatives of the objective function. Here I will take a simplified approach, because building the derivatives are cumbersome. For now we will return the objective function ONLY. A two-point central difference should give us the first and diagonal second derivative anyhow.

**Todo** Parallelization over snapshots is not implemented yet

```
@param[in] mvals Mathematical parameter values
@param[in] AGrad Switch to turn on analytic gradient
@param[in] AHess Switch to turn on analytic Hessian
@return Answer Contribution to the objective function
```

Definition at line 588 of file abinitio.py.

Here is the call graph for this function:



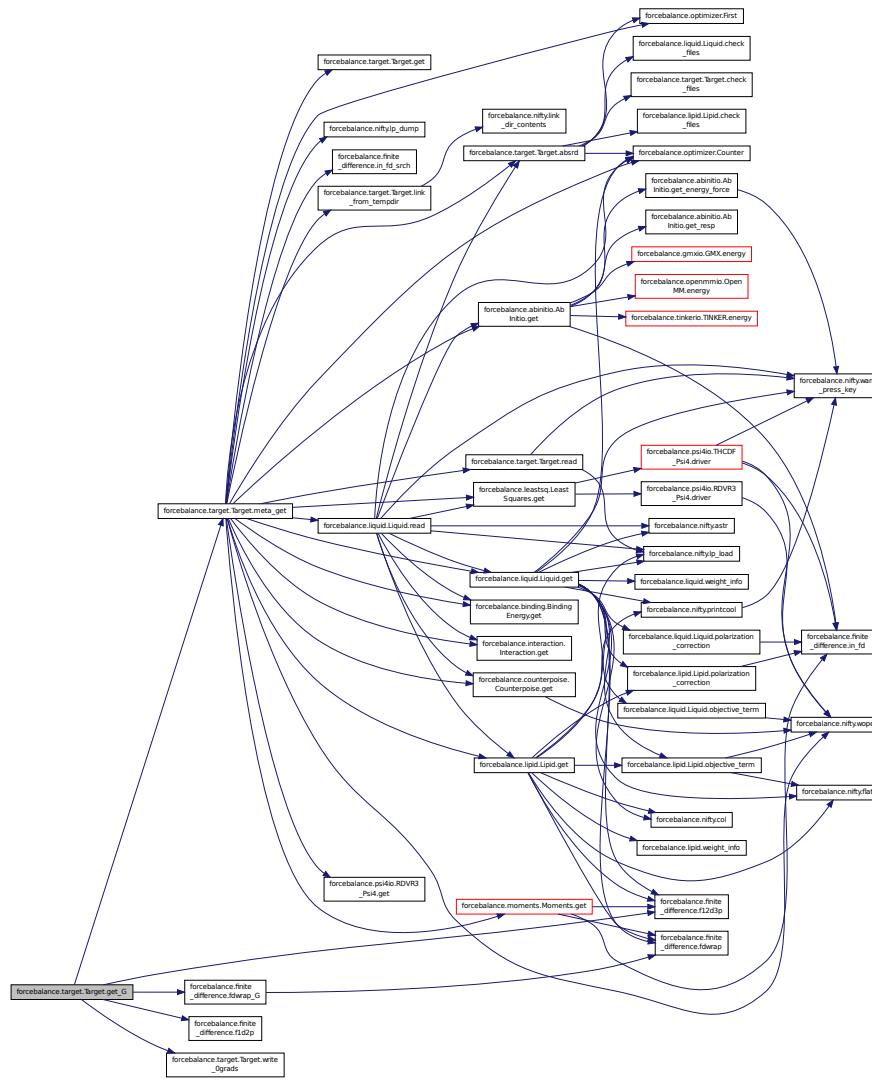
**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



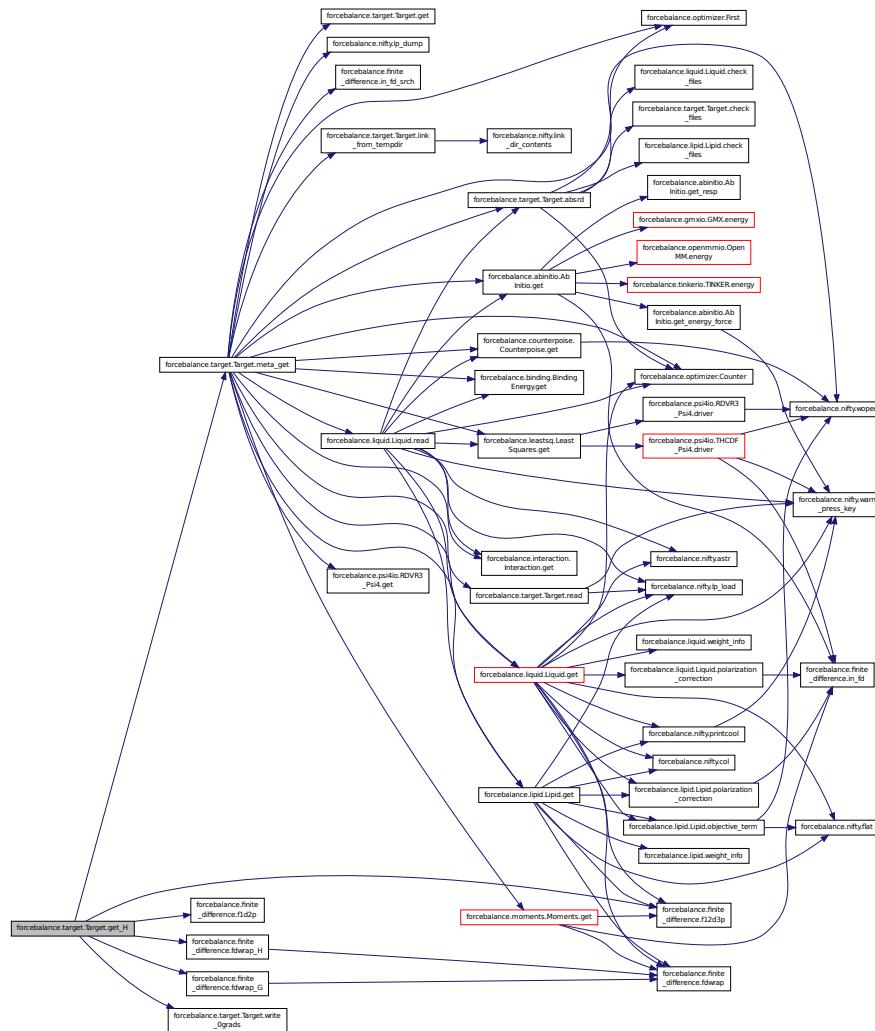
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.AbInitio.get_resp ( self, mvals, AGrad = False, AHess = False )
[inherited] Electrostatic potential fitting.
```

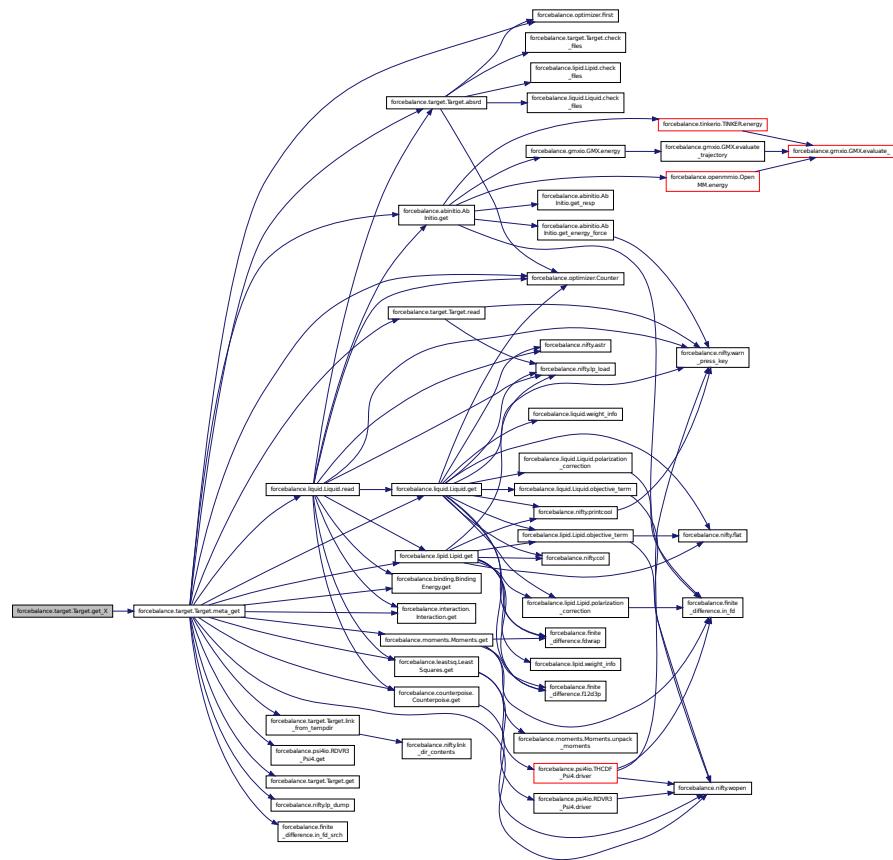
Implements the RESP objective function. (In Python so obviously not optimized.) This function takes the mathematical parameter values and returns the charges on the ATOMS (fancy mapping going on)

Definition at line 1053 of file abinitio.py.

```
def forcebalance.target.Target.get_X ( self, mvals = None ) [inherited] Computes the objective function contribution without any parametric derivatives.
```

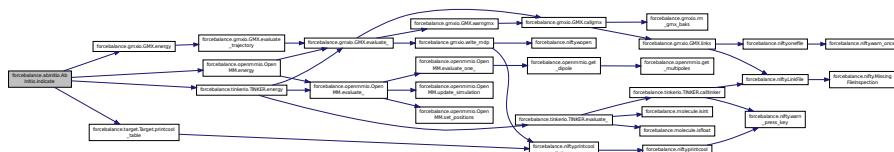
Definition at line 184 of file target.py.

Here is the call graph for this function:



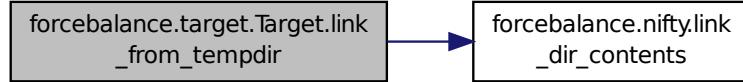
**def forcebalance.abinitio.AblInitio.indicate ( self ) [inherited]** Definition at line 448 of file abinitio.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

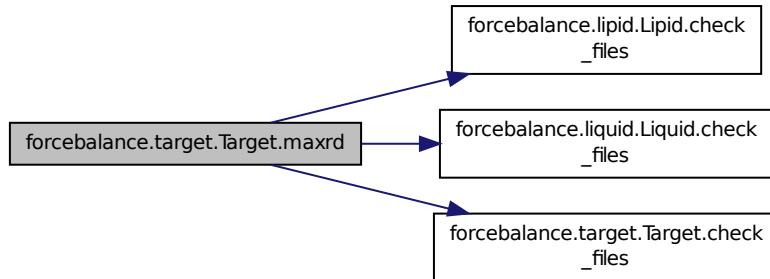
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

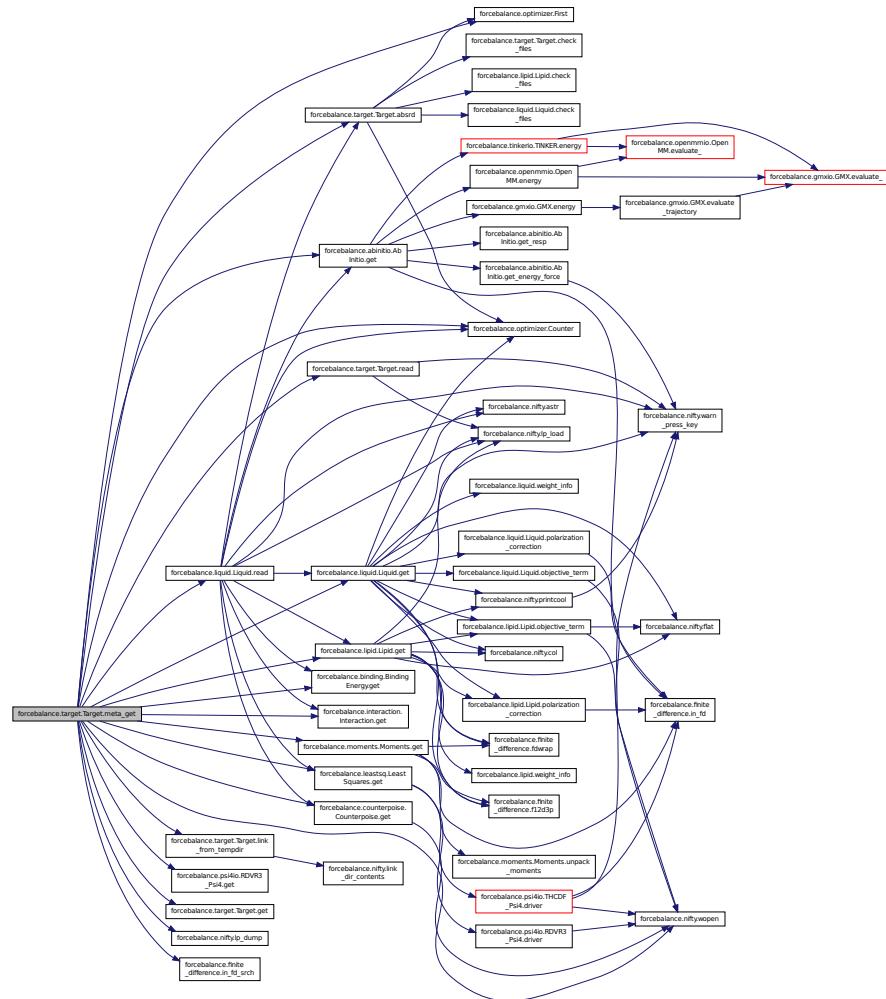


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

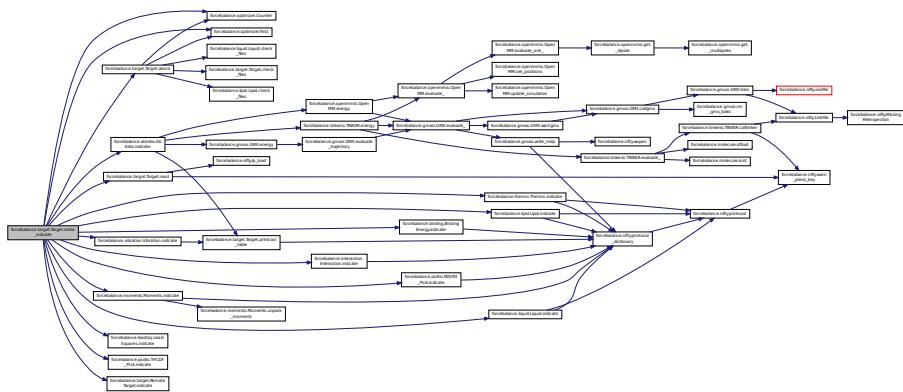
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

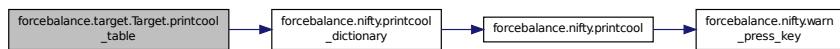
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

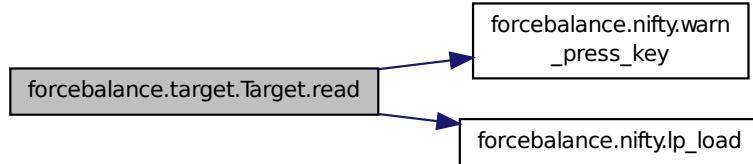
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.abinitio.AblInitio.read\_reference\_data ( self ) [inherited]** Read the reference ab initio data from a file such as qdata.txt.

**Todo** Add an option for picking any slice out of qdata.txt, helpful for cross-validation

**Todo** Closer integration of reference data with program - leave behind the qdata.txt format? (For now, I like the readability of qdata.txt)

After reading in the information from qdata.txt, it is converted into the GROMACS energy units (kind of an arbitrary choice); forces (kind of a misnomer in qdata.txt) are multiplied by -1 to convert gradients to forces.

We also subtract out the mean energies of all energy arrays because energy/force matching does not account for zero-point energy differences between MM and QM (i.e. energy of electrons in core orbitals).

The configurations in force/energy matching typically come from a the thermodynamic ensemble of the MM force field at some temperature (by running MD, for example), and for many reasons it is helpful to introduce non-Boltzmann weights in front of these configurations. There are two options: WHAM Boltzmann weights (for combining the weights of several simulations together) and QM Boltzmann weights (for converting MM weights into QM weights). Note that the two sets of weights 'stack'; i.e. they can be used at the same time.

A 'hybrid' ensemble is possible where we use 50% MM and 50% QM weights. Please read more in LPW and Troy Van Voorhis, JCP Vol. 133, Pg. 231101 (2010), doi:10.1063/1.3519043.

**Todo** The WHAM Boltzmann weights are generated by external scripts (wanalyze.py and make-wham-data.sh) and passed in; perhaps these scripts can be added to the ForceBalance distribution or integrated more tightly.

Finally, note that using non-Boltzmann weights degrades the statistical information content of the snapshots. This problem will generally become worse if the ensemble to which we're reweighting is dramatically different from the one we're sampling from. We end up with a set of Boltzmann weights like [1e-9, 1e-9, 1.0, 1e-9, 1e-9 ... ] and this is essentially just one snapshot. I believe Troy is working on something to cure this problem.

Here, we have a measure for the information content of our snapshots, which comes easily from the definition of information entropy:

```
S = -1*Sum_i(P_i*log(P_i))
InfoContent = exp(-S)
```

With uniform weights, InfoContent is equal to the number of snapshots; with horrible weights, InfoContent is closer to one.

Definition at line 329 of file abinitio.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

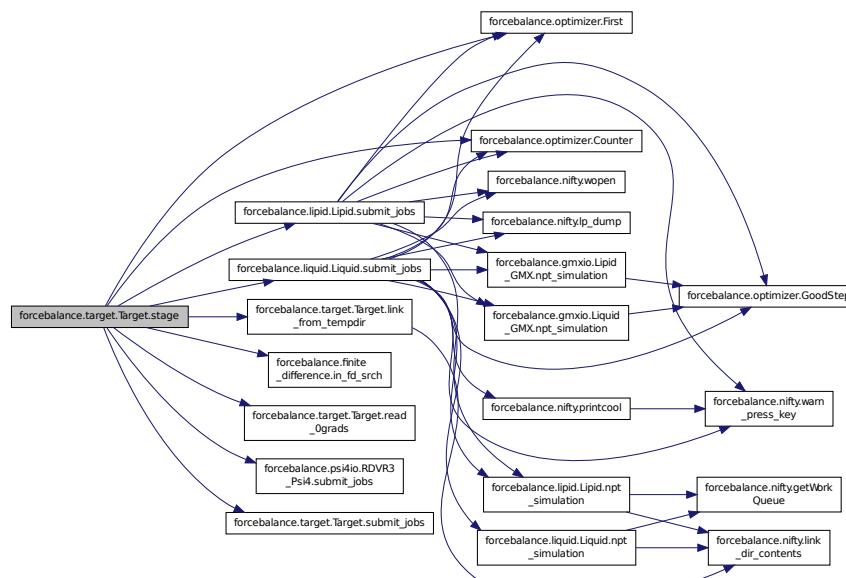
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

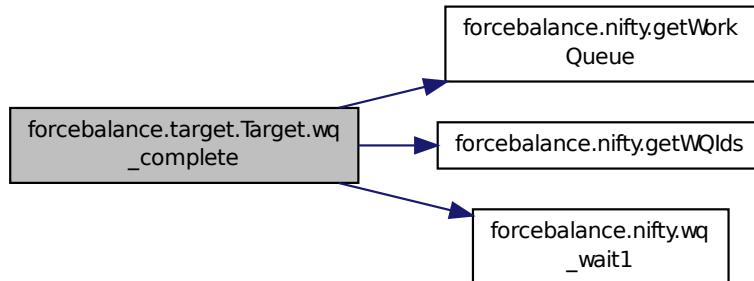


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work  
Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.write_Ograds ( self, Ans ) [inherited] Write a file to the target directory  
containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

#### 8.5.4 Member Data Documentation

```
forcebalance.abinitio.ABInitio.AtomLists [inherited] Lists of atoms to do net force/torque fitting and excluding virtual sites.
```

Definition at line 160 of file abinitio.py.

```
forcebalance.abinitio.ABInitio.AtomMask [inherited] Definition at line 161 of file abinitio.py.
```

```
forcebalance.abinitio.ABInitio.boltz_wts [inherited] Initialize the base class.
```

Number of snapshots Whether to use WHAM Boltzmann weights Whether to use the Sampling Correction Whether to match Absolute Energies (make sure you know what you're doing) Whether to use the Covariance Matrix Whether to use QM Boltzmann weights The temperature for QM Boltzmann weights Number of atoms that we are fitting Whether to fit Energies. Whether to fit Forces. Whether to fit Electrostatic Potential. Weights for the three components. Option for how much data to write to disk. Whether to do energy and force calculations for the whole trajectory, or to do one calculation per snapshot. OpenMM-only option - whether to run the energies and forces internally. Whether we have virtual sites (set at the global option level) Attenuate the weights as a function of energy What is the energy denominator? (Valid for 'attenuate') Set upper cutoff energy WHAM Boltzmann weights

Definition at line 116 of file abinitio.py.

```
forcebalance.abinitio.ABInitio.e_err [inherited] Qualitative Indicator: average energy error (in kJ/mol)
```

Definition at line 134 of file abinitio.py.

```
forcebalance.abinitio.ABInitio.e_err_pct [inherited] Definition at line 135 of file abinitio.py.
```

**forcebalance.abinitio.AbInitio.e.ref** [inherited] Definition at line 1030 of file abinitio.py.

**forcebalance.abinitio.AbInitio.emd0** [inherited] Energies of the sampling simulation.  
Definition at line 122 of file abinitio.py.

**forcebalance.abinitio.AbInitio.engine** [inherited] Build keyword dictionaries to pass to engine.  
Create engine object.  
Definition at line 158 of file abinitio.py.

**forcebalance.openmmio.AbInitio\_OpenMM.engine** Default file names for coordinates and key file.  
Definition at line 1173 of file openmmio.py.

**forcebalance.abinitio.AbInitio.eqm** [inherited] Reference (QM) energies.  
Definition at line 120 of file abinitio.py.

**forcebalance.abinitio.AbInitio.esp\_err** [inherited] Qualitative Indicator: "relative RMS" for electrostatic potential.  
Definition at line 140 of file abinitio.py.

**forcebalance.abinitio.AbInitio.espval** [inherited] ESP values.  
Definition at line 128 of file abinitio.py.

**forcebalance.abinitio.AbInitio.espxyz** [inherited] ESP grid points.  
Definition at line 126 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err** [inherited] Qualitative Indicator: average force error (fractional)  
Definition at line 137 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err\_pct** [inherited] Definition at line 138 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_ref** [inherited] Definition at line 1034 of file abinitio.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)  
Definition at line 160 of file target.py.

**forcebalance.abinitio.AbInitio.fitatoms** [inherited] Definition at line 366 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force** [inherited] Definition at line 361 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force\_map** [inherited] Definition at line 209 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fqm** [inherited] Reference (QM) forces.  
Definition at line 124 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fref** [inherited] Definition at line 442 of file abinitio.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.  
Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.abinitio.ABInitio.invdist [inherited]** Definition at line 1061 of file abinitio.py.

**forcebalance.abinitio.ABInitio.mol [inherited]** Definition at line 148 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nesp [inherited]** Definition at line 363 of file abinitio.py.

**forcebalance.abinitio.ABInitio.new\_vsites [inherited]** Read in the reference data.

The below two options are related to whether we want to rebuild virtual site positions. Rebuild the distance matrix if virtual site positions have changed

Definition at line 166 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_err [inherited]** Definition at line 141 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_err\_pct [inherited]** Definition at line 142 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_ref [inherited]** Definition at line 1038 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nftqm [inherited]** Definition at line 438 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nnf [inherited]** Definition at line 267 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nparticles [inherited]** The number of (atoms + drude particles + virtual sites)  
Definition at line 153 of file abinitio.py.

**forcebalance.abinitio.ABInitio.ns [inherited]** Read in the trajectory file.

Definition at line 147 of file abinitio.py.

**forcebalance.abinitio.ABInitio.ntq [inherited]** Definition at line 268 of file abinitio.py.

**forcebalance.abinitio.ABInitio.objective [inherited]** Definition at line 1172 of file abinitio.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.qfnm [inherited]** The qdata.txt file that contains the QM energies and forces.  
Definition at line 130 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmatoms [inherited]** The number of atoms in the QM calculation (Irrelevant if not fitting forces)

Definition at line 132 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmboltz\_wts [inherited]** QM Boltzmann weights.

Definition at line 118 of file abinitio.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.abinitio.ABInitio.respterm [inherited]** Definition at line 1140 of file abinitio.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.abinitio.ABInitio.save\_vmvales [inherited]** Save the mvales from the last time we updated the vsites.

Definition at line 168 of file abinitio.py.

**forcebalance.target.Target.tempbase [inherited]** Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir [inherited]** Definition at line 155 of file target.py.

**forcebalance.abinitio.ABInitio.tq\_err [inherited]** Definition at line 1042 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_err\_pct [inherited]** Definition at line 143 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_ref [inherited]** Definition at line 1041 of file abinitio.py.

**forcebalance.abinitio.ABInitio.use\_nft [inherited]** Whether to compute net forces and torques, or not.

Definition at line 145 of file abinitio.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.w\_energy [inherited]** Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_force** [inherited] Definition at line 362 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_netforce** [inherited] Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_resp** [inherited] Definition at line 1054 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_torque** [inherited] Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz** [inherited] Definition at line 382 of file abinitio.py.

**forcebalance.target.Target.write\_indicate** [inherited] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

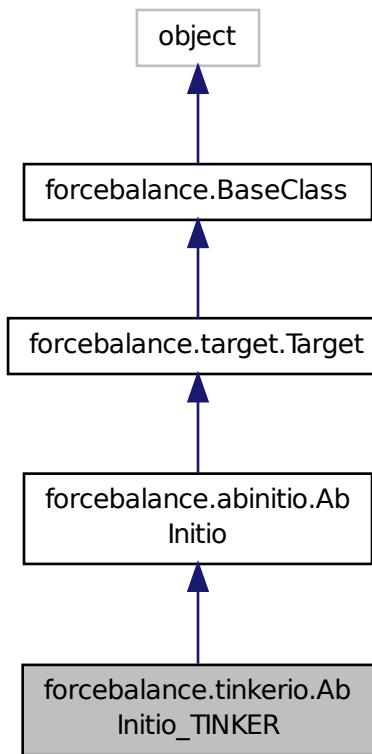
The documentation for this class was generated from the following file:

- [openmmio.py](#)

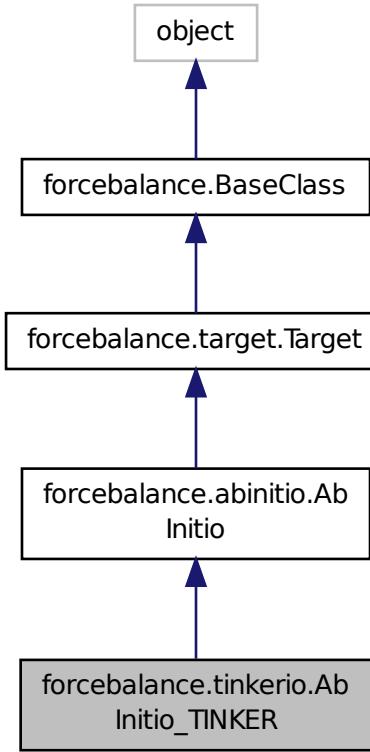
## 8.6 forcebalance.tinkerio.ABInitio\_TINKER Class Reference

Subclass of Target for force and energy matching using [TINKER](#).

Inheritance diagram for forcebalance.tinkerio.AbInitio\_TINKER:



Collaboration diagram for forcebalance.tinkerio.AbInitio\_TINKER:



#### Public Member Functions

- def `_init_`
- def `build_invdist`
- def `compute_netforce_torque`
- def `read_reference_data`

*Read the reference ab initio data from a file such as qdata.txt.*
- def `indicate`
- def `energy_all`
- def `energy_force_all`
- def `energy_force_transform`
- def `energy_one`
- def `energy_force_one`
- def `energy_force_transform_one`
- def `get_energy_force`

*LPW 06-30-2013.*
- def `get_resp`

*Electrostatic potential fitting.*

- def `get`  
 Computes the objective function contribution without any parametric derivatives.
- def `get_X`  
 Read a file from the target directory containing names of parameters that don't contribute to the gradient.
- def `read_0grads`  
 Write a file to the target directory containing names of parameters that don't contribute to the gradient.
- def `write_0grads`  
 Write a file to the target directory containing names of parameters that don't contribute to the gradient.
- def `get_G`  
 Computes the objective function contribution and its gradient.
- def `get_H`  
 Computes the objective function contribution and its gradient / Hessian.
- def `link_from_tempdir`
- def `refresh_temp_directory`  
 Back up the temporary directory if desired, delete it and then create a new one.
- def `check_files`  
 Check this directory for the presence of readable files when the 'read' option is set.
- def `read`  
 Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.
- def `absrd`  
 Supply the correct directory specified by user's "read" option.
- def `maxrd`  
 Supply the latest existing temp-directory containing valid data.
- def `meta_indicate`  
 Wrap around the indicate function, so it can print to screen and also to a file.
- def `meta_get`  
 Wrapper around the get function.
- def `submit_jobs`
- def `stage`  
 Stages the directory for the target, and then launches Work Queue processes if any.
- def `wq_complete`  
 This method determines whether the Work Queue tasks for the current target have completed.
- def `printcool_table`  
 Print target information in an organized table format.
- def `__setattr__`
- def `set_option`

## Public Attributes

- `engine_`  
 Default file names for coordinates and key file.
- `boltz_wts`  
 Initialize the base class.
- `qmboltz_wts`  
 QM Boltzmann weights.
- `eqm`  
 Reference (QM) energies.
- `emd0`  
 Energies of the sampling simulation.

- **fqm**  
*Reference (QM) forces.*
- **espxyz**  
*ESP grid points.*
- **espval**  
*ESP values.*
- **qfnm**  
*The qdata.txt file that contains the QM energies and forces.*
- **qmatoms**  
*The number of atoms in the QM calculation (Irrelevant if not fitting forces)*
- **e\_err**  
*Qualitative Indicator: average energy error (in kJ/mol)*
- **e\_err\_pct**
- **f\_err**  
*Qualitative Indicator: average force error (fractional)*
- **f\_err\_pct**
- **esp\_err**  
*Qualitative Indicator: "relative RMS" for electrostatic potential.*
- **nf\_err**
- **nf\_err\_pct**
- **tq\_err\_pct**
- **use\_nft**  
*Whether to compute net forces and torques, or not.*
- **ns**  
*Read in the trajectory file.*
- **mol**
- **nparticles**  
*The number of (atoms + drude particles + virtual sites)*
- **engine**  
*Build keyword dictionaries to pass to engine.*
- **AtomLists**  
*Lists of atoms to do net force/torque fitting and excluding virtual sites.*
- **AtomMask**
- **new\_vsites**  
*Read in the reference data.*
- **save\_vmvalls**  
*Save the mvalls from the last time we updated the vsites.*
- **force\_map**
- **nnf**
- **ntq**
- **force**
- **w\_force**
- **nesp**
- **fitatoms**
- **whamboltz**
- **nftqm**
- **fref**
- **w\_energy**

- `w_netforce`
- `w_torque`
- `e_ref`
- `f_ref`
- `nf_ref`
- `tq_ref`
- `tq_err`
- `w_resp`
- `invdists`
- `respterm`
- `objective`
- `rd`

*Root directory of the whole project.*
- `pgrad`

*Iteration where we turn on zero-gradient skipping.*
- `tempbase`

*Relative directory of target.*
- `tempdir`
- `rundir`

*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- `FF`

*Need the forcefield (here for now)*
- `xct`

*Counts how often the objective function was computed.*
- `gct`

*Counts how often the gradient was computed.*
- `hct`

*Counts how often the Hessian was computed.*
- `read_indicate`

*Whether to read indicate.log from file when restarting an aborted run.*
- `write_indicate`

*Whether to write indicate.log at every iteration (true for all but remote.)*
- `read_objective`

*Whether to read objective.p from file when restarting an aborted run.*
- `write_objective`

*Whether to write objective.p at every iteration (true for all but remote.)*
- `verbose_options`
- `PrintOptionDict`

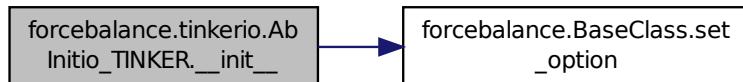
### 8.6.1 Detailed Description

Subclass of Target for force and energy matching using [TINKER](#).  
 Definition at line 1067 of file `tinkerio.py`.

## 8.6.2 Constructor & Destructor Documentation

```
def forcebalance.tinkerio.AbInitio_TINKER.__init__ ( self, options, tgt_opts, forcefield ) Definition at line 1068 of file tinkerio.py.
```

Here is the call graph for this function:



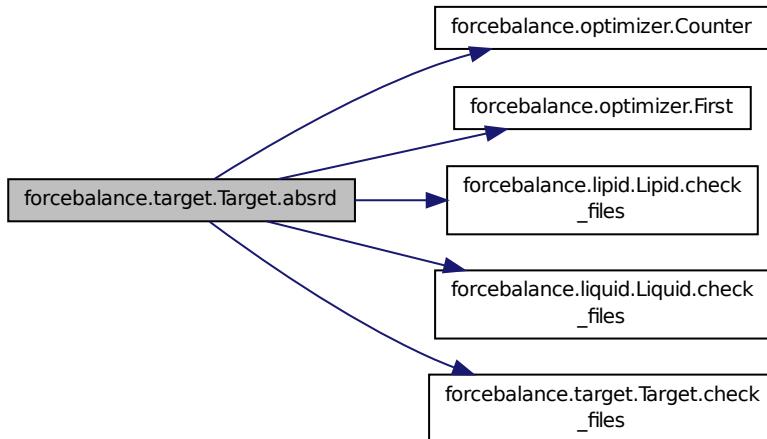
## 8.6.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.AbInitio.build_invdist ( self, mvals ) [inherited] Definition at line 171 of file abinitio.py.
```

```
def forcebalance.target.Target.check_files ( self, there ) [inherited] Check this directory for the presence of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.abinitio.AbInitio.compute_netforce_torque ( self, xyz, force, QM = False )  
[inherited] Definition at line 200 of file abinitio.py.
```

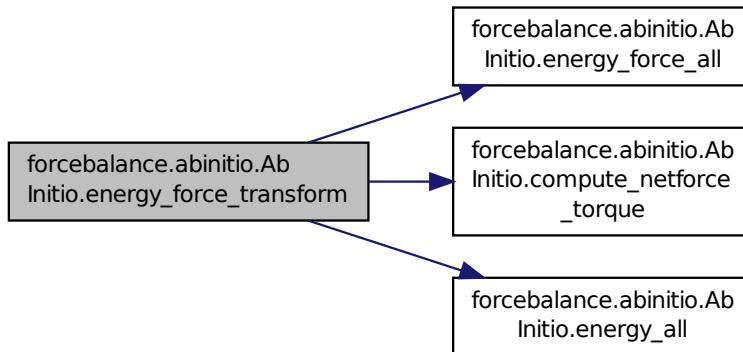
```
def forcebalance.abinitio.AbInitio.energy_all ( self ) [inherited] Definition at line 472 of file abinitio.py.
```

```
def forcebalance.abinitio.AbInitio.energy_force_all ( self ) [inherited] Definition at line 478 of file abinitio.py.
```

```
def forcebalance.abinitio.AbInitio.energy_force_one ( self, i ) [inherited] Definition at line 507 of file abinitio.py.
```

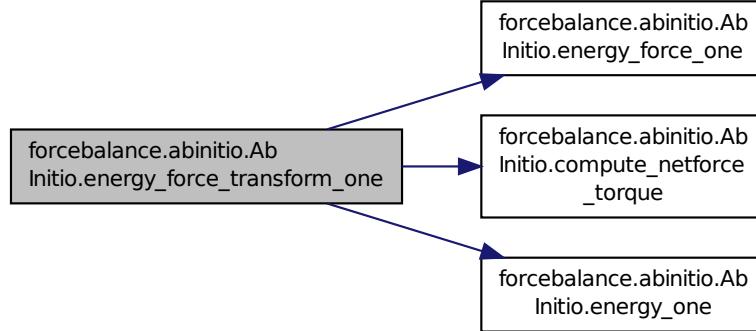
```
def forcebalance.abinitio.AbInitio.energy_force_transform ( self ) [inherited] Definition at line 484 of file abinitio.py.
```

Here is the call graph for this function:



```
def forcebalance.abinitio.AbInitio.energy_force_transform_one ( self, i ) [inherited] Definition at line 513 of file abinitio.py.
```

Here is the call graph for this function:

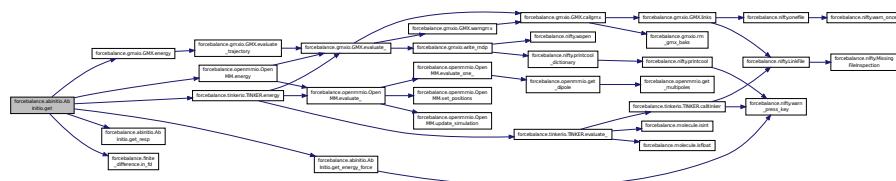


**def forcebalance.abinitio.AbInitio.energy\_one ( self, i ) [inherited]** Definition at line 501 of file abinitio.py.

**def forcebalance.abinitio.AbInitio.get ( self, mvals, AGrad = False, AHess = False ) [inherited]**

Definition at line 1152 of file abinitio.py.

Here is the call graph for this function:



**def forcebalance.abinitio.AbInitio.get\_energy\_force ( self, mvals, AGrad = False, AHess = False ) [inherited]** LPW 06-30-2013.

This subroutine builds the objective function (and optionally its derivatives) from a general simulation software. This is in contrast to using GROMACS-X2, which computes the objective function and prints it out; then 'get' only needs to call GROMACS and read it in.

This subroutine interfaces with simulation software 'drivers'. The driver is only expected to give the energy and forces.

Now this subroutine may sound trivial since the objective function is simply a least-squares quantity  $(M-Q)^2$  - but there are a number of nontrivial considerations. I will list them here.

0) Polytensor formulation: Because there may exist covariance between different components of the force (or covariance between the energy and the force), we build the objective function by taking outer products of vectors that have the form  $[E \ F_{1x} \ F_{1y} \ F_{1z} \ F_{2x} \ F_{2y} \dots]$ , and then we trace it with the inverse of the covariance matrix to get the objective function.

This version implements both the polytensor formulation and the standard formulation.

1) Boltzmann weights and normalization: Each snapshot has its own Boltzmann weight, which may or may not be normalized. This subroutine does the normalization automatically.

2) Subtracting out the mean energy gap: The zero-point energy difference between reference data and simulation is meaningless. This subroutine subtracts it out.

3) Hybrid ensembles: This program builds a combined objective function from both MM and QM ensembles, which is rigorously better than using a single ensemble.

Note that this subroutine does not do EVERYTHING that GROMACS-X2 can do, which includes:

1) Internal coordinate systems 2) 'Sampling correction' (deprecated, since it doesn't seem to work) 3) Analytic derivatives

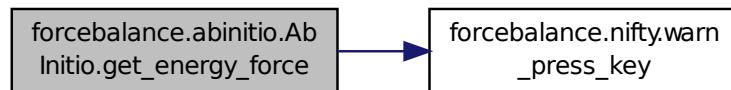
In the previous code (ForTune) this subroutine used analytic first derivatives of the energy and force to build the derivatives of the objective function. Here I will take a simplified approach, because building the derivatives are cumbersome. For now we will return the objective function ONLY. A two-point central difference should give us the first and diagonal second derivative anyhow.

**Todo** Parallelization over snapshots is not implemented yet

```
@param[in] mvals Mathematical parameter values
@param[in] AGrad Switch to turn on analytic gradient
@param[in] AHess Switch to turn on analytic Hessian
@return Answer Contribution to the objective function
```

Definition at line 588 of file abinitio.py.

Here is the call graph for this function:



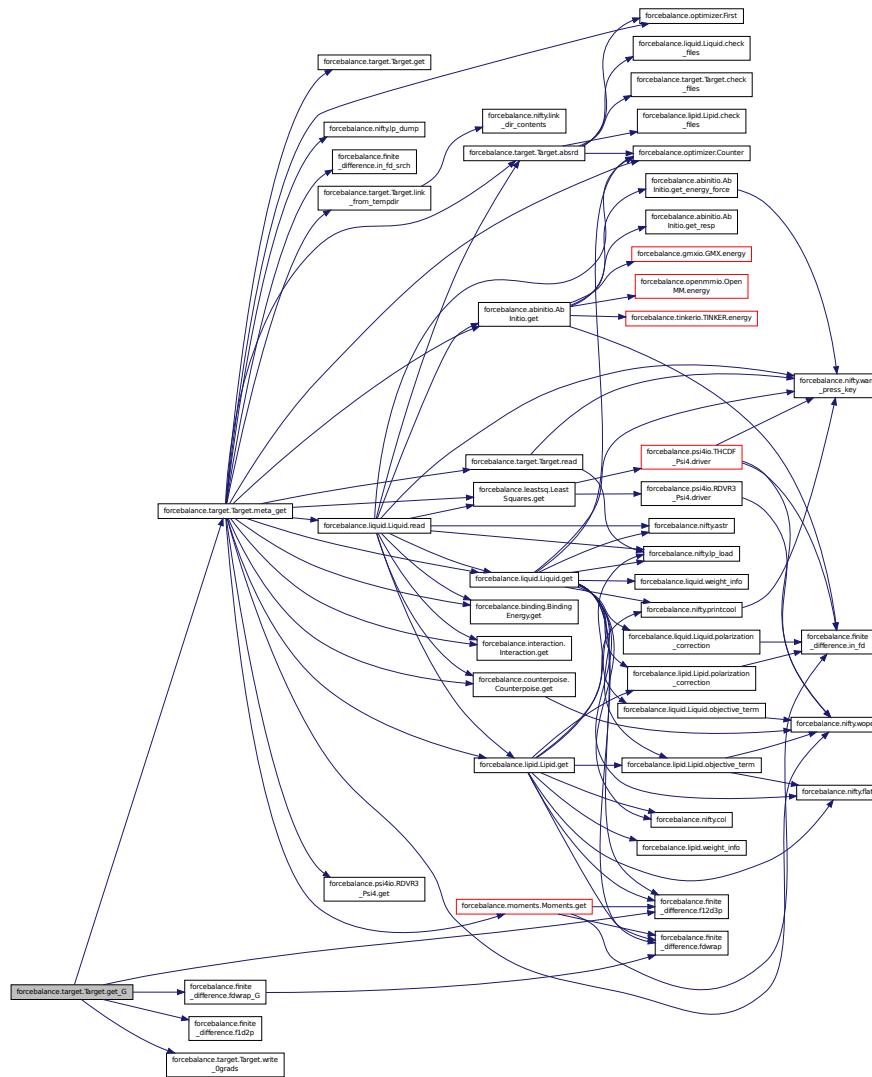
**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



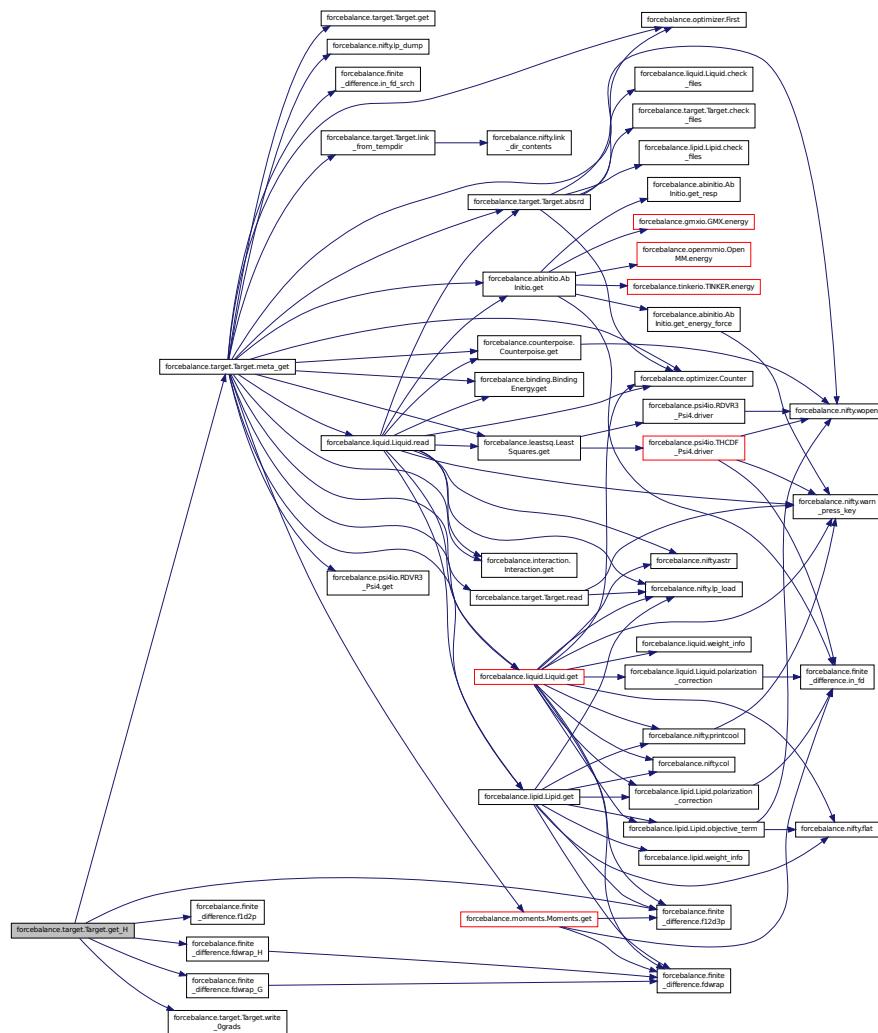
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.AbInitio.get_resp ( self, mvals, AGrad = False, AHess = False )  
[inherited] Electrostatic potential fitting.
```

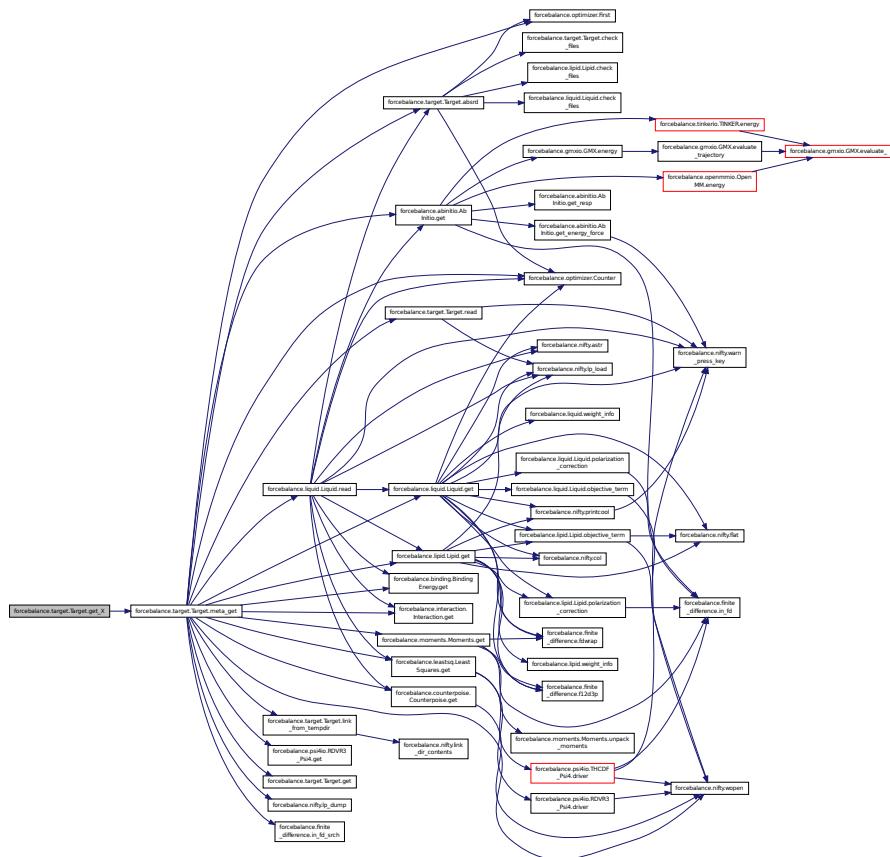
Implements the RESP objective function. (In Python so obviously not optimized.) This function takes the mathematical parameter values and returns the charges on the ATOMS (fancy mapping going on)

Definition at line 1053 of file abinitio.py.

**def forcebalance.target.Target.get\_X( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

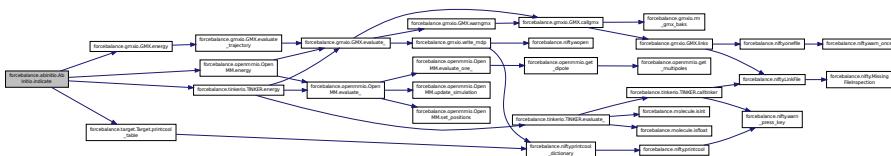
Definition at line 184 of file target.py.

Here is the call graph for this function:



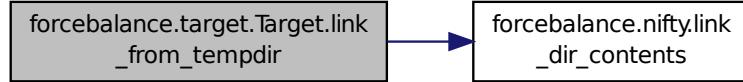
**def forcebalance.abinitio.ABInitio.indicate ( self ) [inherited]** Definition at line 448 of file abinitio.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

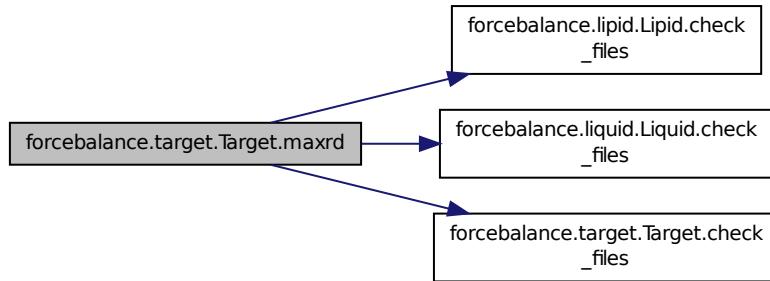
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

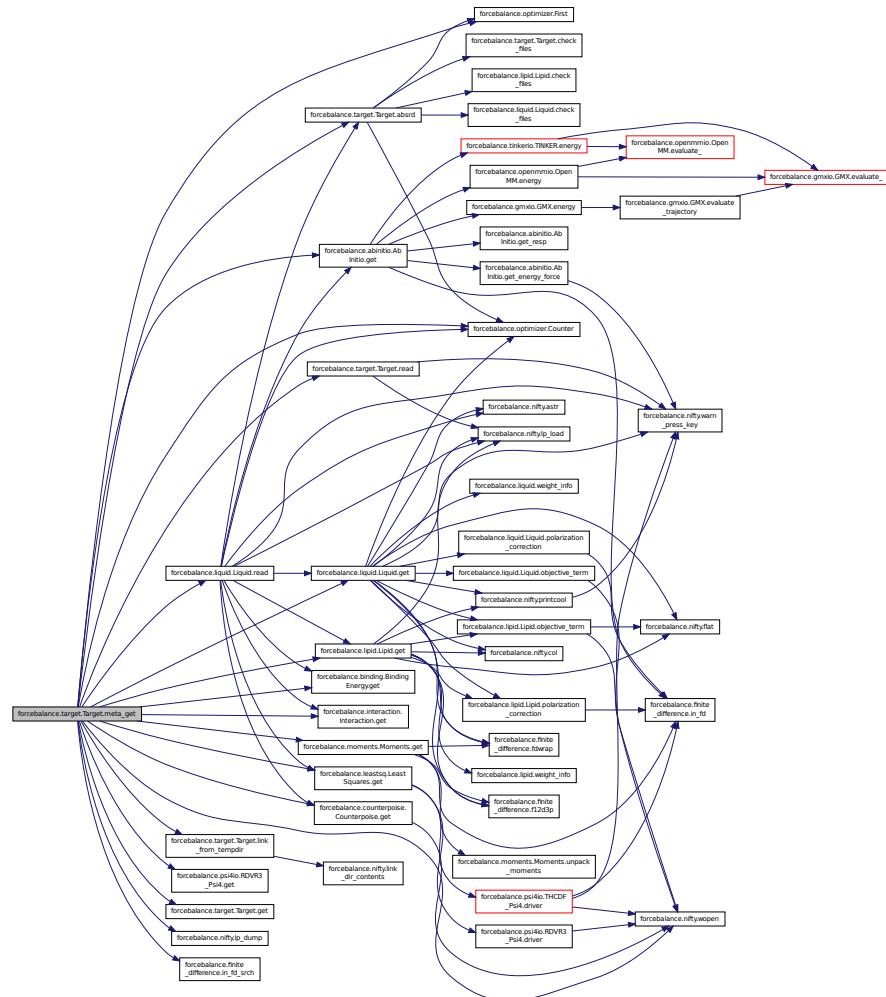


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

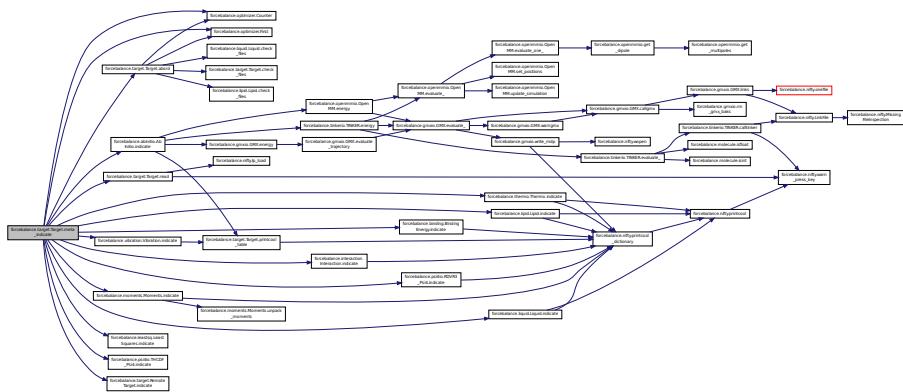
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

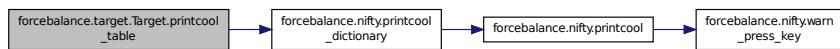
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

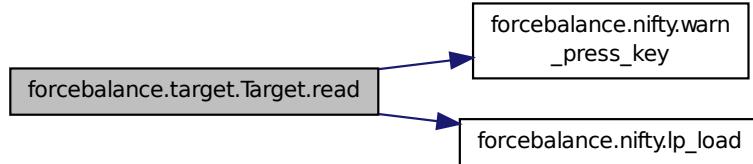
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.abinitio.AblInitio.read\_reference\_data ( self ) [inherited]** Read the reference ab initio data from a file such as qdata.txt.

**Todo** Add an option for picking any slice out of qdata.txt, helpful for cross-validation

**Todo** Closer integration of reference data with program - leave behind the qdata.txt format? (For now, I like the readability of qdata.txt)

After reading in the information from qdata.txt, it is converted into the GROMACS energy units (kind of an arbitrary choice); forces (kind of a misnomer in qdata.txt) are multiplied by -1 to convert gradients to forces.

We also subtract out the mean energies of all energy arrays because energy/force matching does not account for zero-point energy differences between MM and QM (i.e. energy of electrons in core orbitals).

The configurations in force/energy matching typically come from a the thermodynamic ensemble of the MM force field at some temperature (by running MD, for example), and for many reasons it is helpful to introduce non-Boltzmann weights in front of these configurations. There are two options: WHAM Boltzmann weights (for combining the weights of several simulations together) and QM Boltzmann weights (for converting MM weights into QM weights). Note that the two sets of weights 'stack'; i.e. they can be used at the same time.

A 'hybrid' ensemble is possible where we use 50% MM and 50% QM weights. Please read more in LPW and Troy Van Voorhis, JCP Vol. 133, Pg. 231101 (2010), doi:10.1063/1.3519043.

**Todo** The WHAM Boltzmann weights are generated by external scripts (wanalyze.py and make-wham-data.sh) and passed in; perhaps these scripts can be added to the ForceBalance distribution or integrated more tightly.

Finally, note that using non-Boltzmann weights degrades the statistical information content of the snapshots. This problem will generally become worse if the ensemble to which we're reweighting is dramatically different from the one we're sampling from. We end up with a set of Boltzmann weights like [1e-9, 1e-9, 1.0, 1e-9, 1e-9 ... ] and this is essentially just one snapshot. I believe Troy is working on something to cure this problem.

Here, we have a measure for the information content of our snapshots, which comes easily from the definition of information entropy:

```
S = -1*Sum_i(P_i*log(P_i))
InfoContent = exp(-S)
```

With uniform weights, InfoContent is equal to the number of snapshots; with horrible weights, InfoContent is closer to one.

Definition at line 329 of file abinitio.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

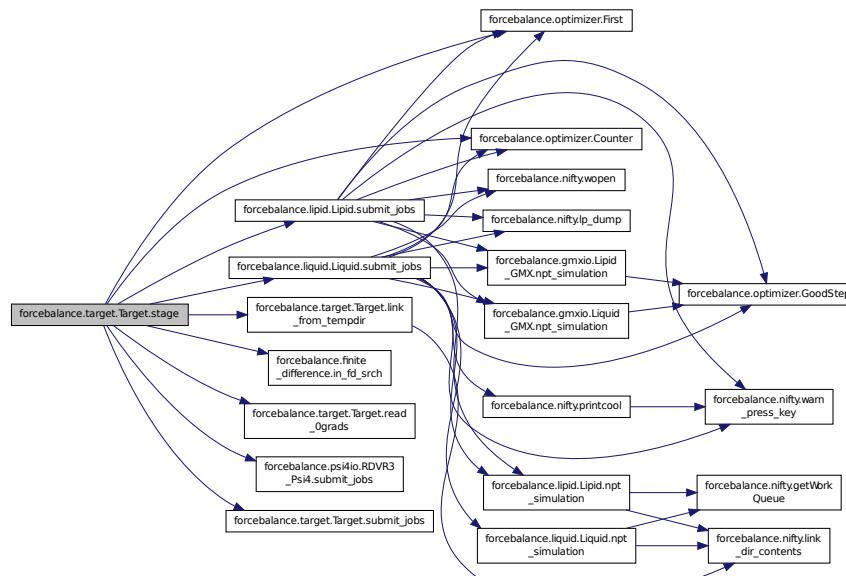
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

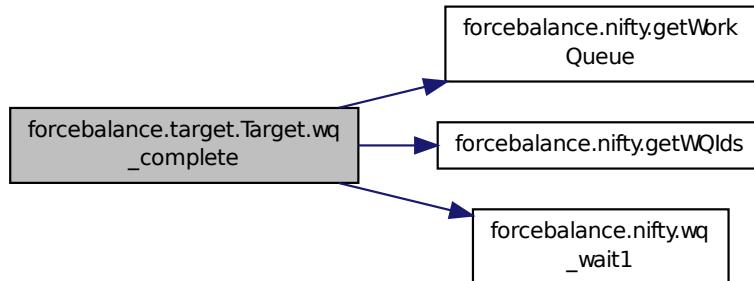


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work  
Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.write_Ograds ( self, Ans ) [inherited] Write a file to the target directory  
containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

#### 8.6.4 Member Data Documentation

```
forcebalance.abinitio.ABInitio.AtomLists [inherited] Lists of atoms to do net force/torque fitting and excluding virtual sites.
```

Definition at line 160 of file abinitio.py.

```
forcebalance.abinitio.ABInitio.AtomMask [inherited] Definition at line 161 of file abinitio.py.
```

```
forcebalance.abinitio.ABInitio.boltz_wts [inherited] Initialize the base class.
```

Number of snapshots Whether to use WHAM Boltzmann weights Whether to use the Sampling Correction Whether to match Absolute Energies (make sure you know what you're doing) Whether to use the Covariance Matrix Whether to use QM Boltzmann weights The temperature for QM Boltzmann weights Number of atoms that we are fitting Whether to fit Energies. Whether to fit Forces. Whether to fit Electrostatic Potential. Weights for the three components. Option for how much data to write to disk. Whether to do energy and force calculations for the whole trajectory, or to do one calculation per snapshot. OpenMM-only option - whether to run the energies and forces internally. Whether we have virtual sites (set at the global option level) Attenuate the weights as a function of energy What is the energy denominator? (Valid for 'attenuate') Set upper cutoff energy WHAM Boltzmann weights

Definition at line 116 of file abinitio.py.

```
forcebalance.abinitio.ABInitio.e_err [inherited] Qualitative Indicator: average energy error (in kJ/mol)
```

Definition at line 134 of file abinitio.py.

```
forcebalance.abinitio.ABInitio.e_err_pct [inherited] Definition at line 135 of file abinitio.py.
```

**forcebalance.abinitio.AbInitio.e.ref** [inherited] Definition at line 1030 of file abinitio.py.

**forcebalance.abinitio.AbInitio.emd0** [inherited] Energies of the sampling simulation.  
Definition at line 122 of file abinitio.py.

**forcebalance.abinitio.AbInitio.engine** [inherited] Build keyword dictionaries to pass to engine.  
Create engine object.  
Definition at line 158 of file abinitio.py.

**forcebalance.tinkerio.AbInitio\_TINKER.engine** Default file names for coordinates and key file.  
Definition at line 1072 of file tinkerio.py.

**forcebalance.abinitio.AbInitio.eqm** [inherited] Reference (QM) energies.  
Definition at line 120 of file abinitio.py.

**forcebalance.abinitio.AbInitio.esp\_err** [inherited] Qualitative Indicator: "relative RMS" for electrostatic potential.  
Definition at line 140 of file abinitio.py.

**forcebalance.abinitio.AbInitio.espval** [inherited] ESP values.  
Definition at line 128 of file abinitio.py.

**forcebalance.abinitio.AbInitio.espxyz** [inherited] ESP grid points.  
Definition at line 126 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err** [inherited] Qualitative Indicator: average force error (fractional)  
Definition at line 137 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err\_pct** [inherited] Definition at line 138 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_ref** [inherited] Definition at line 1034 of file abinitio.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)  
Definition at line 160 of file target.py.

**forcebalance.abinitio.AbInitio.fitatoms** [inherited] Definition at line 366 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force** [inherited] Definition at line 361 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force\_map** [inherited] Definition at line 209 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fqm** [inherited] Reference (QM) forces.  
Definition at line 124 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fref** [inherited] Definition at line 442 of file abinitio.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.  
Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.abinitio.ABInitio.invdist [inherited]** Definition at line 1061 of file abinitio.py.

**forcebalance.abinitio.ABInitio.mol [inherited]** Definition at line 148 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nesp [inherited]** Definition at line 363 of file abinitio.py.

**forcebalance.abinitio.ABInitio.new\_vsites [inherited]** Read in the reference data.

The below two options are related to whether we want to rebuild virtual site positions. Rebuild the distance matrix if virtual site positions have changed

Definition at line 166 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_err [inherited]** Definition at line 141 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_err\_pct [inherited]** Definition at line 142 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_ref [inherited]** Definition at line 1038 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nftqm [inherited]** Definition at line 438 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nnf [inherited]** Definition at line 267 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nparticles [inherited]** The number of (atoms + drude particles + virtual sites)  
Definition at line 153 of file abinitio.py.

**forcebalance.abinitio.ABInitio.ns [inherited]** Read in the trajectory file.

Definition at line 147 of file abinitio.py.

**forcebalance.abinitio.ABInitio.ntq [inherited]** Definition at line 268 of file abinitio.py.

**forcebalance.abinitio.ABInitio.objective [inherited]** Definition at line 1172 of file abinitio.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.qfnm [inherited]** The qdata.txt file that contains the QM energies and forces.  
Definition at line 130 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmatoms [inherited]** The number of atoms in the QM calculation (Irrelevant if not fitting forces)

Definition at line 132 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmboltz\_wts [inherited]** QM Boltzmann weights.

Definition at line 118 of file abinitio.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.abinitio.ABInitio.respterm [inherited]** Definition at line 1140 of file abinitio.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.abinitio.ABInitio.save\_vmvales [inherited]** Save the mvales from the last time we updated the vsites.

Definition at line 168 of file abinitio.py.

**forcebalance.target.Target.tempbase [inherited]** Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir [inherited]** Definition at line 155 of file target.py.

**forcebalance.abinitio.ABInitio.tq\_err [inherited]** Definition at line 1042 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_err\_pct [inherited]** Definition at line 143 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_ref [inherited]** Definition at line 1041 of file abinitio.py.

**forcebalance.abinitio.ABInitio.use\_nft [inherited]** Whether to compute net forces and torques, or not.

Definition at line 145 of file abinitio.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.w\_energy [inherited]** Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_force** [inherited] Definition at line 362 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_netforce** [inherited] Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_resp** [inherited] Definition at line 1054 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_torque** [inherited] Definition at line 630 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz** [inherited] Definition at line 382 of file abinitio.py.

**forcebalance.target.Target.write\_indicate** [inherited] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

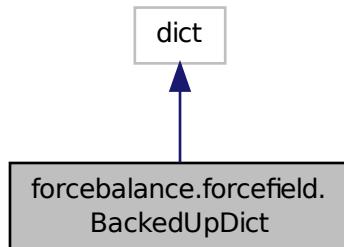
Definition at line 162 of file target.py.

The documentation for this class was generated from the following file:

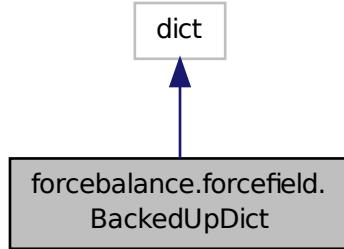
- [tinkerio.py](#)

## 8.7 forcebalance.forcefield.BackedUpDict Class Reference

Inheritance diagram for forcebalance.forcefield.BackedUpDict:



Collaboration diagram for forcebalance.forcefield.BackedUpDict:



### Public Member Functions

- def [\\_\\_init\\_\\_](#)
- def [\\_\\_missing\\_\\_](#)

### Public Attributes

- [backup\\_dict](#)

#### 8.7.1 Detailed Description

Definition at line 176 of file forcefield.py.

#### 8.7.2 Constructor & Destructor Documentation

**def forcebalance.forcefield.BackedUpDict.\_\_init\_\_ ( self, backup\_dict )** Definition at line 177 of file forcefield.py.

#### 8.7.3 Member Function Documentation

**def forcebalance.forcefield.BackedUpDict.\_\_missing\_\_ ( self, key )** Definition at line 180 of file forcefield.py.

#### 8.7.4 Member Data Documentation

**forcebalance.forcefield.BackedUpDict.backup\_dict** Definition at line 179 of file forcefield.py.

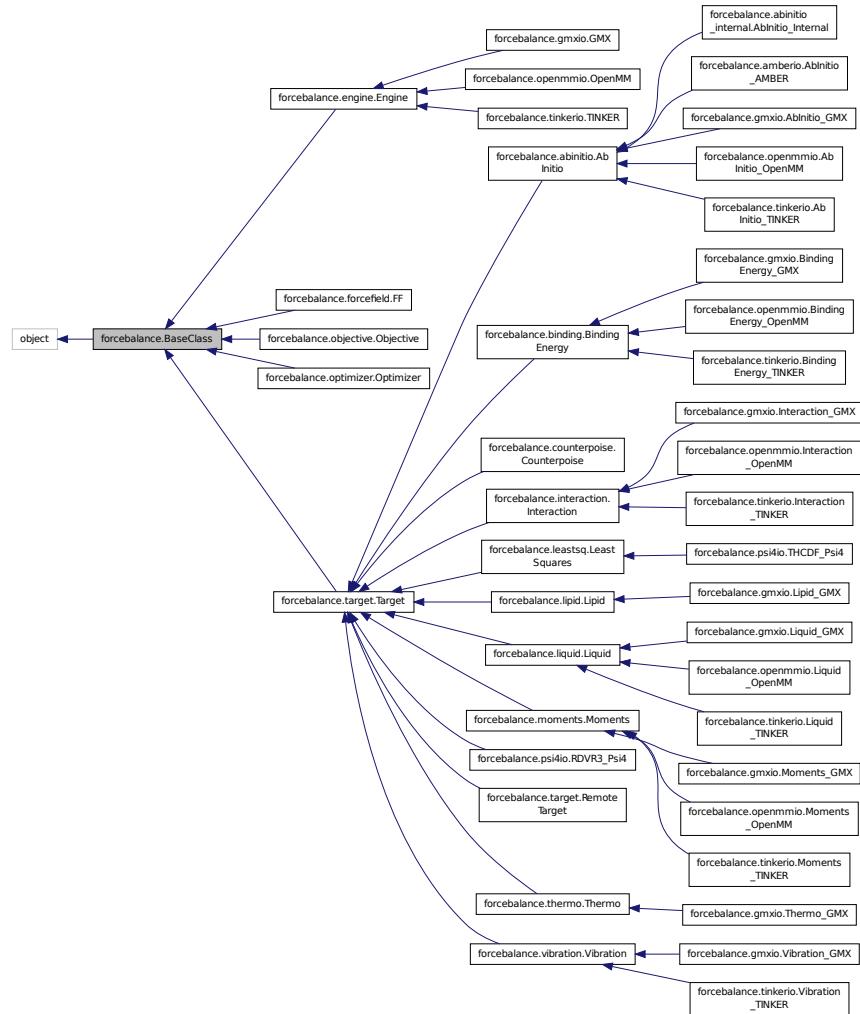
The documentation for this class was generated from the following file:

- [forcefield.py](#)

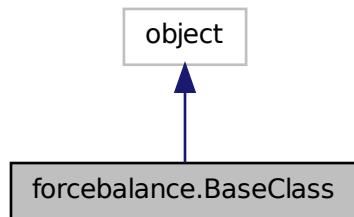
## 8.8 forcebalance.BaseClass Class Reference

Provides some nifty functions that are common to all ForceBalance classes.

Inheritance diagram for forcebalance.BaseClass:



Collaboration diagram for forcebalance.BaseClass:



## Public Member Functions

- def `__setattr__`
- def `__init__`
- def `set_option`

## Public Attributes

- `verbose_options`
- `PrintOptionDict`

### 8.8.1 Detailed Description

Provides some nifty functions that are common to all ForceBalance classes.

Definition at line 26 of file `__init__.py`.

### 8.8.2 Constructor & Destructor Documentation

`def forcebalance.BaseClass.__init__( self, options )` Definition at line 39 of file `__init__.py`.

### 8.8.3 Member Function Documentation

`def forcebalance.BaseClass.__setattr__( self, key, value )` Definition at line 28 of file `__init__.py`.

`def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False )` Definition at line 42 of file `__init__.py`.

### 8.8.4 Member Data Documentation

`forcebalance.BaseClass.PrintOptionDict` Definition at line 44 of file `__init__.py`.

`forcebalance.BaseClass.verbose_options` Definition at line 40 of file `__init__.py`.

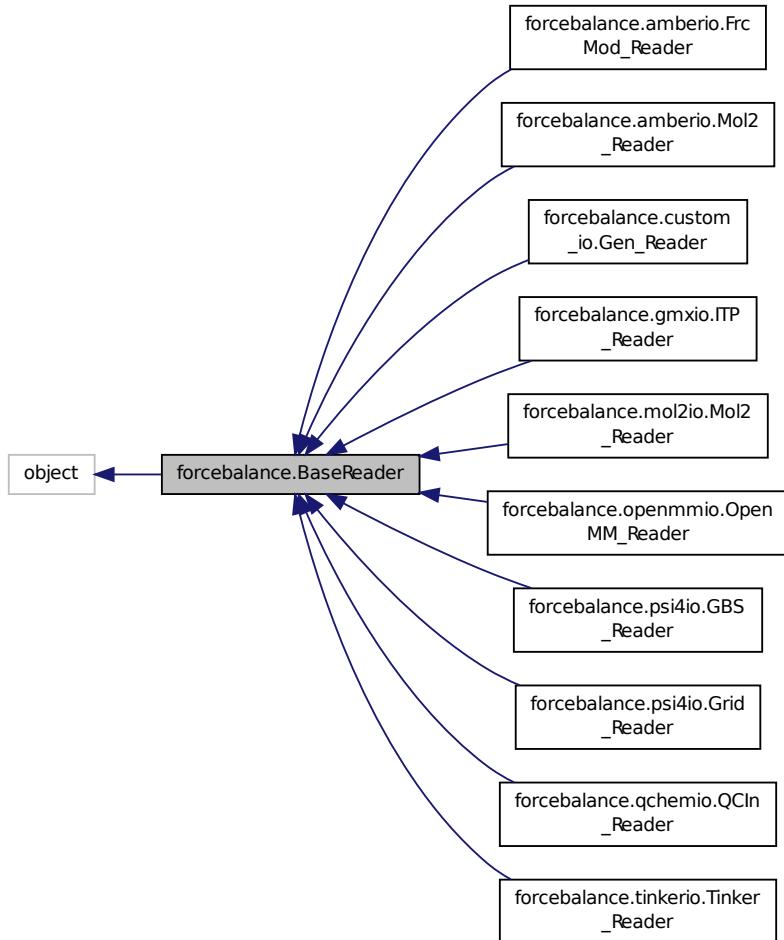
The documentation for this class was generated from the following file:

- `__init__.py`

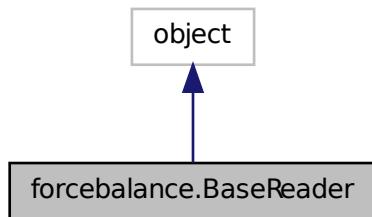
## 8.9 forcebalance.BaseReader Class Reference

The 'reader' class.

Inheritance diagram for forcebalance.BaseReader:



Collaboration diagram for forcebalance.BaseReader:



## Public Member Functions

- def `_init_`
- def `Split`
- def `Whites`
- def `feed`
- def `build_pid`

*Returns the parameter type (e.g.*

## Public Attributes

- `In`
- `itype`
- `suffix`
- `pdict`
- `adict`

*The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.*

- `molatom`

*The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.*

- `Molecules`
- `AtomTypes`

### 8.9.1 Detailed Description

The 'reader' class.

It serves two main functions:

1) When parsing a text force field file, the 'feed' method is called once for every line. Calling the 'feed' method stores the internal variables that are needed for making the unique parameter identifier.

2) The 'reader' also stores the 'pdict' dictionary, which is needed for building the matrix of rescaling factors. This is not needed for the XML force fields, so in XML force fields pdict is replaced with a string called "XML\_Override".

Definition at line 81 of file `__init__.py`.

### 8.9.2 Constructor & Destructor Documentation

`def forcebalance.BaseReader.__init__( self, fnm )` Definition at line 83 of file `__init__.py`.

### 8.9.3 Member Function Documentation

`def forcebalance.BaseReader.build_pid( self, pfd )` Returns the parameter type (e.g.

K in BONDSK) based on the current interaction type.

Both the 'pdict' dictionary (see `gmlio.pdict`) and the interaction type 'state' (here, BONDS) are needed to get the parameter type.

If, however, 'pdict' does not contain the ptype value, a suitable substitute is simply the field number.

Note that if the interaction type state is not set, then it defaults to the file name, so a generic parameter ID is 'filename.line\_num.field\_num'

Definition at line 124 of file `__init__.py`.

`def forcebalance.BaseReader.feed( self, line )` Definition at line 105 of file `__init__.py`.

`def forcebalance.BaseReader.Split( self, line )` Definition at line 99 of file `__init__.py`.

`def forcebalance.BaseReader.Whites( self, line )` Definition at line 102 of file `__init__.py`.

#### 8.9.4 Member Data Documentation

**forcebalance.BaseReader.adict** The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.

Definition at line 89 of file `__init__.py`.

**forcebalance.BaseReader.AtomTypes** Definition at line 97 of file `__init__.py`.

**forcebalance.BaseReader.itype** Definition at line 85 of file `__init__.py`.

**forcebalance.BaseReader.In** Definition at line 84 of file `__init__.py`.

**forcebalance.BaseReader.molatom** The mapping of (molecule name) to a dictionary of of atom types for the atoms in that residue.

`self.moleculerdict = OrderedDict()` The listing of 'RES:ATOMNAMES' for atom names in the line This is obviously a placeholder.

Definition at line 94 of file `__init__.py`.

**forcebalance.BaseReader.Molecules** Definition at line 96 of file `__init__.py`.

**forcebalance.BaseReader.pdict** Definition at line 87 of file `__init__.py`.

**forcebalance.BaseReader.suffix** Definition at line 86 of file `__init__.py`.

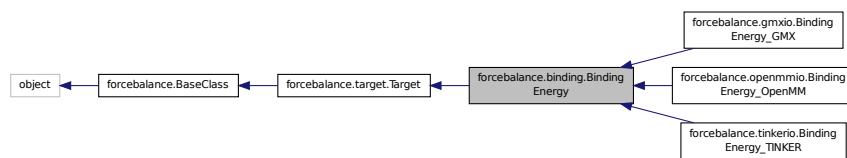
The documentation for this class was generated from the following file:

- [\\_\\_init\\_\\_.py](#)

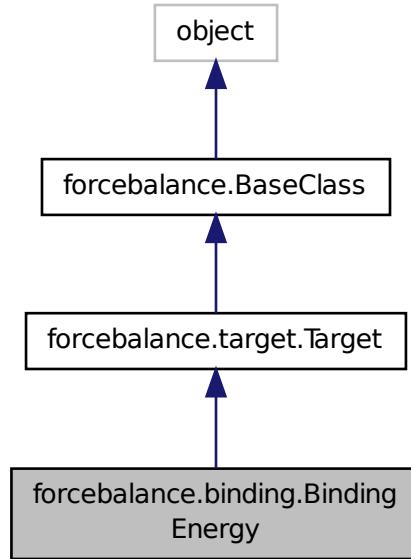
## 8.10 forcebalance.binding.BindingEnergy Class Reference

Improved subclass of Target for fitting force fields to binding energies.

Inheritance diagram for `forcebalance.binding.BindingEnergy`:



Collaboration diagram for forcebalance.binding.BindingEnergy:



## Public Member Functions

- def `__init__`
- def `system_driver`
- def `indicate`
- def `get`
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*
- def `get_H`

*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`

*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`

*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`

*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*

- def **absrd**  
*Supply the correct directory specified by user's "read" option.*
- def **maxrd**  
*Supply the latest existing temp-directory containing valid data.*
- def **meta.\_indicate**  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def **meta.get**  
*Wrapper around the get function.*
- def **submit\_jobs**
- def **stage**  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def **wq.complete**  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def **printcool.table**  
*Print target information in an organized table format.*
- def **\_\_setattr\_\_**
- def **set\_option**

## Public Attributes

- **inter\_opts**  
**engines**  
*Build keyword dictionaries to pass to engine.*
- **PrintDict**
- **RMSDDict**
- **rmsd\_part**
- **energy\_part**
- **objective**
- **rd**  
*Root directory of the whole project.*
- **pgrad**  
*Iteration where we turn on zero-gradient skipping.*
- **tempbase**  
*Relative directory of target.*
- **tempdir**
- **rundir**  
*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- **FF**  
*Need the forcefield (here for now)*
- **xct**  
*Counts how often the objective function was computed.*
- **gct**  
*Counts how often the gradient was computed.*
- **hct**  
*Counts how often the Hessian was computed.*
- **read\_indicate**  
*Whether to read indicate.log from file when restarting an aborted run.*
- **write\_indicate**

*Whether to write indicate.log at every iteration (true for all but remote.)*

- [read\\_objective](#)

*Whether to read objective.p from file when restarting an aborted run.*

- [write\\_objective](#)

*Whether to write objective.p at every iteration (true for all but remote.)*

- [verbose\\_options](#)
- [PrintOptionDict](#)

### 8.10.1 Detailed Description

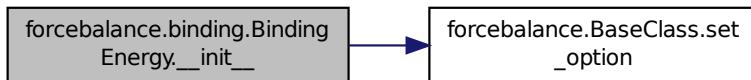
Improved subclass of Target for fitting force fields to binding energies.

Definition at line 122 of file binding.py.

### 8.10.2 Constructor & Destructor Documentation

**def forcebalance.binding.BindingEnergy.\_\_init\_\_ ( self, options, tgt\_opts, forcefield )** Definition at line 125 of file binding.py.

Here is the call graph for this function:



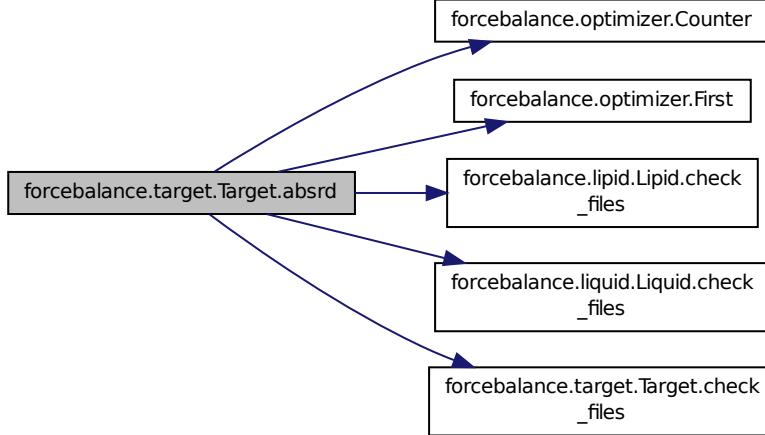
### 8.10.3 Member Function Documentation

**def forcebalance.BaseClass.\_\_setattr\_\_ ( self, key, value ) [inherited]** Definition at line 28 of file \_\_init\_\_.py.

**def forcebalance.target.Target.absrd ( self, inum = None ) [inherited]** Supply the correct directory specified by user's "read" option.

Definition at line 393 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.check\_files( self, there ) [inherited]** Check this directory for the presence of readable files when the 'read' option is set.

Definition at line 364 of file target.py.

**def forcebalance.binding.BindingEnergy.get( self, mvals, AGrad = False, AHess = False )** Definition at line 184 of file binding.py.

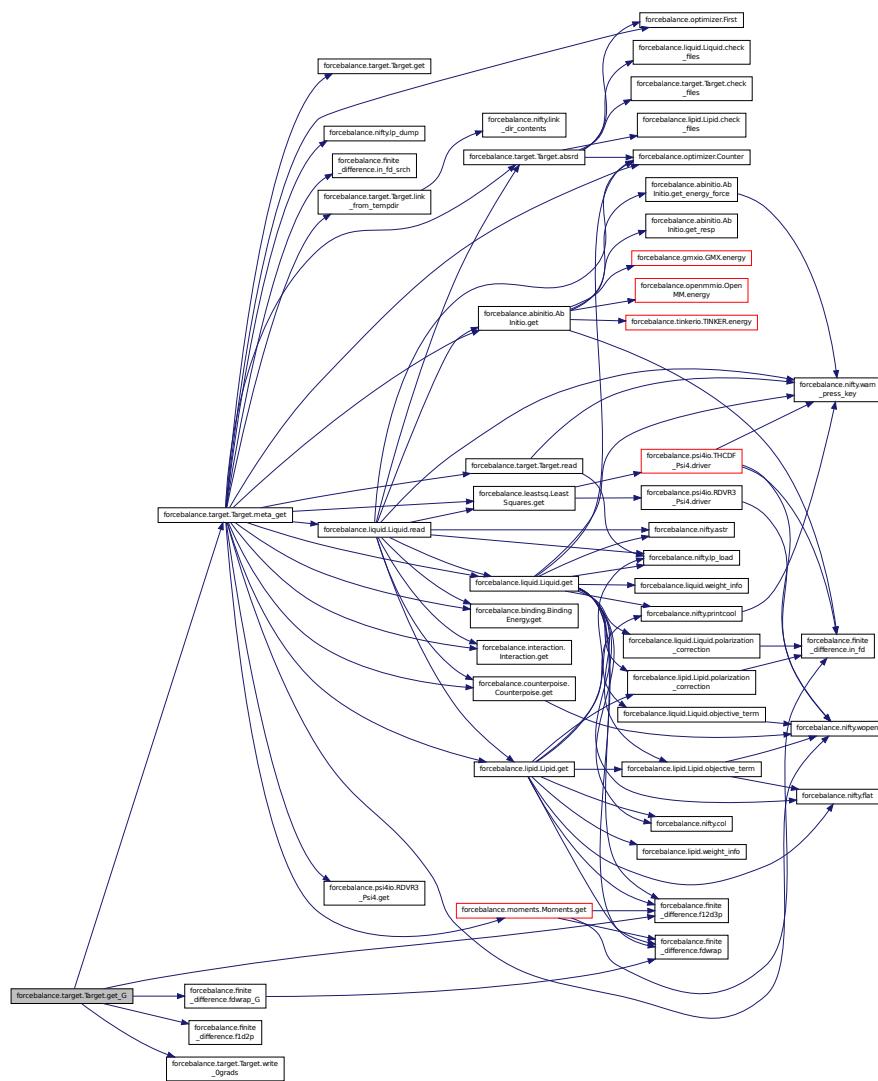
**def forcebalance.target.Target.get\_G( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



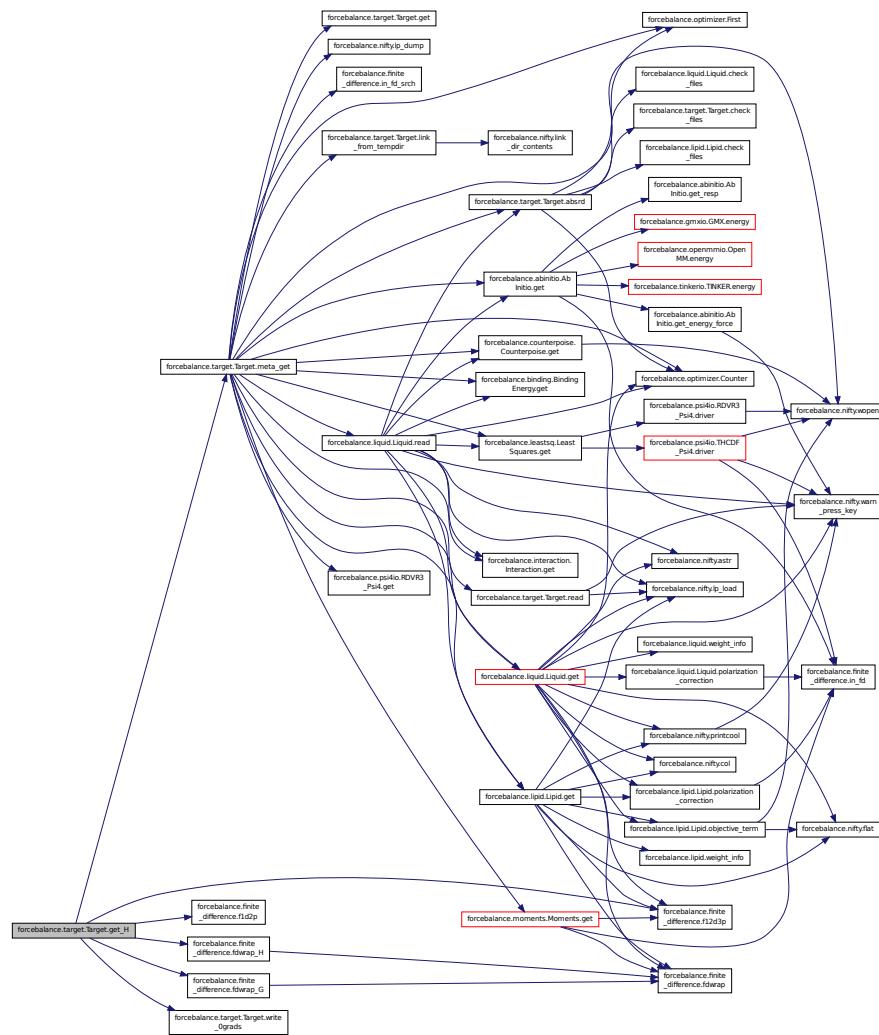
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

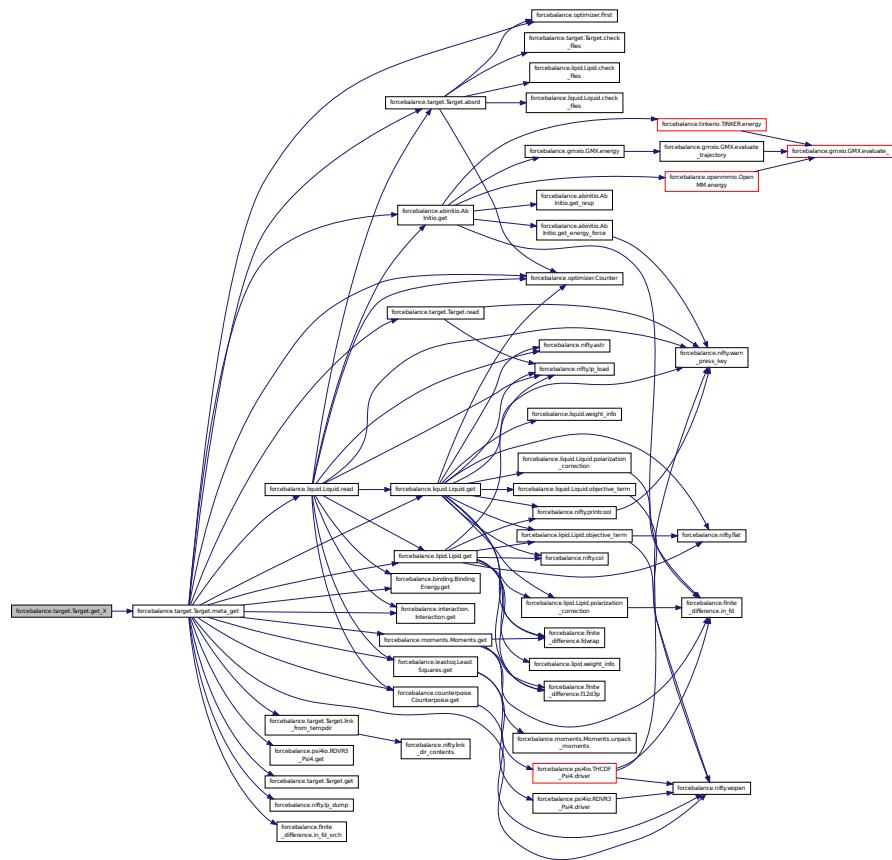
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

Here is the call graph for this function:



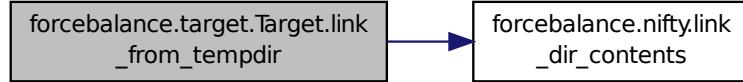
**def forcebalance.binding.BindingEnergy.indicate ( self )** Definition at line 178 of file binding.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

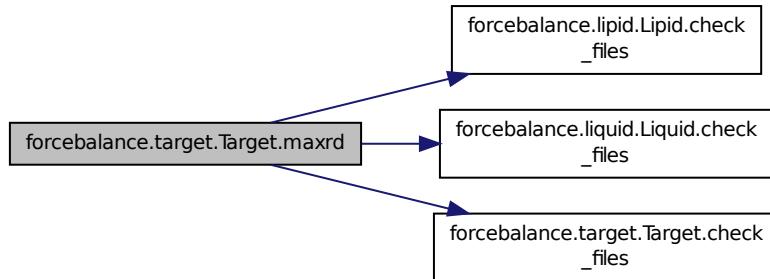
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

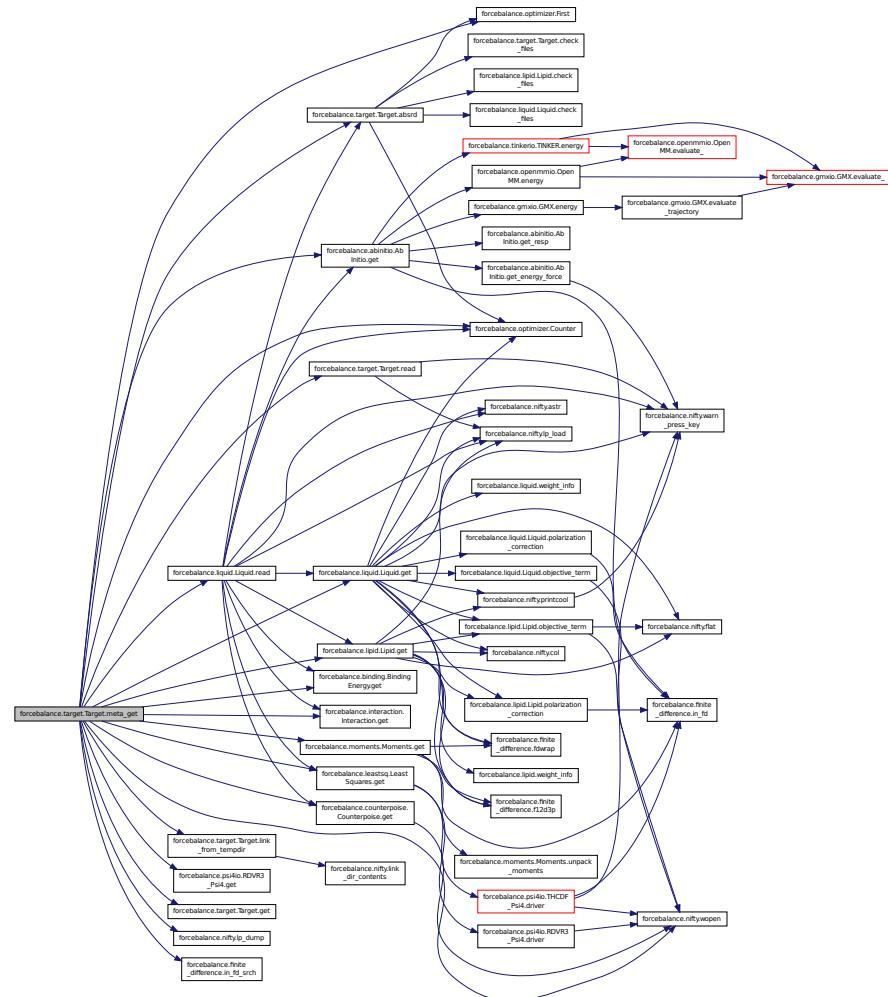


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

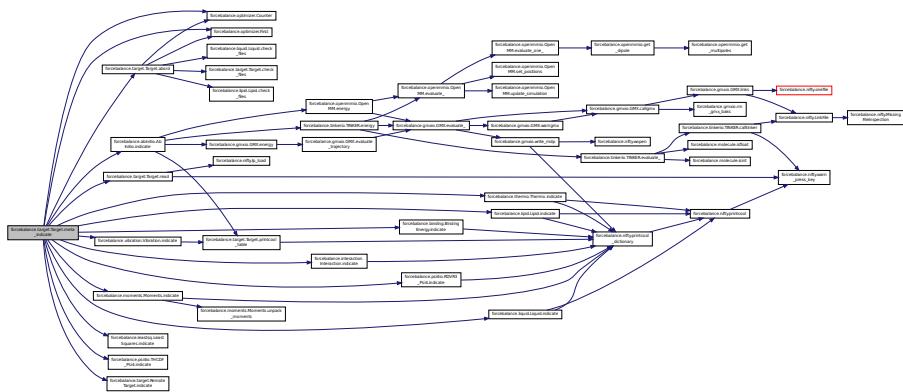
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

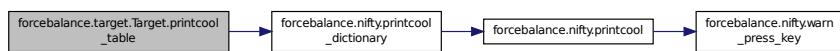
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

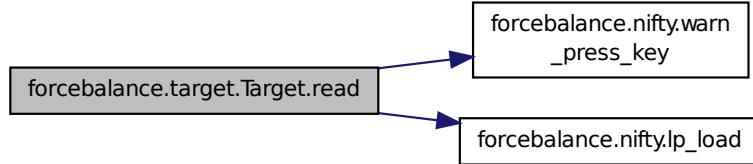
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

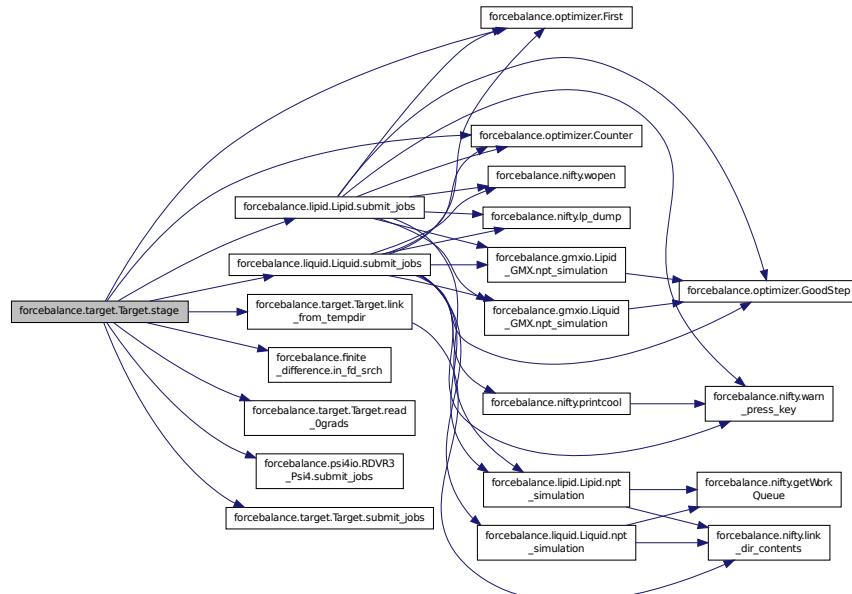
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



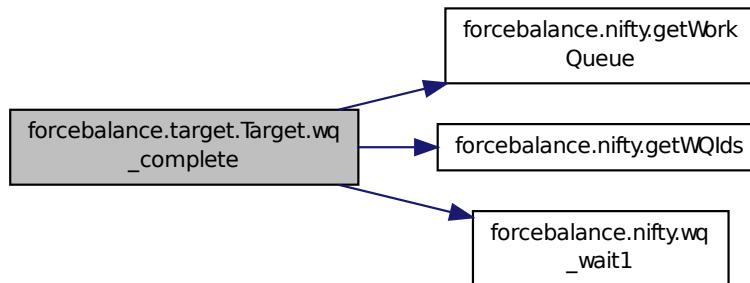
```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.binding.BindingEnergy.system_driver ( self, sysname ) Definition at line 174 of file binding.py.
```

```
def forcebalance.target.Target.wq_complete ( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.10.4 Member Data Documentation

**forcebalance.binding.BindingEnergy.energy\_part** Definition at line 228 of file binding.py.

**forcebalance.binding.BindingEnergy.engines** Build keyword dictionaries to pass to engine.

Create engine objects.

Definition at line 165 of file binding.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.binding.BindingEnergy.inter\_opts** Definition at line 128 of file binding.py.

**forcebalance.binding.BindingEnergy.objective** Definition at line 252 of file binding.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.binding.BindingEnergy.PrintDict** Definition at line 186 of file binding.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.binding.BindingEnergy.rmsd\_part** Definition at line 226 of file binding.py.

**forcebalance.binding.BindingEnergy.RMSDDict** Definition at line 187 of file binding.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number. The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase [inherited]** Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir [inherited]** Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate [inherited]** Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective [inherited]** Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 162 of file target.py.

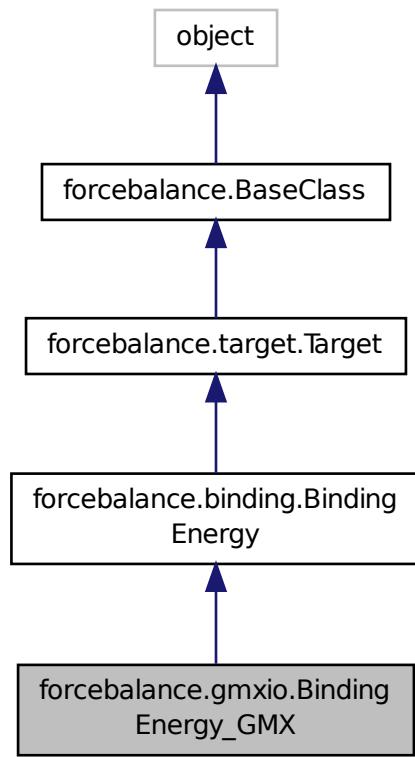
The documentation for this class was generated from the following file:

- [binding.py](#)

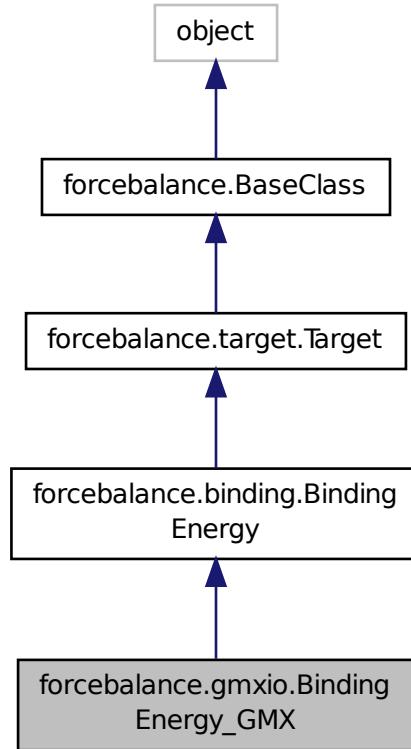
## 8.11 forcebalance.gmxio.BindingEnergy\_GMX Class Reference

Binding energy matching using Gromacs.

Inheritance diagram for forcebalance.gmxio.BindingEnergy\_GMX:



Collaboration diagram for forcebalance.gmxio.BindingEnergy\_GMX:



### Public Member Functions

- def `_init_`
- def `system_driver`
- def `indicate`
- def `get`
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*
- def `get_H`

*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`

*Back up the temporary directory if desired, delete it and then create a new one.*

- def [check\\_files](#)

*Check this directory for the presence of readable files when the 'read' option is set.*

- def [read](#)

*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*

- def [absrd](#)

*Supply the correct directory specified by user's "read" option.*

- def [maxrd](#)

*Supply the latest existing temp-directory containing valid data.*

- def [meta\\_indicate](#)

*Wrap around the indicate function, so it can print to screen and also to a file.*

- def [meta\\_get](#)

*Wrapper around the get function.*

- def [submit\\_jobs](#)

- def [stage](#)

*Stages the directory for the target, and then launches Work Queue processes if any.*

- def [wq\\_complete](#)

*This method determines whether the Work Queue tasks for the current target have completed.*

- def [printcool\\_table](#)

*Print target information in an organized table format.*

- def [\\_\\_setattr\\_\\_](#)

- def [set\\_option](#)

## Public Attributes

- [engine\\_](#)

- [inter\\_opts](#)

- [engines](#)

*Build keyword dictionaries to pass to engine.*

- [PrintDict](#)

- [RMSDDict](#)

- [rmsd\\_part](#)

- [energy\\_part](#)

- [objective](#)

- [rd](#)

*Root directory of the whole project.*

- [pgrad](#)

*Iteration where we turn on zero-gradient skipping.*

- [tempbase](#)

*Relative directory of target.*

- [tempdir](#)

- [rundir](#)

*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*

- [FF](#)

*Need the forcefield (here for now)*

- [xct](#)

*Counts how often the objective function was computed.*

- [gct](#)

- `hct`  
Counts how often the gradient was computed.
- `read_indicate`  
Whether to read indicate.log from file when restarting an aborted run.
- `write_indicate`  
Whether to write indicate.log at every iteration (true for all but remote.)
- `read_objective`  
Whether to read objective.p from file when restarting an aborted run.
- `write_objective`  
Whether to write objective.p at every iteration (true for all but remote.)
- `verbose_options`
- `PrintOptionDict`

### 8.11.1 Detailed Description

Binding energy matching using Gromacs.

Definition at line 1461 of file gmxio.py.

### 8.11.2 Constructor & Destructor Documentation

```
def forcebalance.gmxio.BindingEnergy_GMX.__init__ ( self, options, tgt_opts, forcefield )
```

Definition at line 1461 of file gmxio.py.

### 8.11.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited]
```

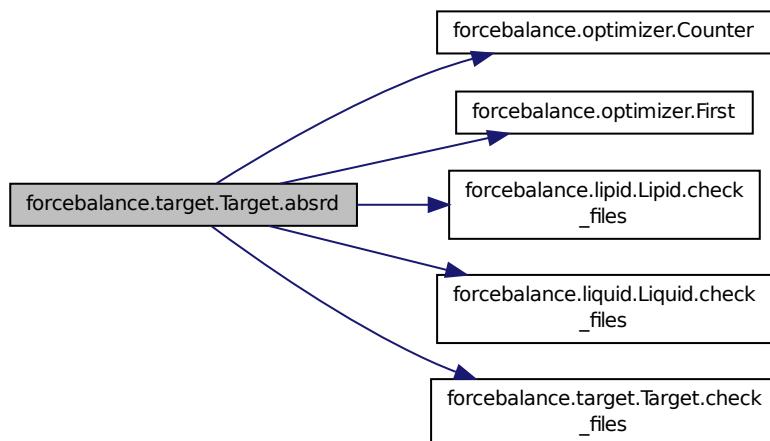
Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited]
```

Supply the correct directory specified by user's "read" option.

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.check_files( self, there ) [inherited] Check this directory for the presence  
of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.binding.BindingEnergy.get ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 184 of file binding.py.
```

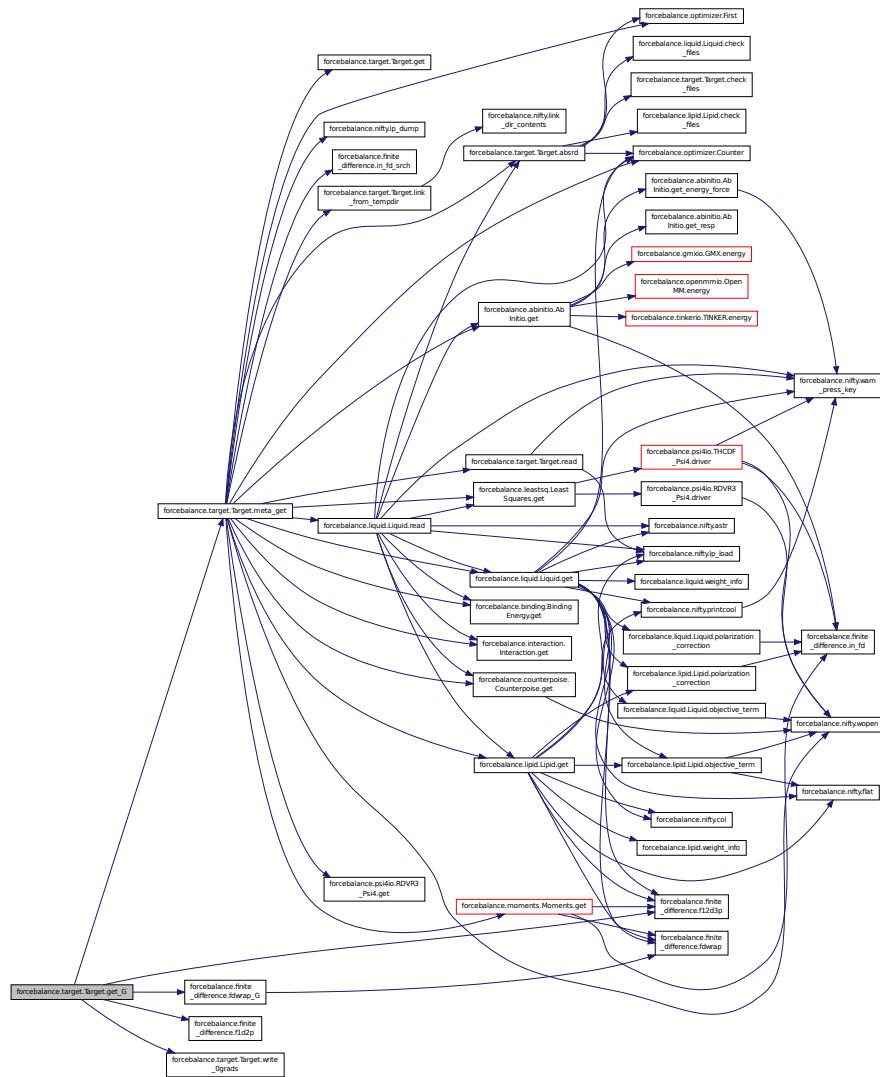
```
def forcebalance.target.Target.get_G( self, mvals = None ) [inherited] Computes the objective function  
contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1.pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



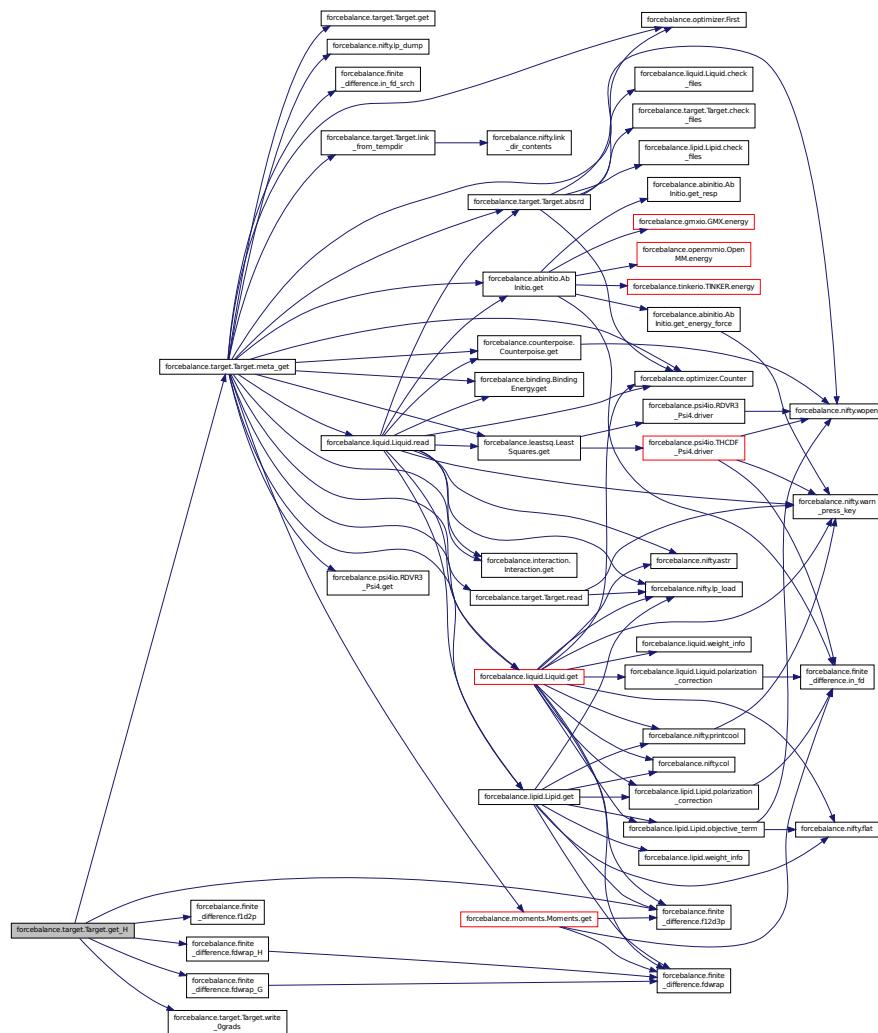
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the `fd2.pids` and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

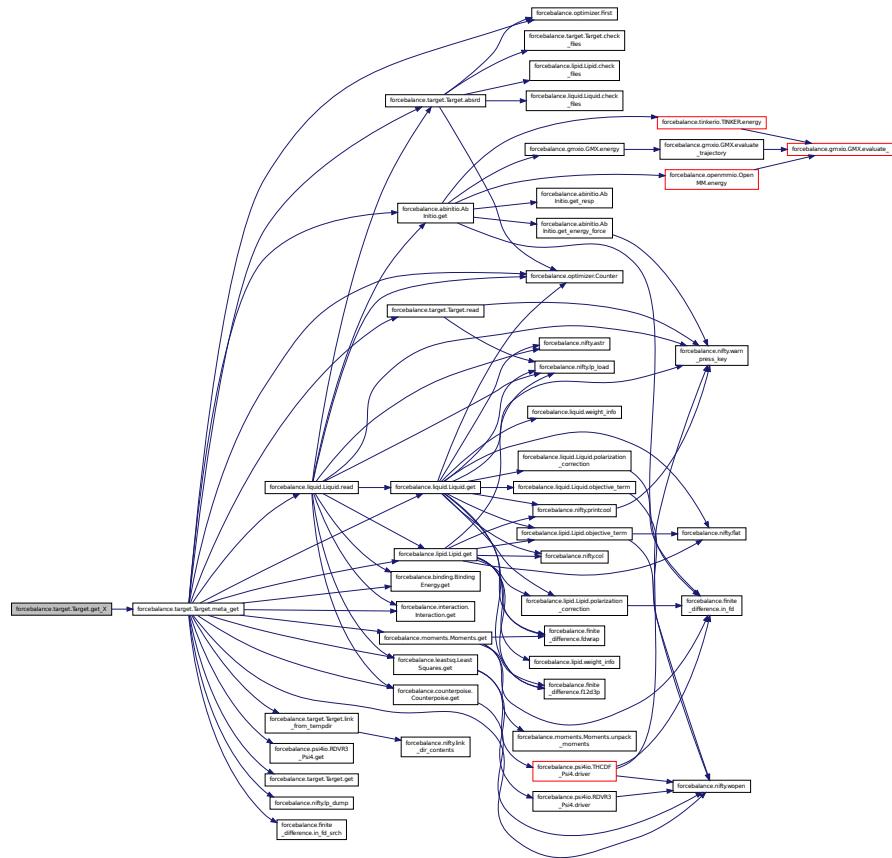
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

Here is the call graph for this function:



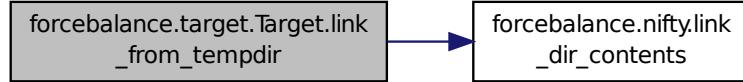
**def forcebalance.binding.BindingEnergy.indicate ( self ) [inherited]** Definition at line 178 of file binding.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

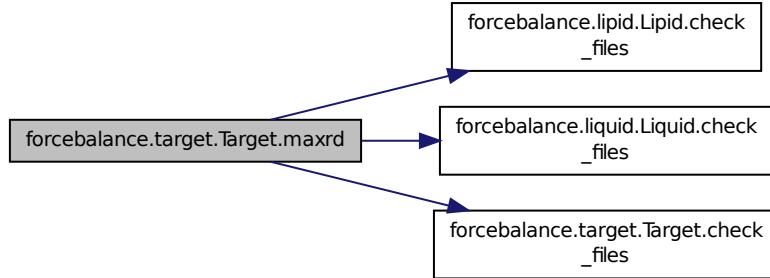
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

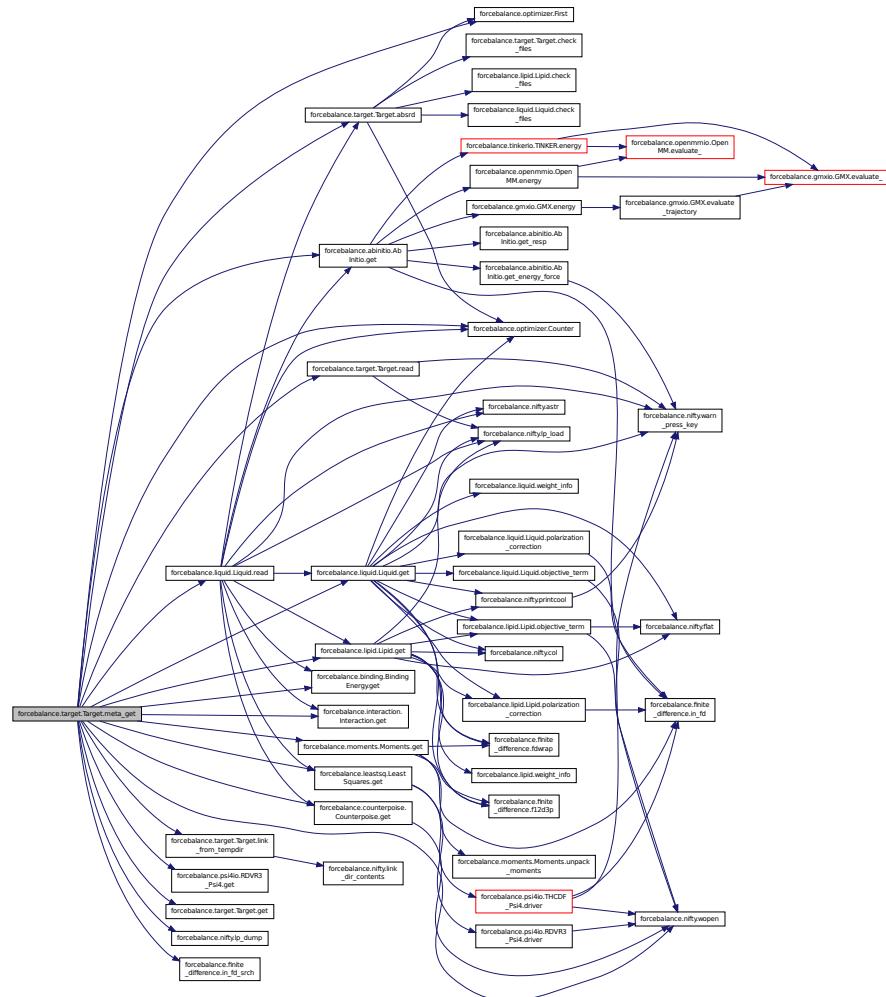


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

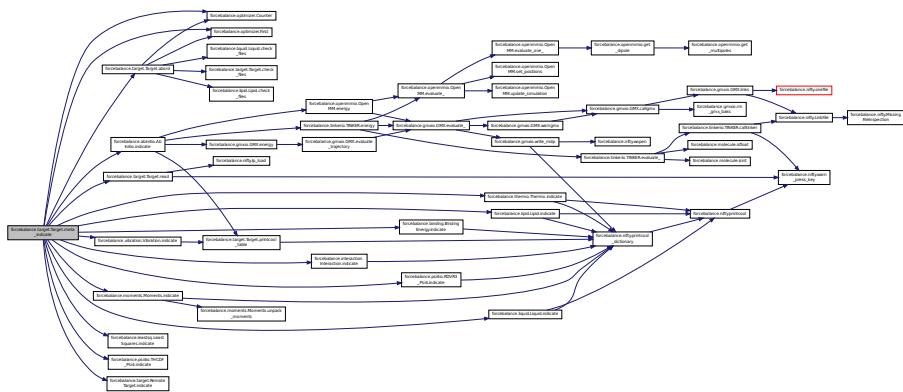
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

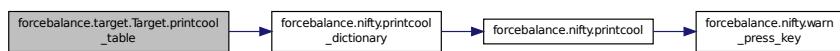
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

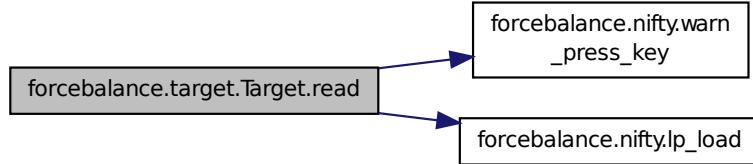
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

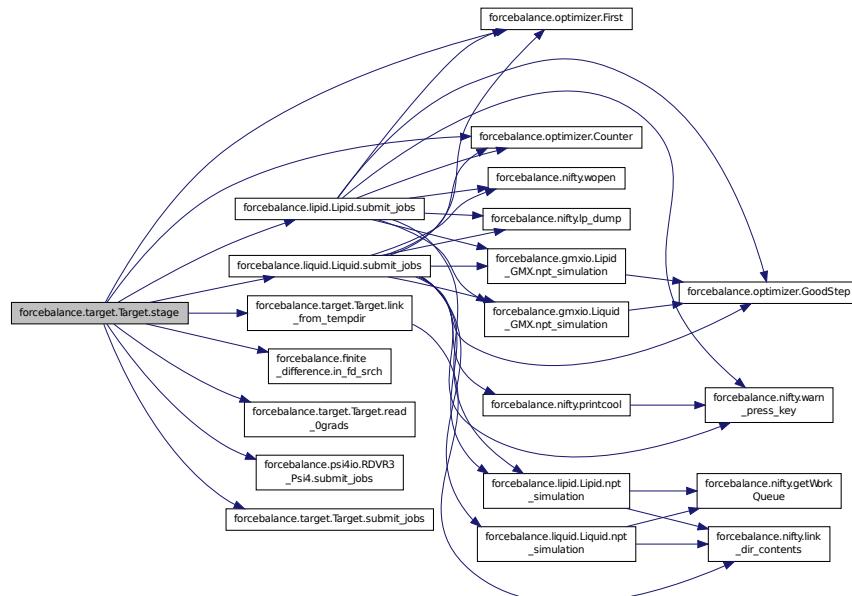
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



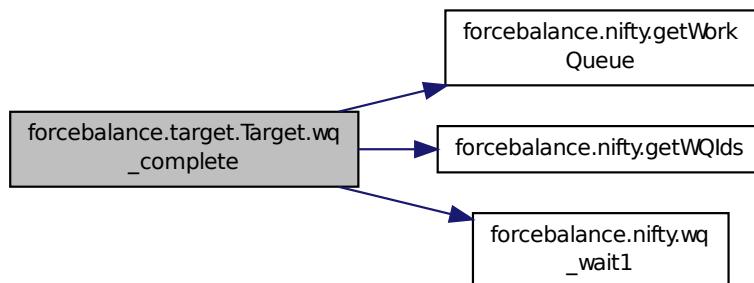
```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.binding.BindingEnergy.system_driver ( self, sysname ) [inherited] Definition at line 174 of file binding.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.11.4 Member Data Documentation

**forcebalance.binding.BindingEnergy.energy\_part [inherited]** Definition at line 228 of file binding.py.

**forcebalance.gmxio.BindingEnergy\_GMX.engine\_** Definition at line 1463 of file gmxio.py.

**forcebalance.binding.BindingEnergy.engines [inherited]** Build keyword dictionaries to pass to engine.

Create engine objects.

Definition at line 165 of file binding.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.binding.BindingEnergy.inter\_opts [inherited]** Definition at line 128 of file binding.py.

**forcebalance.binding.BindingEnergy.objective [inherited]** Definition at line 252 of file binding.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.binding.BindingEnergy.PrintDict [inherited]** Definition at line 186 of file binding.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.binding.BindingEnergy.rmsd.part** [inherited] Definition at line 226 of file binding.py.

**forcebalance.binding.BindingEnergy.RMSDDict** [inherited] Definition at line 187 of file binding.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number  
The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate** [inherited] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

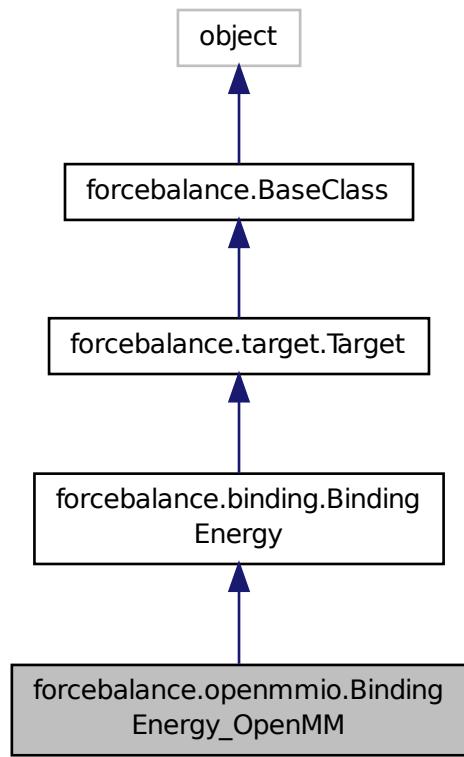
The documentation for this class was generated from the following file:

- [gmxio.py](#)

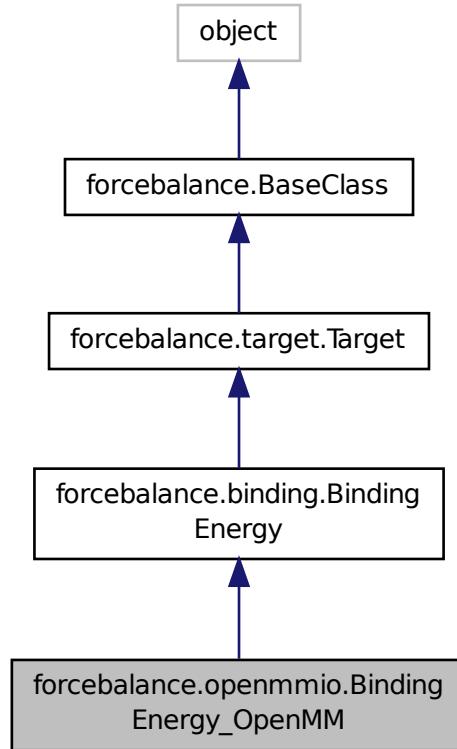
## 8.12 forcebalance.openmmio.BindingEnergy\_OpenMM Class Reference

Binding energy matching using [OpenMM](#).

Inheritance diagram for forcebalance.openmmio.BindingEnergy\_OpenMM:



Collaboration diagram for forcebalance.openmmio.BindingEnergy\_OpenMM:



### Public Member Functions

- def `_init_`
- def `system_driver`
- def `indicate`
- def `get`
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*
- def `get_H`

*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`

*Back up the temporary directory if desired, delete it and then create a new one.*

- def [check\\_files](#)

*Check this directory for the presence of readable files when the 'read' option is set.*

- def [read](#)

*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*

- def [absrd](#)

*Supply the correct directory specified by user's "read" option.*

- def [maxrd](#)

*Supply the latest existing temp-directory containing valid data.*

- def [meta\\_indicate](#)

*Wrap around the indicate function, so it can print to screen and also to a file.*

- def [meta\\_get](#)

*Wrapper around the get function.*

- def [submit\\_jobs](#)

- def [stage](#)

*Stages the directory for the target, and then launches Work Queue processes if any.*

- def [wq\\_complete](#)

*This method determines whether the Work Queue tasks for the current target have completed.*

- def [printcool\\_table](#)

*Print target information in an organized table format.*

- def [\\_\\_setattr\\_\\_](#)

- def [set\\_option](#)

## Public Attributes

- [engine\\_](#)

- [inter\\_opts](#)

- [engines](#)

*Build keyword dictionaries to pass to engine.*

- [PrintDict](#)

- [RMSDDict](#)

- [rmsd\\_part](#)

- [energy\\_part](#)

- [objective](#)

- [rd](#)

*Root directory of the whole project.*

- [pgrad](#)

*Iteration where we turn on zero-gradient skipping.*

- [tempbase](#)

*Relative directory of target.*

- [tempdir](#)

- [rundir](#)

*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*

- [FF](#)

*Need the forcefield (here for now)*

- [xct](#)

*Counts how often the objective function was computed.*

- [gct](#)

- `hct`  
Counts how often the gradient was computed.
- `read_indicate`  
Whether to read indicate.log from file when restarting an aborted run.
- `write_indicate`  
Whether to write indicate.log at every iteration (true for all but remote.)
- `read_objective`  
Whether to read objective.p from file when restarting an aborted run.
- `write_objective`  
Whether to write objective.p at every iteration (true for all but remote.)
- `verbose_options`
- `PrintOptionDict`

### 8.12.1 Detailed Description

Binding energy matching using [OpenMM](#).

Definition at line 1179 of file openmmio.py.

### 8.12.2 Constructor & Destructor Documentation

`def forcebalance.openmmio.BindingEnergy_OpenMM.__init__( self, options, tgt_opts, forcefield )` Definition at line 1181 of file openmmio.py.

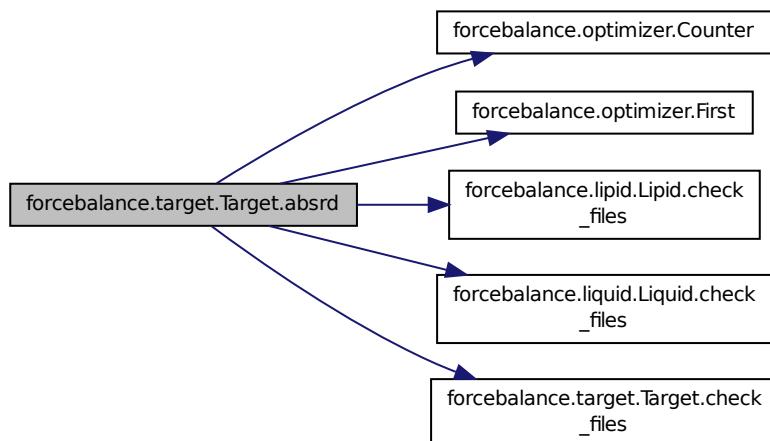
### 8.12.3 Member Function Documentation

`def forcebalance.BaseClass.__setattr__( self, key, value ) [inherited]` Definition at line 28 of file \_\_init\_\_.py.

`def forcebalance.target.Target.absrd( self, inum = None ) [inherited]` Supply the correct directory specified by user's "read" option.

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.check_files( self, there ) [inherited] Check this directory for the presence  
of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.binding.BindingEnergy.get ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 184 of file binding.py.
```

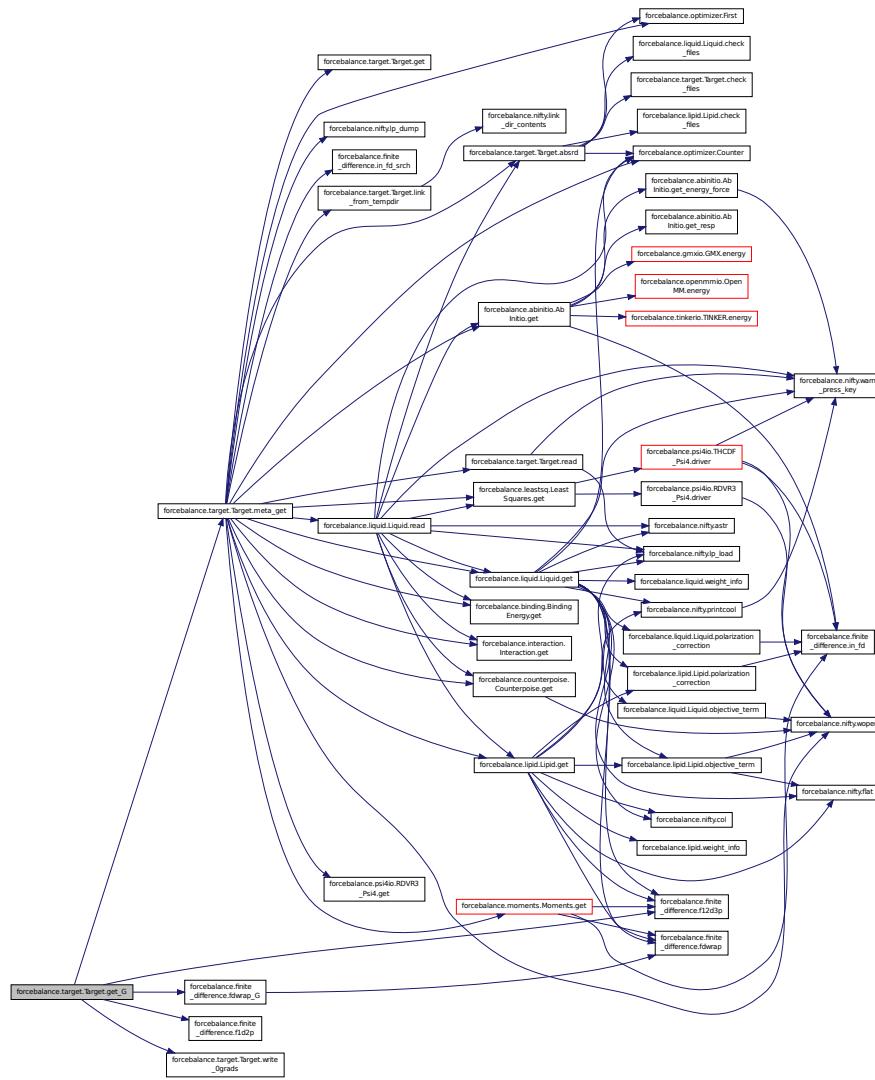
```
def forcebalance.target.Target.get_G( self, mvals = None ) [inherited] Computes the objective function  
contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1.pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



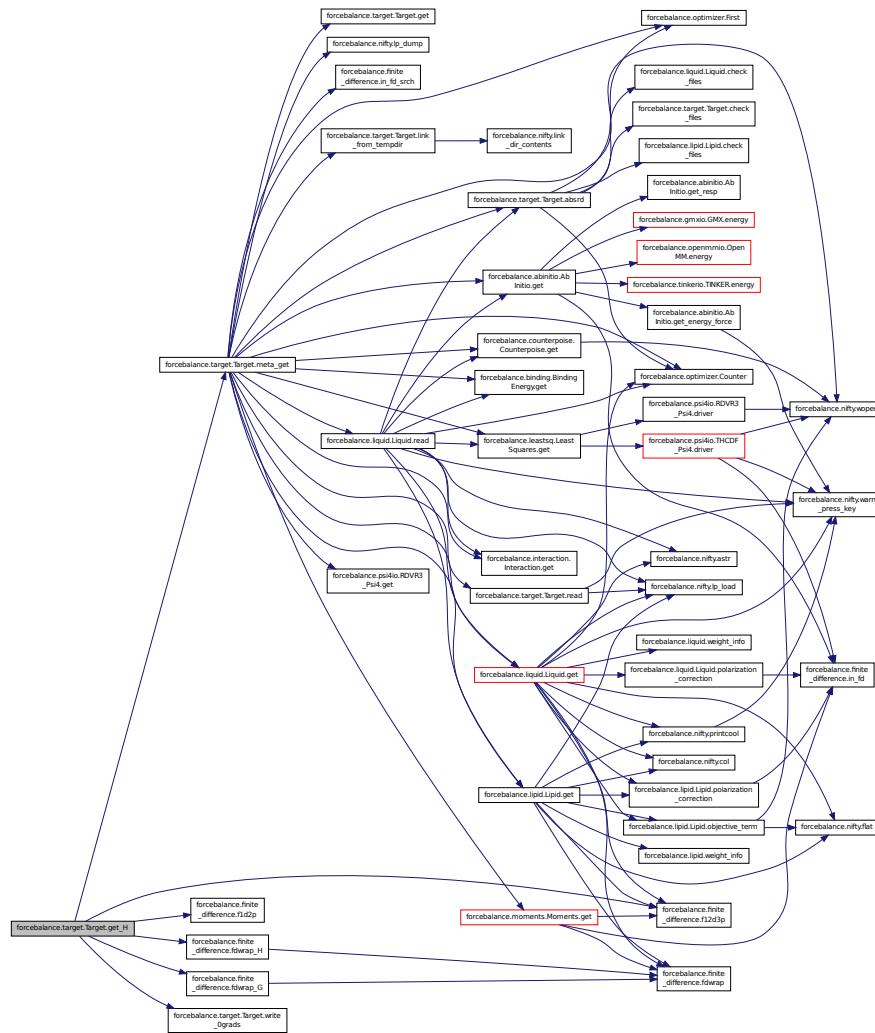
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

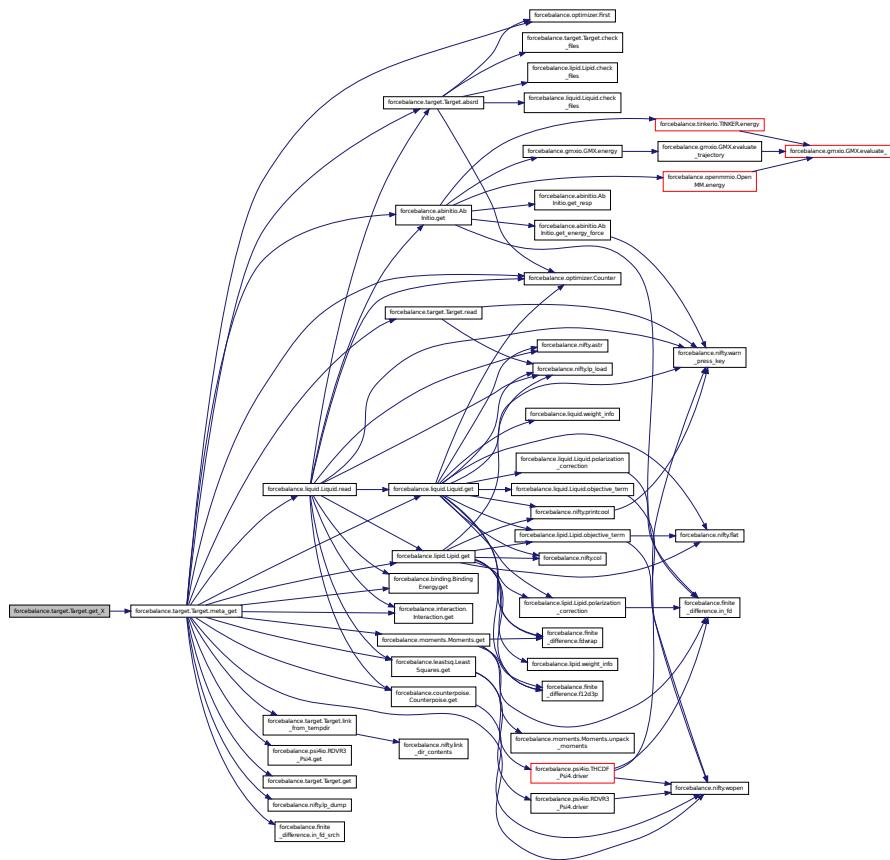
Here is the call graph for this function:



**def forcebalance.target.Target.get.X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

Here is the call graph for this function:



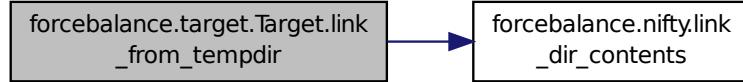
```
def forcebalance.binding.BindingEnergy.indicate( self ) [inherited] Definition at line 178 of file binding.py.
```

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

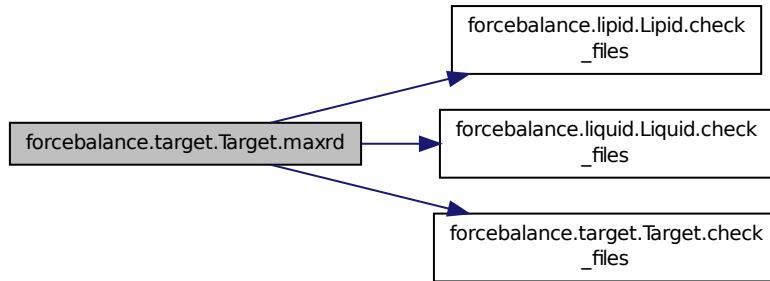
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

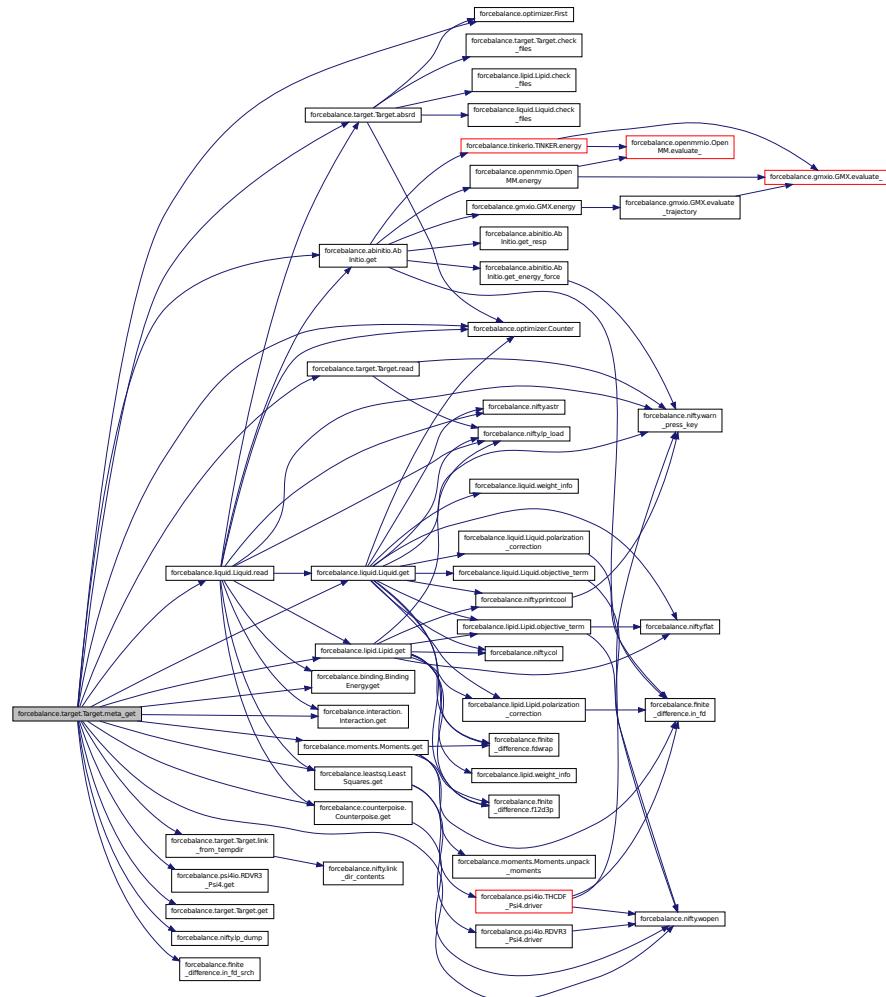


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

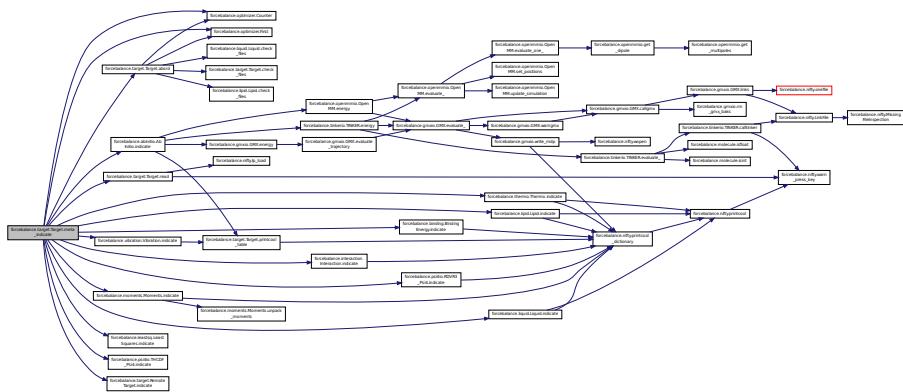
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

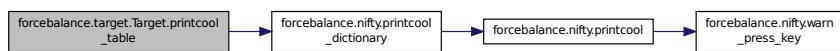
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

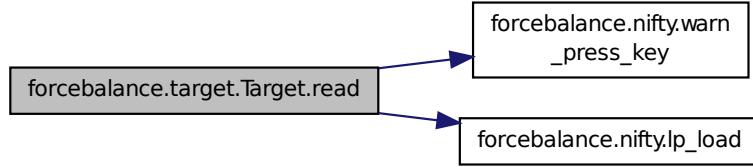
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

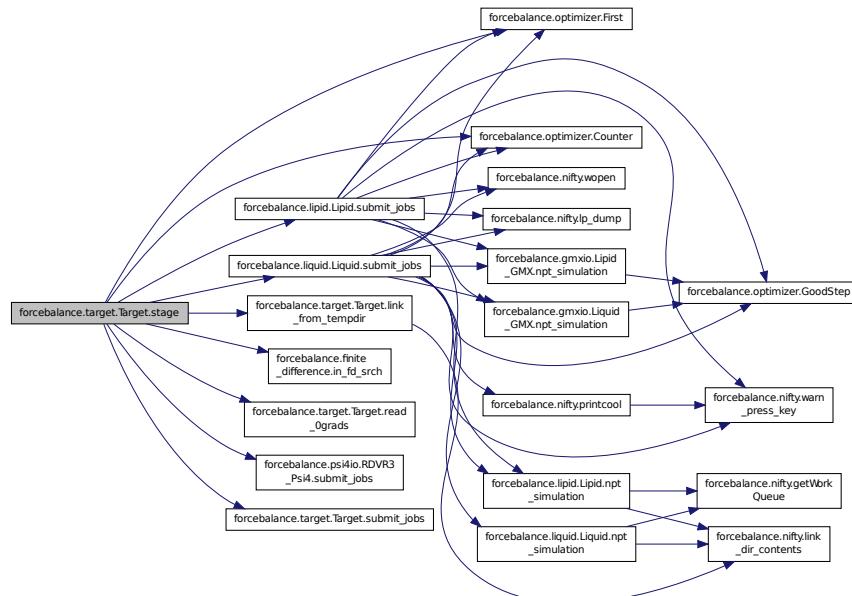
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



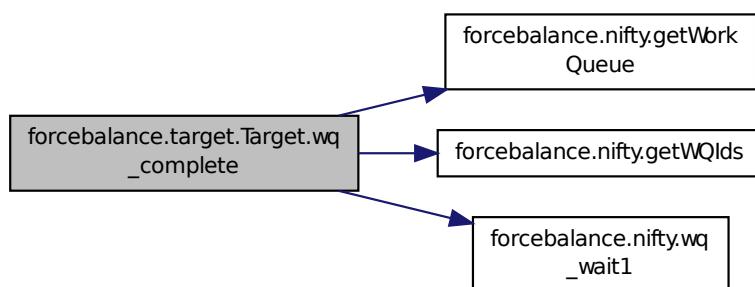
```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.binding.BindingEnergy.system_driver ( self, sysname ) [inherited] Definition at line 174 of file binding.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.12.4 Member Data Documentation

**forcebalance.binding.BindingEnergy.energy\_part [inherited]** Definition at line 228 of file binding.py.

**forcebalance.openmmio.BindingEnergy\_OpenMM.engine\_** Definition at line 1182 of file openmmio.py.

**forcebalance.binding.BindingEnergy.engines [inherited]** Build keyword dictionaries to pass to engine.

Create engine objects.

Definition at line 165 of file binding.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.binding.BindingEnergy.inter\_opts [inherited]** Definition at line 128 of file binding.py.

**forcebalance.binding.BindingEnergy.objective [inherited]** Definition at line 252 of file binding.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.binding.BindingEnergy.PrintDict [inherited]** Definition at line 186 of file binding.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.binding.BindingEnergy.rmsd.part** [inherited] Definition at line 226 of file binding.py.

**forcebalance.binding.BindingEnergy.RMSDDict** [inherited] Definition at line 187 of file binding.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number  
The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate** [inherited] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

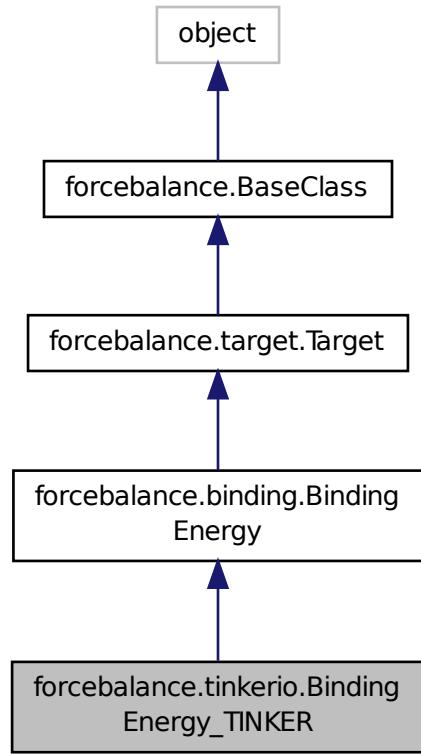
The documentation for this class was generated from the following file:

- [openmmio.py](#)

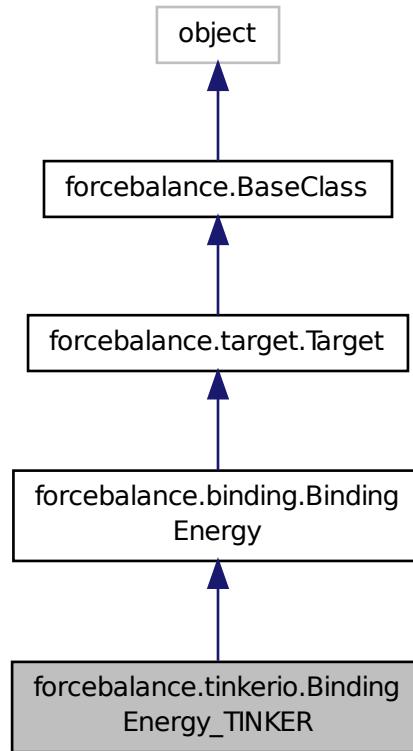
## 8.13 forcebalance.tinkerio.BindingEnergy\_TINKER Class Reference

Binding energy matching using [TINKER](#).

Inheritance diagram for forcebalance.tinkerio.BindingEnergy\_TINKER:



Collaboration diagram for forcebalance.tinkerio.BindingEnergy\_TINKER:



### Public Member Functions

- def `_init_`
- def `system_driver`
- def `indicate`
- def `get`
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*

- def `read_Ograds`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*

- def `write_Ograds`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*

- def `get_G`

*Computes the objective function contribution and its gradient.*

- def `get_H`

*Computes the objective function contribution and its gradient / Hessian.*

- def `link_from_tempdir`

- def `refresh_temp_directory`

*Back up the temporary directory if desired, delete it and then create a new one.*

- def [check\\_files](#)

*Check this directory for the presence of readable files when the 'read' option is set.*

- def [read](#)

*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*

- def [absrd](#)

*Supply the correct directory specified by user's "read" option.*

- def [maxrd](#)

*Supply the latest existing temp-directory containing valid data.*

- def [meta\\_indicate](#)

*Wrap around the indicate function, so it can print to screen and also to a file.*

- def [meta\\_get](#)

*Wrapper around the get function.*

- def [submit\\_jobs](#)

- def [stage](#)

*Stages the directory for the target, and then launches Work Queue processes if any.*

- def [wq\\_complete](#)

*This method determines whether the Work Queue tasks for the current target have completed.*

- def [printcool\\_table](#)

*Print target information in an organized table format.*

- def [\\_\\_setattr\\_\\_](#)

- def [set\\_option](#)

## Public Attributes

- [engine\\_](#)

- [inter\\_opts](#)

- [engines](#)

*Build keyword dictionaries to pass to engine.*

- [PrintDict](#)

- [RMSDDict](#)

- [rmsd\\_part](#)

- [energy\\_part](#)

- [objective](#)

- [rd](#)

*Root directory of the whole project.*

- [pgrad](#)

*Iteration where we turn on zero-gradient skipping.*

- [tempbase](#)

*Relative directory of target.*

- [tempdir](#)

- [rundir](#)

*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*

- [FF](#)

*Need the forcefield (here for now)*

- [xct](#)

*Counts how often the objective function was computed.*

- [gct](#)

- `hct`  
Counts how often the gradient was computed.
- `read_indicate`  
Whether to read `indicate.log` from file when restarting an aborted run.
- `write_indicate`  
Whether to write `indicate.log` at every iteration (true for all but remote.)
- `read_objective`  
Whether to read `objective.p` from file when restarting an aborted run.
- `write_objective`  
Whether to write `objective.p` at every iteration (true for all but remote.)
- `verbose_options`
- `PrintOptionDict`

### 8.13.1 Detailed Description

Binding energy matching using [TINKER](#).

Definition at line 1078 of file `tinkerio.py`.

### 8.13.2 Constructor & Destructor Documentation

```
def forcebalance.tinkerio.BindingEnergy_TINKER.__init__( self, options, tgt_opts, forcefield )
```

Definition at line 1079 of file `tinkerio.py`.

### 8.13.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__( self, key, value ) [inherited]
```

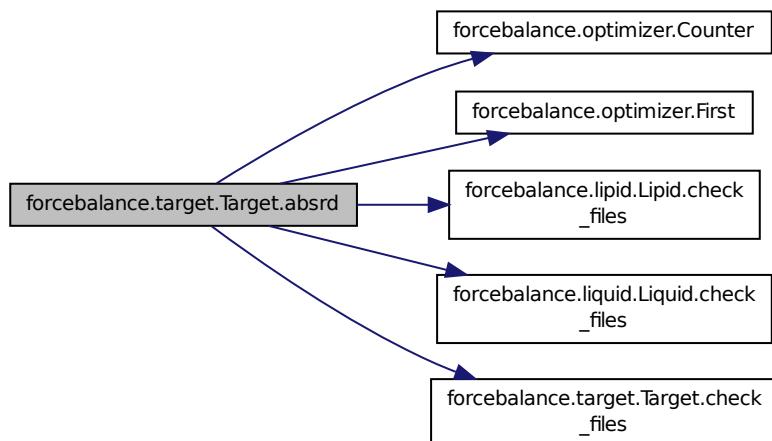
Definition at line 28 of file `__init__.py`.

```
def forcebalance.target.Target.absrd( self, inum = None ) [inherited]
```

Supply the correct directory specified by user's "read" option.

Definition at line 393 of file `target.py`.

Here is the call graph for this function:



```
def forcebalance.target.Target.check_files( self, there ) [inherited] Check this directory for the presence  
of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.binding.BindingEnergy.get ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 184 of file binding.py.
```

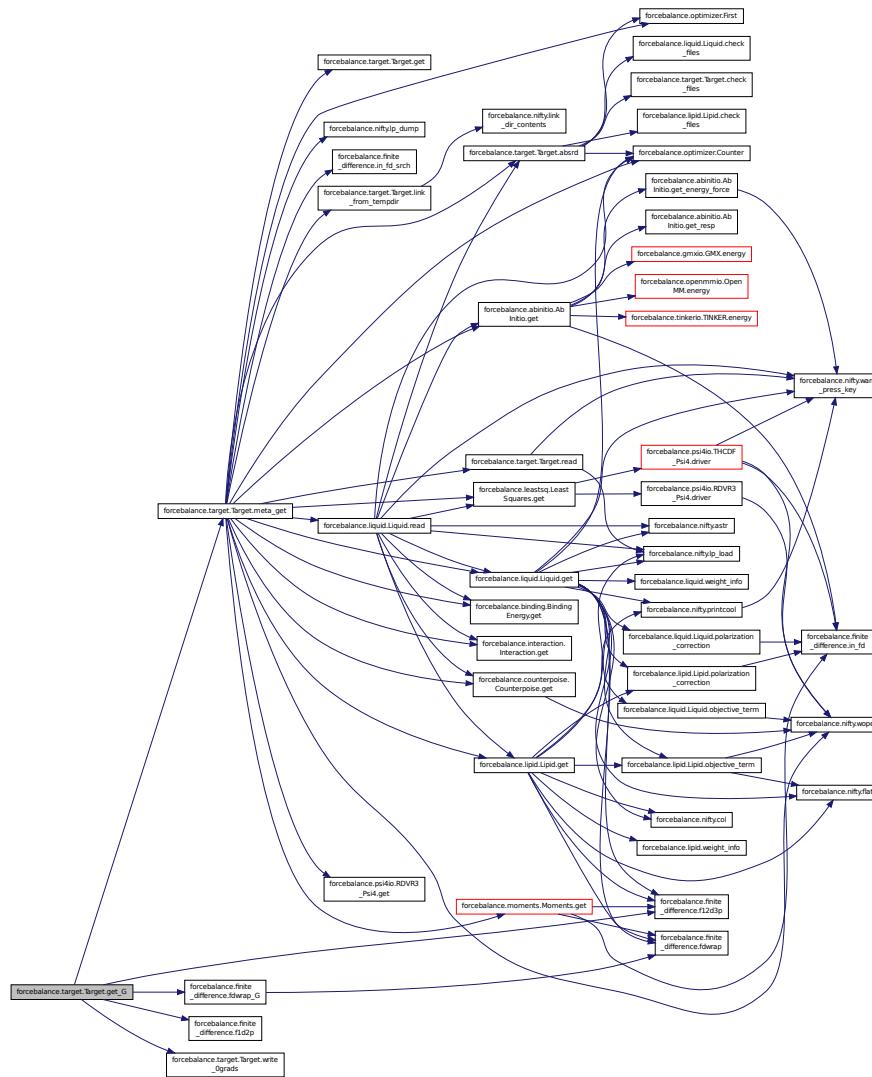
```
def forcebalance.target.Target.get_G( self, mvals = None ) [inherited] Computes the objective function  
contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1.pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



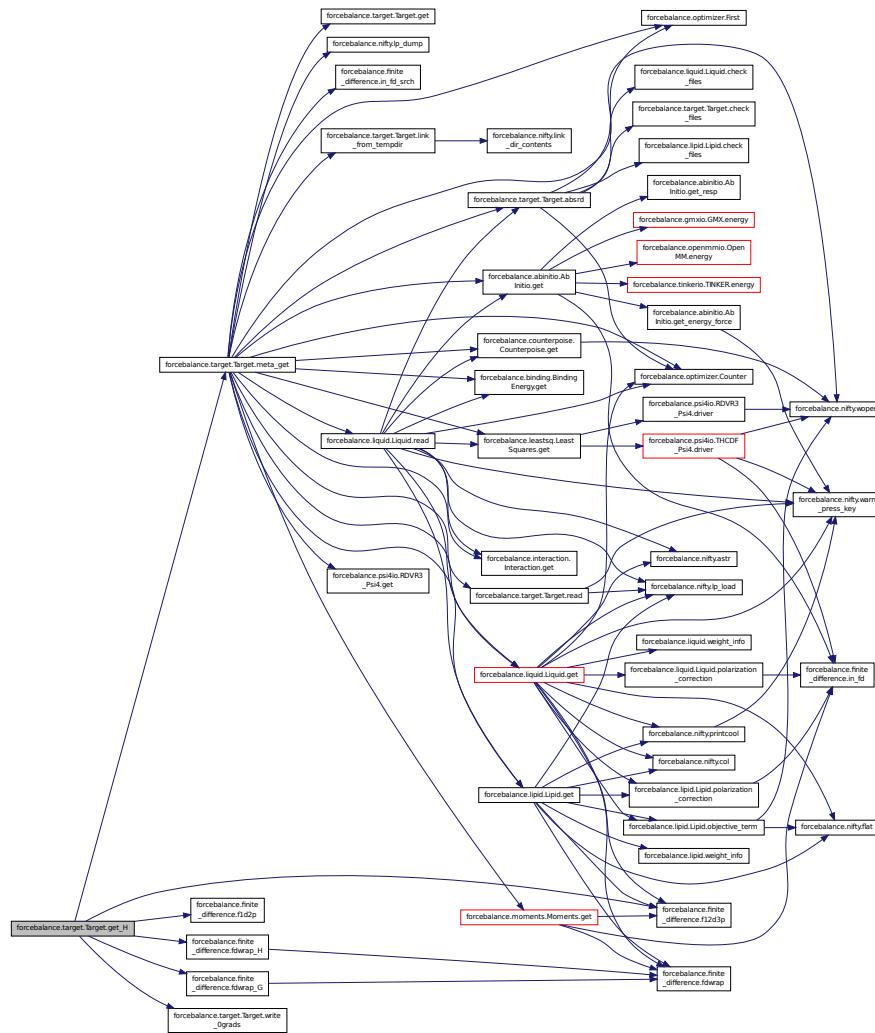
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

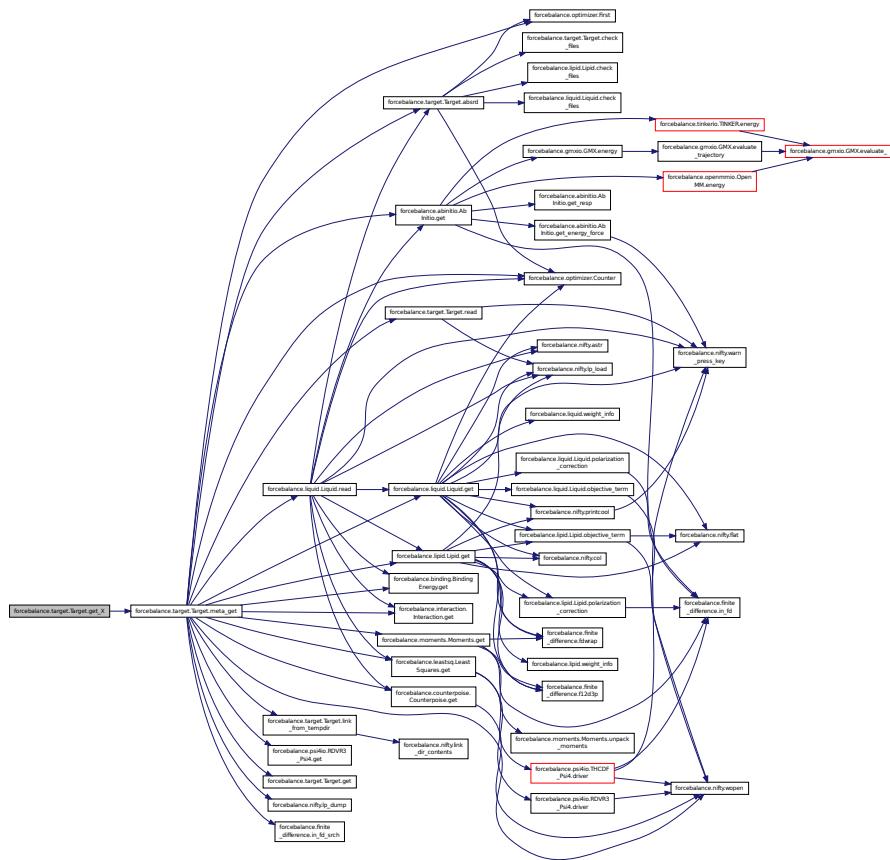
Here is the call graph for this function:



**def forcebalance.target.Target.get.X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

Here is the call graph for this function:



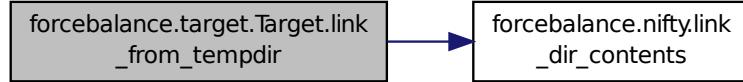
```
def forcebalance.binding.BindingEnergy.indicate( self ) [inherited] Definition at line 178 of file binding.py.
```

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

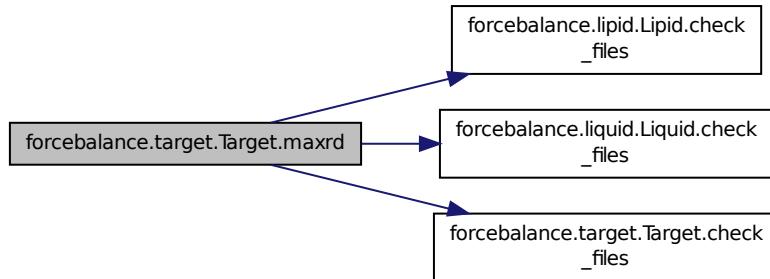
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

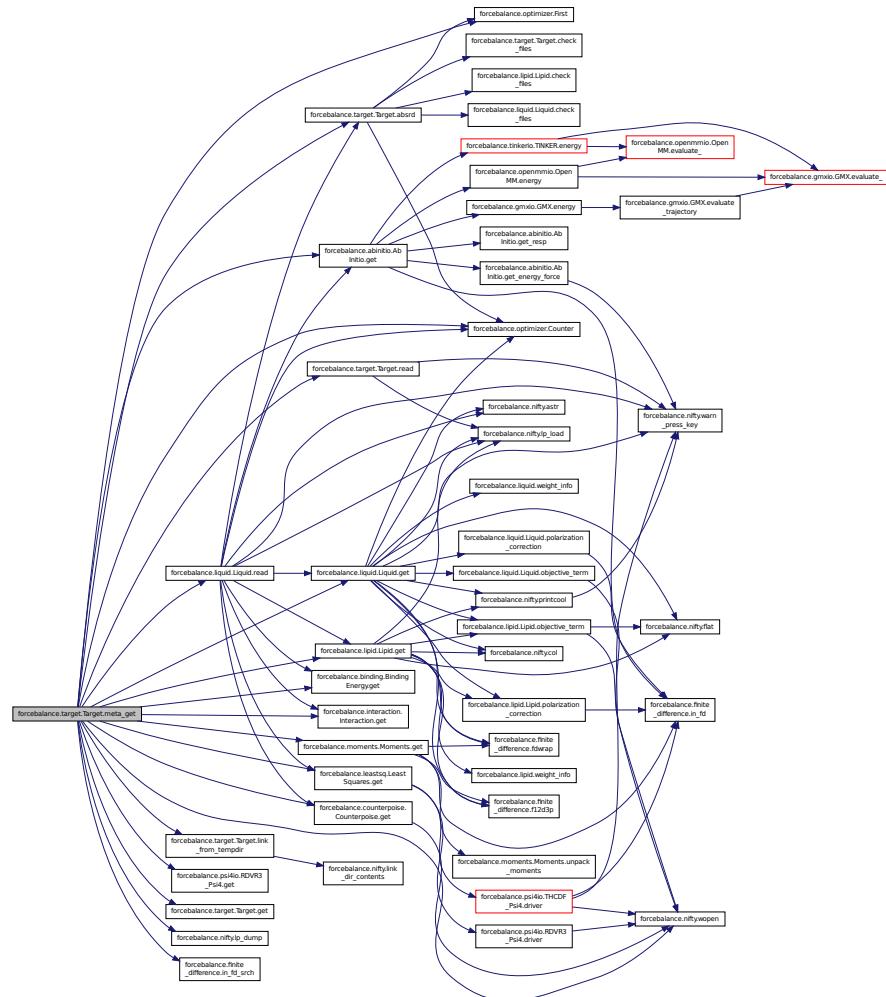


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

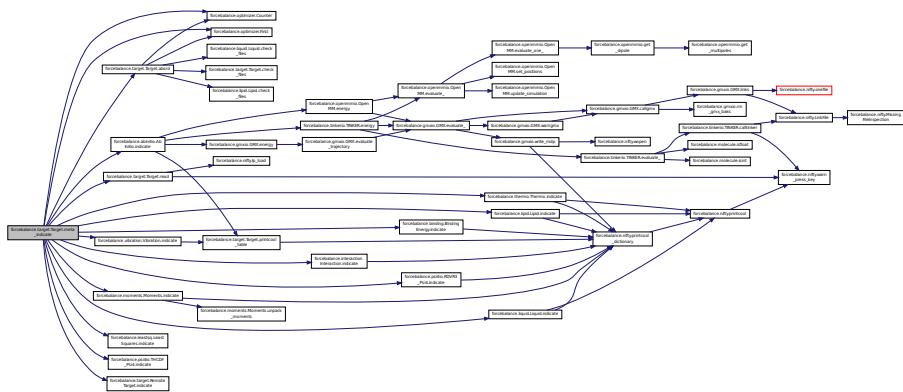
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

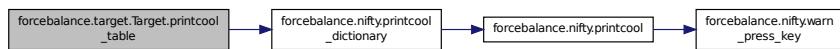
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

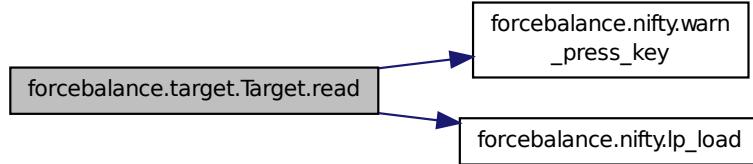
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

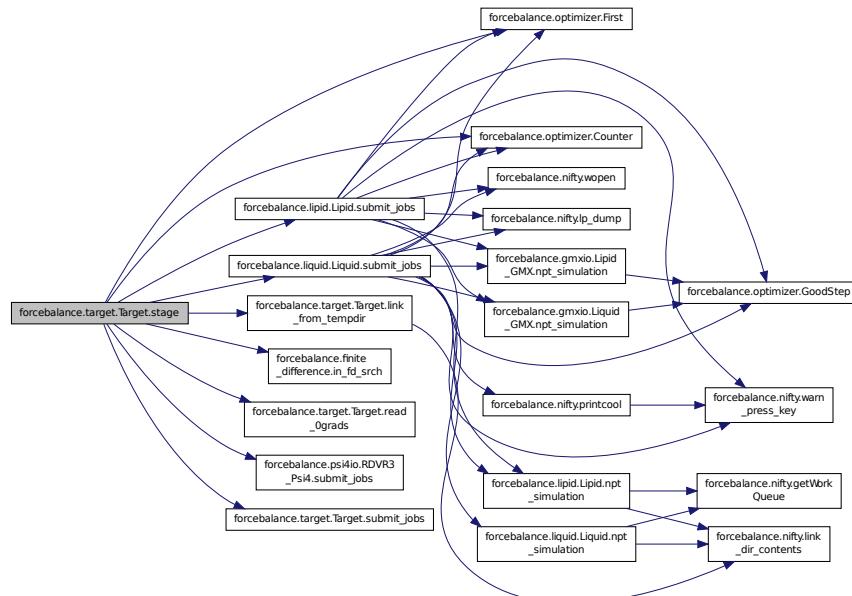
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



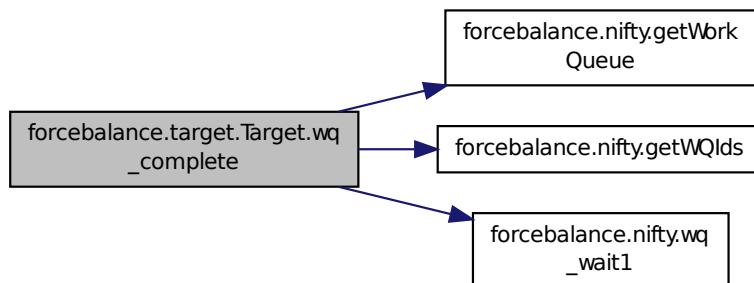
```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.binding.BindingEnergy.system_driver ( self, sysname ) [inherited] Definition at line 174 of file binding.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.13.4 Member Data Documentation

**forcebalance.binding.BindingEnergy.energy\_part [inherited]** Definition at line 228 of file binding.py.

**forcebalance.tinkerio.BindingEnergy\_TINKER.engine\_** Definition at line 1080 of file tinkerio.py.

**forcebalance.binding.BindingEnergy.engines [inherited]** Build keyword dictionaries to pass to engine.

Create engine objects.

Definition at line 165 of file binding.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.binding.BindingEnergy.inter\_opts [inherited]** Definition at line 128 of file binding.py.

**forcebalance.binding.BindingEnergy.objective [inherited]** Definition at line 252 of file binding.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.binding.BindingEnergy.PrintDict [inherited]** Definition at line 186 of file binding.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.binding.BindingEnergy.rmsd.part** [inherited] Definition at line 226 of file binding.py.

**forcebalance.binding.BindingEnergy.RMSDDict** [inherited] Definition at line 187 of file binding.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number  
The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate** [inherited] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

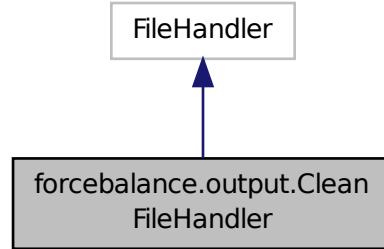
The documentation for this class was generated from the following file:

- [tinkerio.py](#)

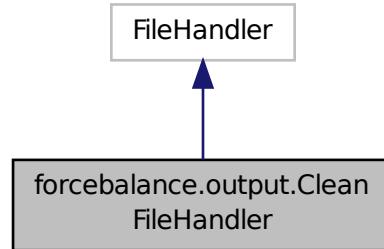
## 8.14 forcebalance.output.CleanFileHandler Class Reference

File handler that does not write terminal escape codes and carriage returns to files.

Inheritance diagram for forcebalance.output.CleanFileHandler:



Collaboration diagram for forcebalance.output.CleanFileHandler:



## Public Member Functions

- def [emit](#)

### 8.14.1 Detailed Description

File handler that does not write terminal escape codes and carriage returns to files.

Use this when writing to a file that will probably not be viewed in a terminal

Definition at line 69 of file [output.py](#).

### 8.14.2 Member Function Documentation

**def forcebalance.output.CleanFileHandler.emit ( self, record )** Definition at line 70 of file [output.py](#).

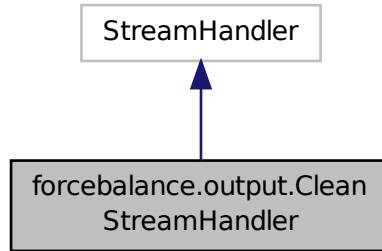
The documentation for this class was generated from the following file:

- [output.py](#)

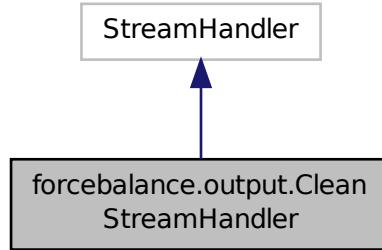
## 8.15 forcebalance.output.CleanStreamHandler Class Reference

Similar to [RawStreamHandler](#) except it does not write terminal escape codes.

Inheritance diagram for forcebalance.output.CleanStreamHandler:



Collaboration diagram for forcebalance.output.CleanStreamHandler:



### Public Member Functions

- def `__init__`
- def `emit`

#### 8.15.1 Detailed Description

Similar to [RawStreamHandler](#) except it does not write terminal escape codes.

Use this for 'plain' terminal output without any fancy colors or formatting

Definition at line 56 of file `output.py`.

#### 8.15.2 Constructor & Destructor Documentation

**def forcebalance.output.CleanStreamHandler.\_\_init\_\_ ( self, stream = sys.stdout )** Definition at line 57 of file `output.py`.

### 8.15.3 Member Function Documentation

```
def forcebalance.output.CleanStreamHandler.emit( self, record )
```

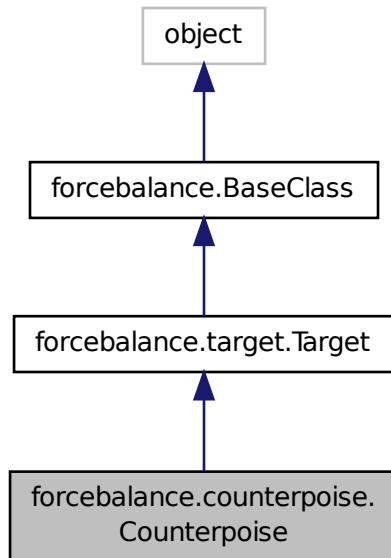
Definition at line 60 of file output.py.  
The documentation for this class was generated from the following file:

- [output.py](#)

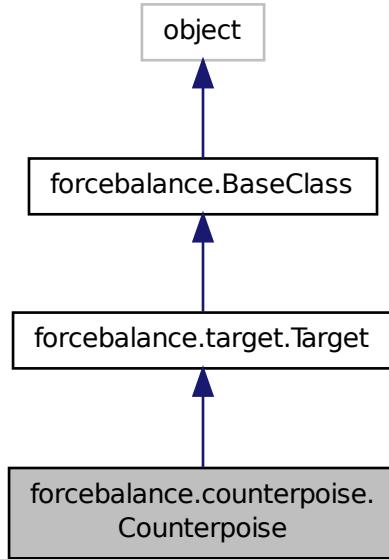
## 8.16 forcebalance.counterpoise.Counterpoise Class Reference

Target subclass for matching the counterpoise correction.

Inheritance diagram for forcebalance.counterpoise.Counterpoise:



Collaboration diagram for forcebalance.counterpoise.Counterpoise:



## Public Member Functions

- def `__init__`  
To instantiate `Counterpoise`, we read the coordinates and counterpoise data.
- def `loadxyz`  
Parse an XYZ file which contains several xyz coordinates, and return their elements.
- def `load_cp`  
Load in the counterpoise data, which is easy; the file consists of floating point numbers separated by newlines.
- def `get`  
Gets the objective function for fitting the counterpoise correction.
- def `get_X`  
Computes the objective function contribution without any parametric derivatives.
- def `read_Ograds`  
Read a file from the target directory containing names of parameters that don't contribute to the gradient.
- def `write_Ograds`  
Write a file to the target directory containing names of parameters that don't contribute to the gradient.
- def `get_G`  
Computes the objective function contribution and its gradient.
- def `get_H`  
Computes the objective function contribution and its gradient / Hessian.
- def `link_from_tempdir`
- def `refresh_temp_directory`  
Back up the temporary directory if desired, delete it and then create a new one.

- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `_setattr_`
- def `set_option`

## Public Attributes

- `xyzs`  
*Number of snapshots.*
- `cpqm`  
*Counterpoise correction data.*
- `na`  
*Number of atoms.*
- `ns`
- `rd`  
*Root directory of the whole project.*
- `pgrad`  
*Iteration where we turn on zero-gradient skipping.*
- `tempbase`  
*Relative directory of target.*
- `tempdir`
- `rundir`  
`self.tempdir = os.path.join('temp',self.name)` *The directory in which the simulation is running - this can be updated.*
- `FF`  
*Need the forcefield (here for now)*
- `xct`  
*Counts how often the objective function was computed.*
- `gct`  
*Counts how often the gradient was computed.*
- `hct`  
*Counts how often the Hessian was computed.*

- `read_indicate`  
*Whether to read indicate.log from file when restarting an aborted run.*
- `write_indicate`  
*Whether to write indicate.log at every iteration (true for all but remote.)*
- `read_objective`  
*Whether to read objective.p from file when restarting an aborted run.*
- `write_objective`  
*Whether to write objective.p at every iteration (true for all but remote.)*
- `verbose_options`
- `PrintOptionDict`

### 8.16.1 Detailed Description

Target subclass for matching the counterpoise correction.

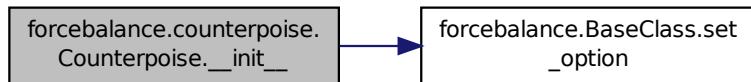
Definition at line 34 of file counterpoise.py.

### 8.16.2 Constructor & Destructor Documentation

`def forcebalance.counterpoise.Counterpoise.__init__ ( self, options, tgt_opts, forcefield )` To instantiate Counterpoise, we read the coordinates and counterpoise data.

Definition at line 38 of file counterpoise.py.

Here is the call graph for this function:



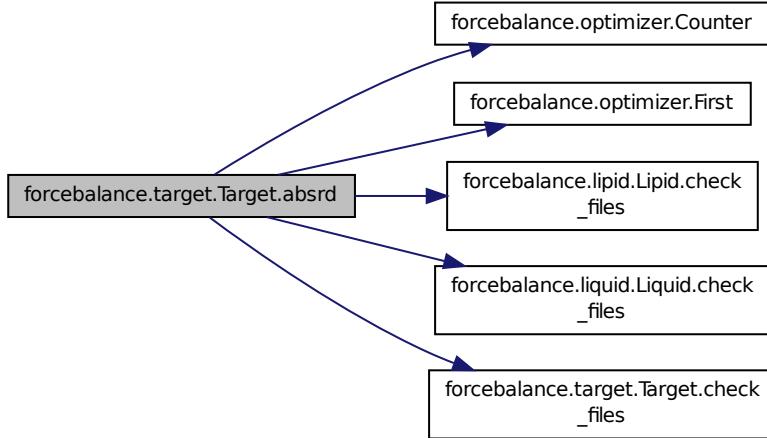
### 8.16.3 Member Function Documentation

`def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited]` Definition at line 28 of file \_\_init\_\_.py.

`def forcebalance.target.Target.absrd ( self, inum = None ) [inherited]` Supply the correct directory specified by user's "read" option.

Definition at line 393 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.check\_files( self, there ) [inherited]** Check this directory for the presence of readable files when the 'read' option is set.

Definition at line 364 of file target.py.

**def forcebalance.counterpoise.Counterpoise.get( self, mvals, AGrad = False, AHess = False )** Gets the objective function for fitting the counterpoise correction.

As opposed to AbInitio\_GMXX2, which calls an external program, this script actually computes the empirical interaction given the force field parameters.

It loops through the snapshots and atom pairs, and computes pairwise contributions to an energy term according to hard-coded functional forms.

One potential issue is that we go through all atom pairs instead of looking only at atom pairs between different fragments. This means that even for two infinitely separated fragments it will predict a finite CP correction. While it might be okay to apply such a potential in practice, there will be some issues for the fitting. Thus, we assume the last snapshot to be CP-free and subtract that value of the potential back out.

Note that forces and parametric derivatives are not implemented.

#### Parameters

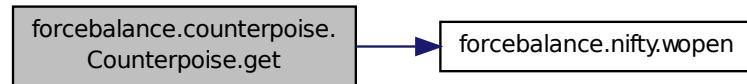
in	<i>mvals</i>	Mathematical parameter values
in	<i>AGrad</i>	Switch to turn on analytic gradient (not implemented)
in	<i>AHess</i>	Switch to turn on analytic Hessian (not implemented)

Returns

Answer Contribution to the objective function

Definition at line 125 of file counterpoise.py.

Here is the call graph for this function:



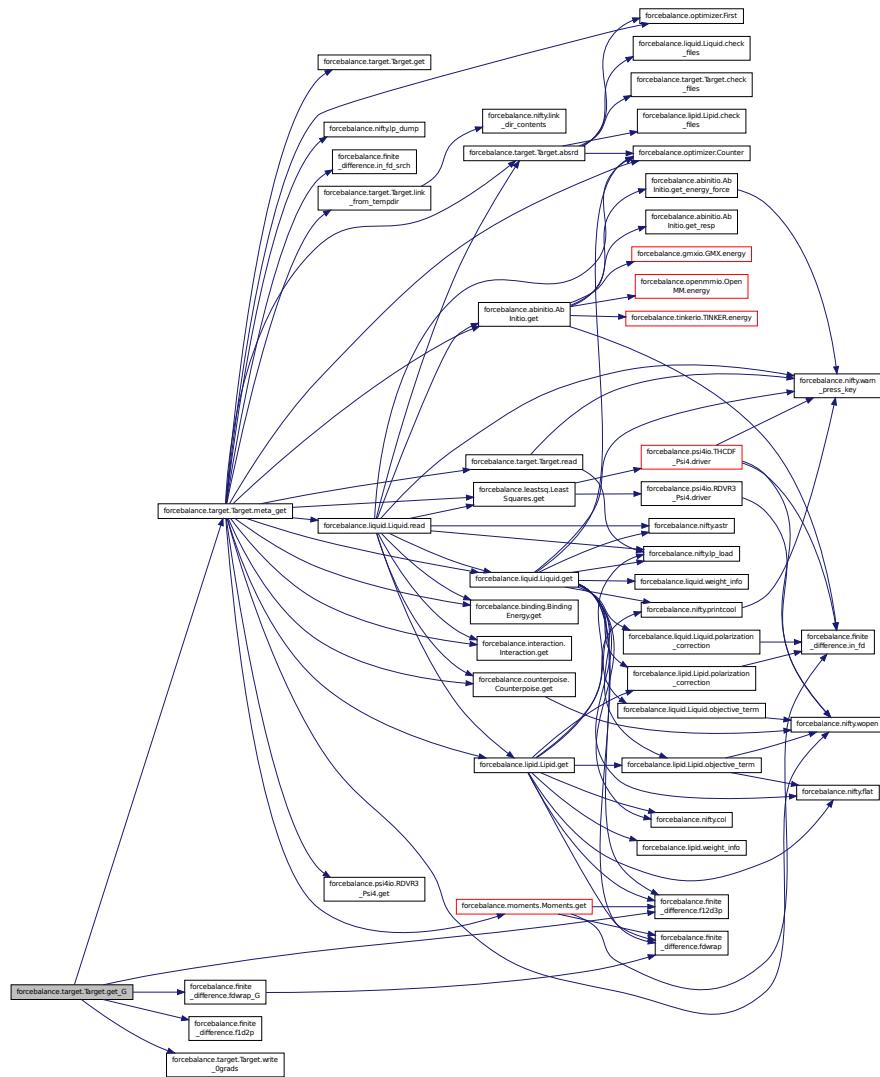
**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1.pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



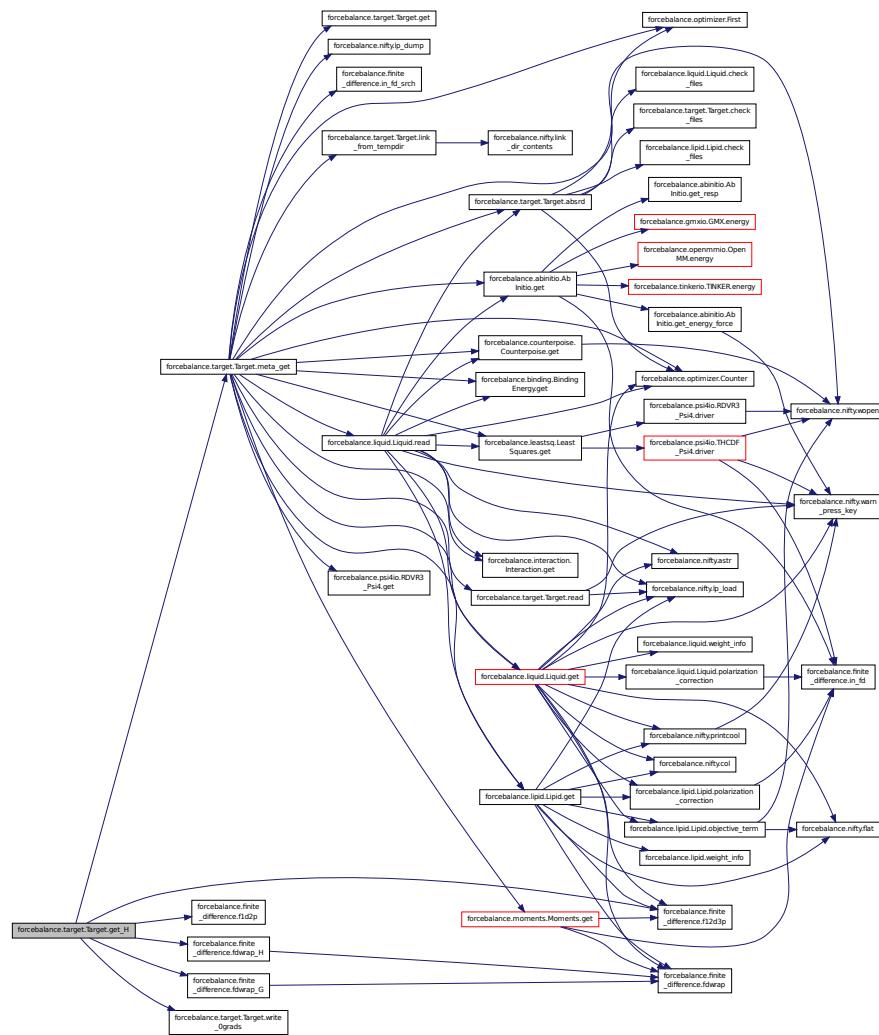
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

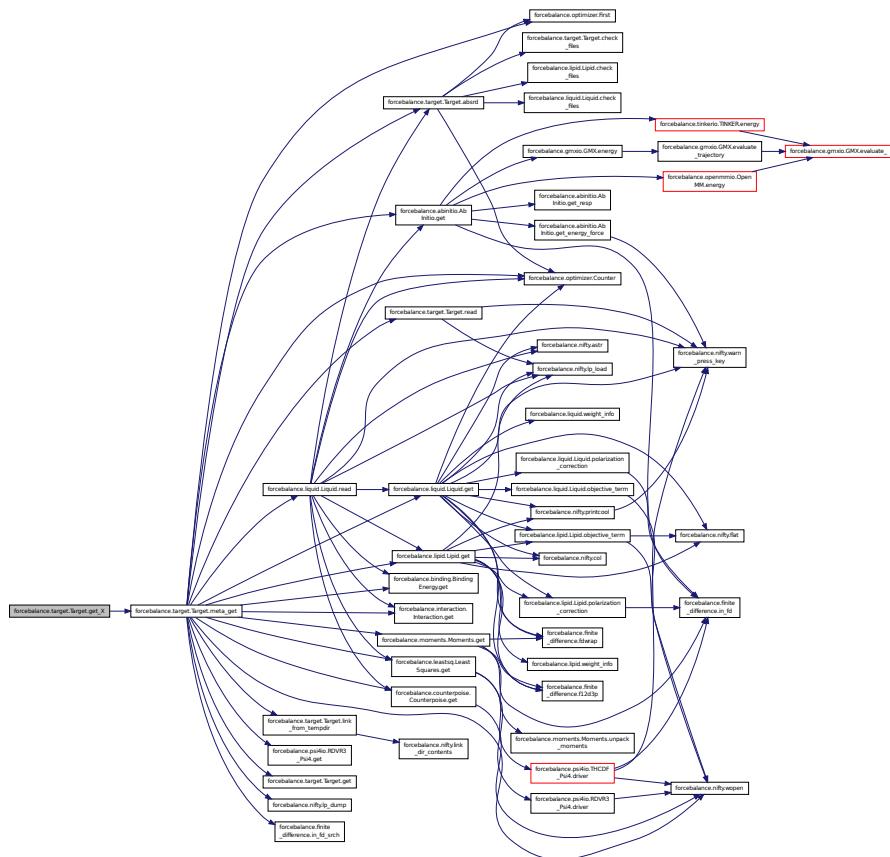
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

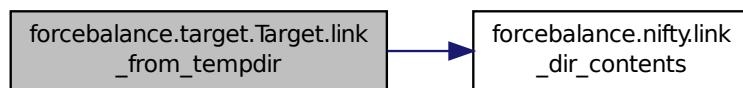
Definition at line 184 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.link_from_tempdir( self, absdestdir ) [inherited] Definition at line 315 of file target.py.
```

Here is the call graph for this function:



**def forcebalance.counterpoise.Counterpoise.load\_cp ( self, fnm )** Load in the counterpoise data, which is easy; the file consists of floating point numbers separated by newlines.

Definition at line 96 of file counterpoise.py.

```
def forcebalance.counterpoise.Counterpoise.loadxyz ( self, fnm ) Parse an XYZ file which contains several xyz coordinates, and return their elements.
```

## Parameters

in	<i>fnm</i>	The input XYZ file name
----	------------	-------------------------

## Returns

`elem` A list of chemical elements in the XYZ file  
`xyzs` A list of XYZ coordinates (number of snapshots times number of atoms)

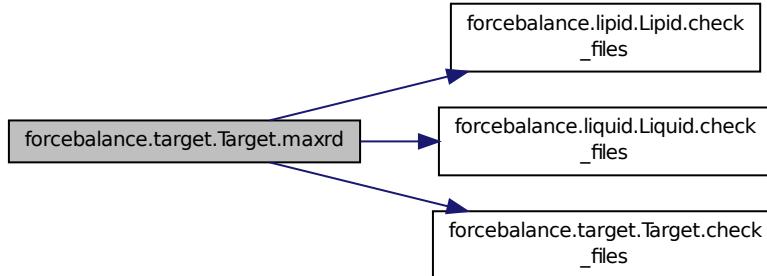
**Todo** I should probably put this into a more general library for reading coordinates.

Definition at line 64 of file counterpoise.py.

**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

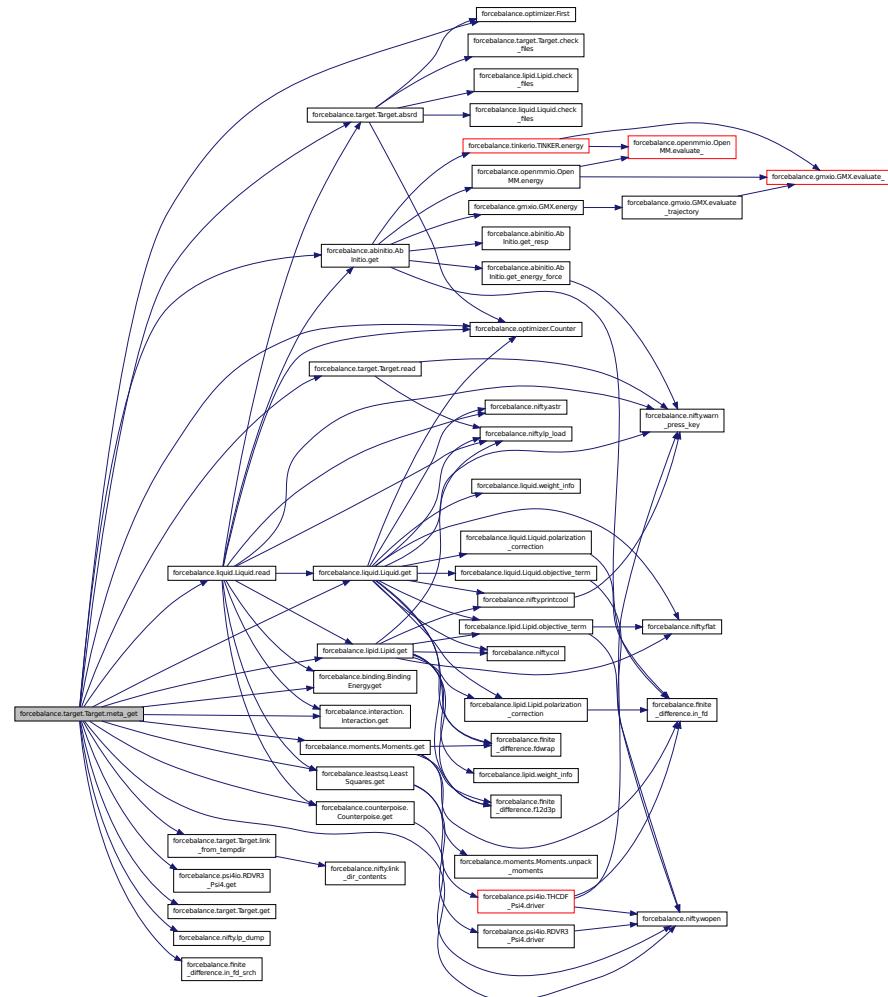


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad=False, AHess=False, customdir=None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call `read()` instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

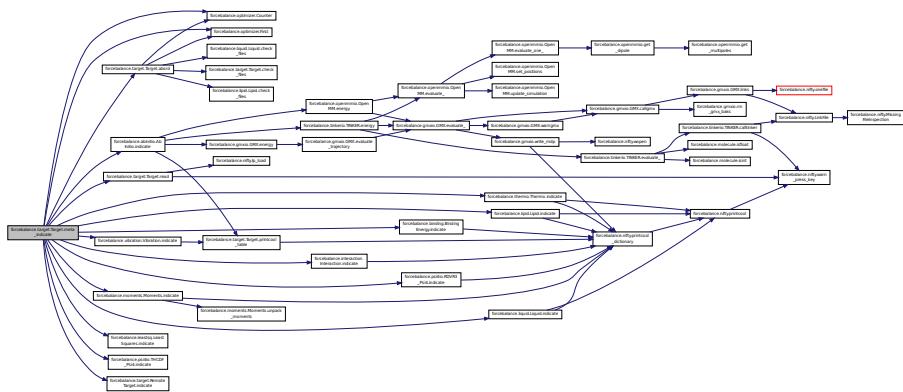
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

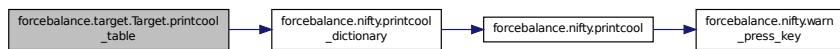
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

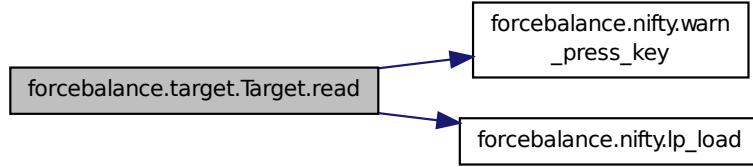
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

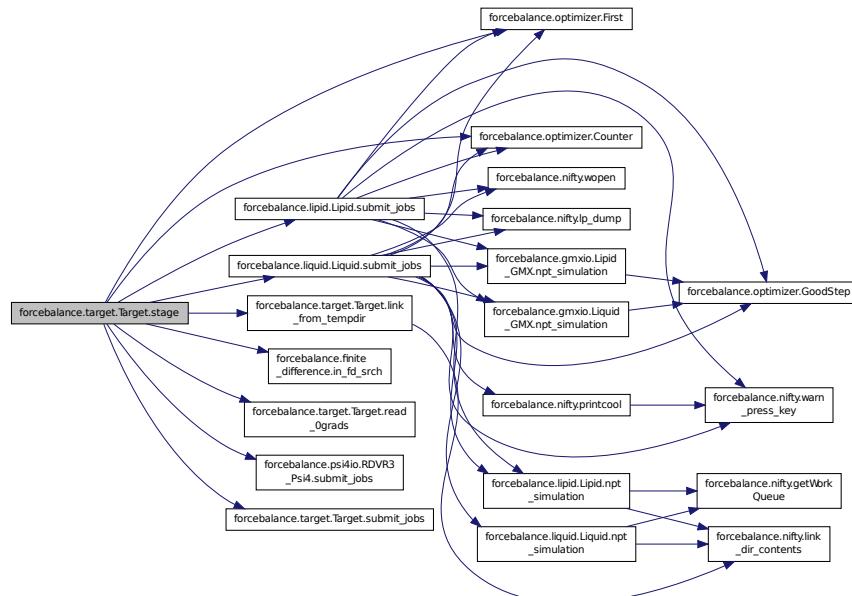
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

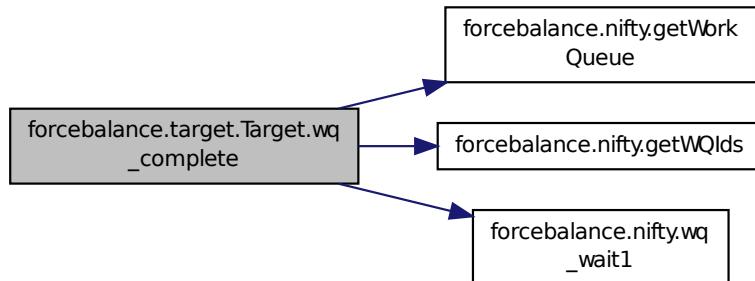


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.write_0grads( self, Ans ) [inherited] Write a file to the target directory containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

#### 8.16.4 Member Data Documentation

**forcebalance.counterpoise.Counterpoise.cpqm** [Counterpoise](#) correction data.

Definition at line 54 of file counterpoise.py.

**forcebalance.target.Target.FF** [\[inherited\]](#) Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct** [\[inherited\]](#) Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct** [\[inherited\]](#) Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.counterpoise.Counterpoise.na** Number of atoms.

Definition at line 77 of file counterpoise.py.

**forcebalance.counterpoise.Counterpoise.ns** Definition at line 90 of file counterpoise.py.

**forcebalance.target.Target.pgrad** [\[inherited\]](#) Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict** [\[inherited\]](#) Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd** [\[inherited\]](#) Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate** [\[inherited\]](#) Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [\[inherited\]](#) Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.target.Target.rundir** [\[inherited\]](#) self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [**inherited**] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [**inherited**] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate** [**inherited**] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

**forcebalance.counterpoise.Counterpoise.xyzs** Number of snapshots.

XYZ elements and coordinates

Definition at line 52 of file counterpoise.py.

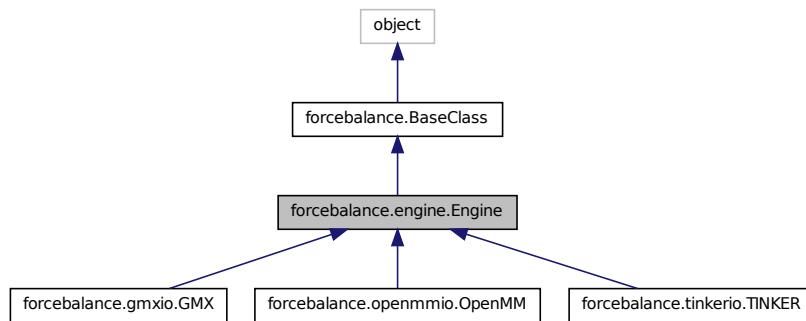
The documentation for this class was generated from the following file:

- [counterpoise.py](#)

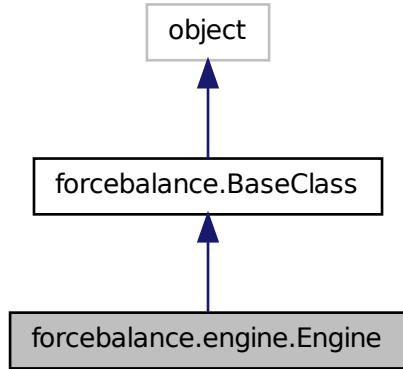
## 8.17 forcebalance.engine.Engine Class Reference

Base class for all engines.

Inheritance diagram for forcebalance.engine.Engine:



Collaboration diagram for forcebalance.engine.Engine:



### Public Member Functions

- def `_init_`
- def `setopts`
- def `readsrc`
- def `prepare`
- def `__setattr__`
- def `set_option`

### Public Attributes

- `name`
  - `verbose`
  - `target`
- Engines can get properties from the Target that creates them.*
- `root`
  - `srcdir`
  - `tempdir`
  - `FF`
  - `verbose_options`
  - `PrintOptionDict`

#### 8.17.1 Detailed Description

Base class for all engines.

##### 1. Introduction

In ForceBalance an `Engine` represents a molecular dynamics code and the calculations that may be carried out with that code.

##### 1. Purpose

Previously system calls to MD software have been made by the Target. Duplication of code was occurring, because different Targets were carrying out the same type of calculation.

#### 1. Also

Target objects should contain [Engine](#) objects, because OpenMM [Engine](#) objects need to be initialized at the start of a calculation.

Definition at line 41 of file engine.py.

### 8.17.2 Constructor & Destructor Documentation

```
def forcebalance.engine.Engine.__init__ ( self, name = "engine", kwargs ) Definition at line 44 of file engine.py.
```

### 8.17.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.engine.Engine.prepare ( self, kwargs ) Definition at line 95 of file engine.py.
```

```
def forcebalance.engine.Engine.readsrc ( self, kwargs ) Definition at line 92 of file engine.py.
```

```
def forcebalance.BaseClass.set_option ( self, in_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

```
def forcebalance.engine.Engine.setopts ( self, kwargs ) Definition at line 89 of file engine.py.
```

### 8.17.4 Member Data Documentation

**forcebalance.engine.Engine.FF** Definition at line 65 of file engine.py.

**forcebalance.engine.Engine.name** Definition at line 48 of file engine.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.engine.Engine.root** Definition at line 56 of file engine.py.

**forcebalance.engine.Engine.srcdir** Definition at line 57 of file engine.py.

**forcebalance.engine.Engine.target** Engines can get properties from the Target that creates them.  
Definition at line 55 of file engine.py.

**forcebalance.engine.Engine.tempdir** Definition at line 58 of file engine.py.

**forcebalance.engine.Engine.verbose** Definition at line 50 of file engine.py.

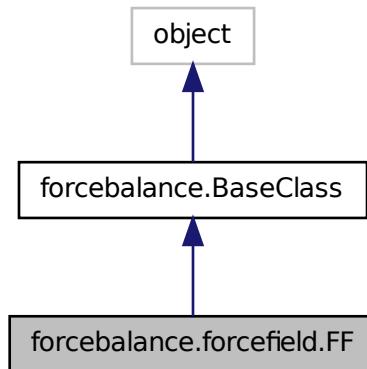
**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.  
The documentation for this class was generated from the following file:

- [engine.py](#)

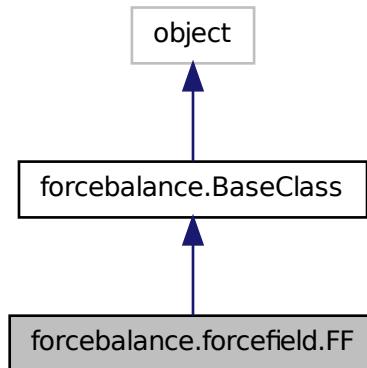
## 8.18 forcebalance.forcefield.FF Class Reference

Force field class.

Inheritance diagram for forcebalance.forcefield.FF:



Collaboration diagram for forcebalance.forcefield.FF:



### Public Member Functions

- def `__init__`  
*Instantiation of force field class.*
- def `addff`  
*Parse a force field file and add it to the class.*
- def `addff_txt`

- def `addff_xml`  
*Parse a text force field and create several important instance variables.*
- def `make`  
*Create a new force field using provided parameter values.*
- def `make_redirect`
- def `find_spacings`
- def `create_pvals`  
*Converts mathematical to physical parameters.*
- def `create_mvals`  
*Converts physical to mathematical parameters.*
- def `rsmake`  
*Create the rescaling factors for the coordinate transformation in parameter space.*
- def `mktransmat`  
*Create the transformation matrix to rescale and rotate the mathematical parameters.*
- def `list_map`  
*Create the plist, which is like a reversed version of the parameter map.*
- def `print_map`  
*Prints out the (physical or mathematical) parameter indices, IDs and values in a visually appealing way.*
- def `sprint_map`  
*Prints out the (physical or mathematical) parameter indices, IDs and values to a string.*
- def `assign_p0`  
*Assign physical parameter values to the 'pvals0' array.*
- def `assign_field`  
*Record the locations of a parameter in a txt file; [[file name, line number, field number, and multiplier]].*
- def `__eq__`
- def `__setattr__`
- def `set_option`

## Public Attributes

- `ffdata`  
*As these options proliferate, the force field class becomes less standalone.*
- `ffdata_isxml`
- `map`  
*The mapping of interaction type -> parameter number.*
- `plist`  
*The listing of parameter number -> interaction types.*
- `patoms`  
*A listing of parameter number -> atoms involved.*
- `pfields`  
*A list where pfields[pnum] = [file',line,field,mult,cmd], basically a new way to modify force field files; when we modify the force field file, we go to the specific line/field in a given file and change the number.*
- `rs`  
*List of rescaling factors.*
- `tm`  
*The transformation matrix for mathematical -> physical parameters.*
- `tml`

*The transpose of the transformation matrix.*

- [excision](#)

*Indices to exclude from optimization / Hessian inversion.*

- [np](#)

*The total number of parameters.*

- [pvals0](#)

*Initial value of physical parameters.*

- [Readers](#)

*A dictionary of force field reader classes.*

- [atomnames](#)

*A list of atom names (this is new, for ESP fitting)*

- [FFAtomTypes](#)

*WORK IN PROGRESS ## This is a dictionary of { 'AtomType':{ 'Mass' : float, 'Charge' : float, 'ParticleType' : string ('A', 'S', or 'D'), 'AtomicNumber' : int } }.*

- [FFMolecules](#)

- [redirect](#)

*Creates plist from map.*

- [linedestroy\\_save](#)

*Destruction dictionary (experimental).*

- [prmdestroy\\_save](#)

- [linedestroy\\_this](#)

- [prmdestroy\\_this](#)

- [tinkerprm](#)

- [openmmxml](#)

- [qmap](#)

- [qid](#)

- [qid2](#)

- [verbose\\_options](#)

- [PrintOptionDict](#)

### 8.18.1 Detailed Description

Force field class.

This class contains all methods for force field manipulation. To create an instance of this class, an input file is required containing the list of force field file names. Everything else inside this class pertaining to force field generation is self-contained.

For details on force field parsing, see the detailed documentation for addff.

Definition at line 197 of file forcefield.py.

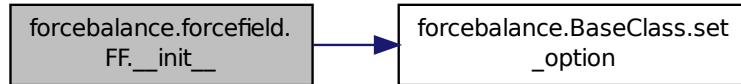
### 8.18.2 Constructor & Destructor Documentation

**def forcebalance.forcefield.FF.\_\_init\_\_( self, options, verbose = True )** Instantiation of force field class.

Many variables here are initialized to zero, but they are filled out by methods like addff, rsmake, and mktransmat.

Definition at line 205 of file forcefield.py.

Here is the call graph for this function:



### 8.18.3 Member Function Documentation

**def forcebalance.forcefield.FF.eq\_( self, other )** Definition at line 1185 of file forcefield.py.

**def forcebalance.BaseClass.\_\_setattr\_\_( self, key, value ) [inherited]** Definition at line 28 of file \_\_init\_\_.py.

**def forcebalance.forcefield.FF.addff( self, fname )** Parse a force field file and add it to the class.

First, figure out the type of force field file. This is done either by explicitly specifying the type using for example, `ffname force_field.xml:openmm` or we figure it out by looking at the file extension.

Next, parse the file. Currently we support two classes of files - text and XML. The two types are treated very differently; for XML we use the parsers in libxml (via the python lxml module), and for text files we have our own in-house parsing class. Within text files, there is also a specialized GROMACS and TINKER parser as well as a generic text parser.

The job of the parser is to determine the following things: 1) Read the user-specified selection of parameters being fitted 2) Build a mapping (dictionary) of parameter identifier  $\rightarrow$  index in parameter vector 3) Build a list of physical parameter values 4) Figure out where to replace the parameter values in the force field file when the values are changed 5) Figure out which parameters need to be repeated or sign-flipped

Generally speaking, each parameter value in the force field file has a unique parameter identifier . The identifier consists of three parts - the interaction type, the parameter subtype (within that interaction type), and the atoms involved.

--- If XML: ---

The force field file is read in using the lxml Python module. Specify which parameter you want to fit using by adding a 'parameterize' element to the end of the force field XML file, like so.

```
1 <AmoebaVdwForce type="BUFFERED-14-7">  
2   <Vdw class="74" sigma="0.2655" epsilon="0.056484" reduction="0.910" parameterize="sigma, epsilon,  
     reduction" />
```

In this example, the parameter identifier would look like Vdw/74/epsilon .

--- If GROMACS (.itp) or TINKER (.prm) : ---

Follow the rules in the ITP.Reader or Tinker.Reader derived class. Read the documentation in the class documentation or the 'feed' method to learn more. In all cases the parameter is tagged using # PRM 3 (where # denotes a comment, the word PRM stays the same, and 3 is the field number starting from zero.)

--- If normal text : ---

The parameter identifier is simply built using the file name, line number, and field. Thus, the identifier is unique but completely noninformative (which is not ideal for our purposes, but it works.)

--- Endif ---

## Warning

My program currently assumes that we are only using one MM program per job. If we use CHARMM and GROMACS to perform simulations as part of the same TARGET, we will get messed up. Maybe this needs to be fixed in the future, with program prefixes to parameters like C\_ , G\_ .. or simply unit conversions, you get the idea.

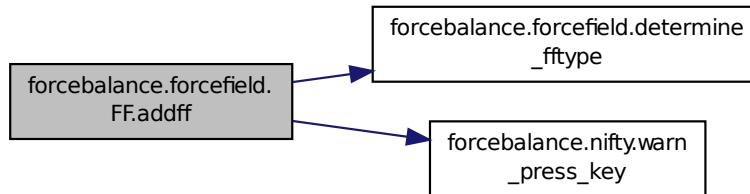
I don't think the multiplier actually works for analytic derivatives unless the interaction calculator knows the multiplier as well. I'm sure I can make this work in the future if necessary.

## Parameters

in	ffname	Name of the force field file
----	--------	------------------------------

Definition at line 398 of file forcefield.py.

Here is the call graph for this function:



**def forcebalance.forcefield.FF.addff\_txt ( self, ffname, ffitype )** Parse a text force field and create several important instance variables.

Each line is processed using the 'feed' method as implemented in the reader class. This essentially allows us to create the correct parameter identifier (pid), because the pid comes from more than the current line, it also depends on the section that we're in.

When 'PRM' or 'RPT' is encountered, we do several things:

- Build the parameter identifier and insert it into the map
- Point to the file name, line number, and field where the parameter may be modified

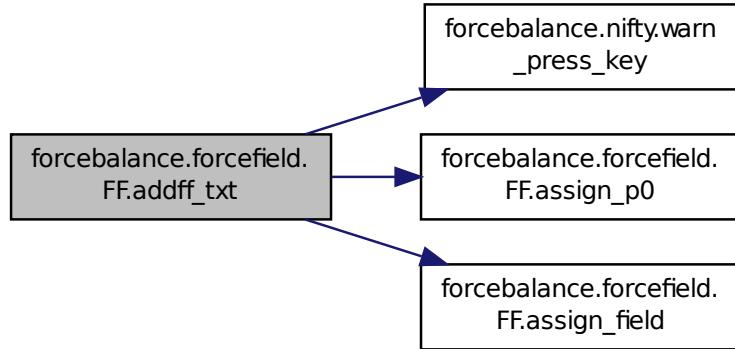
Additionally, when 'PRM' is encountered:

- Store the physical parameter value (this is permanent; it's the original value)
- Increment the total number of parameters

When 'RPT' is encountered we don't expand the parameter vector because this parameter is a copy of an existing one. If the parameter identifier is preceded by MINUS\_, we chop off the prefix but remember that the sign needs to be flipped.

Definition at line 468 of file forcefield.py.

Here is the call graph for this function:



```
def forcebalance.forcefield.FF.addff_xml ( self, fname ) Parse an XML force field file and create important instance variables.
```

This was modeled after addff.txt, but XML and text files are fundamentally different, necessitating two different methods.

We begin with an `_ElementTree` object. We search through the tree for the 'parameterize' and 'parameter\_repeat' keywords. Each time the keyword is encountered, we do the same four actions that I describe in addff.txt.

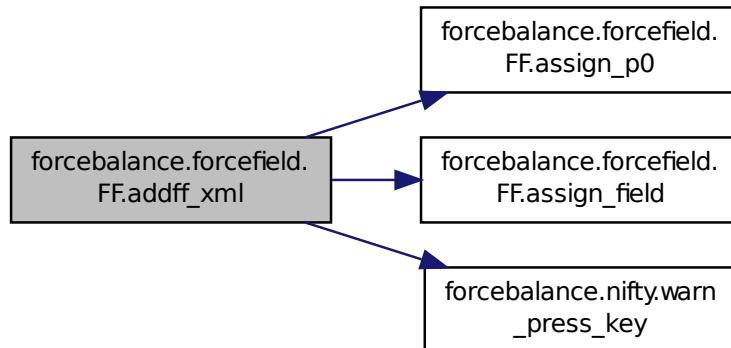
It's hard to specify precisely the location in an XML file to change a force field parameter. I can create a list of tree elements (essentially pointers to elements within a tree), but this method breaks down when I copy the tree because I have no way to refer to the copied tree elements. Fortunately, lxml gives me a way to represent a tree using a flat list, and my XML file 'locations' are represented using the positions in the list.

## Warning

The sign-flip hasn't been implemented yet. This shouldn't matter unless your calculation contains repeated parameters with opposite sign.

Definition at line 588 of file forcefield.py.

Here is the call graph for this function:



**def forcebalance.forcefield.FF.assign\_field ( self, idx, fnm, ln, pfld, mult, cmd = None )** Record the locations of a parameter in a txt file: [[file name, line number, field number, and multiplier]].

Note that parameters can have multiple locations because of the repetition functionality.

### Parameters

in	idx	The index of the parameter.
in	fnm	The file name of the parameter field.
in	ln	The line number within the file (or the node index in the flattened xml)
in	pfld	The field within the line (or the name of the attribute in the xml)
in	mult	The multiplier (this is usually 1.0)

Definition at line 1179 of file forcefield.py.

**def forcebalance.forcefield.FF.assign\_p0 ( self, idx, val )** Assign physical parameter values to the 'pvals0' array.

### Parameters

in	idx	The index to which we assign the parameter value.
in	val	The parameter value to be inserted.

Definition at line 1161 of file forcefield.py.

**def forcebalance.forcefield.FF.create\_mvals ( self, pvals )** Converts physical to mathematical parameters.

We create the inverse transformation matrix using SVD.

#### Parameters

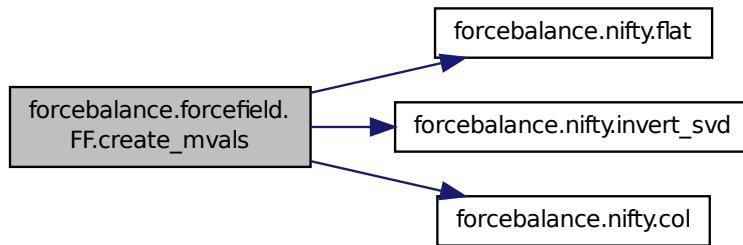
in	<i>pvals</i>	The physical parameters
----	--------------	-------------------------

#### Returns

*mvals* The mathematical parameters

Definition at line 870 of file forcefield.py.

Here is the call graph for this function:



**def forcebalance.forcefield.FF.create\_pvals ( *self*, *mvals* )** Converts mathematical to physical parameters.

First, mathematical parameters are rescaled and rotated by multiplying by the transformation matrix, followed by adding the original physical parameters.

#### Parameters

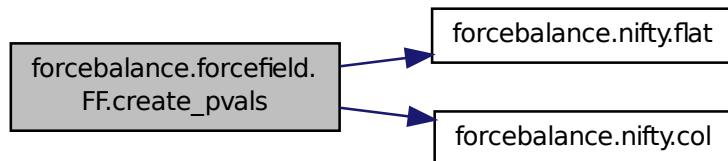
in	<i>mvals</i>	The mathematical parameters
----	--------------	-----------------------------

#### Returns

*pvals* The physical parameters

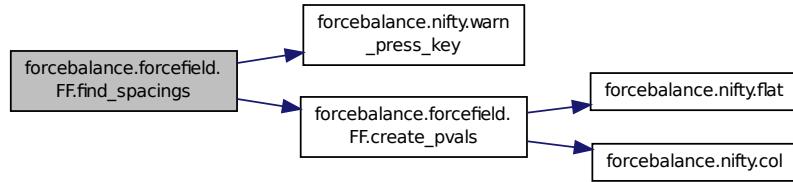
Definition at line 837 of file forcefield.py.

Here is the call graph for this function:



**def forcebalance.forcefield.FF.find\_spacings ( self )** Definition at line 792 of file forcefield.py.

Here is the call graph for this function:

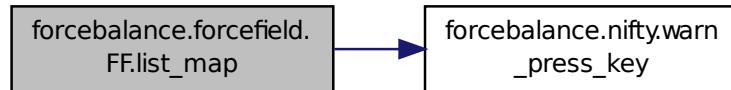


**def forcebalance.forcefield.FF.list\_map ( self )** Create the plist, which is like a reversed version of the parameter map.

More convenient for printing.

Definition at line 1129 of file forcefield.py.

Here is the call graph for this function:



**def forcebalance.forcefield.FF.make ( self, vals = None, use\_pvals = False, printdir = None, precision = 12 )** Create a new force field using provided parameter values.

This big kahuna does a number of things: 1) Creates the physical parameters from the mathematical parameters 2) Creates force fields with physical parameters substituted in 3) Prints the force fields to the specified file.

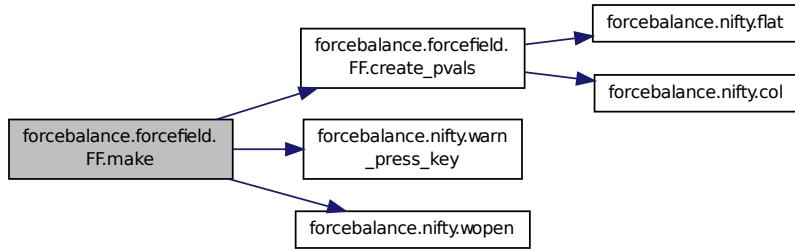
It does NOT store the mathematical parameters in the class state (since we can only hold one set of parameters).

#### Parameters

in	<i>printdir</i>	The directory that the force fields are printed to; as usual this is relative to the project root directory.
in	<i>vals</i>	Input parameters. I previously had an option where it uses stored values in the class state, but I don't think that's a good idea anymore.
in	<i>use_pvals</i>	Switch for whether to bypass the coordinate transformation and use physical parameters directly.

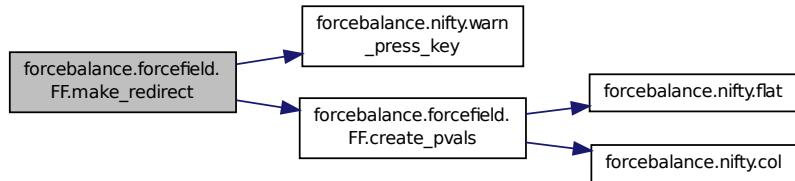
Definition at line 635 of file forcefield.py.

Here is the call graph for this function:



**def forcebalance.forcefield.FF.make\_redirect ( self, mvals )** Definition at line 751 of file forcefield.py.

Here is the call graph for this function:



**def forcebalance.forcefield.FF.mktransmat ( self )** Create the transformation matrix to rescale and rotate the mathematical parameters.

For point charge parameters, project out perturbations that change the total charge.

First build these:

'qmap' : Just a list of parameter indices that point to charges.

'qid' : For each parameter in the qmap, a list of the affected atoms :) A potential target for the molecule-specific thang.

Then make this:

'qtrans2' : A transformation matrix that rotates the charge parameters. The first row is all zeros (because it corresponds to increasing the charge on all atoms) The other rows correspond to changing one of the parameters and decreasing all of the others equally such that the overall charge is preserved.

'qmat2' : An identity matrix with 'qtrans2' pasted into the right place

'transmat' : 'qmat2' with rows and columns scaled using self.rs

'excision' : Parameter indices that need to be 'cut out' because they are irrelevant and mess with the matrix diagonalization

**Todo** Only project out changes in total charge of a molecule, and perhaps generalize to fragments of molecules or other types of parameters.

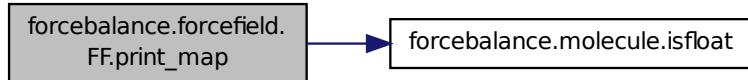
The AMOEBA selection of charge depends not only on the atom type, but what that atom is bonded to.

Definition at line 963 of file forcefield.py.

```
def forcebalance.forcefield.FF.print_map ( self, vals = None, precision = 4 ) Prints out the (physical or mathematical) parameter indices, IDs and values in a visually appealing way.
```

Definition at line 1141 of file forcefield.py.

Here is the call graph for this function:



```
def forcebalance.forcefield.FF.rsmake ( self, printfacs = True ) Create the rescaling factors for the coordinate transformation in parameter space.
```

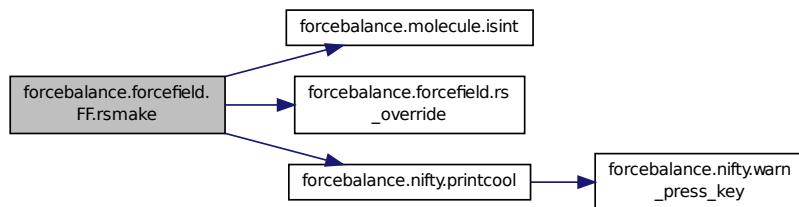
The proper choice of rescaling factors (read: prior widths in maximum likelihood analysis) is still a black art. This is a topic of current research.

**Todo** Pass in rsfactors through the input file

```
@param[in] printfacs List for printing out the resecaling factors
```

Definition at line 887 of file forcefield.py.

Here is the call graph for this function:

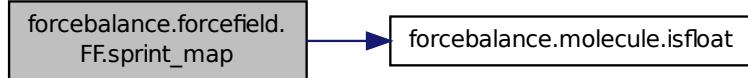


```
def forcebalance.BaseClass.set_option ( self, in_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

```
def forcebalance.forcefield.FF.sprint_map ( self, vals = None, precision = 4 ) Prints out the (physical or mathematical) parameter indices, IDs and values to a string.
```

Definition at line 1149 of file forcefield.py.

Here is the call graph for this function:



#### 8.18.4 Member Data Documentation

**forcebalance.forcefield.FF.atomnames** A list of atom names (this is new, for ESP fitting)  
Definition at line 268 of file forcefield.py.

**forcebalance.forcefield.FF.excision** Indices to exclude from optimization / Hessian inversion.  
Some customized constraints here.  
Quadrupoles must be traceless  
Definition at line 260 of file forcefield.py.

**forcebalance.forcefield.FF.FFAtomTypes** WORK IN PROGRESS ## This is a dictionary of {'AtomType': {'Mass' : float, 'Charge' : float, 'ParticleType' : string ('A', 'S', or 'D'), 'AtomicNumber' : int}}.  
Definition at line 278 of file forcefield.py.

**forcebalance.forcefield.FF.ffdata** As these options proliferate, the force field class becomes less standalone.  
I need to think of a good solution here... The root directory of the project File names of force fields Directory containing force fields, relative to project directory Priors given by the user :) Whether to constrain the charges. Whether to constrain the charges. Switch for AMOEBA direct or mutual. AMOEBA mutual dipole convergence tolerance. Switch for rigid water molecules Bypass the transformation and use physical parameters directly The content of all force field files are stored in memory  
Definition at line 240 of file forcefield.py.

**forcebalance.forcefield.FF.ffdata\_isxml** Definition at line 241 of file forcefield.py.

**forcebalance.forcefield.FF.FFMolecules** Definition at line 291 of file forcefield.py.

**forcebalance.forcefield.FF.linedestroy\_save** Destruction dictionary (experimental).  
Definition at line 319 of file forcefield.py.

**forcebalance.forcefield.FF.linedestroy\_this** Definition at line 321 of file forcefield.py.

**forcebalance.forcefield.FF.map** The mapping of interaction type -> parameter number.  
Definition at line 243 of file forcefield.py.

**forcebalance.forcefield.FF.np** The total number of parameters.  
Definition at line 262 of file forcefield.py.

**forcebalance.forcefield.FF.openmmxml** Definition at line 414 of file forcefield.py.

**forcebalance.forcefield.FF.patoms** A listing of parameter number -> atoms involved.

Definition at line 247 of file forcefield.py.

**forcebalance.forcefield.FF.pfields** A list where pfields[pnum] = [file',line,field,mult,cmd], basically a new way to modify force field files; when we modify the force field file, we go to the specific line/field in a given file and change the number.

Definition at line 252 of file forcefield.py.

**forcebalance.forcefield.FF.plist** The listing of parameter number -> interaction types.

Definition at line 245 of file forcefield.py.

**forcebalance.BaseClass.PrintOptionDict** [**inherited**] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.forcefield.FF.prmdestroy\_save** Definition at line 320 of file forcefield.py.

**forcebalance.forcefield.FF.prmdestroy\_this** Definition at line 322 of file forcefield.py.

**forcebalance.forcefield.FF.pvals0** Initial value of physical parameters.

Definition at line 264 of file forcefield.py.

**forcebalance.forcefield.FF.qid** Definition at line 965 of file forcefield.py.

**forcebalance.forcefield.FF.qid2** Definition at line 966 of file forcefield.py.

**forcebalance.forcefield.FF.qmap** Definition at line 964 of file forcefield.py.

**forcebalance.forcefield.FF.Readers** A dictionary of force field reader classes.

Definition at line 266 of file forcefield.py.

**forcebalance.forcefield.FF.redirect** Creates plist from map.

Prints the plist to screen. Make the rescaling factors. Make the transformation matrix. Redirection dictionary (experimental).

Definition at line 317 of file forcefield.py.

**forcebalance.forcefield.FF.rs** List of rescaling factors.

Takes the dictionary 'BONDS':{3:'B', 4:'K'}, 'VDW':{4:'S', 5:'T'}, and turns it into a list of term types ['BONDSB','B-ONDSK','VDWS','VDWT'].

The array of rescaling factors

Definition at line 254 of file forcefield.py.

**forcebalance.forcefield.FF.tinkerprm** Definition at line 407 of file forcefield.py.

**forcebalance.forcefield.FF.tm** The transformation matrix for mathematical -> physical parameters.

Definition at line 256 of file forcefield.py.

**forcebalance.forcefield.FF.tml** The transpose of the transformation matrix.

Definition at line 258 of file forcefield.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

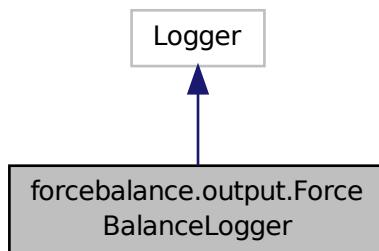
The documentation for this class was generated from the following file:

- [forcefield.py](#)

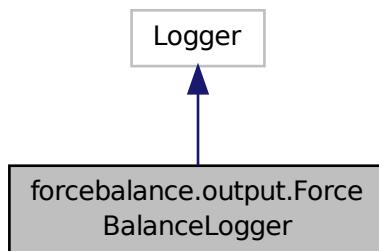
## 8.19 forcebalance.output.ForceBalanceLogger Class Reference

This logger starts out with a default handler that writes to stdout addHandler removes this default the first time another handler is added.

Inheritance diagram for forcebalance.output.ForceBalanceLogger:



Collaboration diagram for forcebalance.output.ForceBalanceLogger:



### Public Member Functions

- def [\\_\\_init\\_\\_](#)
- def [addHandler](#)
- def [removeHandler](#)

### Public Attributes

- [defaultHandler](#)

### 8.19.1 Detailed Description

This logger starts out with a default handler that writes to stdout addHandler removes this default the first time another handler is added.

This is used for forcebalance package level logging, where a logger should always be present unless otherwise specified. To silence, add a NullHandler We also by default set the log level to INFO (logging.Logger starts at WARNING)

Definition at line 10 of file output.py.

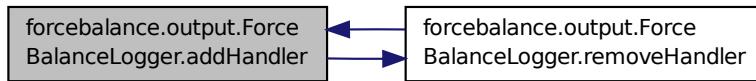
### 8.19.2 Constructor & Destructor Documentation

```
def forcebalance.output.ForceBalanceLogger.__init__ ( self, name ) Definition at line 11 of file output.py.
```

### 8.19.3 Member Function Documentation

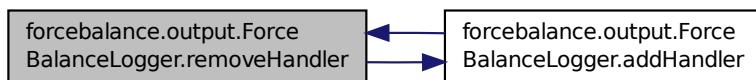
```
def forcebalance.output.ForceBalanceLogger.addHandler ( self, hdlr ) Definition at line 17 of file output.py.
```

Here is the call graph for this function:



```
def forcebalance.output.ForceBalanceLogger.removeHandler ( self, hdlr ) Definition at line 23 of file output.py.
```

Here is the call graph for this function:



### 8.19.4 Member Data Documentation

```
forcebalance.output.ForceBalanceLogger.defaultHandler Definition at line 13 of file output.py.
```

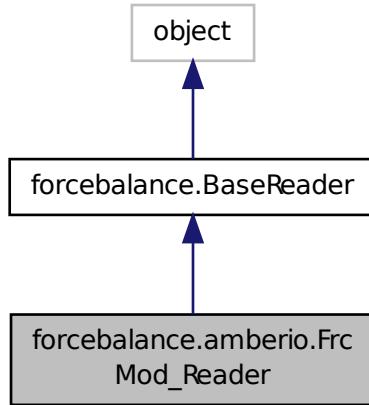
The documentation for this class was generated from the following file:

- [output.py](#)

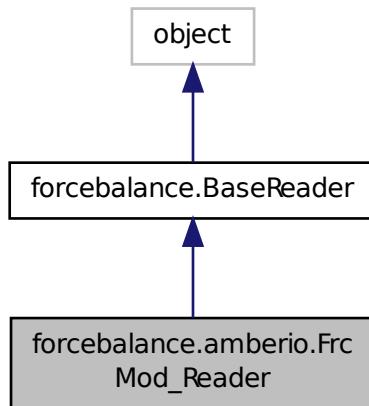
## 8.20 forcebalance.amberio.FrcMod.Reader Class Reference

Finite state machine for parsing FrcMod force field file.

Inheritance diagram for forcebalance.amberio.FrcMod.Reader:



Collaboration diagram for forcebalance.amberio.FrcMod.Reader:



#### Public Member Functions

- def `_init_`
- def `Split`
- def `Whites`
- def `feed`
- def `build.pid`

Returns the parameter type (e.g.

## Public Attributes

- [pdict](#)  
The parameter dictionary (defined in this file)
- [atom](#)  
The atom numbers in the interaction (stored in the parser)
- [dihe](#)  
Whether we're inside the dihedral section.
- [adict](#)  
The frcmod file never has any atoms in it.
- [itype](#)
- [suffix](#)
- [In](#)
- [molatom](#)  
The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.
- [Molecules](#)
- [AtomTypes](#)

### 8.20.1 Detailed Description

Finite state machine for parsing FrcMod force field file.

Definition at line 99 of file amberio.py.

### 8.20.2 Constructor & Destructor Documentation

**def forcebalance.amberio.FrcMod\_Reader.\_\_init\_\_ ( self, fnm )** Definition at line 101 of file amberio.py.

### 8.20.3 Member Function Documentation

**def forcebalance.BaseReader.build\_pid ( self, pfd ) [inherited]** Returns the parameter type (e.g. K in BOND\$K) based on the current interaction type.

Both the 'pdict' dictionary (see [gmlio.pdict](#)) and the interaction type 'state' (here, BONDS) are needed to get the parameter type.

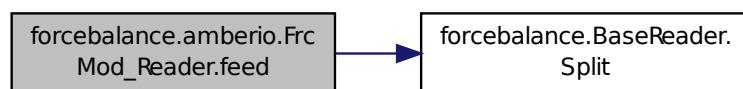
If, however, 'pdict' does not contain the ptype value, a suitable substitute is simply the field number.

Note that if the interaction type state is not set, then it defaults to the file name, so a generic parameter ID is 'filename.line\_num.field\_num'

Definition at line 124 of file \_\_init\_\_.py.

**def forcebalance.amberio.FrcMod\_Reader.feed ( self, line )** Definition at line 119 of file amberio.py.

Here is the call graph for this function:



**def forcebalance.amberio.FrcMod.Reader.Split ( *self*, *line* )** Definition at line 113 of file amberio.py.

**def forcebalance.amberio.FrcMod.Reader.Whites ( *self*, *line* )** Definition at line 116 of file amberio.py.

#### 8.20.4 Member Data Documentation

**forcebalance.amberio.FrcMod.Reader.adict** The frcmod file never has any atoms in it.  
Definition at line 111 of file amberio.py.

**forcebalance.amberio.FrcMod.Reader.atom** The atom numbers in the interaction (stored in the parser)  
Definition at line 107 of file amberio.py.

**forcebalance.BaseReader.AtomTypes [inherited]** Definition at line 97 of file \_\_init\_\_.py.

**forcebalance.amberio.FrcMod.Reader.dihe** Whether we're inside the dihedral section.  
Definition at line 109 of file amberio.py.

**forcebalance.amberio.FrcMod.Reader.itype** Definition at line 130 of file amberio.py.

**forcebalance.BaseReader.In [inherited]** Definition at line 84 of file \_\_init\_\_.py.

**forcebalance.BaseReader.molatom [inherited]** The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.

self.moleculedict = OrderedDict() The listing of 'RES:ATOMNAMES' for atom names in the line This is obviously a placeholder.

Definition at line 94 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules [inherited]** Definition at line 96 of file \_\_init\_\_.py.

**forcebalance.amberio.FrcMod.Reader.pdict** The parameter dictionary (defined in this file)  
Definition at line 105 of file amberio.py.

**forcebalance.amberio.FrcMod.Reader.suffix** Definition at line 165 of file amberio.py.

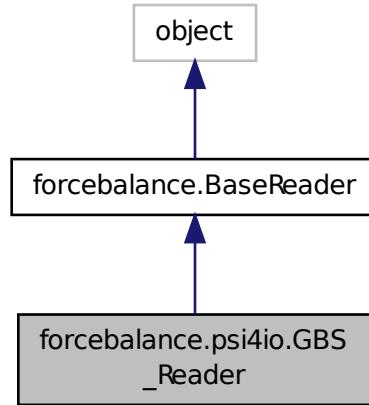
The documentation for this class was generated from the following file:

- [amberio.py](#)

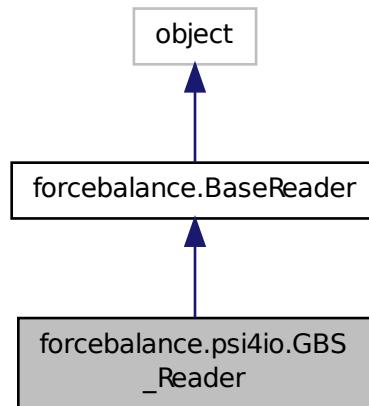
### 8.21 forcebalance.psi4io.GBS\_Reader Class Reference

Interaction type -> Parameter Dictionary.

Inheritance diagram for forcebalance.psi4io.GBS\_Reader:



Collaboration diagram for forcebalance.psi4io.GBS\_Reader:



#### Public Member Functions

- def `_init_`
- def `build_pid`
- def `feed`  
*Feed in a line.*
- def `Split`

- def `Whites`
- def `feed`

### Public Attributes

- `element`
- `amom`
- `last_amom`
- `basis_number`
- `contraction_number`
- `adict`
- `isdata`
- `destroy`
- `In`
- `itype`
- `suffix`
- `pdict`
- `molatom`

*The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.*

- `Molecules`
- `AtomTypes`

#### 8.21.1 Detailed Description

Interaction type -> Parameter Dictionary.

`pdict = {'Exponent':{0:'A', 1:'C'}, 'BASSP' :{0:'A', 1:'B', 2:'C'} }` Finite state machine for parsing basis set files.  
Definition at line 35 of file psi4io.py.

#### 8.21.2 Constructor & Destructor Documentation

`def forcebalance.psi4io.GBS_Reader.__init__ ( self, fnm = None )` Definition at line 37 of file psi4io.py.

#### 8.21.3 Member Function Documentation

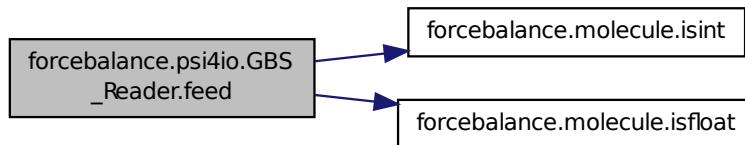
`def forcebalance.psi4io.GBS_Reader.build_pid ( self, pfld )` Definition at line 48 of file psi4io.py.

`def forcebalance.psi4io.GBS_Reader.feed ( self, line, linindep = False )` Feed in a line.  
Parameters

in	line	The line of data
----	------	------------------

Definition at line 61 of file psi4io.py.

Here is the call graph for this function:



```
def forcebalance.BaseReader.feed( self, line ) [inherited] Definition at line 105 of file __init__.py.

def forcebalance.BaseReader.Split( self, line ) [inherited] Definition at line 99 of file __init__.py.

def forcebalance.BaseReader.Whites( self, line ) [inherited] Definition at line 102 of file __init__.py.
```

#### 8.21.4 Member Data Documentation

```
forcebalance.psi4io.GBS_Reader.adict Definition at line 44 of file psi4io.py.

forcebalance.psi4io.GBS_Reader.amom Definition at line 40 of file psi4io.py.

forcebalance.BaseReader.AtomTypes [inherited] Definition at line 97 of file __init__.py.

forcebalance.psi4io.GBS_Reader.basis_number Definition at line 42 of file psi4io.py.

forcebalance.psi4io.GBS_Reader.contraction_number Definition at line 43 of file psi4io.py.

forcebalance.psi4io.GBS_Reader.destroy Definition at line 46 of file psi4io.py.

forcebalance.psi4io.GBS_Reader.element Definition at line 39 of file psi4io.py.

forcebalance.psi4io.GBS_Reader.isdata Definition at line 45 of file psi4io.py.

forcebalance.BaseReader.itype [inherited] Definition at line 85 of file __init__.py.

forcebalance.psi4io.GBS_Reader.last_amom Definition at line 41 of file psi4io.py.

forcebalance.BaseReader.In [inherited] Definition at line 84 of file __init__.py.

forcebalance.BaseReader.molatom [inherited] The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.  
self.molecdict = OrderedDict() The listing of 'RES:ATOMNAMES' for atom names in the line This is obviously a placeholder.  
Definition at line 94 of file __init__.py.

forcebalance.BaseReader.Molecules [inherited] Definition at line 96 of file __init__.py.

forcebalance.BaseReader.pdict [inherited] Definition at line 87 of file __init__.py.

forcebalance.BaseReader.suffix [inherited] Definition at line 86 of file __init__.py.  
The documentation for this class was generated from the following file:

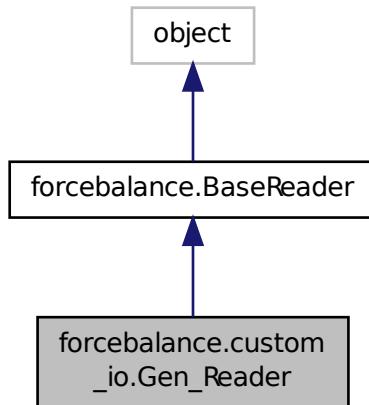
- psi4io.py

```

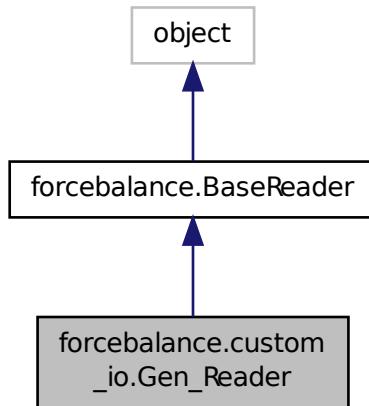
## 8.22 forcebalance.custom.io.Gen\_Reader Class Reference

Finite state machine for parsing custom GROMACS force field files.

Inheritance diagram for forcebalance.custom.io.Gen\_Reader:



Collaboration diagram for forcebalance.custom.io.Gen\_Reader:



### Public Member Functions

- def `__init__`
- def `feed`

*Feed in a line.*

- def [Split](#)
- def [Whites](#)
- def [build.pid](#)

*Returns the parameter type (e.g.*

## Public Attributes

- [sec](#)  
*The current section that we're in.*
- [pdict](#)  
*The parameter dictionary (defined in this file)*
- [itype](#)
- [suffix](#)
- [In](#)
- [adict](#)  
*The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.*
- [molatom](#)  
*The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.*
- [Molecules](#)
- [AtomTypes](#)

### 8.22.1 Detailed Description

Finite state machine for parsing custom GROMACS force field files.

This class is instantiated when we begin to read in a file. The feed(line) method updates the state of the machine, giving it information like the residue we're currently on, the nonbonded interaction type, and the section that we're in. Using this information we can look up the interaction type and parameter type for building the parameter ID.

Definition at line 41 of file `custom.io.py`.

### 8.22.2 Constructor & Destructor Documentation

**def forcebalance.custom.io.Gen\_Reader.\_\_init\_\_ ( self, fnm )** Definition at line 43 of file `custom.io.py`.

### 8.22.3 Member Function Documentation

**def forcebalance.BaseReader.build.pid ( self, pfd ) [inherited]** Returns the parameter type (e.g. K in BONDSK) based on the current interaction type.

Both the 'pdict' dictionary (see `gmlio.pdict`) and the interaction type 'state' (here, BONDS) are needed to get the parameter type.

If, however, 'pdict' does not contain the ptype value, a suitable substitute is simply the field number.

Note that if the interaction type state is not set, then it defaults to the file name, so a generic parameter ID is 'filename.line\_num.field\_num'

Definition at line 124 of file `__init__.py`.

**def forcebalance.custom.io.Gen\_Reader.feed ( self, line )** Feed in a line.

Parameters

---

in	line	The line of data
----	------	------------------

Definition at line 57 of file custom.io.py.

**def forcebalance.BaseReader.Split( self, line ) [inherited]** Definition at line 99 of file \_\_init\_\_.py.

**def forcebalance.BaseReader.Whites( self, line ) [inherited]** Definition at line 102 of file \_\_init\_\_.py.

#### 8.22.4 Member Data Documentation

**forcebalance.BaseReader.adict [inherited]** The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.

Definition at line 89 of file \_\_init\_\_.py.

**forcebalance.BaseReader.AtomTypes [inherited]** Definition at line 97 of file \_\_init\_\_.py.

**forcebalance.custom\_io.Gen\_Reader.itype** Definition at line 60 of file custom.io.py.

**forcebalance.BaseReader.In [inherited]** Definition at line 84 of file \_\_init\_\_.py.

**forcebalance.BaseReader.molatom [inherited]** The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.

self.molecdict = OrderedDict() The listing of 'RES:ATOMNAMES' for atom names in the line This is obviously a placeholder.

Definition at line 94 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules [inherited]** Definition at line 96 of file \_\_init\_\_.py.

**forcebalance.custom\_io.Gen\_Reader.pdict** The parameter dictionary (defined in this file)

Definition at line 49 of file custom.io.py.

**forcebalance.custom\_io.Gen\_Reader.sec** The current section that we're in.

Definition at line 47 of file custom.io.py.

**forcebalance.custom\_io.Gen\_Reader.suffix** Definition at line 79 of file custom.io.py.

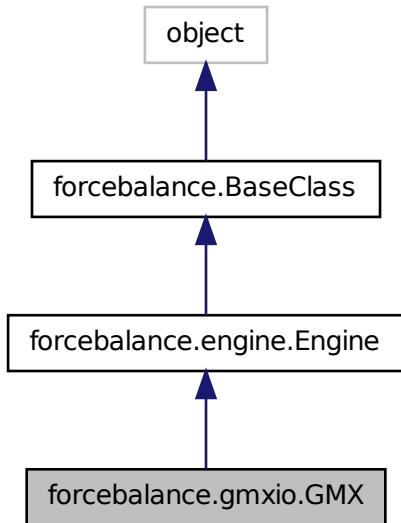
The documentation for this class was generated from the following file:

- [custom.io.py](#)

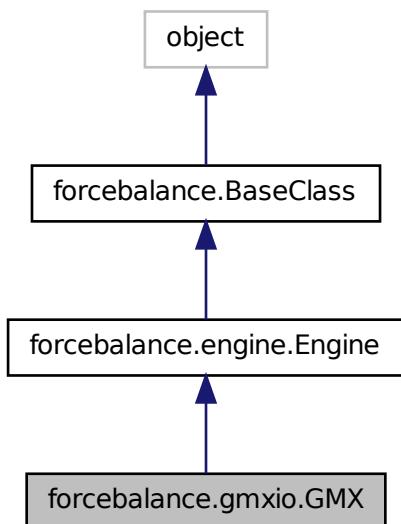
### 8.23 forcebalance.gmxio.GMX Class Reference

Derived from Engine object for carrying out general purpose GROMACS calculations.

Inheritance diagram for forcebalance.gmxio.GMX:



Collaboration diagram for forcebalance.gmxio.GMX:



## Public Member Functions

- def `_init_`  
`Called by init ; Set GROMACS-specific options.`
- def `readsrc`  
`Called by init ; read files from the source directory.`
- def `prepare`  
`Called by init ; prepare the temp directory and figure out the topology.`
- def `links`
- def `callgmx`  
`Call GROMACS; prepend the gmxpath to the call to the GROMACS program.`
- def `warngmx`  
`Call gromacs and allow for certain expected warnings.`
- def `energy_termnames`  
`Get a list of energy term names from the .edr file by parsing a system call to g_energy.`
- def `optimize`  
`Optimize the geometry and align the optimized geometry to the starting geometry.`
- def `evaluate_`  
`Utility function for computing energy, and (optionally) forces and dipoles using GROMACS.`
- def `evaluate_snapshot`  
`Evaluate variables (energies, force and/or dipole) using GROMACS for a single snapshot.`
- def `evaluate_trajectory`  
`Evaluate variables (energies, force and/or dipole) using GROMACS over a trajectory.`
- def `energy_one`  
`Compute the energy using GROMACS for a snapshot.`
- def `energy_force_one`  
`Compute the energy and force using GROMACS for a single snapshot; interfaces with AbInitio target.`
- def `energy`  
`Compute the energy using GROMACS over a trajectory.`
- def `energy_force`  
`Compute the energy and force using GROMACS over a trajectory.`
- def `energy_dipole`
- def `energy_rmsd`  
`Calculate energy of the selected structure (optionally minimize and return the minimized energy and RMSD).`
- def `interaction_energy`  
`Computes the interaction energy between two fragments over a trajectory.`
- def `multipole_moments`  
`Return the multipole moments of the 1st snapshot in Debye and Buckingham units.`
- def `normal_modes`
- def `generate_vsite_positions`
- def `n_snaps`
- def `scd_persnap`
- def `calc_scd`
- def `molecular_dynamics`  
`Method for running a molecular dynamics simulation.`
- def `md`  
`Method for running a molecular dynamics simulation.`
- def `prepare`
- def `__setattr__`
- def `set_option`

## Public Attributes

- **valkw**  
*Valid GROMACS-specific keywords.*
- **gmxsuffix**  
*Disable some optimizations.*
- **gmxpath**  
*The directory containing GROMACS executables (e.g.*
- **top**  
*Attempt to determine file names of .gro, .top, and .mdp files.*
- **mdp**
- **mol**
- **gmx\_defs**
- **pbc**
- **double**  
*Write out the trajectory coordinates to a .gro file.*
- **AtomMask**
- **AtomLists**
- **mdtraj**
- **mdene**
- **name**
- **verbose**
- **target**  
*Engines can get properties from the Target that creates them.*
- **root**
- **srcdir**
- **tempdir**
- **FF**
- **verbose\_options**
- **PrintOptionDict**

### 8.23.1 Detailed Description

Derived from Engine object for carrying out general purpose GROMACS calculations.

Definition at line 490 of file gmxio.py.

### 8.23.2 Constructor & Destructor Documentation

```
def forcebalance.gmxio.GMX.__init__ ( self, name = "gmx", kwargs )
```

 Definition at line 493 of file gmxio.py.

### 8.23.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited]
```

 Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.gmxio.GMX.calc_scd ( self, n_snap, timestep )
```

 Definition at line 1069 of file gmxio.py.

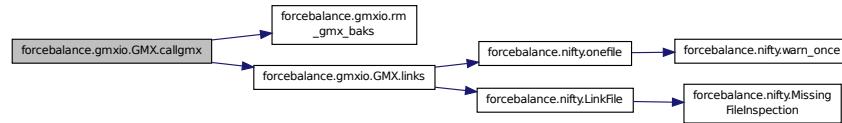
Here is the call graph for this function:



```
def forcebalance.gmxio.GMX.callgmx ( self, command, stdin = None, print_to_screen = False, print_command = False, kwargs ) Call GROMACS; prepend the gmxpath to the call to the GROMACS program.
```

Definition at line 691 of file gmxio.py.

Here is the call graph for this function:



```
def forcebalance.gmxio.GMX.energy ( self, traj = None ) Compute the energy using GROMACS over a trajectory.
```

Definition at line 890 of file gmxio.py.

Here is the call graph for this function:



```
def forcebalance.gmxio.GMX.energy_dipole ( self, traj = None ) Definition at line 903 of file gmxio.py.
```

Here is the call graph for this function:



```
def forcebalance.gmxio.GMX.energy_force ( self, force = True, traj = None ) Compute the energy and force using GROMACS over a trajectory.
```

Definition at line 897 of file gmxio.py.

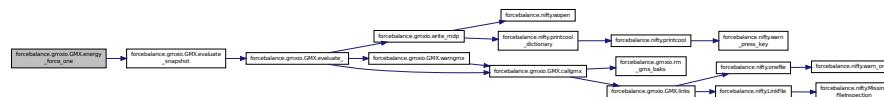
Here is the call graph for this function:



**def forcebalance.gmxio.GMX.energy\_force\_one ( self, shot )** Compute the energy and force using GROMACS for a single snapshot; interfaces with Ablinitio target.

Definition at line 882 of file gmxio.py.

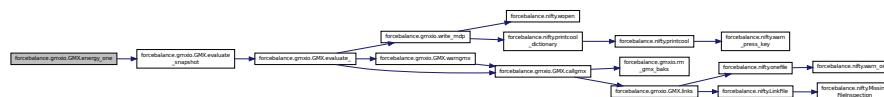
Here is the call graph for this function:



**def forcebalance.gmxio.GMX.energy\_one ( self, shot )** Compute the energy using GROMACS for a snapshot.

Definition at line 875 of file gmxio.py.

Here is the call graph for this function:

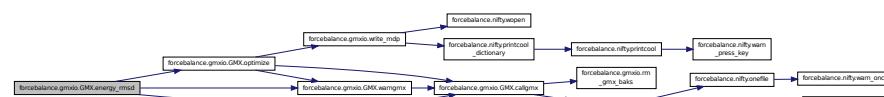


**def forcebalance.gmxio.GMX.energy\_rmsd ( self, shot, optimize = True )** Calculate energy of the selected structure (optionally minimize and return the minimized energy and RMSD).

In kcal/mol.

Definition at line 909 of file gmxio.py.

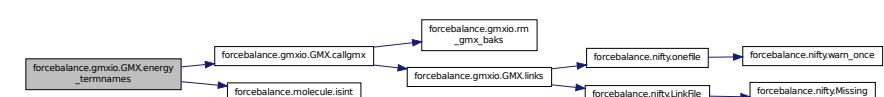
Here is the call graph for this function:



**def forcebalance.gmxio.GMX.energy\_termnames ( self, edrfile = None )** Get a list of energy term names from the .edr file by parsing a system call to g\_energy.

Definition at line 748 of file gmxio.py.

Here is the call graph for this function:



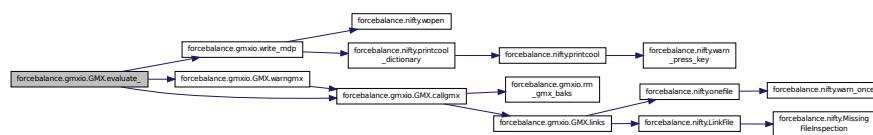
**def forcebalance.gmxio.GMX.evaluate\_( self, force = False, dipole = False, traj = None )** Utility function for computing energy, and (optionally) forces and dipoles using GROMACS.

Inputs: force: Switch for calculating the force. dipole: Switch for calculating the dipole. traj: Trajectory file name. If present, will loop over these snapshots. Otherwise will do a single point evaluation at the current geometry.

Outputs: Result: Dictionary containing energies, forces and/or dipoles.

Definition at line 820 of file gmxio.py.

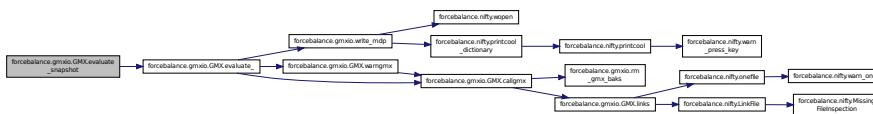
Here is the call graph for this function:



**def forcebalance.gmxio.GMX.evaluate\_snapshot( self, shot, force = False, dipole = False )** Evaluate variables (energies, force and/or dipole) using GROMACS for a single snapshot.

Definition at line 853 of file gmxio.py.

Here is the call graph for this function:

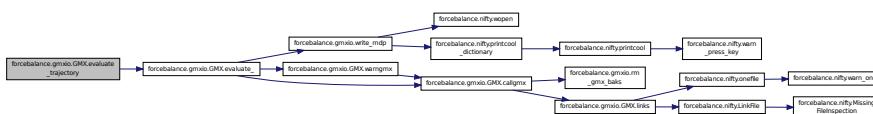


**def forcebalance.gmxio.GMX.evaluate\_trajectory( self, force = False, dipole = False, traj = None )**

Evaluate variables (energies, force and/or dipole) using GROMACS over a trajectory.

Definition at line 862 of file gmxio.py.

Here is the call graph for this function:



**def forcebalance.gmxio.GMX.generate\_vsites\_positions( self )** Definition at line 1047 of file gmxio.py.

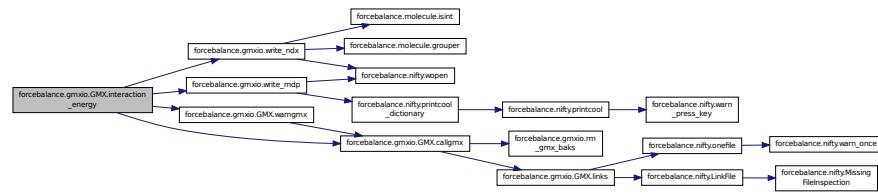
Here is the call graph for this function:



**def forcebalance.gmxio.GMX.interaction\_energy ( self, fraga, fragb )** Computes the interaction energy between two fragments over a trajectory.

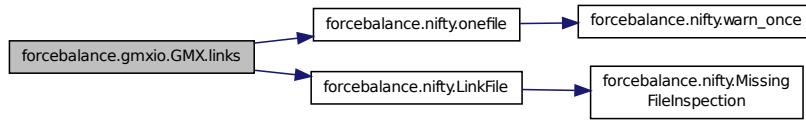
Definition at line 924 of file gmxio.py.

Here is the call graph for this function:



**def forcebalance.gmxio.GMX.links ( self )** Definition at line 675 of file gmxio.py.

Here is the call graph for this function:



**def forcebalance.gmxio.GMX.md ( self, nsteps = 0, nequil = 0, verbose = False, deffnm = None, kwargs )**

) Method for running a molecular dynamics simulation.

A little different than molecular\_dynamics (for [thermo.py](#))

Required arguments:

nsteps = (int) Number of total time steps

nequil = (int) Number of additional time steps at the beginning  
for equilibration

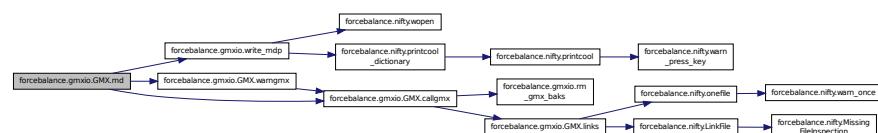
verbose = (bool) Be loud and noisy

deffnm = (string) default names for simulation output files

The simulation data is written to the working directory.

Definition at line 1260 of file gmxio.py.

Here is the call graph for this function:



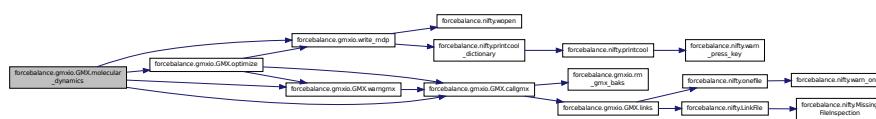
```
def forcebalance.gmxio.GMX.molecular_dynamics ( self, nsteps, timestep, temperature = None, pressure = None, nequil = 0, nsave = 0, minimize = True, threads = None, verbose = False, bilayer = False, kwargs ) Method for running a molecular dynamics simulation.
```

Required arguments: nsteps = (int) Number of total time steps timestep = (float) Time step in FEMTOSECOND-S temperature = (float) Temperature control (Kelvin) pressure = (float) Pressure control (atmospheres) nequil = (int) Number of additional time steps at the beginning for equilibration nsave = (int) Step interval for saving data minimize = (bool) Perform an energy minimization prior to dynamics threads = (int) Number of MPI-threads

Returns simulation data: Rhos = (array) Density in kilogram m<sup>-3</sup> Potentials = (array) Potential energies Kinetics = (array) Kinetic energies Volumes = (array) Box volumes Dips = (3xN array) Dipole moments EComps = (dict) Energy components Als = (array) Average area per lipid in nm<sup>2</sup> Scds = (Nx28 array) Deuterium order parameter

Definition at line 1110 of file gmxio.py.

Here is the call graph for this function:

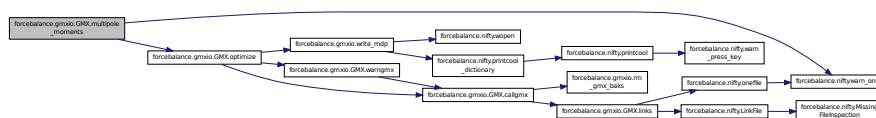


```
def forcebalance.gmxio.GMX.multipole_moments ( self, shot = 0, optimize = True, polarizability = False )
```

) Return the multipole moments of the 1st snapshot in Debye and Buckingham units.

Definition at line 960 of file gmxio.py.

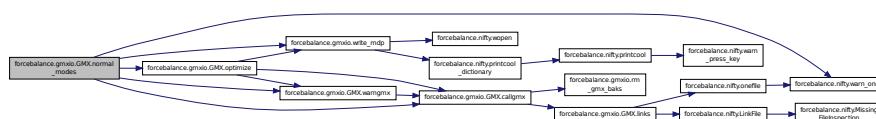
Here is the call graph for this function:



```
def forcebalance.gmxio.GMX.n_snaps ( self, nsteps, step_interval, timestep ) Definition at line 1055 of file gmxio.py.
```

```
def forcebalance.gmxio.GMX.normal_modes ( self, shot = 0, optimize = True ) Definition at line 1011 of file gmxio.py.
```

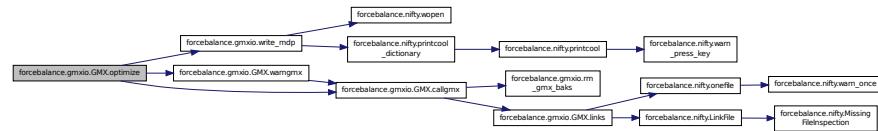
Here is the call graph for this function:



```
def forcebalance.gmxio.GMX.optimize ( self, shot = 0, crit = 1e-4, kwargs ) Optimize the geometry and align the optimized geometry to the starting geometry.
```

Definition at line 779 of file gmxio.py.

Here is the call graph for this function:



**def forcebalance.engine.Engine.prepare( self, kwargs ) [inherited]** Definition at line 95 of file engine.py.

**def forcebalance.gmxio.GMX.prepare( self, pbc = False, kwargs )** Called by **init** ; prepare the temp directory and figure out the topology.

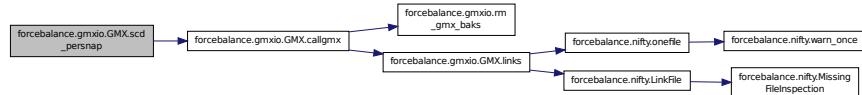
Definition at line 552 of file gmxio.py.

**def forcebalance.gmxio.GMX.readsrc( self, kwargs )** Called by **init** ; read files from the source directory.

Definition at line 536 of file gmxio.py.

**def forcebalance.gmxio.GMX.scd\_persnap( self, ndx, timestep, final\_frame )** Definition at line 1058 of file gmxio.py.

Here is the call graph for this function:



**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.gmxio.GMX.setopts( self, kwargs )** Called by **init** ; Set GROMACS-specific options.

Definition at line 500 of file gmxio.py.

**def forcebalance.gmxio.GMX.warngmx( self, command, warnings = [], maxwarn = 1, kwargs )** Call gromacs and allow for certain expected warnings.

Definition at line 707 of file gmxio.py.

Here is the call graph for this function:



#### 8.23.4 Member Data Documentation

**forcebalance.gmxio.GMX.AtomLists** Definition at line 643 of file gmxio.py.

**forcebalance.gmxio.GMX.AtomMask** Definition at line 642 of file gmxio.py.

**forcebalance.gmxio.GMX.double** Write out the trajectory coordinates to a .gro file.

At this point, we could have gotten a .mdp file from the target folder or as part of the force field. If it still missing, then we may write a default. Call grompp followed by gmxdump to read the trajectory

Definition at line 637 of file gmxio.py.

**forcebalance.engine.Engine.FF [inherited]** Definition at line 65 of file engine.py.

**forcebalance.gmxio.GMX.gmx\_defs** Definition at line 555 of file gmxio.py.

**forcebalance.gmxio.GMX.gmxpath** The directory containing GROMACS executables (e.g.

mdrun)

Definition at line 518 of file gmxio.py.

**forcebalance.gmxio.GMX.gmxsuffix** Disable some optimizations.

The suffix to GROMACS executables, e.g. '\_d' for double precision.

Definition at line 510 of file gmxio.py.

**forcebalance.gmxio.GMX.mdene** Definition at line 1313 of file gmxio.py.

**forcebalance.gmxio.GMX.mdp** Definition at line 541 of file gmxio.py.

**forcebalance.gmxio.GMX.mdtraj** Definition at line 1183 of file gmxio.py.

**forcebalance.gmxio.GMX.mol** Definition at line 543 of file gmxio.py.

**forcebalance.engine.Engine.name [inherited]** Definition at line 48 of file engine.py.

**forcebalance.gmxio.GMX.pbc** Definition at line 560 of file gmxio.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.engine.Engine.root [inherited]** Definition at line 56 of file engine.py.

**forcebalance.engine.Engine.srkdir [inherited]** Definition at line 57 of file engine.py.

**forcebalance.engine.Engine.target [inherited]** Engines can get properties from the Target that creates them.

Definition at line 55 of file engine.py.

**forcebalance.engine.Engine.tempdir [inherited]** Definition at line 58 of file engine.py.

**forcebalance.gmxio.GMX.top** Attempt to determine file names of .gro, .top, and .mdp files.

Link files into the temp directory.

Write the appropriate coordinate files.

Definition at line 540 of file gmxio.py.

**forcebalance.gmxio.GMX.valkwd** Valid GROMACS-specific keywords.

Definition at line 495 of file gmxio.py.

**forcebalance.engine.Engine.verbose [inherited]** Definition at line 50 of file engine.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

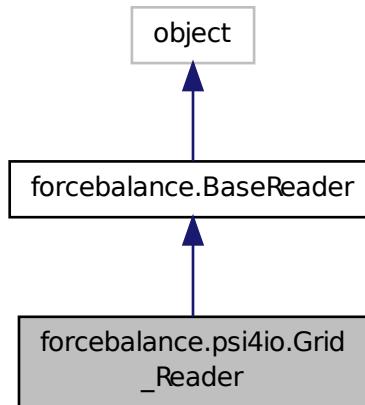
The documentation for this class was generated from the following file:

- [gmxio.py](#)

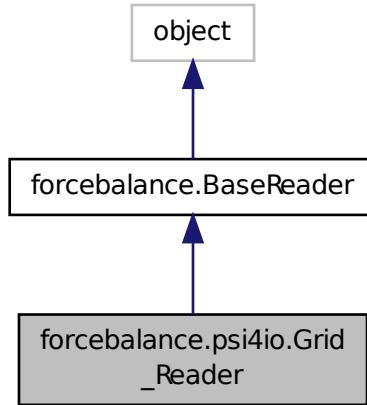
## 8.24 forcebalance.psi4io.Grid\_Reader Class Reference

Finite state machine for parsing DVR grid files.

Inheritance diagram for forcebalance.psi4io.Grid\_Reader:



Collaboration diagram for forcebalance.psi4io.Grid\_Reader:



### Public Member Functions

- def `_init_`
- def `build_pid`
- def `feed`  
*Feed in a line.*
- def `Split`
- def `Whites`
- def `feed`

### Public Attributes

- `element`
- `point`
- `radii`
- `isdata`
- `In`
- `itype`
- `suffix`
- `pdict`
- `adict`

*The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.*

- `molatom`

*The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.*

- `Molecules`
- `AtomTypes`

### 8.24.1 Detailed Description

Finite state machine for parsing DVR grid files.  
Definition at line 250 of file psi4io.py.

### 8.24.2 Constructor & Destructor Documentation

**def forcebalance.psi4io.Grid\_Reader.\_\_init\_\_ ( self, fnm = None )** Definition at line 252 of file psi4io.py.

### 8.24.3 Member Function Documentation

**def forcebalance.psi4io.Grid\_Reader.build\_pid ( self, pfd )** Definition at line 258 of file psi4io.py.

**def forcebalance.BaseReader.feed ( self, line ) [inherited]** Definition at line 105 of file \_\_init\_\_.py.

**def forcebalance.psi4io.Grid\_Reader.feed ( self, line, linindep = False )** Feed in a line.  
Parameters

in	line	The line of data
----	------	------------------

Definition at line 273 of file psi4io.py.

**def forcebalance.BaseReader.Split ( self, line ) [inherited]** Definition at line 99 of file \_\_init\_\_.py.

**def forcebalance.BaseReader.Whites ( self, line ) [inherited]** Definition at line 102 of file \_\_init\_\_.py.

### 8.24.4 Member Data Documentation

**forcebalance.BaseReader.adict [inherited]** The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.  
Definition at line 89 of file \_\_init\_\_.py.

**forcebalance.BaseReader.AtomTypes [inherited]** Definition at line 97 of file \_\_init\_\_.py.

**forcebalance.psi4io.Grid\_Reader.element** Definition at line 254 of file psi4io.py.

**forcebalance.psi4io.Grid\_Reader.isdata** Definition at line 284 of file psi4io.py.

**forcebalance.BaseReader.itype [inherited]** Definition at line 85 of file \_\_init\_\_.py.

**forcebalance.BaseReader.In [inherited]** Definition at line 84 of file \_\_init\_\_.py.

**forcebalance.BaseReader.molatom [inherited]** The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.

self.molecdict = OrderedDict() The listing of 'RES:ATOMNAMES' for atom names in the line This is obviously a placeholder.

Definition at line 94 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules [inherited]** Definition at line 96 of file \_\_init\_\_.py.

**forcebalance.BaseReader.pdict [inherited]** Definition at line 87 of file \_\_init\_\_.py.

**forcebalance.psi4io.Grid\_Reader.point** Definition at line 255 of file psi4io.py.

**forcebalance.psi4io.Grid\_Reader.radius** Definition at line 256 of file psi4io.py.

**forcebalance.BaseReader.suffix [inherited]** Definition at line 86 of file \_\_init\_\_.py.

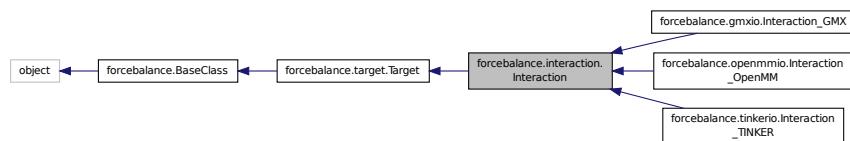
The documentation for this class was generated from the following file:

- [psi4io.py](#)

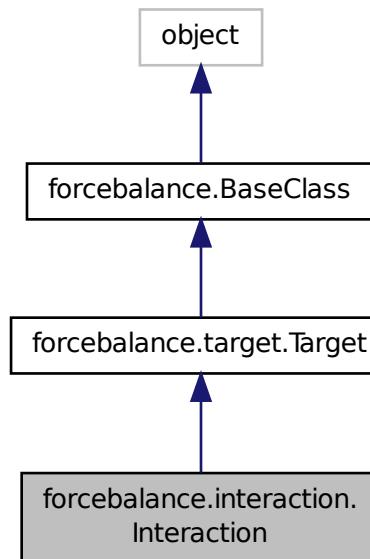
## 8.25 forcebalance.interaction.Interaction Class Reference

Subclass of Target for fitting force fields to interaction energies.

Inheritance diagram for forcebalance.interaction.Interaction:



Collaboration diagram for forcebalance.interaction.Interaction:



### Public Member Functions

- def [\\_\\_init\\_\\_](#)
- def [read\\_reference\\_data](#)

*Read the reference ab initio data from a file such as qdata.txt.*

- def `indicate`
- def `get`  
*Evaluate objective function.*
- def `get_X`  
*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`  
*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`  
*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`  
*Computes the objective function contribution and its gradient.*
- def `get_H`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `select1`  
*Number of snapshots.*
- `select2`  
*Set fragment 2.*
- `eqm`  
*Set upper cutoff energy.*
- `label`  
*Snapshot label, useful for graphing.*

- **qfnm**  
*The qdata.txt file that contains the QM energies and forces.*
- **e\_err**
- **e\_err\_pct**
- **ns**  
*Read in the trajectory file.*
- **mol**
- **engine**  
*Build keyword dictionaries to pass to engine.*
- **divisor**  
*Read in the reference data.*
- **prefactor**
- **weight**
- **emm**
- **objective**
- **rd**  
*Root directory of the whole project.*
- **pgrad**  
*Iteration where we turn on zero-gradient skipping.*
- **tempbase**  
*Relative directory of target.*
- **tempdir**
- **rundir**  
*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- **FF**  
*Need the forcefield (here for now)*
- **xct**  
*Counts how often the objective function was computed.*
- **gct**  
*Counts how often the gradient was computed.*
- **hct**  
*Counts how often the Hessian was computed.*
- **read\_indicate**  
*Whether to read indicate.log from file when restarting an aborted run.*
- **write\_indicate**  
*Whether to write indicate.log at every iteration (true for all but remote.)*
- **read\_objective**  
*Whether to read objective.p from file when restarting an aborted run.*
- **write\_objective**  
*Whether to write objective.p at every iteration (true for all but remote.)*
- **verbose\_options**
- **PrintOptionDict**

### 8.25.1 Detailed Description

Subclass of Target for fitting force fields to interaction energies.

Currently TINKER is supported.

We introduce the following concepts:

- The number of snapshots
- The reference interaction energies and the file they belong in (qdata.txt)

This subclass contains the 'get' method for building the objective function from any simulation software (a driver to run the program and read output is still required).

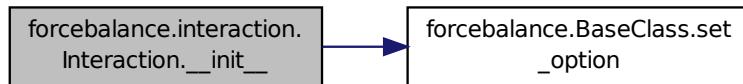
Definition at line 35 of file interaction.py.

### 8.25.2 Constructor & Destructor Documentation

```
def forcebalance.interaction.Interaction.__init__( self, options, tgt_opts, forcefield )
```

 Definition at line 38 of file interaction.py.

Here is the call graph for this function:



### 8.25.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__( self, key, value ) [inherited]
```

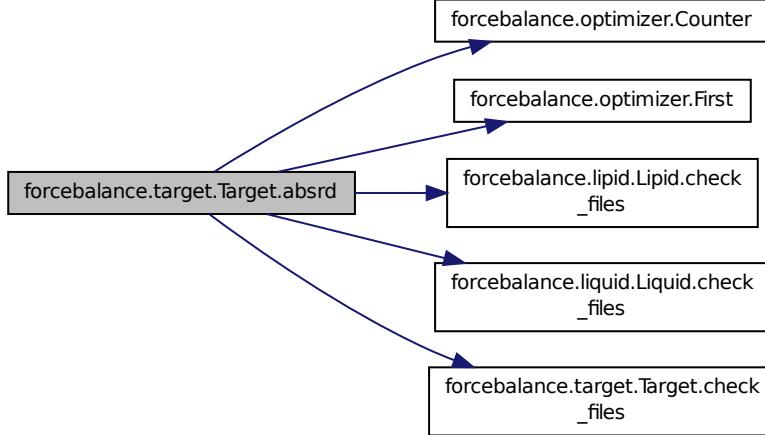
 Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.absrd( self, inum = None ) [inherited]
```

 Supply the correct directory specified by user's "read" option.

Definition at line 393 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.check\_files( self, there ) [inherited]** Check this directory for the presence of readable files when the 'read' option is set.

Definition at line 364 of file target.py.

**def forcebalance.interaction.Interaction.get( self, mvals, AGrad = False, AHess = False )** Evaluate objective function.

Definition at line 157 of file interaction.py.

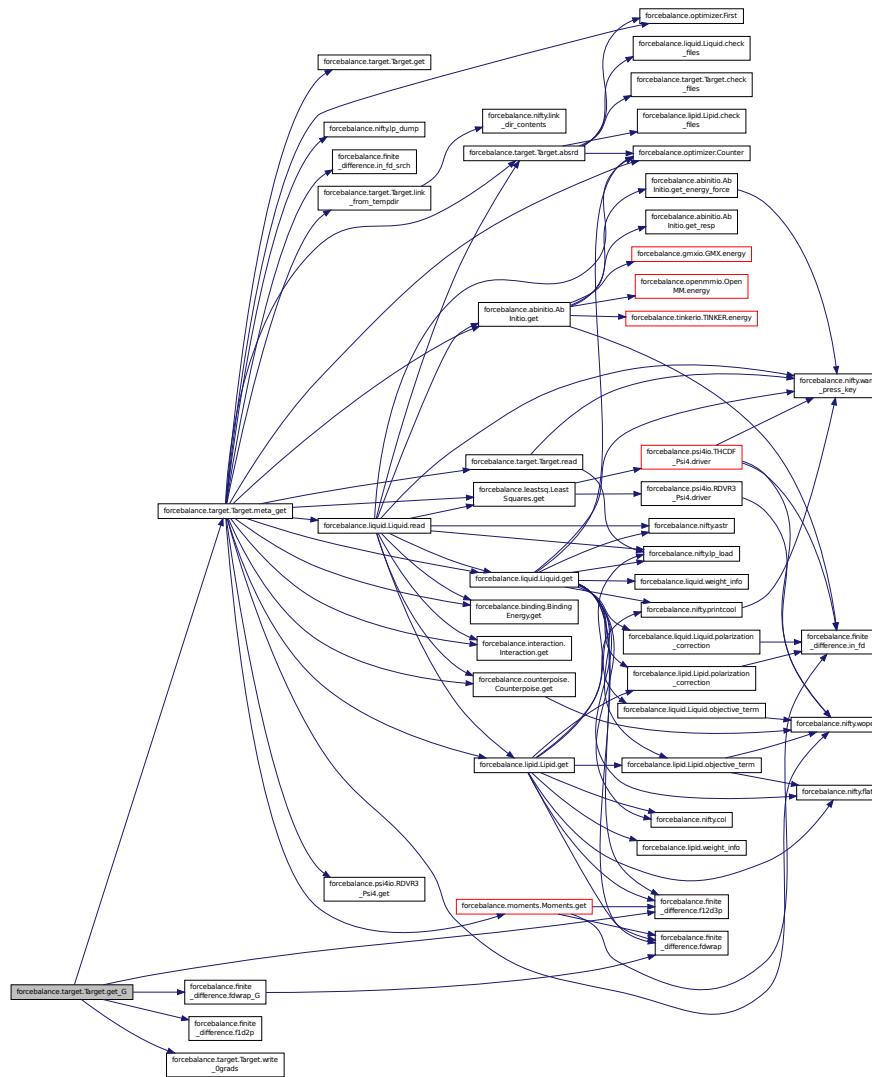
**def forcebalance.target.Target.get\_G( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the `fd1.pids` and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



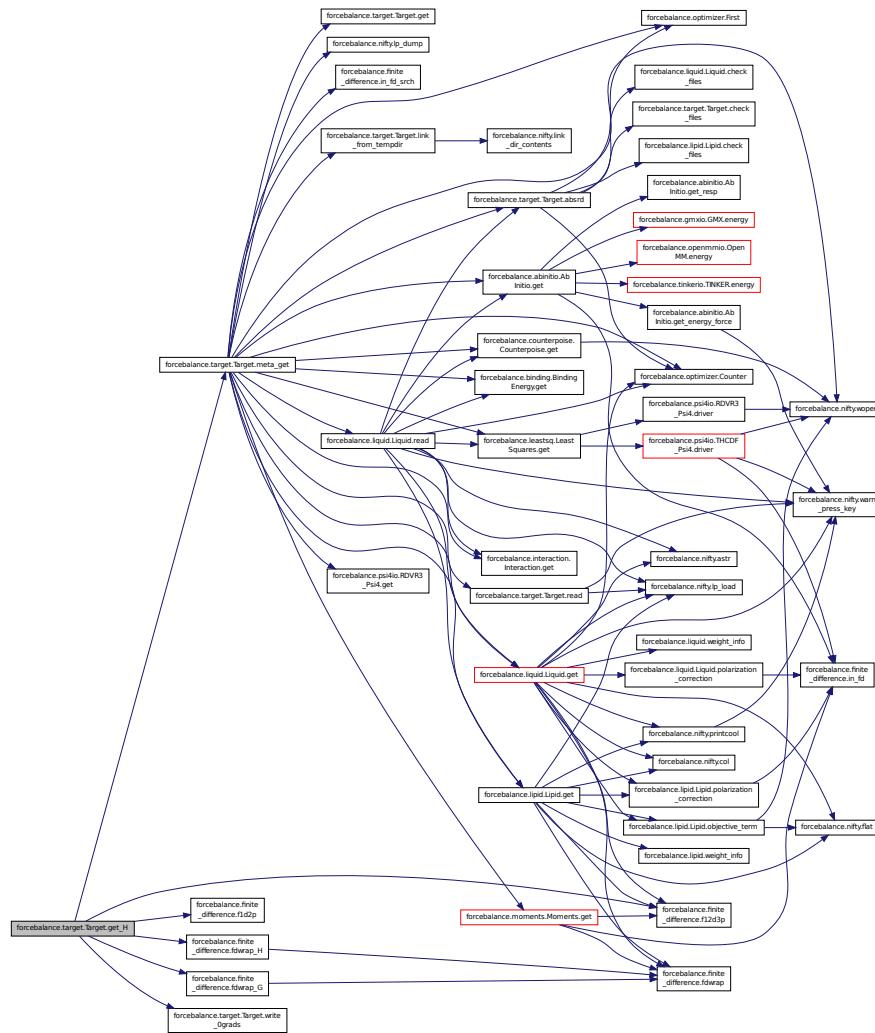
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

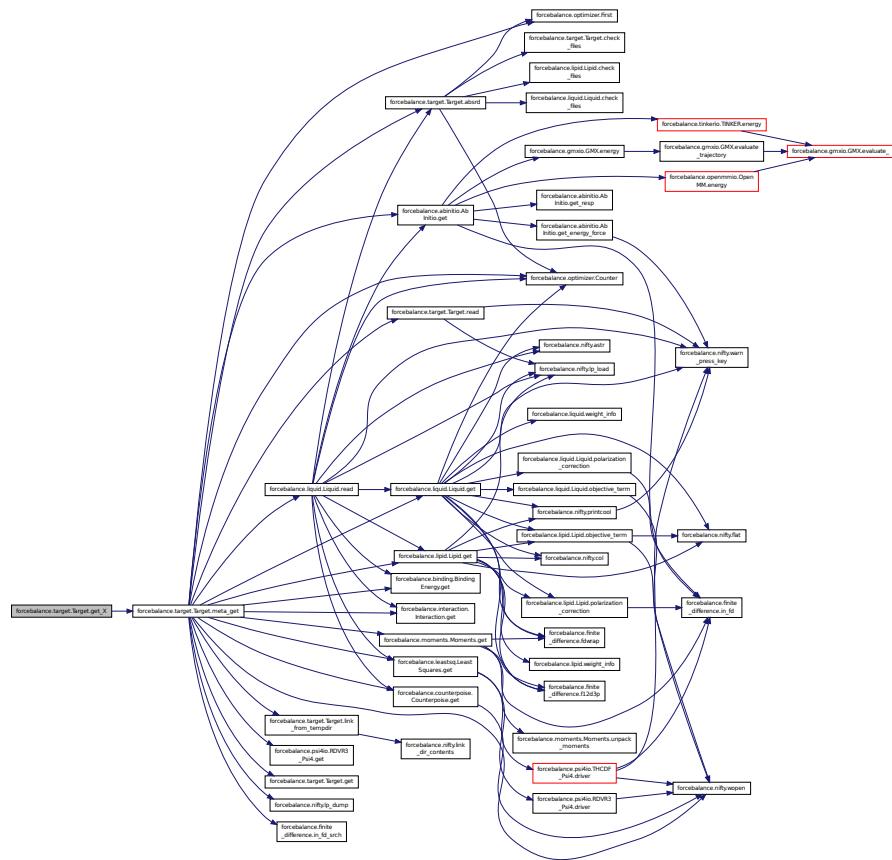
Here is the call graph for this function:



**def forcebalance.target.Target.get.X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

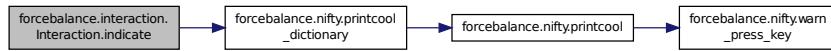
Definition at line 184 of file target.py.

Here is the call graph for this function:



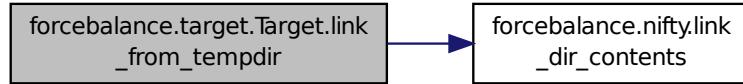
**def forcebalance.interaction.Interaction.indicate ( self )** Definition at line 141 of file interaction.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

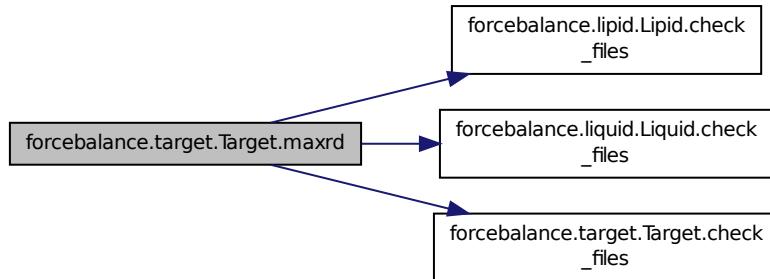
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

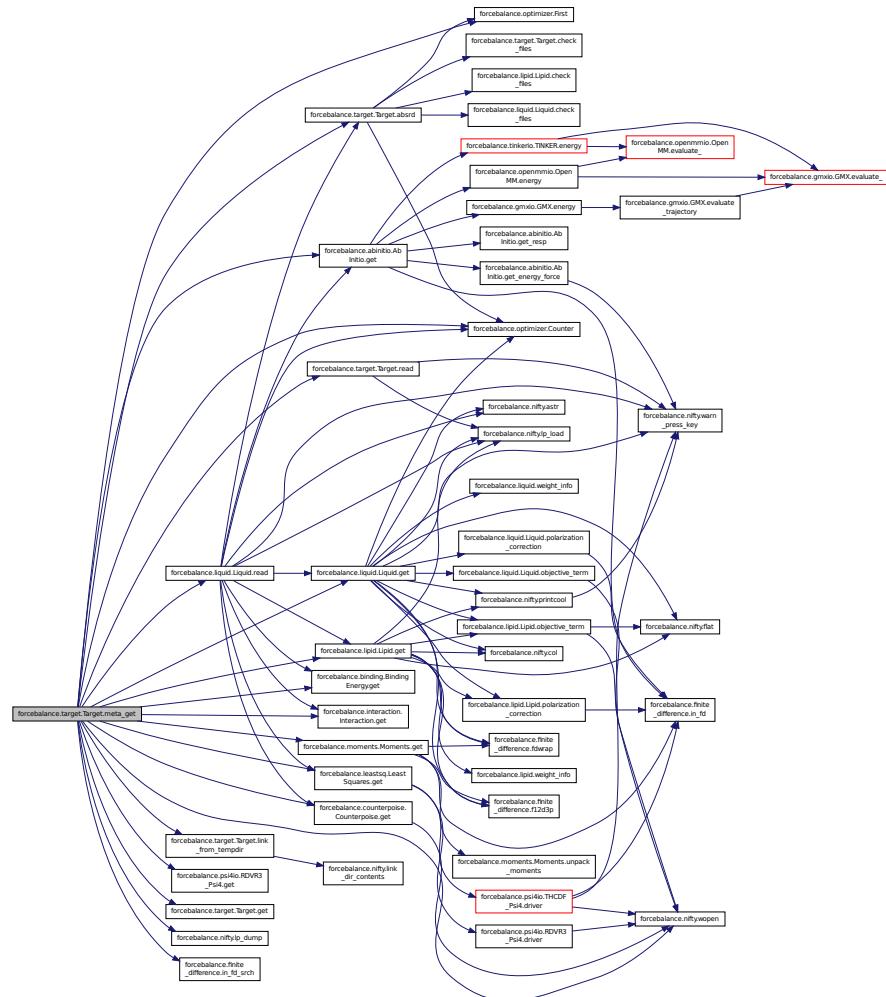


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

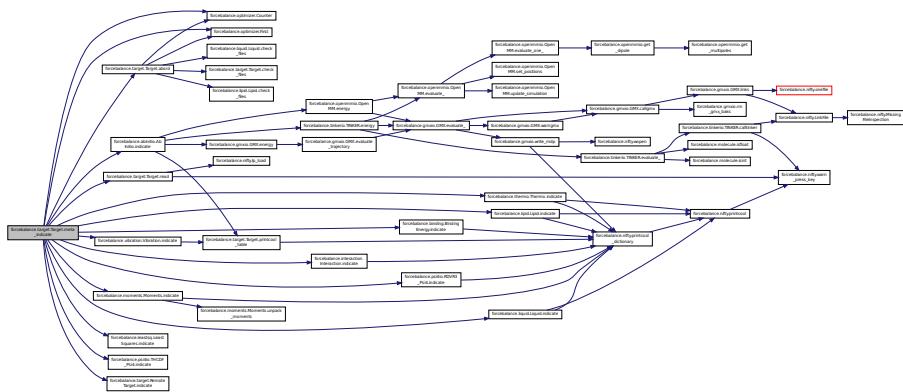
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

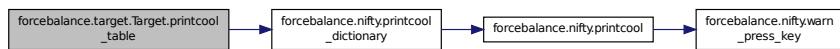
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

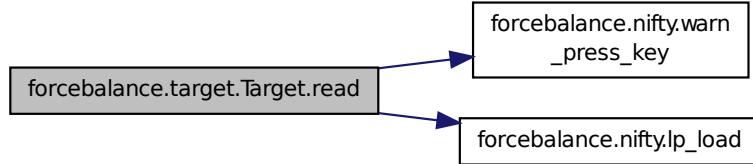
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.interaction.Interaction.read\_reference\_data ( self )** Read the reference ab initio data from a file such as qdata.txt.

After reading in the information from qdata.txt, it is converted into kcal/mol.

Definition at line 124 of file interaction.py.

**def forcebalance.target.Target.refresh\_temp\_directory ( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

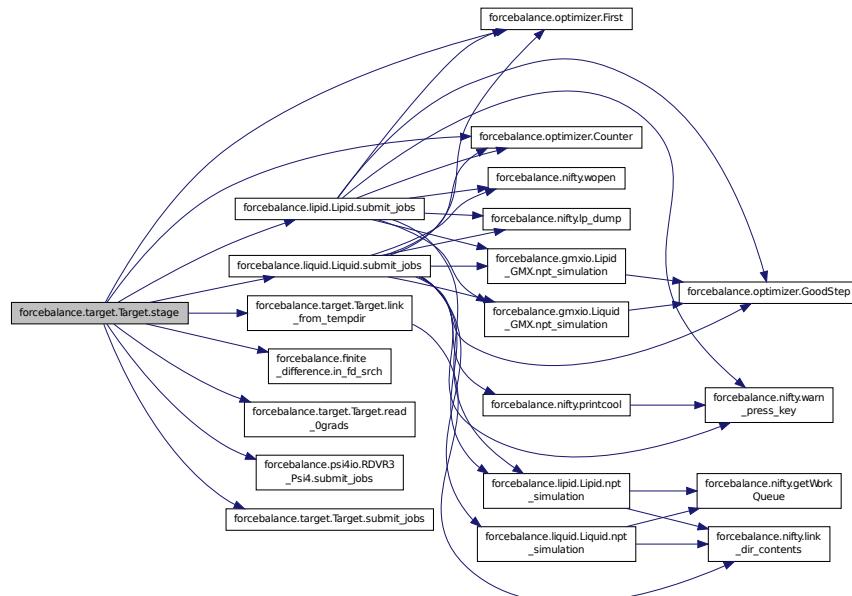
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

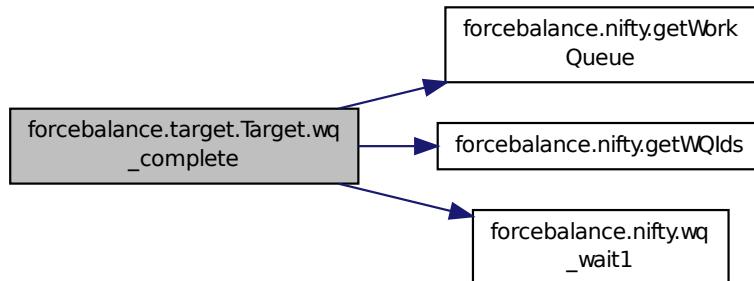


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.write_0grads( self, Ans ) [inherited] Write a file to the target directory containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

#### 8.25.4 Member Data Documentation

**forcebalance.interaction.Interaction.divisor** Read in the reference data.

Definition at line 95 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err** Definition at line 77 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err\_pct** Definition at line 78 of file interaction.py.

**forcebalance.interaction.Interaction.emm** Definition at line 194 of file interaction.py.

**forcebalance.interaction.Interaction.engine** Build keyword dictionaries to pass to engine.

Definition at line 88 of file interaction.py.

**forcebalance.interaction.Interaction.eqm** Set upper cutoff energy.

Reference (QM) interaction energies

Definition at line 72 of file interaction.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.interaction.Interaction.label** Snapshot label, useful for graphing.

Definition at line 74 of file interaction.py.

**forcebalance.interaction.Interaction.mol** Definition at line 81 of file interaction.py.

**forcebalance.interaction.Interaction.ns** Read in the trajectory file.

Definition at line 80 of file interaction.py.

**forcebalance.interaction.Interaction.objective** Definition at line 195 of file interaction.py.

**forcebalance.target.Target.pgrad** [inherited] Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.interaction.Interaction.prefactor** Definition at line 113 of file interaction.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.interaction.Interaction.qfnm** The qdata.txt file that contains the QM energies and forces.

Definition at line 76 of file interaction.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.interaction.Interaction.select1** Number of snapshots.

Do we call Q-Chem for dielectric energies? (Currently needs to be fixed) Do we put the reference energy into the denominator? Do we put the reference energy into the denominator? What is the energy denominator? Set fragment 1

Definition at line 60 of file interaction.py.

**forcebalance.interaction.Interaction.select2** Set fragment 2.

Definition at line 65 of file interaction.py.

**forcebalance.target.Target.tempbase [inherited]** Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir [inherited]** Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.interaction.Interaction.weight** Definition at line 161 of file interaction.py.

**forcebalance.target.Target.write\_indicate [inherited]** Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective [inherited]** Whether to write objective.p at every iteration (true for all but remote.)

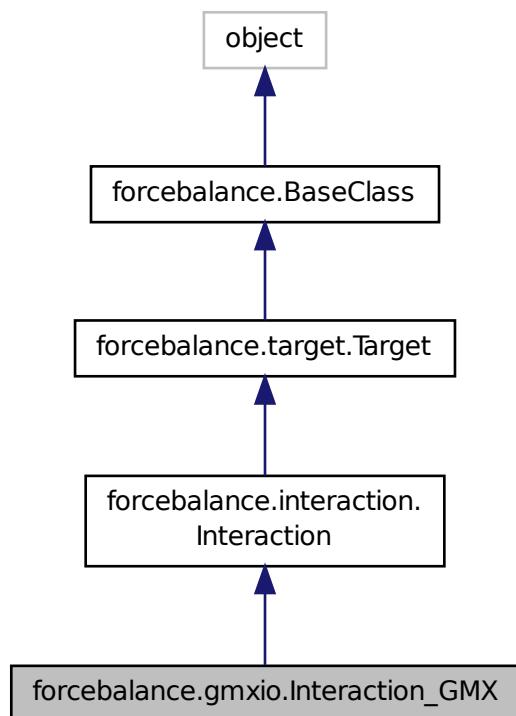
Definition at line 174 of file target.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.  
Definition at line 162 of file target.py.  
The documentation for this class was generated from the following file:  
• [interaction.py](#)

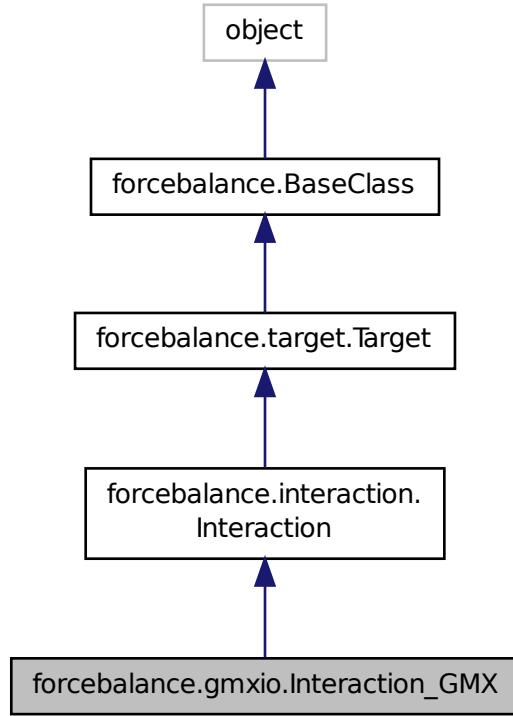
## 8.26 forcebalance.gmxio.Interaction\_GMX Class Reference

Interaction energy matching using GROMACS.

Inheritance diagram for forcebalance.gmxio.Interaction\_GMX:



Collaboration diagram for forcebalance.gmxio.Interaction\_GMX:



### Public Member Functions

- def `__init__`
- def `read_reference_data`

*Read the reference ab initio data from a file such as qdata.txt.*
- def `indicate`
- def `get`

*Evaluate objective function.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_0grads`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_0grads`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*
- def `get_H`

*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`

- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `engine_`  
*Default file names for coordinates, top and mdp files.*
- `select1`  
*Number of snapshots.*
- `select2`  
*Set fragment 2.*
- `eqm`  
*Set upper cutoff energy.*
- `label`  
*Snapshot label, useful for graphing.*
- `qfnm`  
*The qdata.txt file that contains the QM energies and forces.*
- `e_err`
- `e_err_pct`
- `ns`  
*Read in the trajectory file.*
- `mol`
- `engine`  
*Build keyword dictionaries to pass to engine.*
- `divisor`  
*Read in the reference data.*
- `prefactor`

- `weight`
- `emm`
- `objective`
- `rd`  
*Root directory of the whole project.*
- `pgrad`  
*Iteration where we turn on zero-gradient skipping.*
- `tempbase`  
*Relative directory of target.*
- `tempdir`
- `rundir`  
`self.tempdir = os.path.join('temp',self.name)` *The directory in which the simulation is running - this can be updated.*
- `FF`  
*Need the forcefield (here for now)*
- `xct`  
*Counts how often the objective function was computed.*
- `gct`  
*Counts how often the gradient was computed.*
- `hct`  
*Counts how often the Hessian was computed.*
- `read_indicate`  
*Whether to read indicate.log from file when restarting an aborted run.*
- `write_indicate`  
*Whether to write indicate.log at every iteration (true for all but remote.)*
- `read_objective`  
*Whether to read objective.p from file when restarting an aborted run.*
- `write_objective`  
*Whether to write objective.p at every iteration (true for all but remote.)*
- `verbose_options`
- `PrintOptionDict`

### 8.26.1 Detailed Description

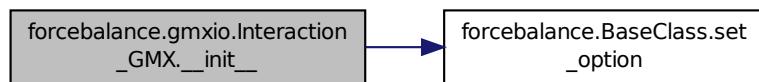
Interaction energy matching using GROMACS.

Definition at line 1469 of file gmxio.py.

### 8.26.2 Constructor & Destructor Documentation

`def forcebalance.gmxio.Interaction_GMX.__init__( self, options, tgt_opts, forcefield )` Definition at line 1470 of file gmxio.py.

Here is the call graph for this function:



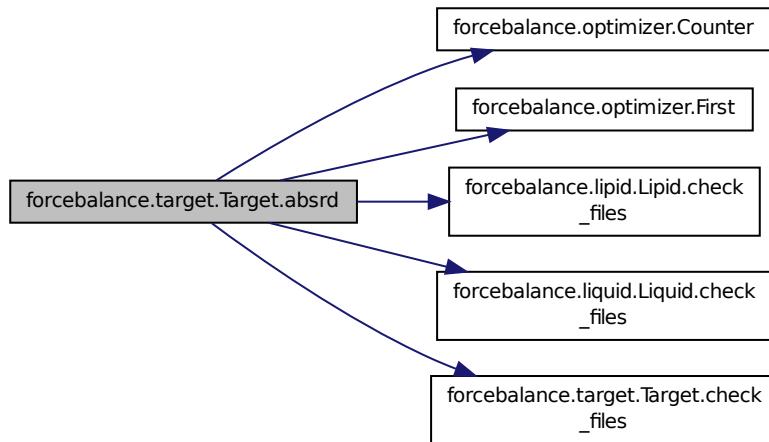
### 8.26.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.check_files( self, there ) [inherited] Check this directory for the presence of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.interaction.Interaction.get( self, mvals, AGrad = False, AHess = False ) [inherited] Evaluate objective function.
```

Definition at line 157 of file interaction.py.

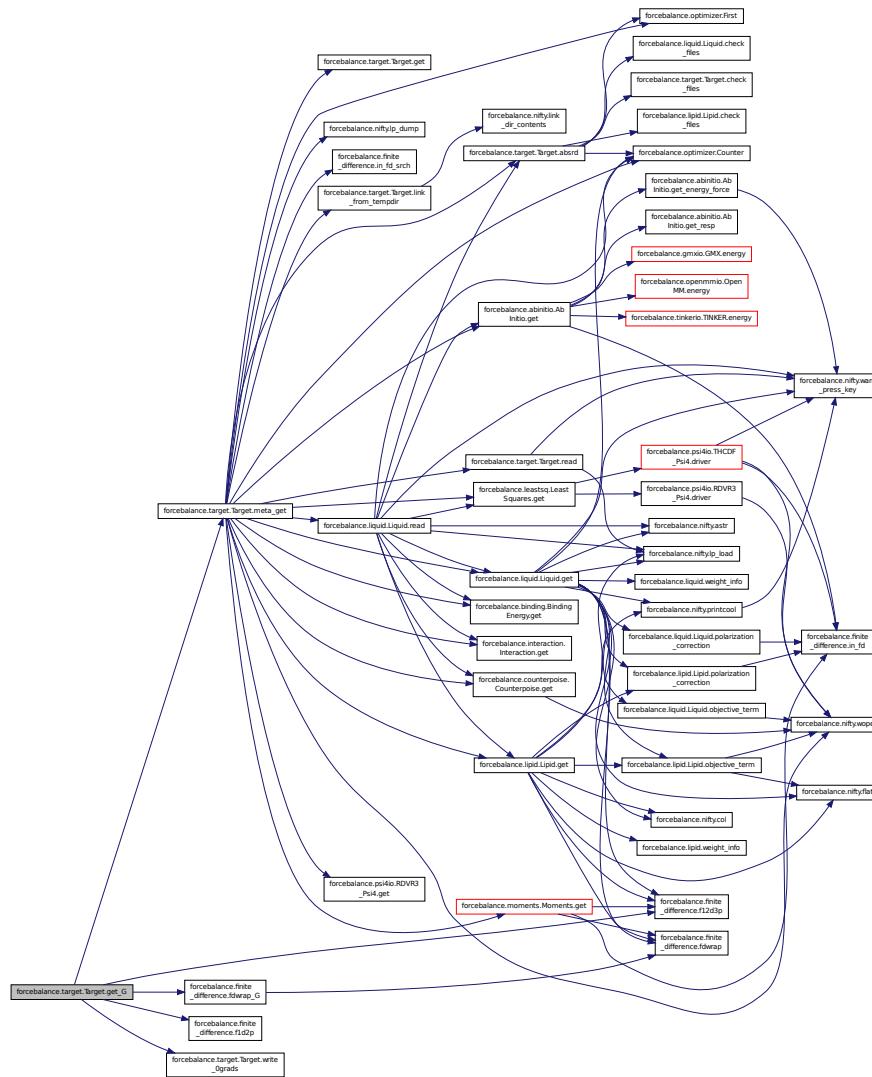
```
def forcebalance.target.Target.get_G( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



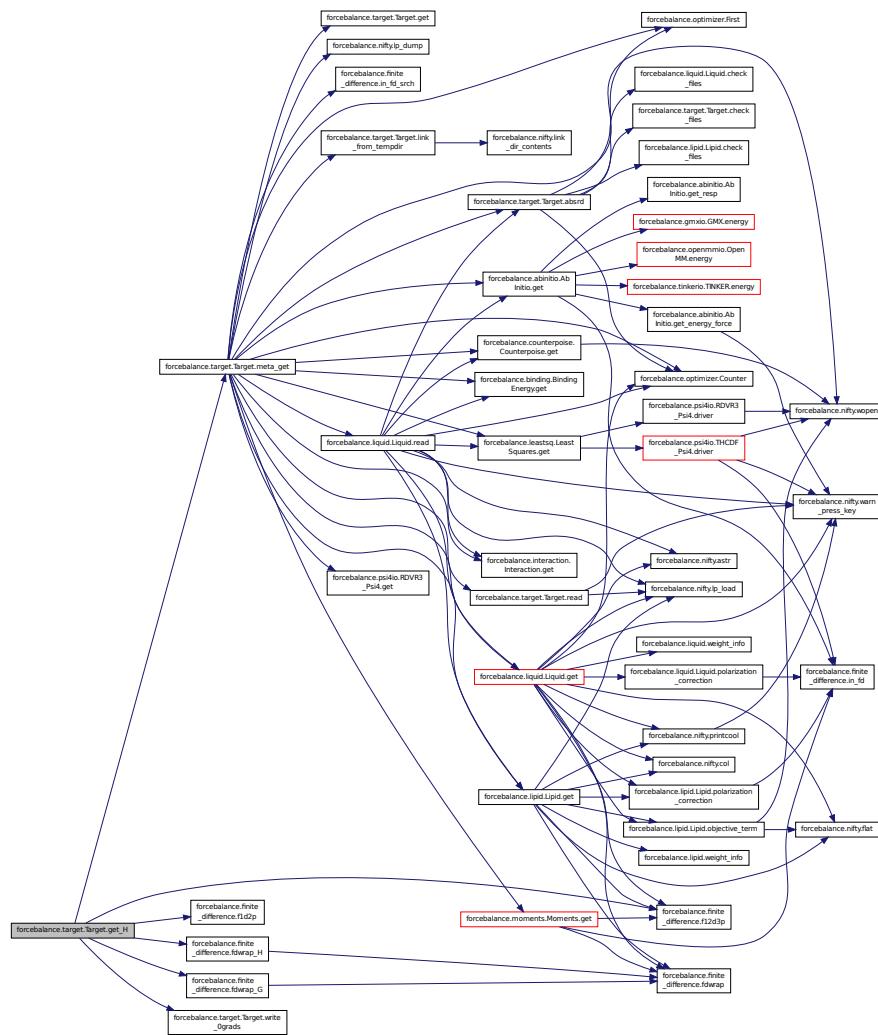
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

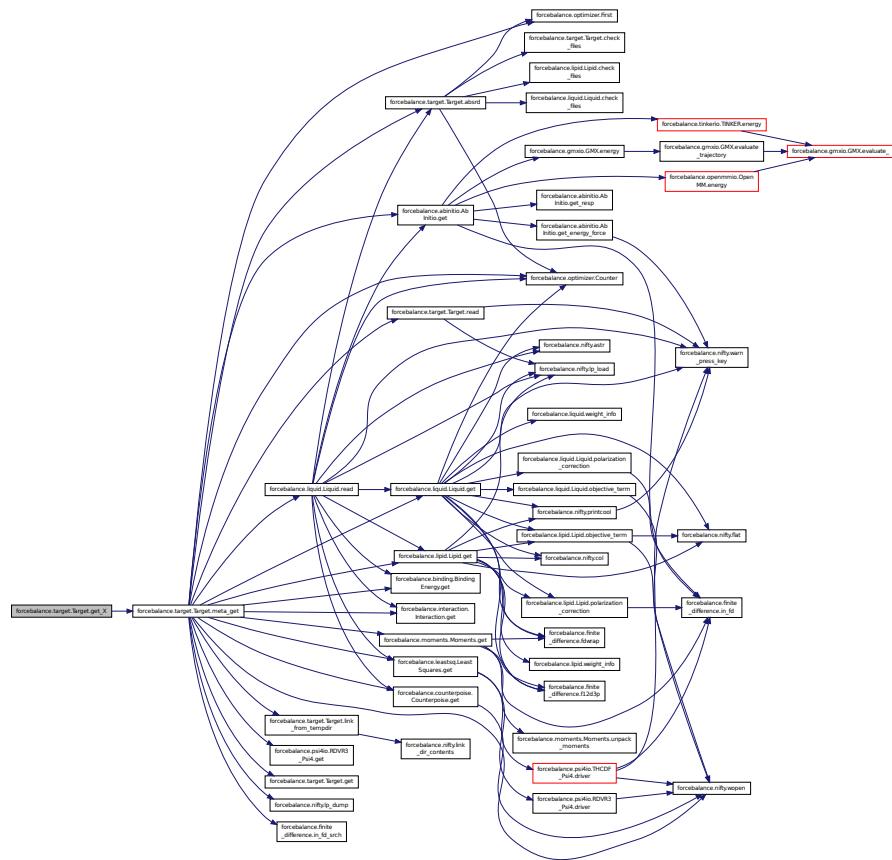
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

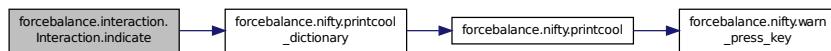
Definition at line 184 of file target.py.

Here is the call graph for this function:



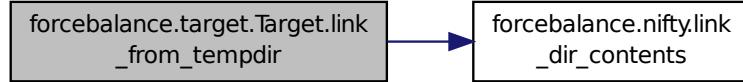
**def forcebalance.interaction.Interaction.indicate ( self ) [inherited]** Definition at line 141 of file interaction.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

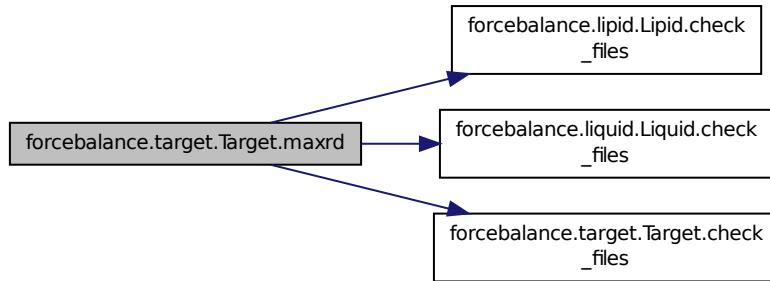
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

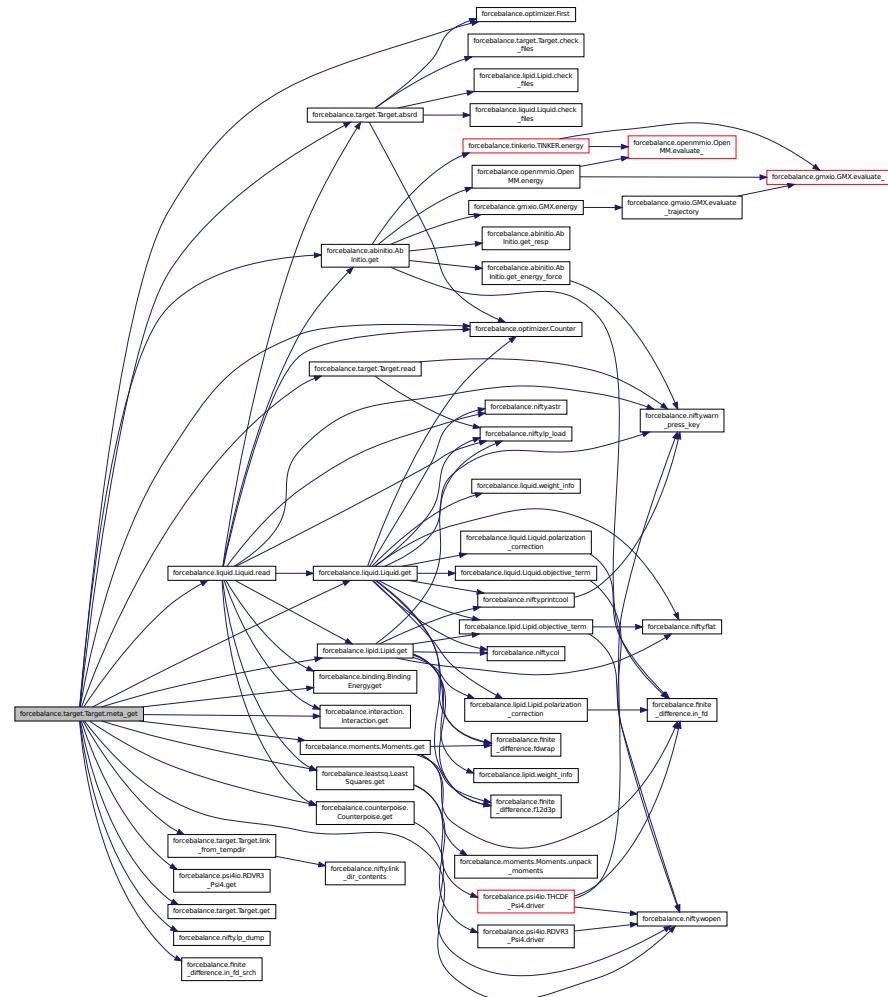


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

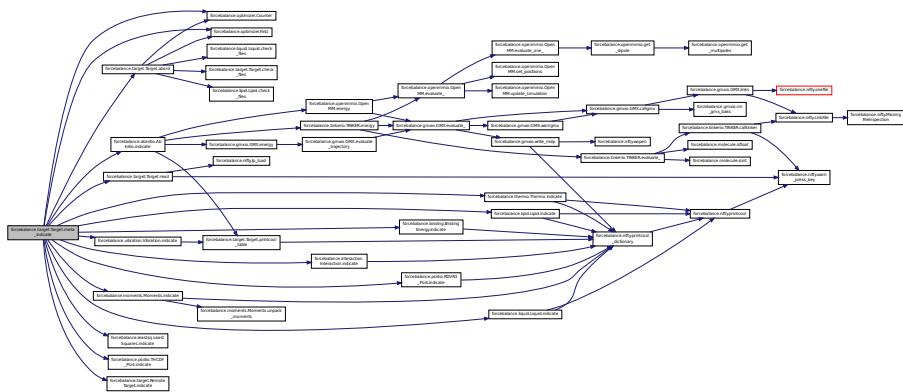
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

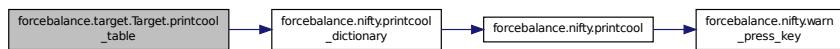
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

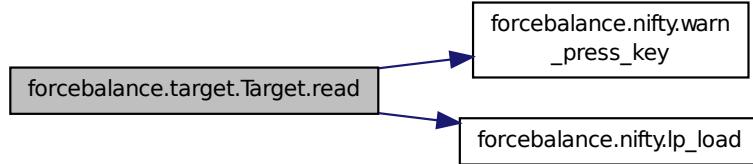
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.interaction.Interaction.read\_reference\_data ( self ) [inherited]** Read the reference ab initio data from a file such as qdata.txt.

After reading in the information from qdata.txt, it is converted into kcal/mol.

Definition at line 124 of file interaction.py.

**def forcebalance.target.Target.refresh\_temp\_directory ( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

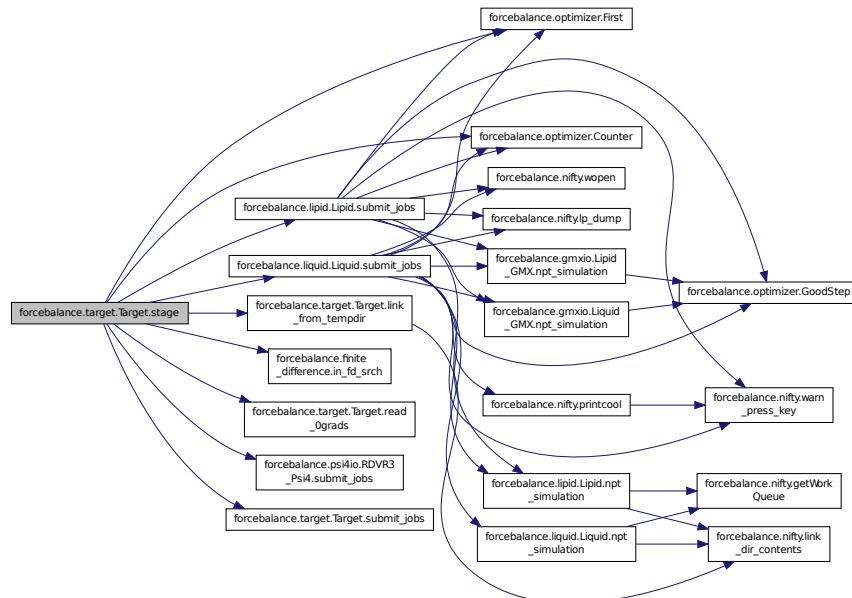
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

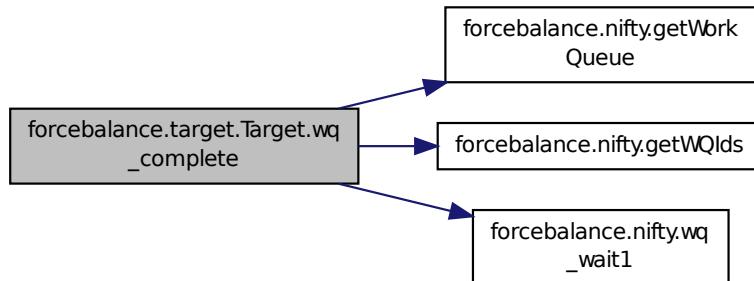


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.write_0grads( self, Ans ) [inherited] Write a file to the target directory containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

#### 8.26.4 Member Data Documentation

**forcebalance.interaction.Interaction.divisor [inherited]** Read in the reference data.

Definition at line 95 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err [inherited]** Definition at line 77 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err\_pct [inherited]** Definition at line 78 of file interaction.py.

**forcebalance.interaction.Interaction.emm [inherited]** Definition at line 194 of file interaction.py.

**forcebalance.interaction.Interaction.engine [inherited]** Build keyword dictionaries to pass to engine.

Definition at line 88 of file interaction.py.

**forcebalance.gmxio.Interaction\_GMX.engine** Default file names for coordinates, top and mdp files.

Definition at line 1475 of file gmxio.py.

**forcebalance.interaction.Interaction.eqm [inherited]** Set upper cutoff energy.

Reference (QM) interaction energies

Definition at line 72 of file interaction.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.interaction.Interaction.label [inherited]** Snapshot label, useful for graphing.

Definition at line 74 of file interaction.py.

**forcebalance.interaction.Interaction.mol [inherited]** Definition at line 81 of file interaction.py.

**forcebalance.interaction.Interaction.ns [inherited]** Read in the trajectory file.

Definition at line 80 of file interaction.py.

**forcebalance.interaction.Interaction.objective [inherited]** Definition at line 195 of file interaction.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.interaction.Interaction.prefactor [inherited]** Definition at line 113 of file interaction.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.interaction.Interaction.qfnm [inherited]** The qdata.txt file that contains the QM energies and forces.

Definition at line 76 of file interaction.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.interaction.Interaction.select1 [inherited]** Number of snapshots.

Do we call Q-Chem for dielectric energies? (Currently needs to be fixed) Do we put the reference energy into the denominator? Do we put the reference energy into the denominator? What is the energy denominator? Set fragment 1

Definition at line 60 of file interaction.py.

**forcebalance.interaction.Interaction.select2 [inherited]** Set fragment 2.

Definition at line 65 of file interaction.py.

**forcebalance.target.Target.tempbase [inherited]** Relative directory of target.

Temporary (working) directory; it is temp/(target\_name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir [inherited]** Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.interaction.Interaction.weight [inherited]** Definition at line 161 of file interaction.py.

**forcebalance.target.Target.write\_indicate [inherited]** Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

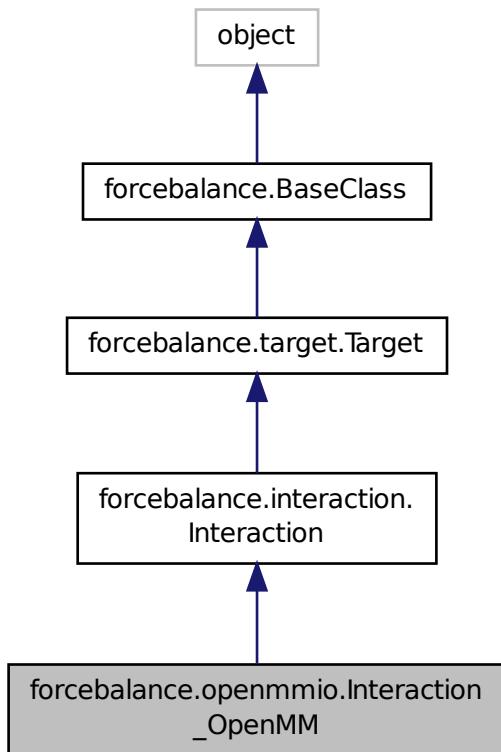
The documentation for this class was generated from the following file:

- [gmxio.py](#)

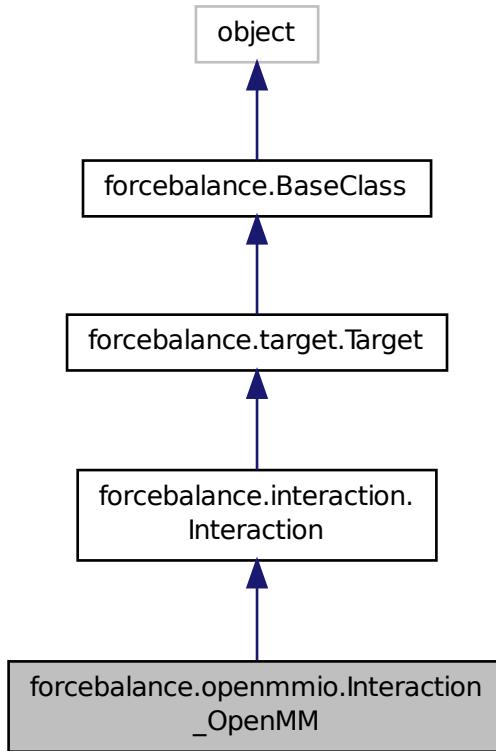
## 8.27 forcebalance.openmmio.Interaction\_OpenMM Class Reference

Interaction matching using [OpenMM](#).

Inheritance diagram for forcebalance.openmmio.Interaction\_OpenMM:



Collaboration diagram for forcebalance.openmmio.Interaction\_OpenMM:



### Public Member Functions

- def `_init_`
- def `read_reference_data`

*Read the reference ab initio data from a file such as qdata.txt.*
- def `indicate`
- def `get`

*Evaluate objective function.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*
- def `get_H`

*Computes the objective function contribution and its gradient / Hessian.*

- def `link_from_tempdir`  
 Back up the temporary directory if desired, delete it and then create a new one.
- def `refresh_temp_directory`  
 Check this directory for the presence of readable files when the 'read' option is set.
- def `read`  
 Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.
- def `absrd`  
 Supply the correct directory specified by user's "read" option.
- def `maxrd`  
 Supply the latest existing temp-directory containing valid data.
- def `meta_indicate`  
 Wrap around the indicate function, so it can print to screen and also to a file.
- def `meta_get`  
 Wrapper around the get function.
- def `submit_jobs`
- def `stage`  
 Stages the directory for the target, and then launches Work Queue processes if any.
- def `wq_complete`  
 This method determines whether the Work Queue tasks for the current target have completed.
- def `printcool_table`  
 Print target information in an organized table format.
- def `__setattr__`
- def `set_option`

## Public Attributes

- `engine_`  
 Default file names for coordinates and key file.
- `select1`  
 Number of snapshots.
- `select2`  
 Set fragment 2.
- `eqm`  
 Set upper cutoff energy.
- `label`  
 Snapshot label, useful for graphing.
- `qfnm`  
 The qdata.txt file that contains the QM energies and forces.
- `e_err`
- `e_err_pct`
- `ns`  
 Read in the trajectory file.
- `mol`
- `engine`  
 Build keyword dictionaries to pass to engine.
- `divisor`  
 Read in the reference data.

- **prefactor**
- **weight**
- **emm**
- **objective**
- **rd**  
*Root directory of the whole project.*
- **pgrad**  
*Iteration where we turn on zero-gradient skipping.*
- **tempbase**  
*Relative directory of target.*
- **tempdir**
- **rundir**  
*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- **FF**  
*Need the forcefield (here for now)*
- **xct**  
*Counts how often the objective function was computed.*
- **gct**  
*Counts how often the gradient was computed.*
- **hct**  
*Counts how often the Hessian was computed.*
- **read\_indicate**  
*Whether to read indicate.log from file when restarting an aborted run.*
- **write\_indicate**  
*Whether to write indicate.log at every iteration (true for all but remote.)*
- **read\_objective**  
*Whether to read objective.p from file when restarting an aborted run.*
- **write\_objective**  
*Whether to write objective.p at every iteration (true for all but remote.)*
- **verbose\_options**
- **PrintOptionDict**

### 8.27.1 Detailed Description

Interaction matching using [OpenMM](#).

Definition at line 1190 of file openmmio.py.

### 8.27.2 Constructor & Destructor Documentation

**def forcebalance.openmmio.Interaction\_OpenMM.\_\_init\_\_( self, options, tgt\_opts, forcefield )** Definition at line 1191 of file openmmio.py.

Here is the call graph for this function:



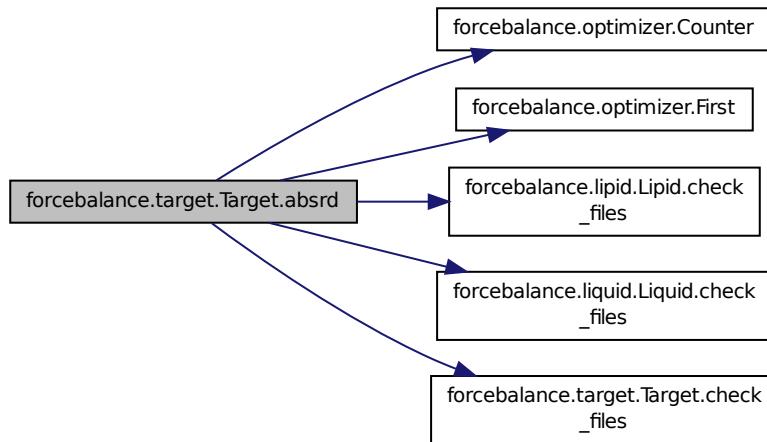
### 8.27.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.check_files( self, there ) [inherited] Check this directory for the presence of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.interaction.Interaction.get( self, mvals, AGrad = False, AHess = False ) [inherited] Evaluate objective function.
```

Definition at line 157 of file interaction.py.

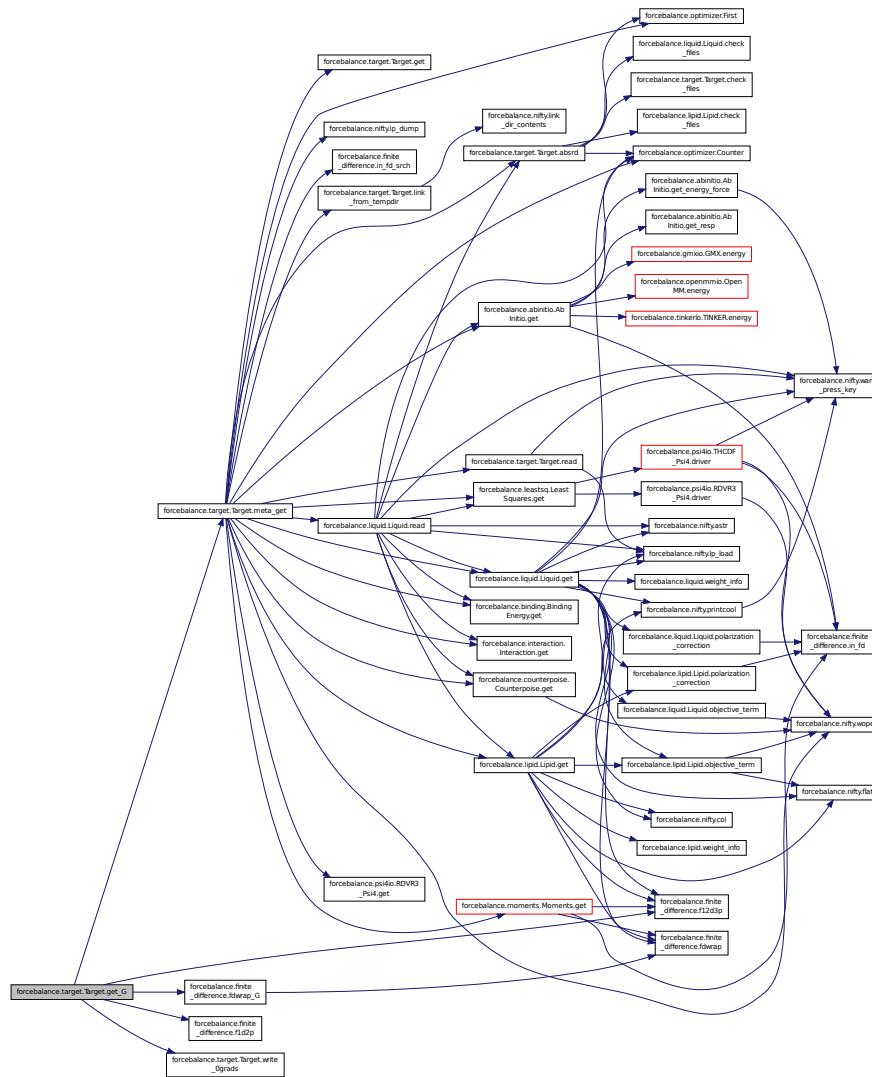
```
def forcebalance.target.Target.get_G( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



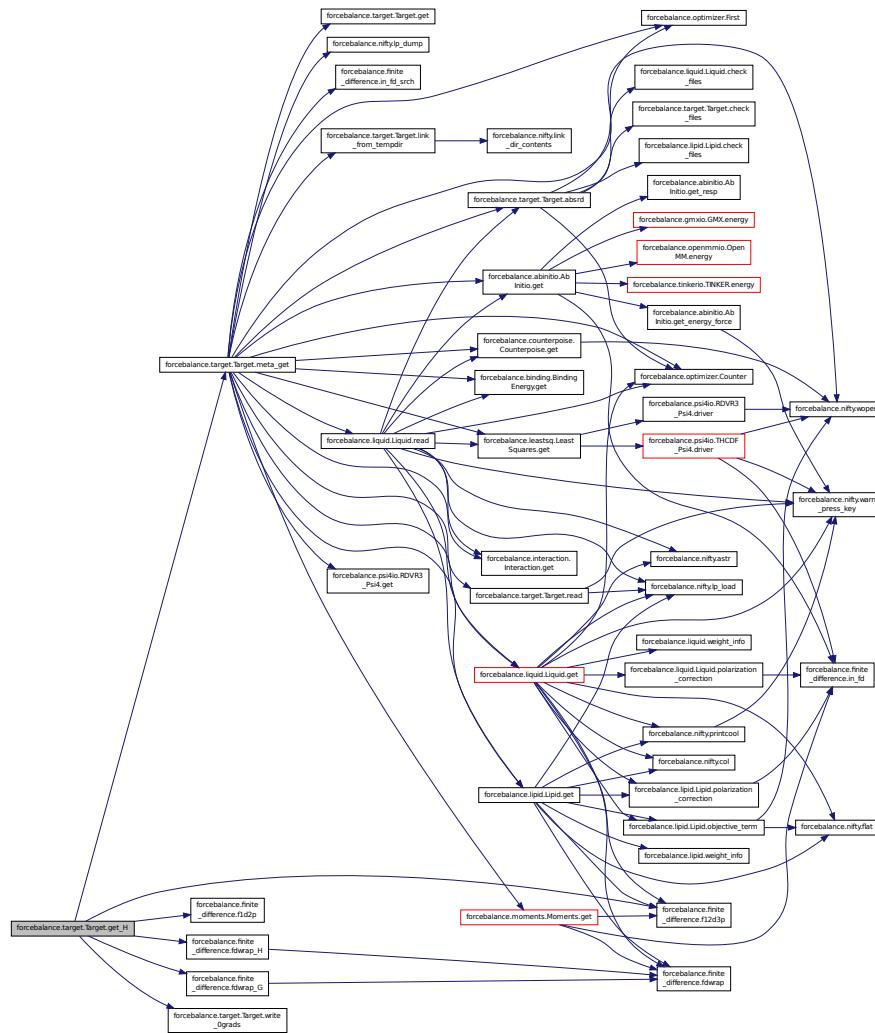
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

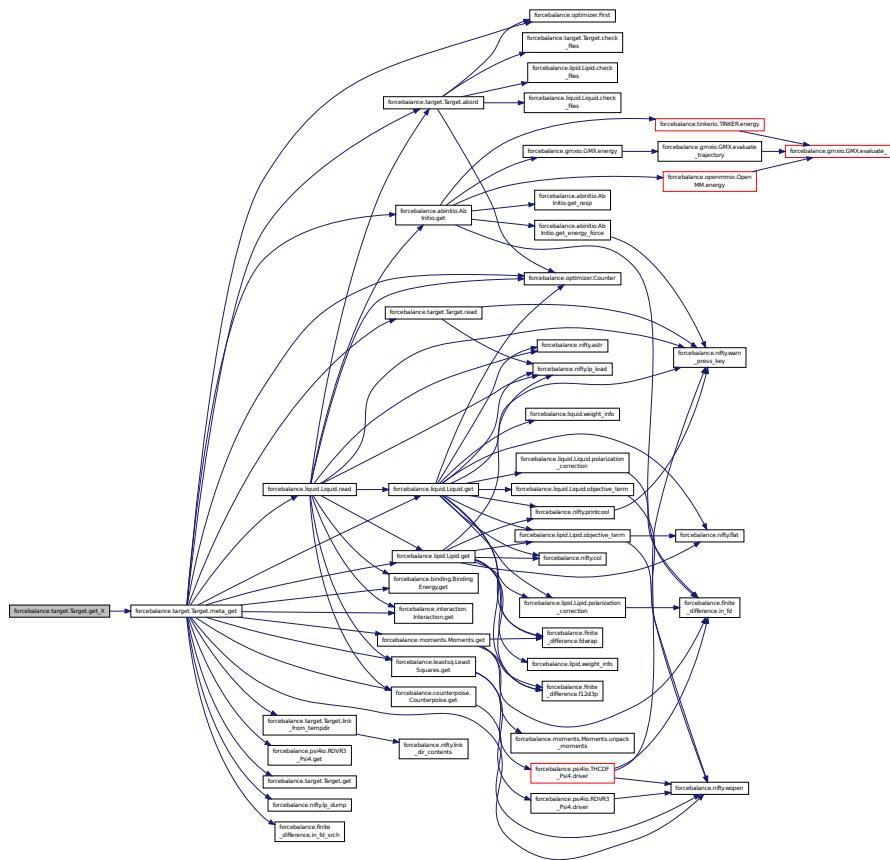
Here is the call graph for this function:



**def forcebalance.target.Target.get.X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

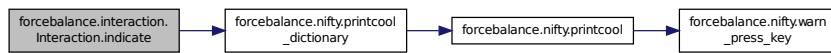
Definition at line 184 of file target.py.

Here is the call graph for this function:



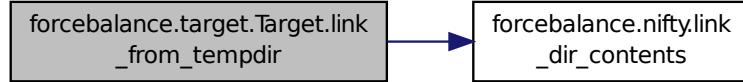
```
def forcebalance.interaction.Interaction.indicate ( self ) [inherited] Definition at line 141 of file interaction.py.
```

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

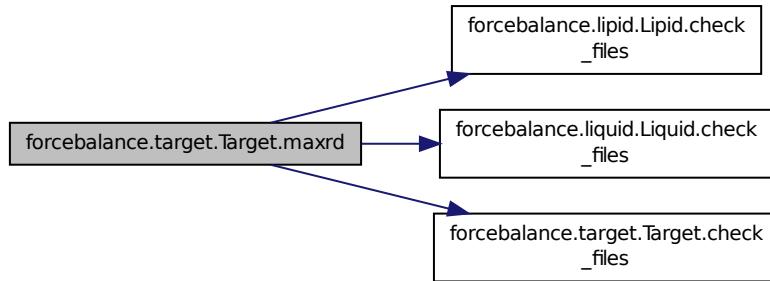
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

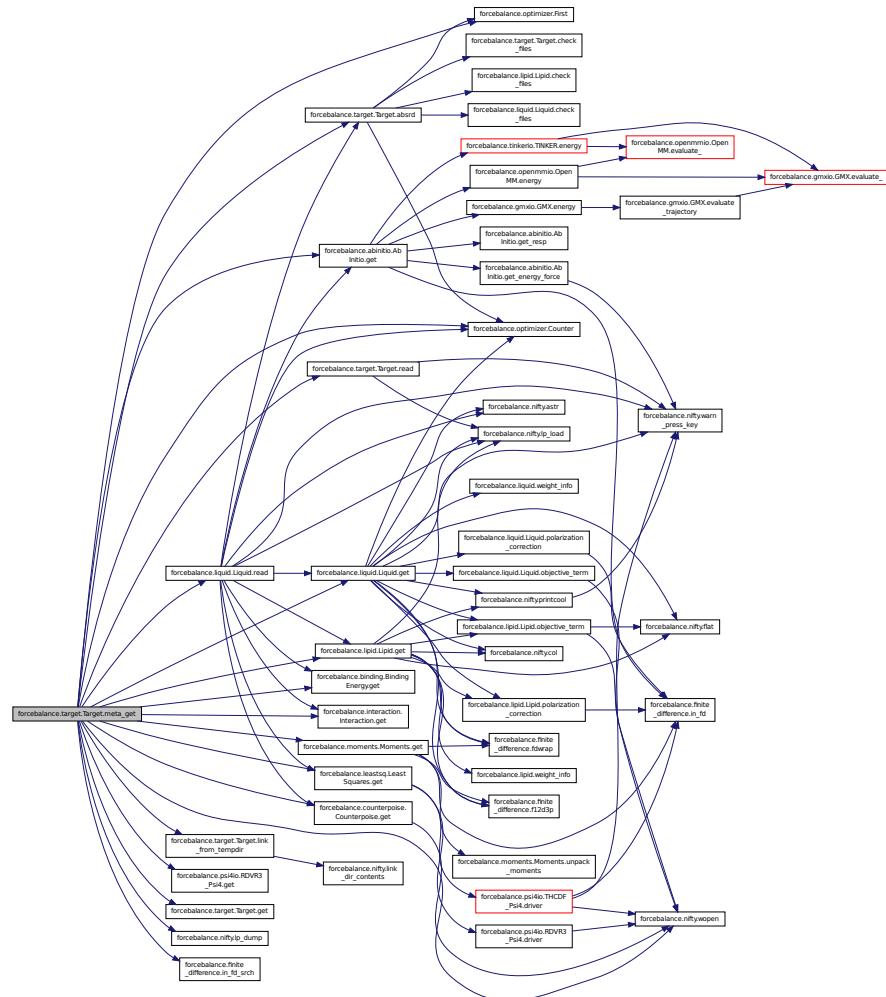


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

Here is the call graph for this function:

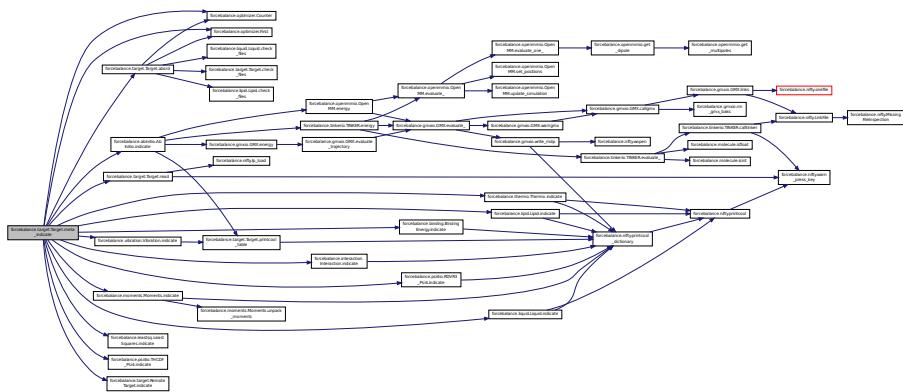


**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.

Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

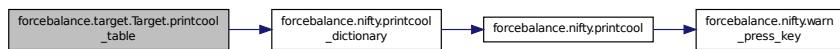
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

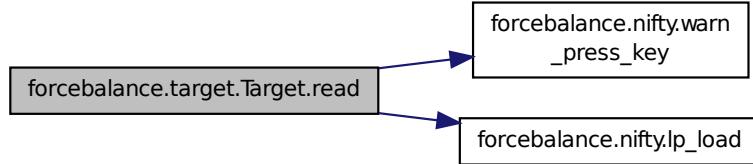
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.interaction.Interaction.read\_reference\_data ( self ) [inherited]** Read the reference ab initio data from a file such as qdata.txt.

After reading in the information from qdata.txt, it is converted into kcal/mol.

Definition at line 124 of file interaction.py.

**def forcebalance.target.Target.refresh\_temp\_directory ( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

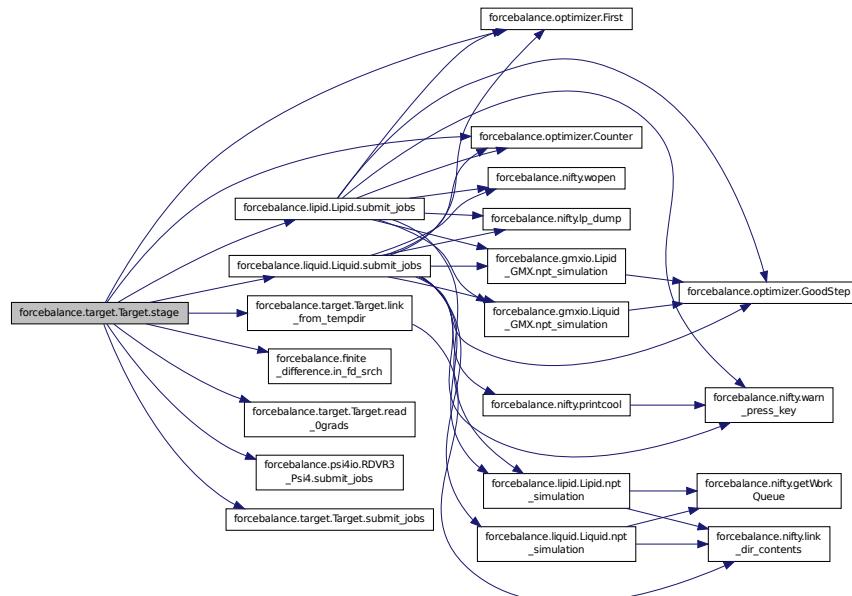
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



```

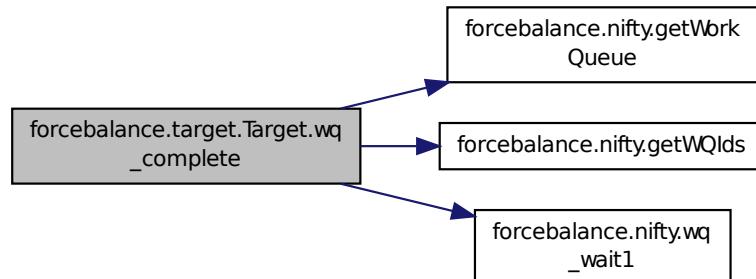
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
  
```

```

def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
  
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```

def forcebalance.target.Target.write_0grads( self, Ans ) [inherited] Write a file to the target directory containing names of parameters that don't contribute to the gradient.
  
```

Definition at line 225 of file target.py.

#### 8.27.4 Member Data Documentation

**forcebalance.interaction.Interaction.divisor [inherited]** Read in the reference data.

Definition at line 95 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err [inherited]** Definition at line 77 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err\_pct [inherited]** Definition at line 78 of file interaction.py.

**forcebalance.interaction.Interaction.emm [inherited]** Definition at line 194 of file interaction.py.

**forcebalance.interaction.Interaction.engine [inherited]** Build keyword dictionaries to pass to engine.

Definition at line 88 of file interaction.py.

**forcebalance.openmmio.Interaction\_OpenMM.engine** Default file names for coordinates and key file.

Definition at line 1196 of file openmmio.py.

**forcebalance.interaction.Interaction.eqm [inherited]** Set upper cutoff energy.

Reference (QM) interaction energies

Definition at line 72 of file interaction.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.interaction.Interaction.label [inherited]** Snapshot label, useful for graphing.

Definition at line 74 of file interaction.py.

**forcebalance.interaction.Interaction.mol [inherited]** Definition at line 81 of file interaction.py.

**forcebalance.interaction.Interaction.ns [inherited]** Read in the trajectory file.

Definition at line 80 of file interaction.py.

**forcebalance.interaction.Interaction.objective [inherited]** Definition at line 195 of file interaction.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.interaction.Interaction.prefactor [inherited]** Definition at line 113 of file interaction.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.interaction.Interaction.qfnm [inherited]** The qdata.txt file that contains the QM energies and forces.

Definition at line 76 of file interaction.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.interaction.Interaction.select1 [inherited]** Number of snapshots.

Do we call Q-Chem for dielectric energies? (Currently needs to be fixed) Do we put the reference energy into the denominator? Do we put the reference energy into the denominator? What is the energy denominator? Set fragment 1

Definition at line 60 of file interaction.py.

**forcebalance.interaction.Interaction.select2 [inherited]** Set fragment 2.

Definition at line 65 of file interaction.py.

**forcebalance.target.Target.tempbase [inherited]** Relative directory of target.

Temporary (working) directory; it is temp/(target\_name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir [inherited]** Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.interaction.Interaction.weight [inherited]** Definition at line 161 of file interaction.py.

**forcebalance.target.Target.write\_indicate [inherited]** Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

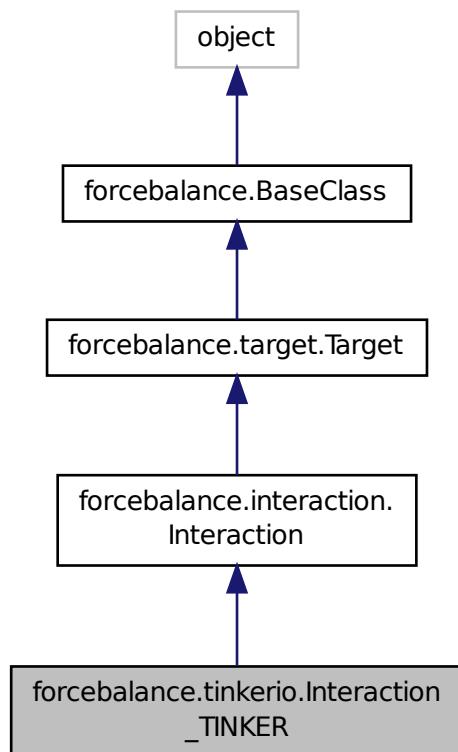
The documentation for this class was generated from the following file:

- [openmmio.py](#)

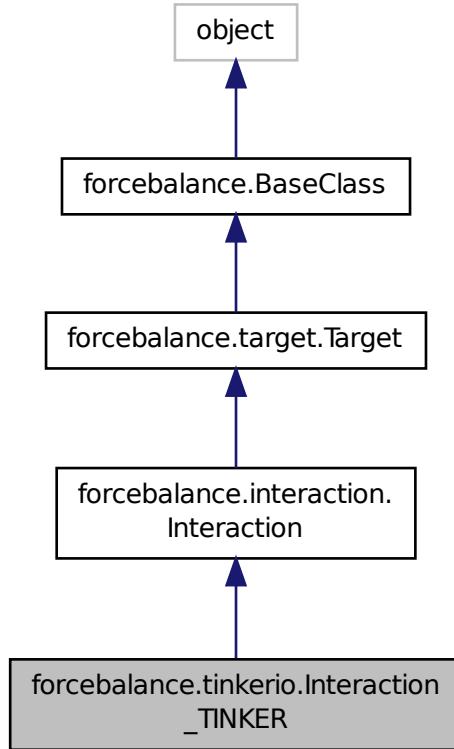
## 8.28 forcebalance.tinkerio.Interaction\_TINKER Class Reference

Subclass of Target for interaction matching using [TINKER](#).

Inheritance diagram for forcebalance.tinkerio.Interaction\_TINKER:



Collaboration diagram for forcebalance.tinkerio.Interaction\_TINKER:



### Public Member Functions

- def `_init_`
- def `read_reference_data`

*Read the reference ab initio data from a file such as qdata.txt.*
- def `indicate`
- def `get`

*Evaluate objective function.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*
- def `get_H`

*Computes the objective function contribution and its gradient / Hessian.*

- def `link_from_tempdir`  
 Back up the temporary directory if desired, delete it and then create a new one.
- def `refresh_temp_directory`  
 Check this directory for the presence of readable files when the 'read' option is set.
- def `read`  
 Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.
- def `absrd`  
 Supply the correct directory specified by user's "read" option.
- def `maxrd`  
 Supply the latest existing temp-directory containing valid data.
- def `meta_indicate`  
 Wrap around the indicate function, so it can print to screen and also to a file.
- def `meta_get`  
 Wrapper around the get function.
- def `submit_jobs`
- def `stage`  
 Stages the directory for the target, and then launches Work Queue processes if any.
- def `wq_complete`  
 This method determines whether the Work Queue tasks for the current target have completed.
- def `printcool_table`  
 Print target information in an organized table format.
- def `__setattr__`
- def `set_option`

## Public Attributes

- `engine_`  
 Default file names for coordinates and key file.
- `select1`  
 Number of snapshots.
- `select2`  
 Set fragment 2.
- `eqm`  
 Set upper cutoff energy.
- `label`  
 Snapshot label, useful for graphing.
- `qfnm`  
 The qdata.txt file that contains the QM energies and forces.
- `e_err`
- `e_err_pct`
- `ns`  
 Read in the trajectory file.
- `mol`
- `engine`  
 Build keyword dictionaries to pass to engine.
- `divisor`  
 Read in the reference data.

- `prefactor`
- `weight`
- `emm`
- `objective`
- `rd`  
`Root directory of the whole project.`
- `pgrad`  
`Iteration where we turn on zero-gradient skipping.`
- `tempbase`  
`Relative directory of target.`
- `tempdir`
- `rundir`  
`self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.`
- `FF`  
`Need the forcefield (here for now)`
- `xct`  
`Counts how often the objective function was computed.`
- `gct`  
`Counts how often the gradient was computed.`
- `hct`  
`Counts how often the Hessian was computed.`
- `read_indicate`  
`Whether to read indicate.log from file when restarting an aborted run.`
- `write_indicate`  
`Whether to write indicate.log at every iteration (true for all but remote.)`
- `read_objective`  
`Whether to read objective.p from file when restarting an aborted run.`
- `write_objective`  
`Whether to write objective.p at every iteration (true for all but remote.)`
- `verbose_options`
- `PrintOptionDict`

### 8.28.1 Detailed Description

Subclass of Target for interaction matching using [TINKER](#).

Definition at line 1086 of file tinkerio.py.

### 8.28.2 Constructor & Destructor Documentation

`def forcebalance.tinkerio.Interaction_TINKER.__init__ ( self, options, tgt_opts, forcefield )` Definition at line 1087 of file tinkerio.py.

Here is the call graph for this function:



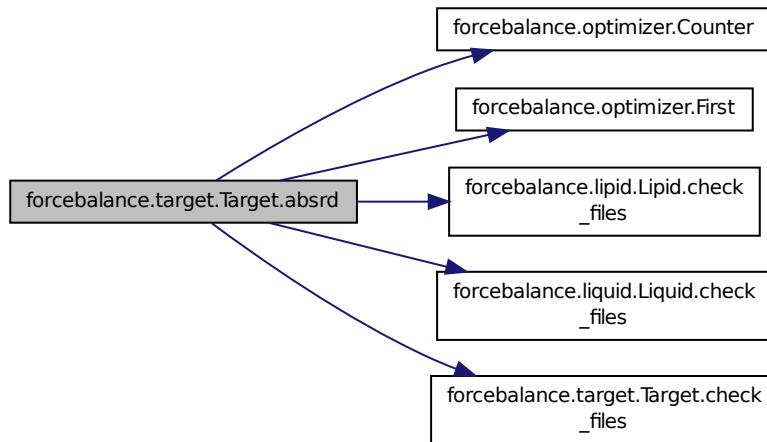
### 8.28.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.check_files( self, there ) [inherited] Check this directory for the presence of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.interaction.Interaction.get( self, mvals, AGrad = False, AHess = False ) [inherited] Evaluate objective function.
```

Definition at line 157 of file interaction.py.

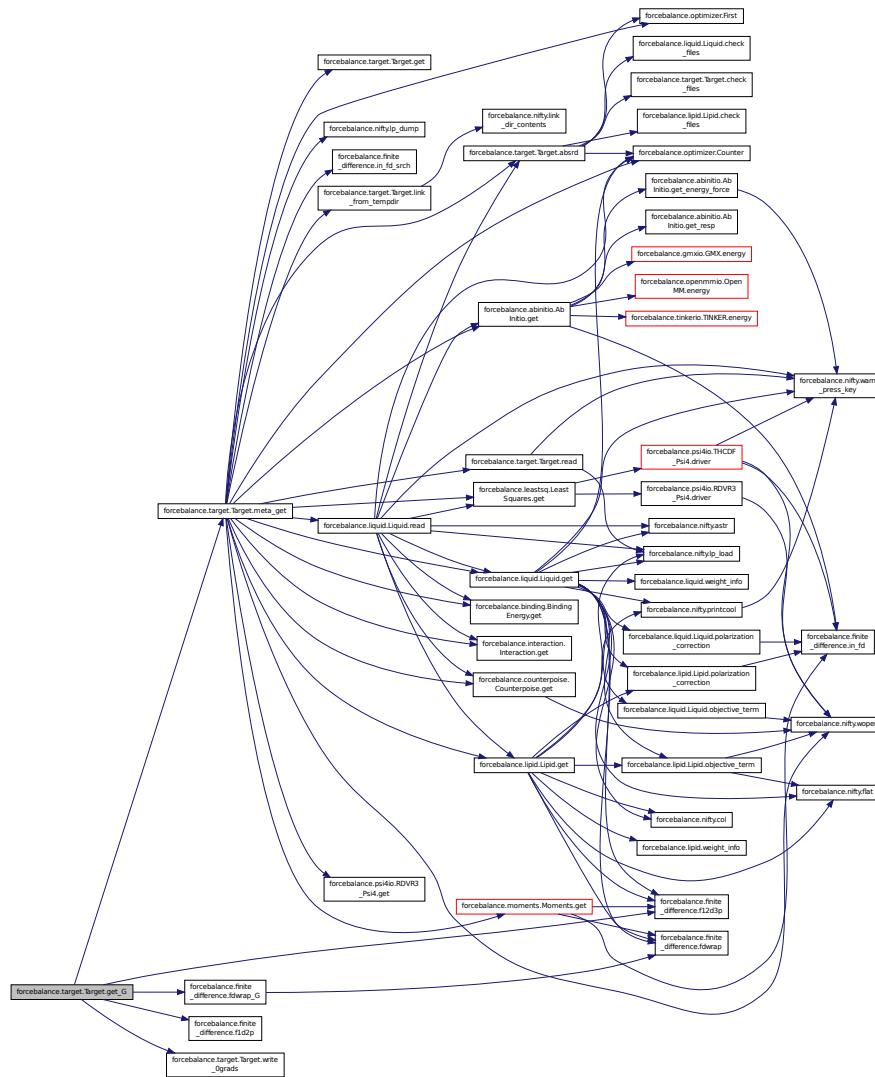
```
def forcebalance.target.Target.get_G( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



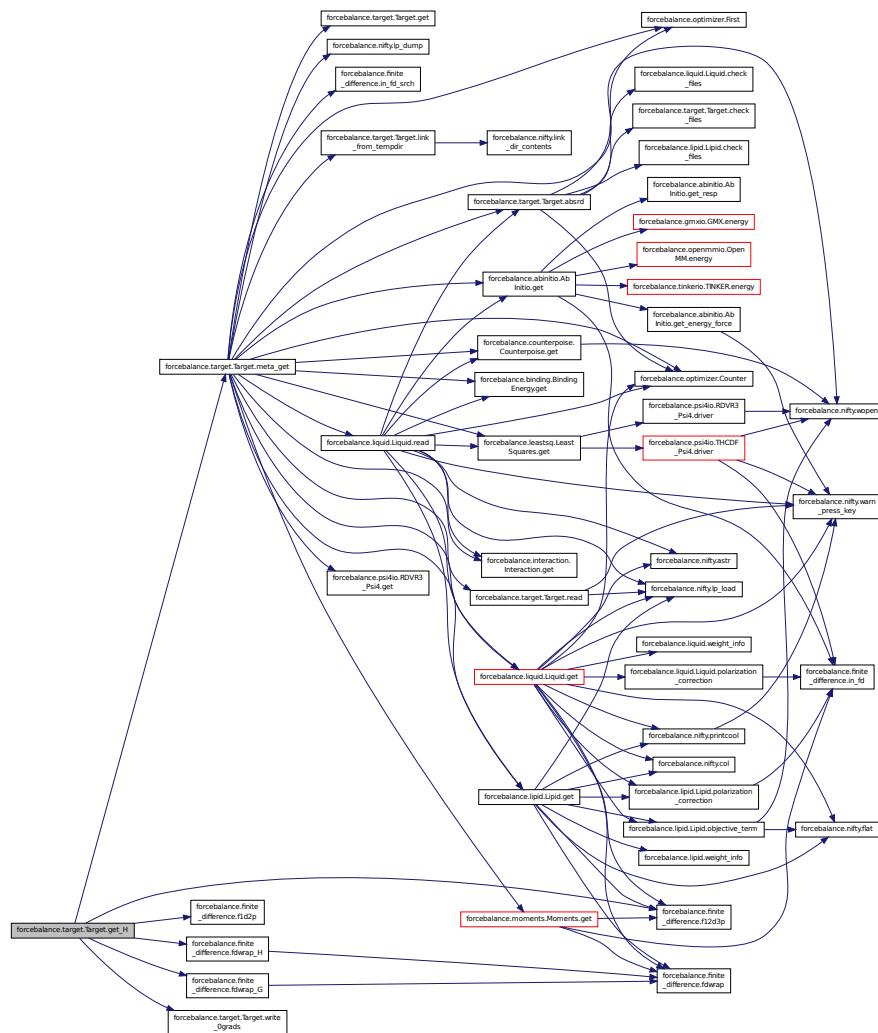
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

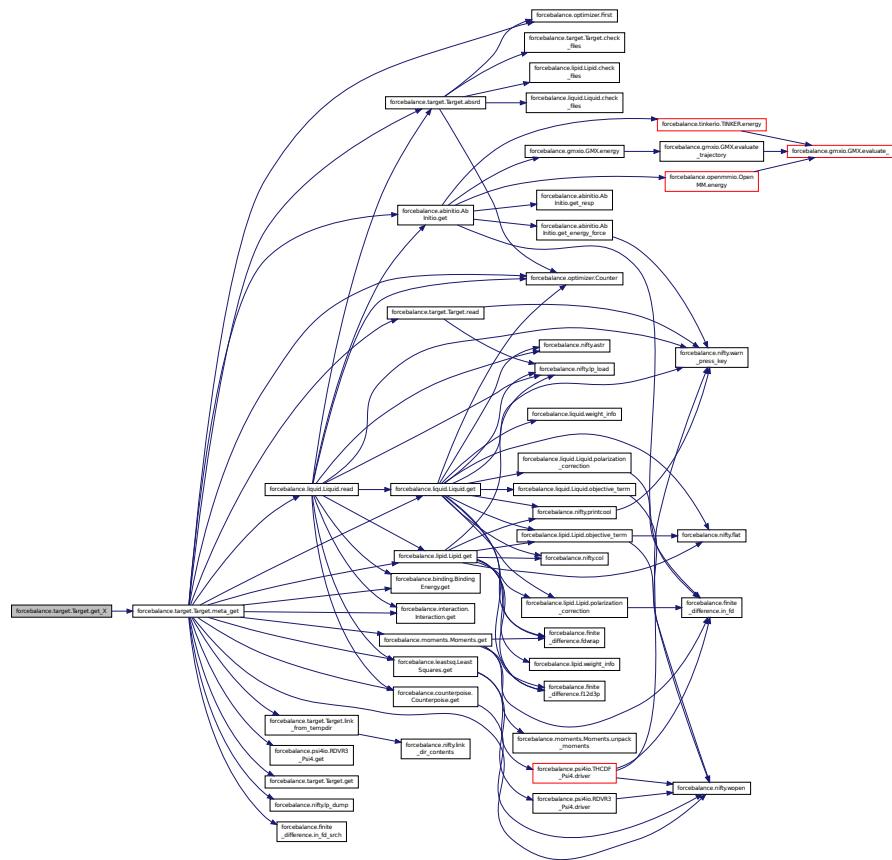
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

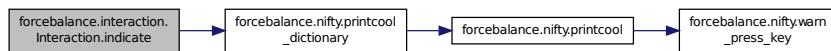
Definition at line 184 of file target.py.

Here is the call graph for this function:



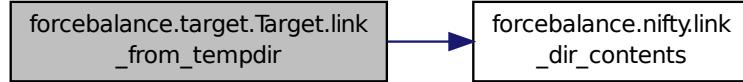
```
def forcebalance.optimizer.optimize ( self ) [inherited] Definition at line 141 of file
optimizer.py.
```

Here is the call graph for this function:



```
def forcebalance.target.Target.link_from_tempdir( self, absdestdir ) [inherited] Definition at line 315 of
file target.py.
```

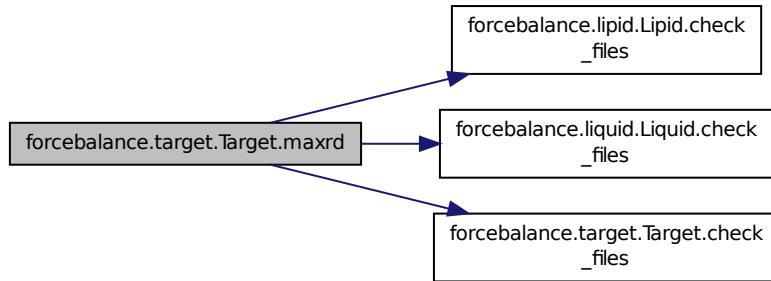
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

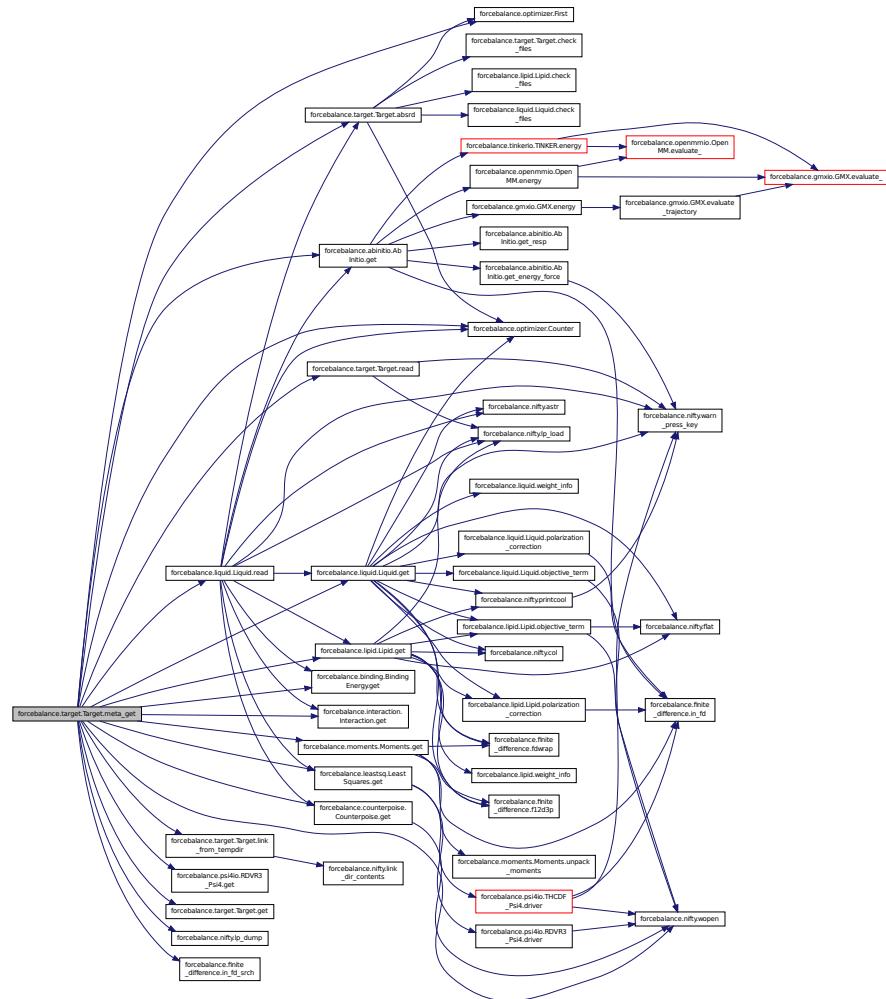


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

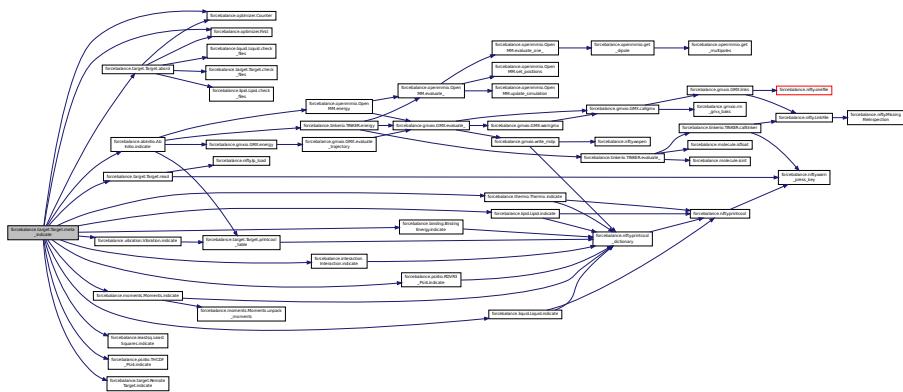
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

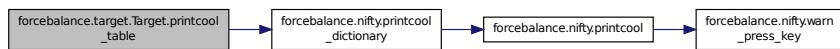
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

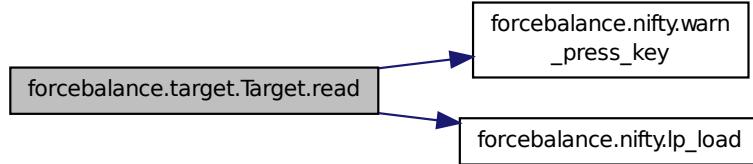
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.interaction.Interaction.read\_reference\_data ( self ) [inherited]** Read the reference ab initio data from a file such as qdata.txt.

After reading in the information from qdata.txt, it is converted into kcal/mol.

Definition at line 124 of file interaction.py.

**def forcebalance.target.Target.refresh\_temp\_directory ( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

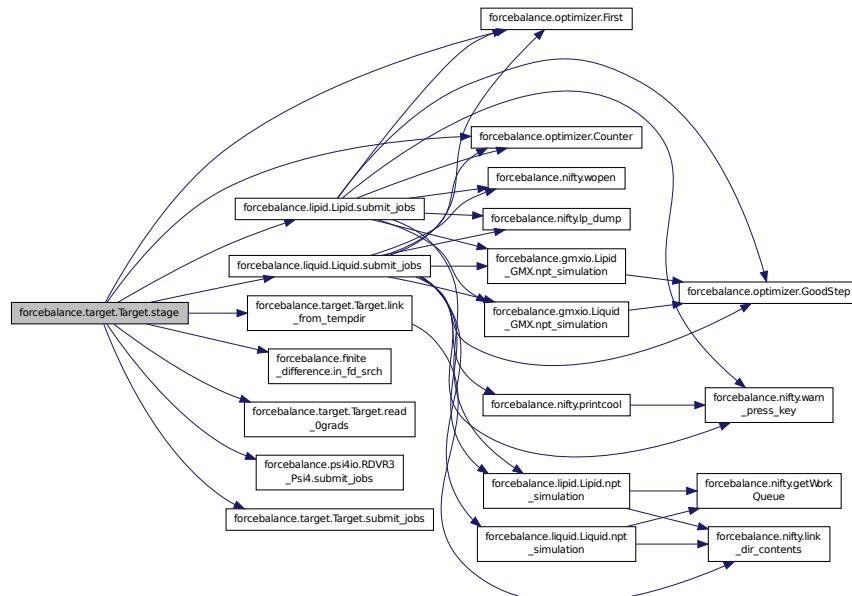
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

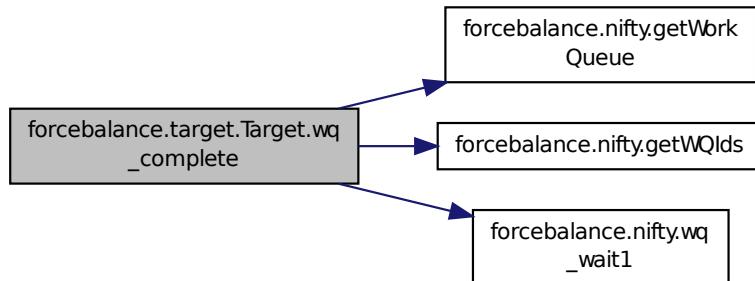


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.write_0grads( self, Ans ) [inherited] Write a file to the target directory containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

#### 8.28.4 Member Data Documentation

**forcebalance.interaction.Interaction.divisor [inherited]** Read in the reference data.

Definition at line 95 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err [inherited]** Definition at line 77 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err\_pct [inherited]** Definition at line 78 of file interaction.py.

**forcebalance.interaction.Interaction.emm [inherited]** Definition at line 194 of file interaction.py.

**forcebalance.interaction.Interaction.engine [inherited]** Build keyword dictionaries to pass to engine.

Definition at line 88 of file interaction.py.

**forcebalance.tinkerio.Interaction\_TINKER.engine** Default file names for coordinates and key file.

Definition at line 1091 of file tinkerio.py.

**forcebalance.interaction.Interaction.eqm [inherited]** Set upper cutoff energy.

Reference (QM) interaction energies

Definition at line 72 of file interaction.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.interaction.Interaction.label [inherited]** Snapshot label, useful for graphing.

Definition at line 74 of file interaction.py.

**forcebalance.interaction.Interaction.mol [inherited]** Definition at line 81 of file interaction.py.

**forcebalance.interaction.Interaction.ns [inherited]** Read in the trajectory file.

Definition at line 80 of file interaction.py.

**forcebalance.interaction.Interaction.objective [inherited]** Definition at line 195 of file interaction.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.interaction.Interaction.prefactor [inherited]** Definition at line 113 of file interaction.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.interaction.Interaction.qfnm [inherited]** The qdata.txt file that contains the QM energies and forces.

Definition at line 76 of file interaction.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.interaction.Interaction.select1 [inherited]** Number of snapshots.

Do we call Q-Chem for dielectric energies? (Currently needs to be fixed) Do we put the reference energy into the denominator? Do we put the reference energy into the denominator? What is the energy denominator? Set fragment 1

Definition at line 60 of file interaction.py.

**forcebalance.interaction.Interaction.select2 [inherited]** Set fragment 2.

Definition at line 65 of file interaction.py.

**forcebalance.target.Target.tempbase [inherited]** Relative directory of target.

Temporary (working) directory; it is temp/(target\_name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir [inherited]** Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.interaction.Interaction.weight [inherited]** Definition at line 161 of file interaction.py.

**forcebalance.target.Target.write\_indicate [inherited]** Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

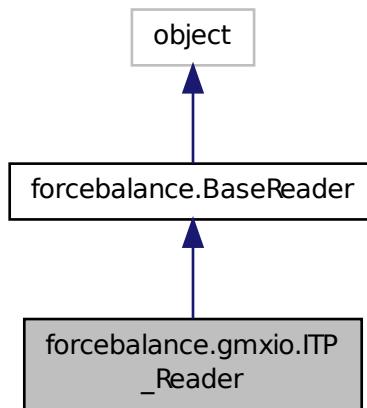
The documentation for this class was generated from the following file:

- [tinkerio.py](#)

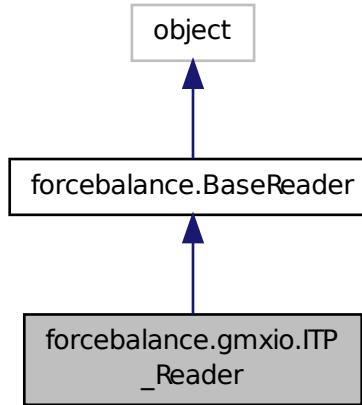
## 8.29 forcebalance.gmxio.ITP\_Reader Class Reference

Finite state machine for parsing GROMACS force field files.

Inheritance diagram for forcebalance.gmxio.ITP\_Reader:



Collaboration diagram for forcebalance.gmxio.ITP\_Reader:



## Public Member Functions

- def `__init__`
- def `feed`

*Given a line, determine the interaction type and the atoms involved (the suffix).*
- def `Split`
- def `Whites`
- def `build.pid`

*Returns the parameter type (e.g.*

## Public Attributes

- `sec`

*The current section that we're in.*
- `nbttype`

*Nonbonded type.*
- `mol`

*The current molecule (set by the moleculetype keyword)*
- `pdict`

*The parameter dictionary (defined in this file)*
- `atomnames`

*Listing of all atom names in the file, (probably unnecessary)*
- `atomtypes`

*Listing of all atom types in the file, (probably unnecessary)*
- `atomtype_to_mass`

*A dictionary of atomic masses.*
- `itype`
- `suffix`

- [molatom](#)
- [In](#)
- [adict](#)

*The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.*

- [Molecules](#)
- [AtomTypes](#)

### 8.29.1 Detailed Description

Finite state machine for parsing GROMACS force field files.

We open the force field file and read all of its lines. As we loop through the force field file, we look for two types of tags: (1) section markers, in [GMX](#) indicated by [ section\_name ], which allows us to determine the section, and (2) parameter tags, indicated by the 'PRM' or 'RPT' keywords.

As we go through the file, we figure out the atoms involved in the interaction described on each line.

When a 'PRM' keyword is indicated, it is followed by a number which is the field in the line to be modified, starting with zero. Based on the field number and the section name, we can figure out the parameter type. With the parameter type and the atoms in hand, we construct a 'parameter identifier' or pid which uniquely identifies that parameter. We also store the physical parameter value in an array called 'pvals0' and the precise location of that parameter (by filename, line number, and field number) in a list called 'pfields'.

An example: Suppose in 'my\_ff.itp' I encounter the following on lines 146 and 147:

```
1 [ angletypes ]
2 CA  CB  O  1  109.47  350.00 ; PRM 4 5
```

From reading [ angletypes ] I know I'm in the 'angletypes' section.

On the next line, I notice two parameters on fields 4 and 5.

From the atom types, section type and field number I know the parameter IDs are 'ANGLESBCACBO' and 'ANGLESKCACBO'.

After building map={'ANGLESBCACBO':1,'ANGLESKCACBO':2}, I store the values in an array: pvals0=array([109.47,350.00]), and I put the parameter locations in pfields: pfields=[['my\_ff.itp',147,4,1.0],['my\_ff.itp',146,5,1.0]]. The 1.0 is a 'multiplier' and I will explain it below.

Note that in the creation of parameter IDs, we run into the issue that the atoms involved in the interaction may be labeled in reverse order (e.g. OCACB). Thus, we store both the normal and the reversed parameter ID in the map.

Parameter repetition and multiplier:

If 'RPT' is encountered in the line, it is always in the syntax: 'RPT 4 ANGLESBCACAH 5 MINUS\_ANGLES-KCACAH /RPT'. In this case, field 4 is replaced by the stored parameter value corresponding to ANGLESBCACAH and field 5 is replaced by -1 times the stored value of ANGLESKCACAH. Now I just picked this as an example, I don't think people actually want a negative angle force constant .. :) the MINUS keyword does come in handy for assigning atomic charges and virtual site positions. In order to achieve this, a multiplier of -1.0 is stored into pfields instead of 1.0.

**Todo** Note that I can also create the opposite virtual site position by changing the atom labeling, woo!

Definition at line 328 of file gmxio.py.

### 8.29.2 Constructor & Destructor Documentation

**def forcebalance.gmxio.ITP\_Reader.\_\_init\_\_ ( self, fnm )** Definition at line 331 of file gmxio.py.

### 8.29.3 Member Function Documentation

**def forcebalance.BaseReader.build.pid ( self, pfd ) [inherited]** Returns the parameter type (e.g. K in BONDSK) based on the current interaction type.

Both the 'pdic' dictionary (see [gmxio.pdic](#)) and the interaction type 'state' (here, BONDS) are needed to get the parameter type.

If, however, 'pdic' does not contain the ptype value, a suitable substitute is simply the field number.

Note that if the interaction type state is not set, then it defaults to the file name, so a generic parameter ID is 'filename.line\_num.field\_num'

Definition at line 124 of file \_\_init\_\_.py.

**def forcebalance.gmxio.ITP\_Reader.feed ( self, line )** Given a line, determine the interaction type and the atoms involved (the suffix).

For example, we want

```
H O H 5 1.231258497536e+02 4.269161426840e+02 -1.033397697685e-02 1.304674117410e+04  
; PRM 4 5 6 7
```

to give us itype = 'UREY\_BRADLEY' and suffix = 'HOH'

If we are in a TypeSection, it returns a list of atom types;

If we are in a TopolSection, it returns a list of atom names.

The section is essentially a case statement that picks out the appropriate interaction type and makes a list of the atoms involved

Note that we can call gmxdump for this as well, but I prefer to read the force field file directly.

ToDo: [ atoms ] section might need to be more flexible to accommodate optional fields

Definition at line 369 of file gmxio.py.

**def forcebalance.BaseReader.Split ( self, line ) [inherited]** Definition at line 99 of file \_\_init\_\_.py.

**def forcebalance.BaseReader.Whites ( self, line ) [inherited]** Definition at line 102 of file \_\_init\_\_.py.

#### 8.29.4 Member Data Documentation

**forcebalance.BaseReader.adict [inherited]** The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.

Definition at line 89 of file \_\_init\_\_.py.

**forcebalance.gmxio.ITP\_Reader.atomnames** Listing of all atom names in the file, (probably unnecessary)

Definition at line 343 of file gmxio.py.

**forcebalance.gmxio.ITP\_Reader.atomtype\_to\_mass** A dictionary of atomic masses.

Definition at line 347 of file gmxio.py.

**forcebalance.BaseReader.AtomTypes [inherited]** Definition at line 97 of file \_\_init\_\_.py.

**forcebalance.gmxio.ITP\_Reader.atomtypes** Listing of all atom types in the file, (probably unnecessary)

Definition at line 345 of file gmxio.py.

**forcebalance.gmxio.ITP\_Reader.itype** Definition at line 372 of file gmxio.py.

**forcebalance.BaseReader.in [inherited]** Definition at line 84 of file \_\_init\_\_.py.

**forcebalance.gmxio.ITP\_Reader.mol** The current molecule (set by the moleculetype keyword)

Definition at line 339 of file gmxio.py.

**forcebalance.gmxio.ITP\_Reader.molatom** Definition at line 479 of file gmxio.py.

**forcebalance.BaseReader.Molecules [inherited]** Definition at line 96 of file \_\_init\_\_.py.

**forcebalance.gmxio.ITP\_Reader.nbtype** Nonbonded type.

Definition at line 337 of file gmxio.py.

**forcebalance.gmxio.ITP\_Reader.pdict** The parameter dictionary (defined in this file)

Definition at line 341 of file gmxio.py.

**forcebalance.gmxio.ITP\_Reader.sec** The current section that we're in.

Definition at line 335 of file gmxio.py.

**forcebalance.gmxio.ITP\_Reader.suffix** Definition at line 474 of file gmxio.py.

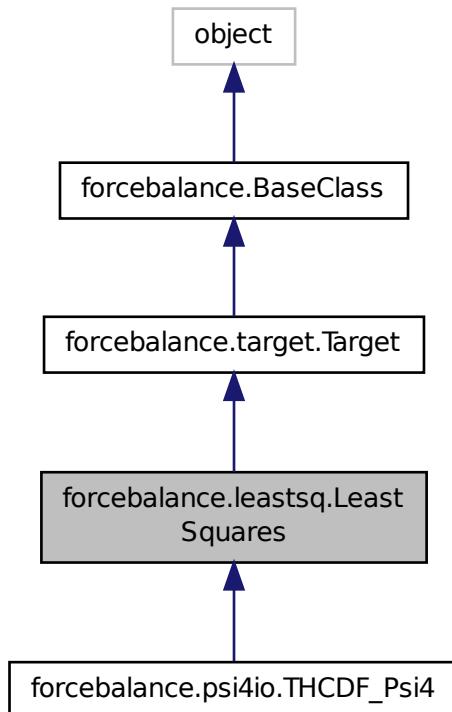
The documentation for this class was generated from the following file:

- [gmxio.py](#)

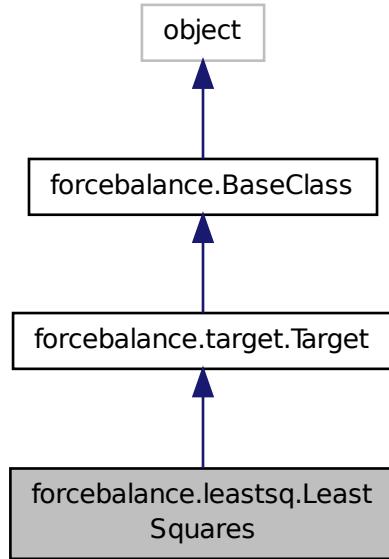
## 8.30 forcebalance.leastsq.LeastSquares Class Reference

Subclass of Target for general least squares fitting.

Inheritance diagram for forcebalance.leastsq.LeastSquares:



Collaboration diagram for forcebalance.leastsq.LeastSquares:



### Public Member Functions

- def `_init_`
- def `indicate`
- def `get`  
    *LPW 05-30-2012.*
- def `get_X`  
    *Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`  
    *Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`  
    *Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`  
    *Computes the objective function contribution and its gradient.*
- def `get_H`  
    *Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`  
    *Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
    *Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
    *Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*

- def **absrd**  
*Supply the correct directory specified by user's "read" option.*
- def **maxrd**  
*Supply the latest existing temp-directory containing valid data.*
- def **meta\_indicate**  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def **meta\_get**  
*Wrapper around the get function.*
- def **submit\_jobs**
- def **stage**  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def **wq\_complete**  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def **printcool\_table**  
*Print target information in an organized table format.*
- def **\_\_setattr\_\_**
- def **set\_option**

## Public Attributes

- **MAQ**  
*Dictionary for derivative terms.*
- **D**
- **objective**
- **rd**  
*Root directory of the whole project.*
- **pgrad**  
*Iteration where we turn on zero-gradient skipping.*
- **tempbase**  
*Relative directory of target.*
- **tempdir**
- **rundir**  
*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- **FF**  
*Need the forcefield (here for now)*
- **xct**  
*Counts how often the objective function was computed.*
- **gct**  
*Counts how often the gradient was computed.*
- **hct**  
*Counts how often the Hessian was computed.*
- **read\_indicate**  
*Whether to read indicate.log from file when restarting an aborted run.*
- **write\_indicate**  
*Whether to write indicate.log at every iteration (true for all but remote.)*
- **read\_objective**  
*Whether to read objective.p from file when restarting an aborted run.*
- **write\_objective**  
*Whether to write objective.p at every iteration (true for all but remote.)*
- **verbose\_options**
- **PrintOptionDict**

### 8.30.1 Detailed Description

Subclass of Target for general least squares fitting.  
Definition at line 35 of file leastsq.py.

### 8.30.2 Constructor & Destructor Documentation

```
def forcebalance.leastsq.LeastSquares.__init__ ( self, options, tgt_opts, forcefield )
```

 Definition at line 38 of file leastsq.py.

### 8.30.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited]
```

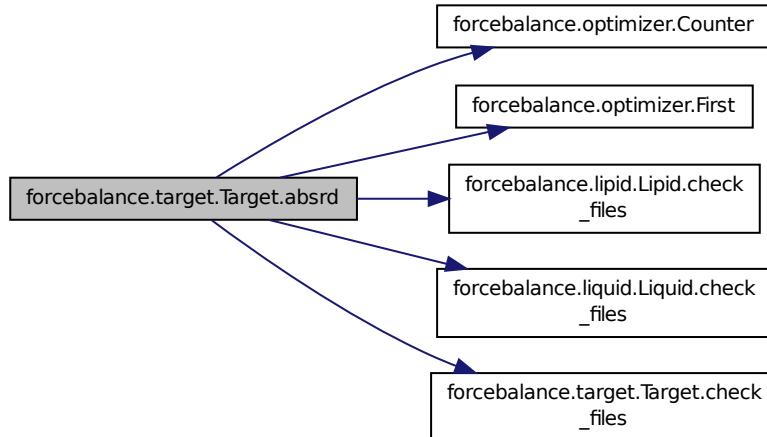
 Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited]
```

 Supply the correct directory specified by user's "read" option.

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.check_files ( self, there ) [inherited]
```

 Check this directory for the presence of readable files when the 'read' option is set.

Definition at line 364 of file target.py.

```
def forcebalance.leastsq.LeastSquares.get ( self, mvals, AGrad = False, AHess = False )
```

 LPW 05-30-2012.

This subroutine builds the objective function (and optionally its derivatives) from a general software.

This subroutine interfaces with simulation software 'drivers'. The driver is expected to give exact values, fitting values, and weights.

## Parameters

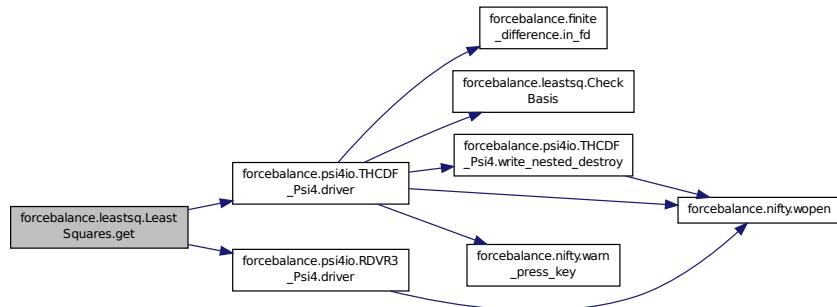
in	<i>mvals</i>	Mathematical parameter values
in	<i>AGrad</i>	Switch to turn on analytic gradient
in	<i>AHess</i>	Switch to turn on analytic Hessian

## Returns

Answer Contribution to the objective function

Definition at line 62 of file leastsq.py.

Here is the call graph for this function:



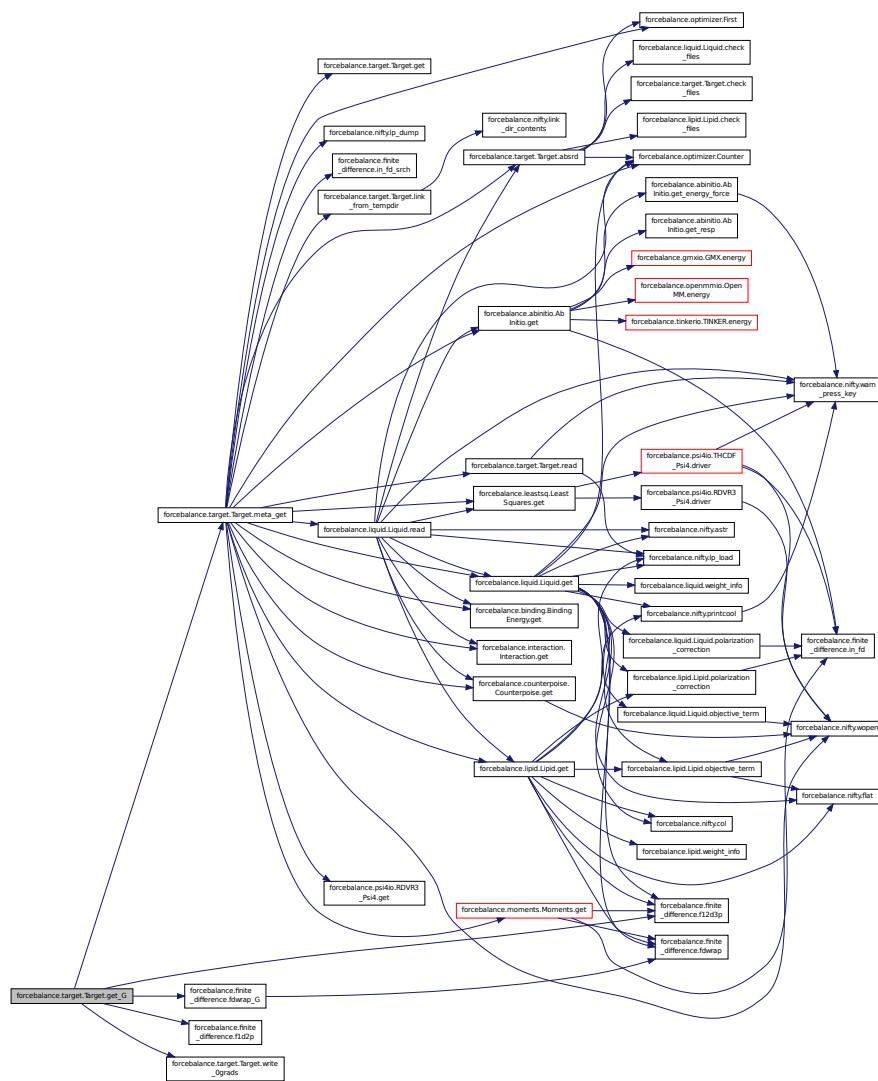
**def forcebalance.target.Target.get\_G ( *self*, *mvals = None* ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1.pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



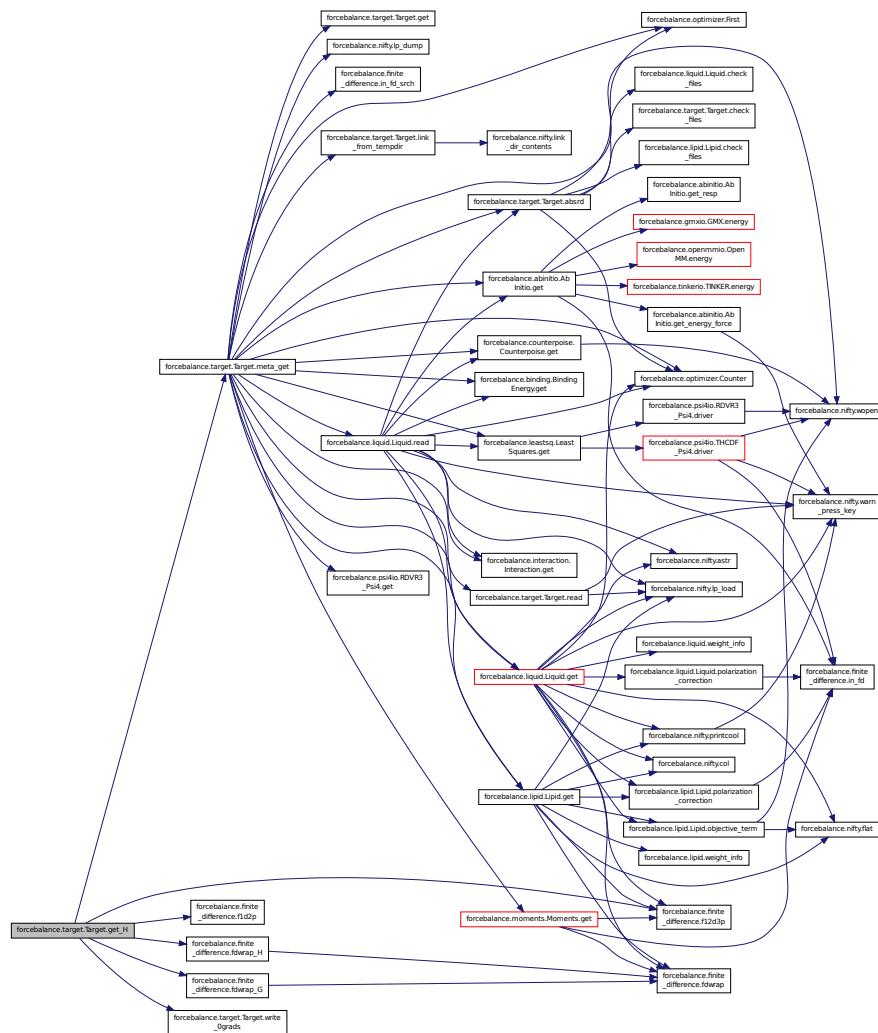
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

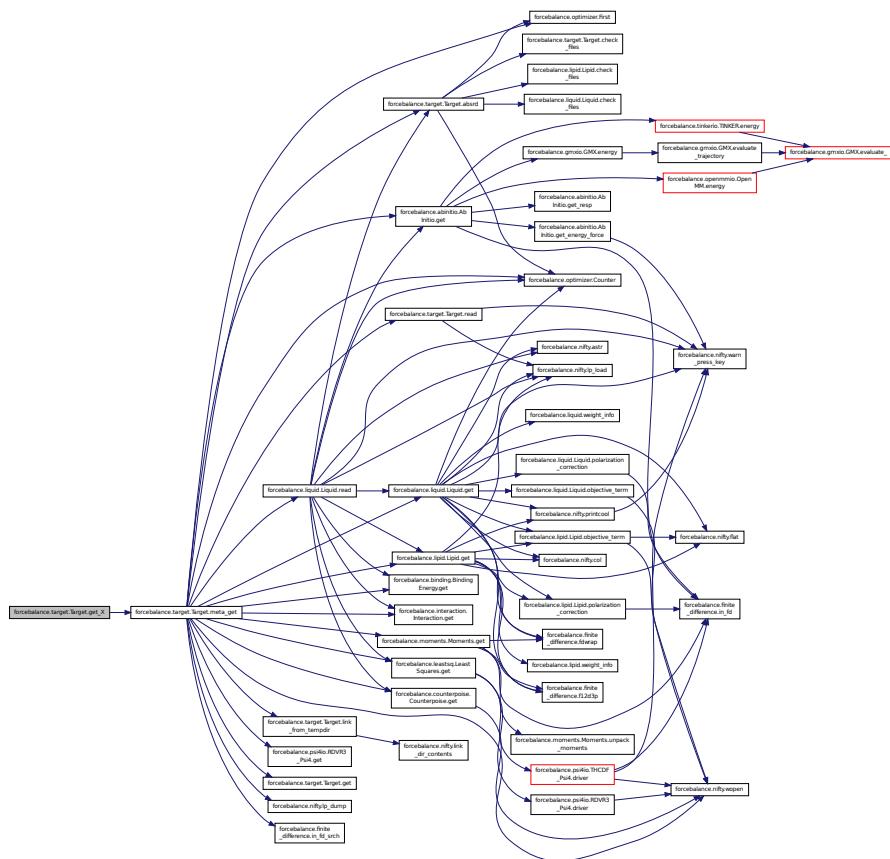
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

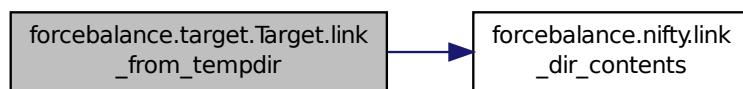
Here is the call graph for this function:



**def forcebalance.leastsq.LeastSquares.indicate ( self )** Definition at line 41 of file leastsq.py.

**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

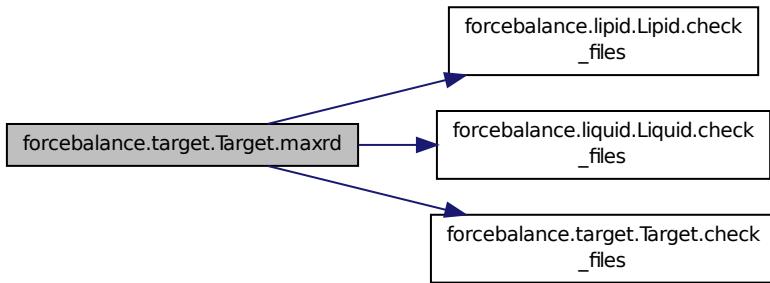
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

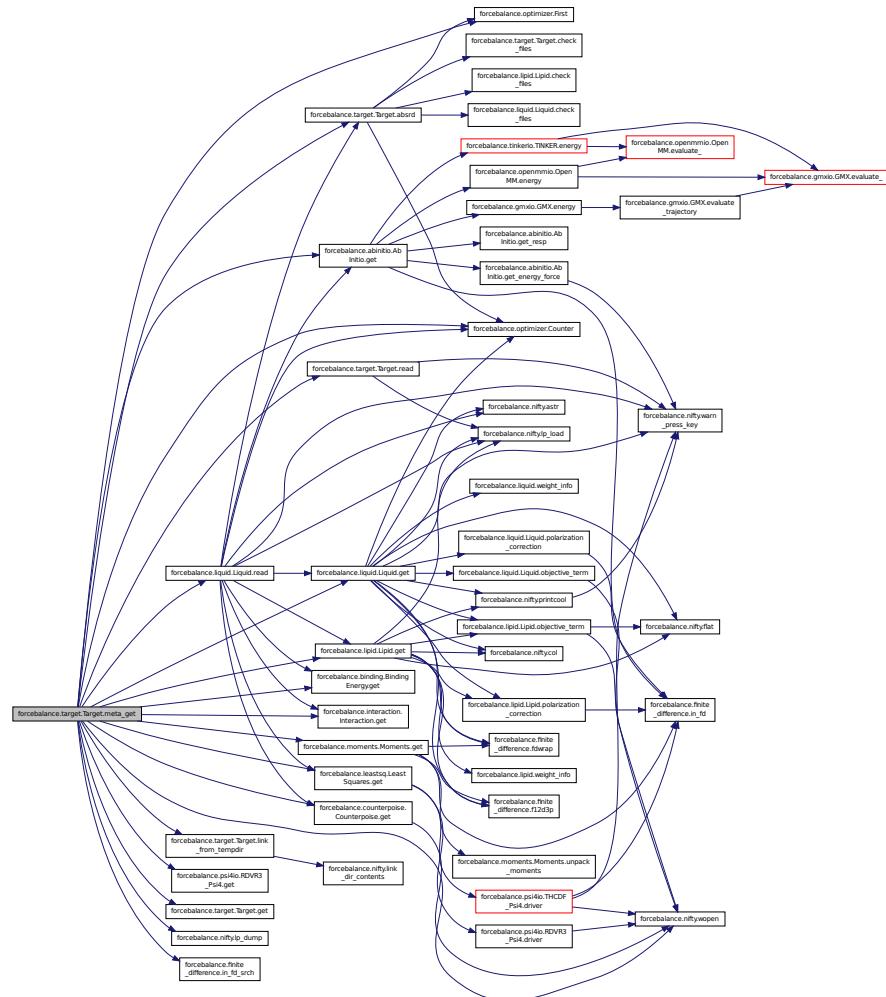
Definition at line 447 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.meta_get( self, mvals, AGrad=False, AHess=False, customdir=None ) [inherited] Wrapper around the get function.  
Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call read\(\) instead. The 'get' method should not worry about the directory that it's running in.  
Definition at line 511 of file target.py.
```

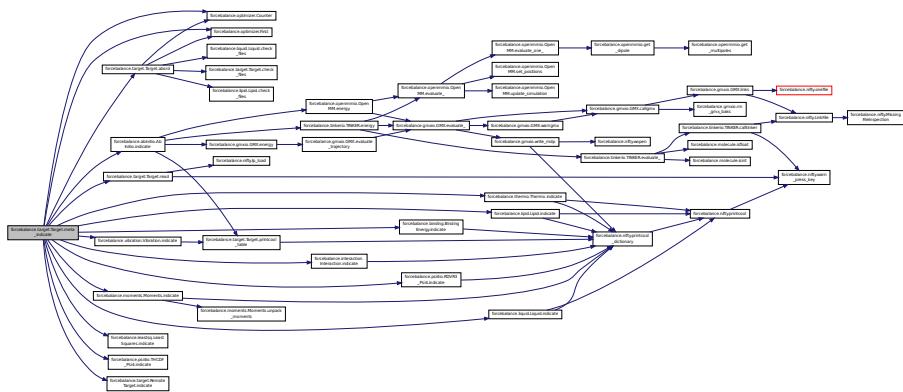
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

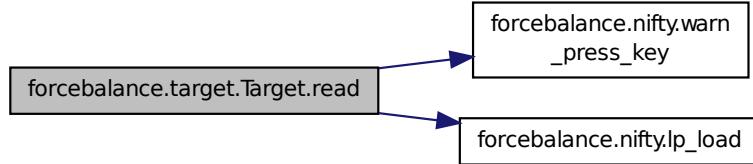
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

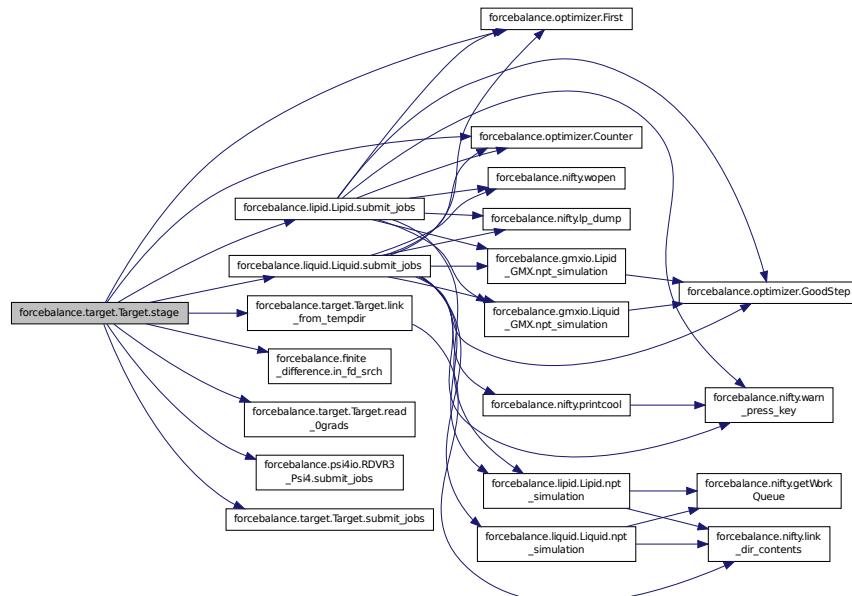
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

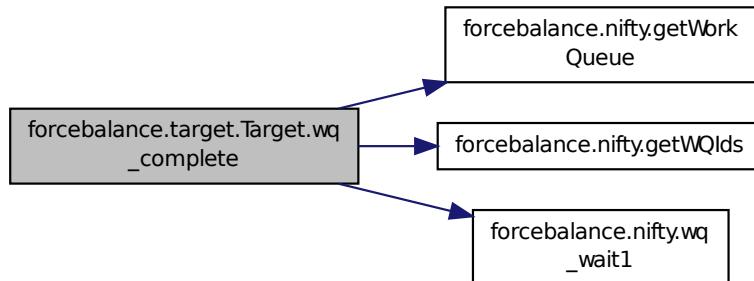


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.write_0grads( self, Ans ) [inherited] Write a file to the target directory containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

#### 8.30.4 Member Data Documentation

**forcebalance.leastsq.LeastSquares.D** Definition at line 126 of file leastsq.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)  
Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.  
Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.leastsq.LeastSquares.MAQ** Dictionary for derivative terms.  
Definition at line 88 of file leastsq.py.

**forcebalance.leastsq.LeastSquares.objective** Definition at line 127 of file leastsq.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.  
Dictionary of whether to call the derivatives.  
Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.  
Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.  
Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.  
Definition at line 172 of file target.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.  
Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number  
The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [**inherited**] Relative directory of target.

Temporary (working) directory; it is temp/(target\_name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [**inherited**] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate** [**inherited**] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

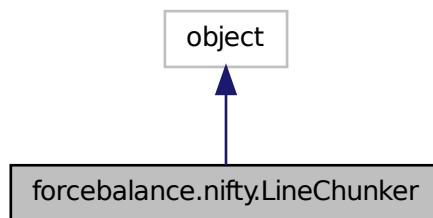
Definition at line 162 of file target.py.

The documentation for this class was generated from the following file:

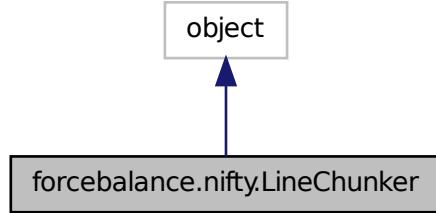
- [leastsq.py](#)

## 8.31 forcebalance.nifty.LineChunker Class Reference

Inheritance diagram for forcebalance.nifty.LineChunker:



Collaboration diagram for forcebalance.nifty.LineChunker:



#### Public Member Functions

- def `__init__`
- def `push`
- def `close`
- def `nomnom`
- def `__enter__`
- def `__exit__`

#### Public Attributes

- `callback`
- `buf`

#### 8.31.1 Detailed Description

Definition at line 943 of file nifty.py.

#### 8.31.2 Constructor & Destructor Documentation

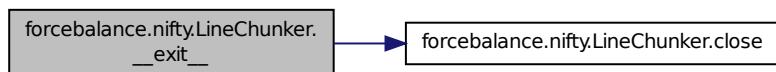
`def forcebalance.nifty.LineChunker.__init__( self, callback )` Definition at line 944 of file nifty.py.

#### 8.31.3 Member Function Documentation

`def forcebalance.nifty.LineChunker.__enter__( self )` Definition at line 963 of file nifty.py.

`def forcebalance.nifty.LineChunker.__exit__( self, args, kwargs )` Definition at line 966 of file nifty.py.

Here is the call graph for this function:



```
def forcebalance.nifty.LineChunker.close ( self ) Definition at line 952 of file nifty.py.  
  
def forcebalance.nifty.LineChunker.nomnom ( self ) Definition at line 956 of file nifty.py.  
  
def forcebalance.nifty.LineChunker.push ( self, data ) Definition at line 948 of file nifty.py.  
Here is the call graph for this function:
```



#### 8.31.4 Member Data Documentation

**forcebalance.nifty.LineChunker.buf** Definition at line 946 of file nifty.py.

**forcebalance.nifty.LineChunker.callback** Definition at line 945 of file nifty.py.

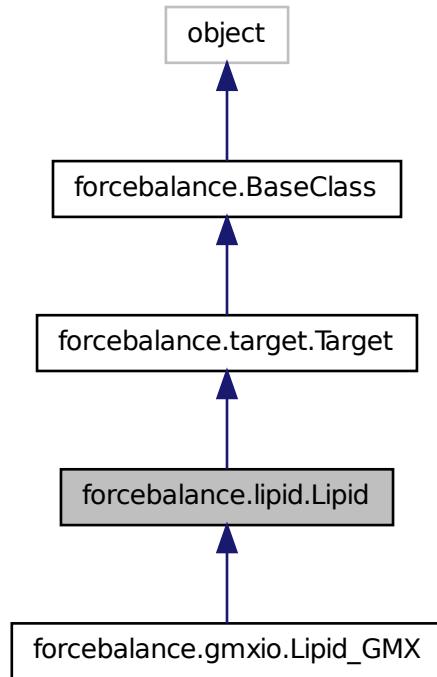
The documentation for this class was generated from the following file:

- [nifty.py](#)

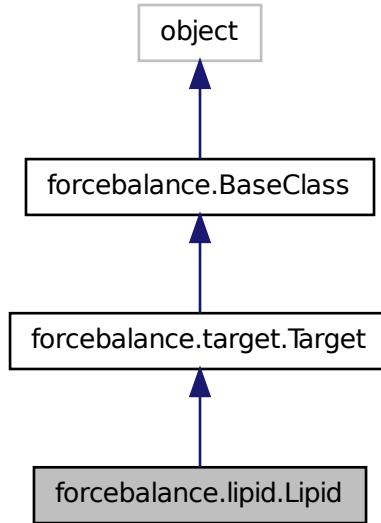
### 8.32 forcebalance.lipid.Lipid Class Reference

Subclass of Target for lipid property matching.

Inheritance diagram for forcebalance.lipid.Lipid:



Collaboration diagram for forcebalance.lipid.Lipid:



## Public Member Functions

- def `_init_`
- def `prepare_temp_directory`

*Prepare the temporary directory by copying in important files.*
- def `read_data`
- def `check_files`
- def `npt_simulation`

*Submit a NPT simulation to the Work Queue.*
- def `polarization_correction`
- def `indicate`
- def `objective_term`
- def `submit_jobs`
- def `get`

*Fitting of lipid bulk properties.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*
- def `get_H`

- def `link_from_tempdir`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `do_self_pol`
- `lipid_mol`
- `last_traj`
- `extra_output`
- `gas_engine`  
*Read the reference data.*
- `read_indicate`
- `write_indicate`
- `read_objective`
- `SavedMVal`  
*Saved force field mvals for all iterations.*
- `SavedTraj`  
*Saved trajectories for all iterations and all temperatures.*
- `MBarEnergy`  
*Evaluated energies for all trajectories (i.e.*
- `RefData`
- `PhasePoints`
- `Labels`
- `engname`
- `w_rho`  
*Density.*
- `w_alpha`
- `w_kappa`
- `w_cp`

- `w_eps0`
- `w_al`
- `w_scd`
- `Xp`
- `Wp`
- `Pp`
- `Gp`
- `Objective`
- `rd`  
*Root directory of the whole project.*
- `pgrad`  
*Iteration where we turn on zero-gradient skipping.*
- `tempbase`  
*Relative directory of target.*
- `tempdir`
- `rundir`  
`self.tempdir = os.path.join('temp',self.name)` *The directory in which the simulation is running - this can be updated.*
- `FF`  
*Need the forcefield (here for now)*
- `xct`  
*Counts how often the objective function was computed.*
- `gct`  
*Counts how often the gradient was computed.*
- `hct`  
*Counts how often the Hessian was computed.*
- `write_objective`  
*Whether to write objective.p at every iteration (true for all but remote.)*
- `verbose_options`
- `PrintOptionDict`

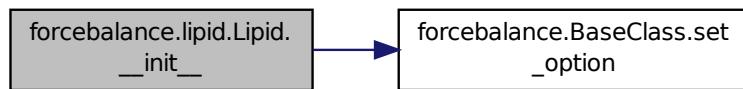
### 8.32.1 Detailed Description

Subclass of Target for lipid property matching.

Definition at line 51 of file lipid.py.

### 8.32.2 Constructor & Destructor Documentation

`def forcebalance.lipid.Lipid.__init__( self, options, tgt_opts, forcefield )` Definition at line 54 of file lipid.py.  
 Here is the call graph for this function:



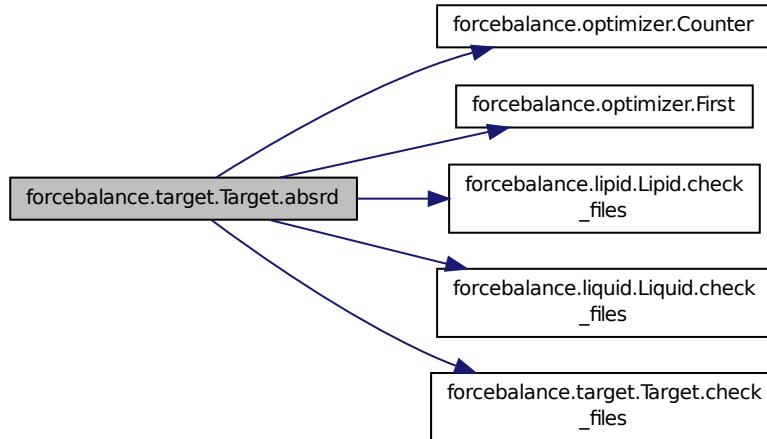
### 8.32.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.lipid.Lipid.check_files( self, there ) Definition at line 252 of file lipid.py.
```

```
def forcebalance.lipid.Lipid.get( self, mvals, AGrad = True, AHess = True ) Fitting of lipid bulk properties.
```

This is the current major direction of development for ForceBalance. Basically, fitting the QM energies / forces alone does not always give us the best simulation behavior. In many cases it makes more sense to try and reproduce some experimentally known data as well.

In order to reproduce experimentally known data, we need to run a simulation and compare the simulation result to experiment. The main challenge here is that the simulations are computationally intensive (i.e. they require energy and force evaluations), and furthermore the results are noisy. We need to run the simulations automatically and remotely (i.e. on clusters) and a good way to calculate the derivatives of the simulation results with respect to the parameter values.

This function contains some experimentally known values of the density and enthalpy of vaporization (Hvap) of lipid water. It launches the density and Hvap calculations on the cluster, and gathers the results / derivatives. The actual calculation of results / derivatives is done in a separate file.

After the results come back, they are gathered together to form an objective function.

#### Parameters

in	mvals	Mathematical parameter values
----	-------	-------------------------------

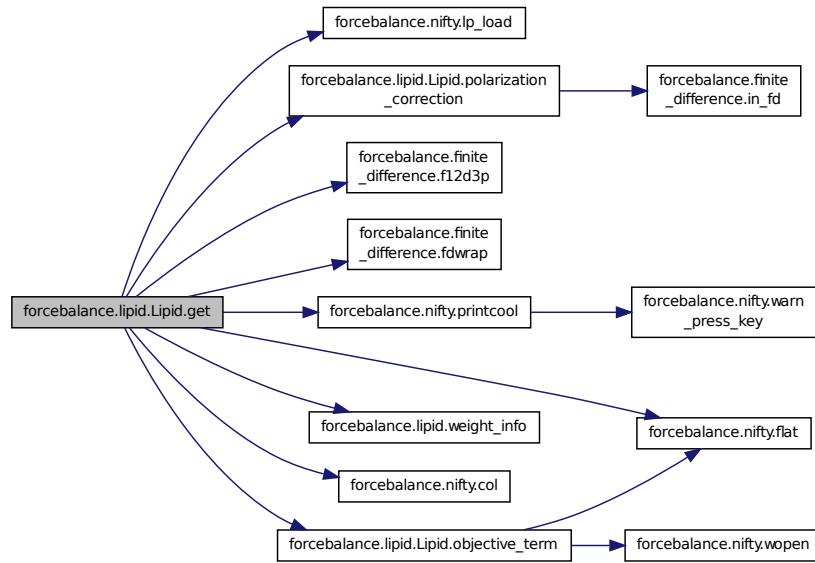
in	<i>AGrad</i>	Switch to turn on analytic gradient
in	<i>AHess</i>	Switch to turn on analytic Hessian

Returns

Answer Contribution to the objective function Fill in the weight matrix with MBAR weights where MBAR was run, and equal weights otherwise.

Definition at line 479 of file lipid.py.

Here is the call graph for this function:



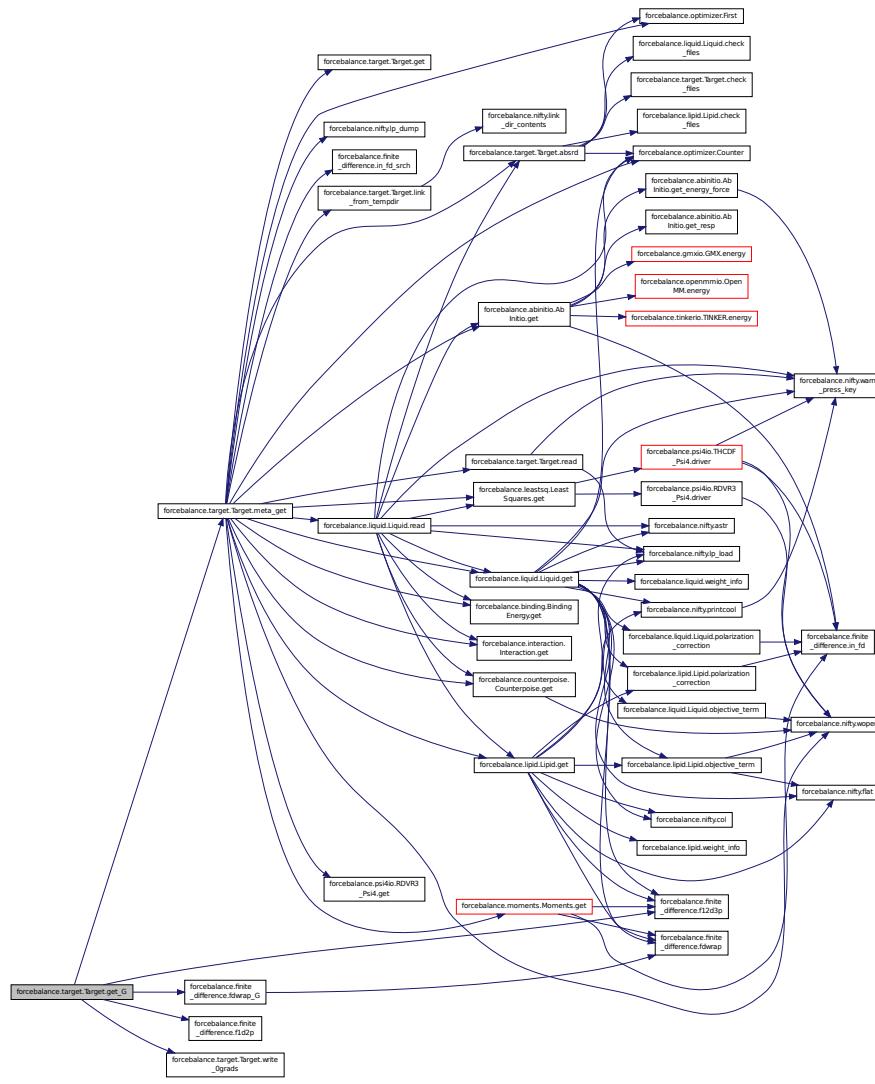
**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1.pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



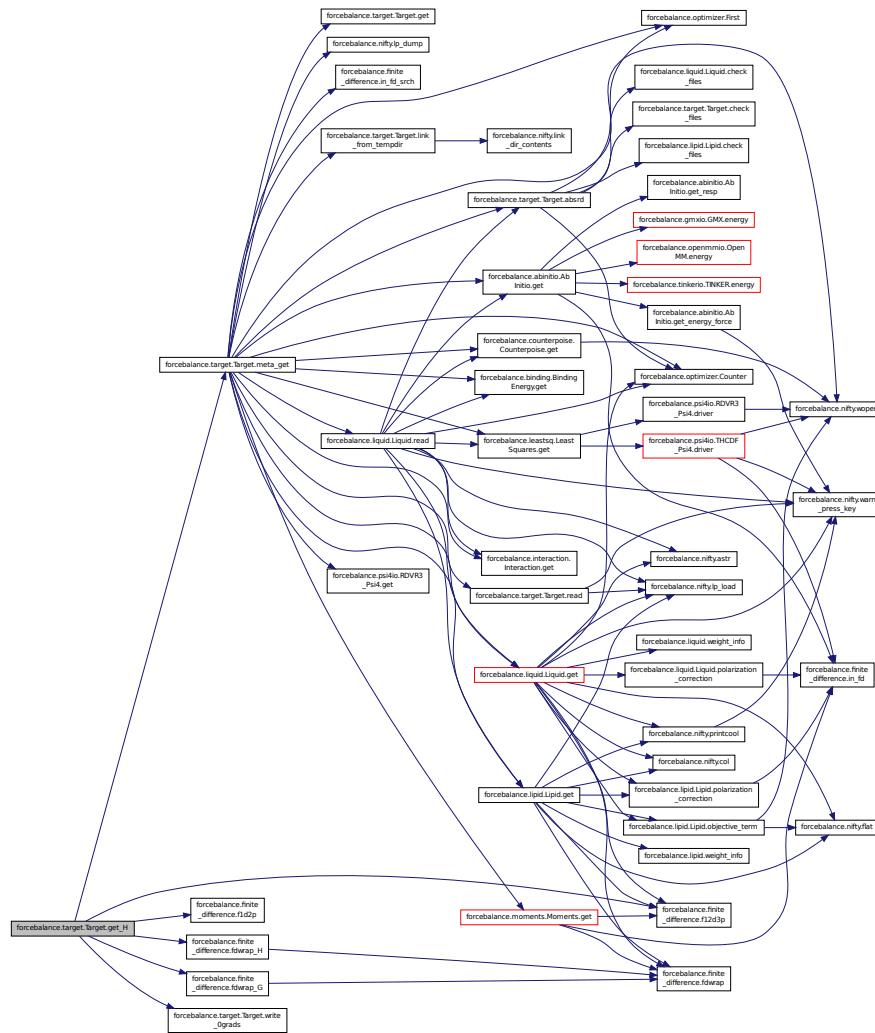
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2.pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

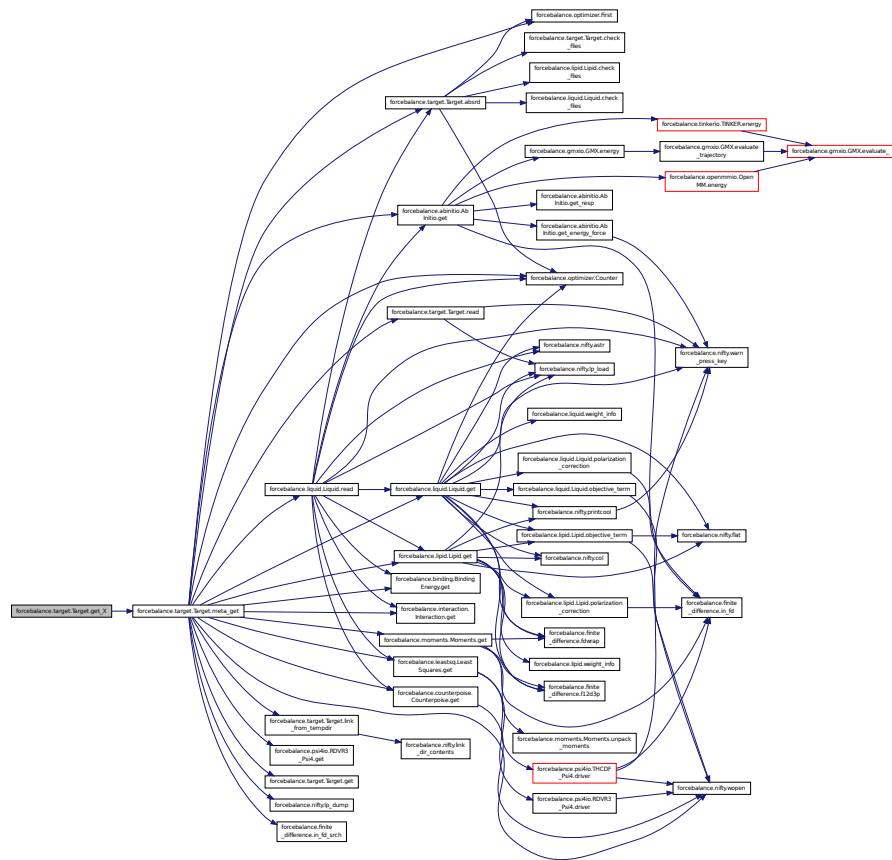
Here is the call graph for this function:



**def forcebalance.target.Target.get.X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

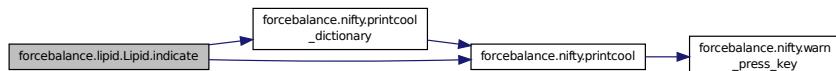
Definition at line 184 of file target.py.

Here is the call graph for this function:



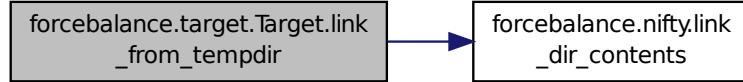
**def forcebalance.lipid.Lipid.indicate ( self )** Definition at line 303 of file lipid.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

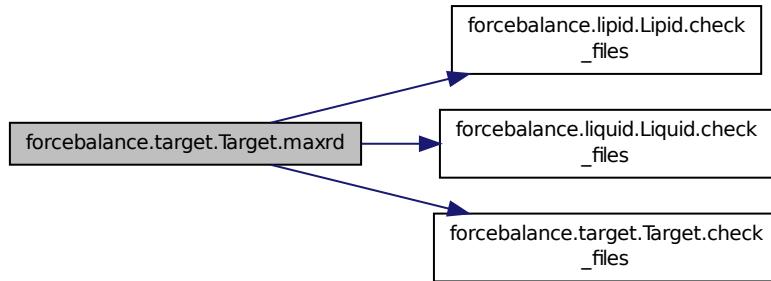
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

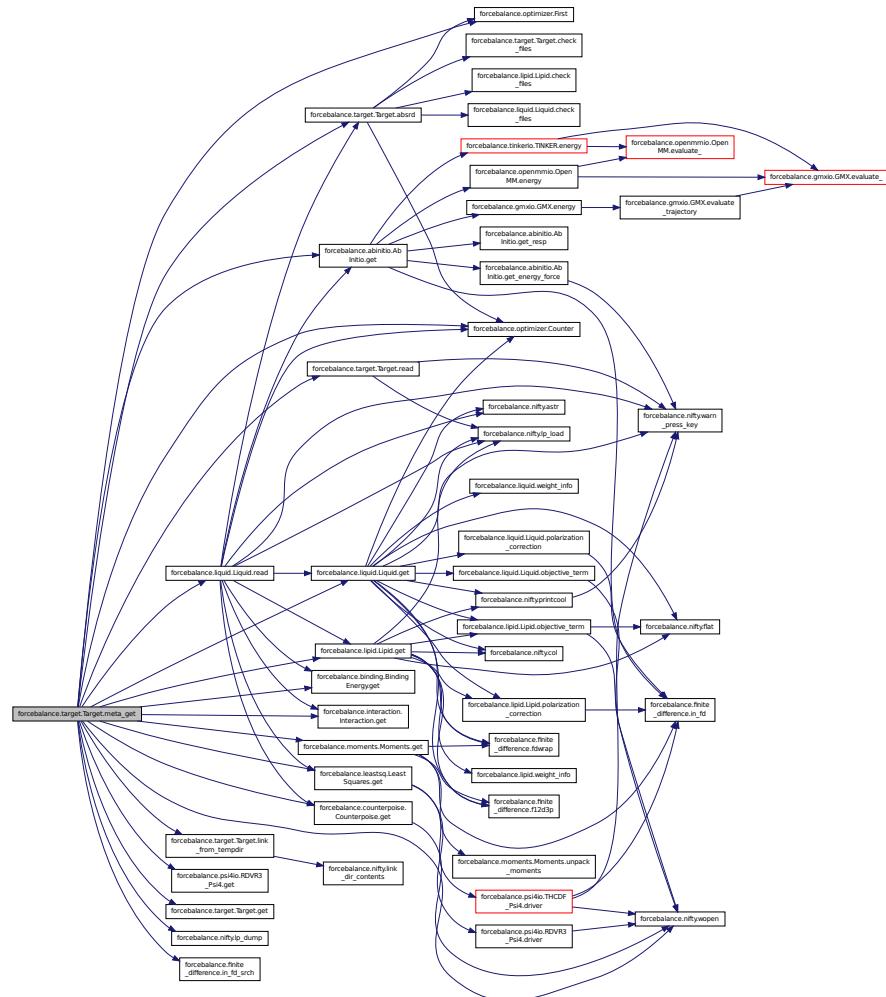


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

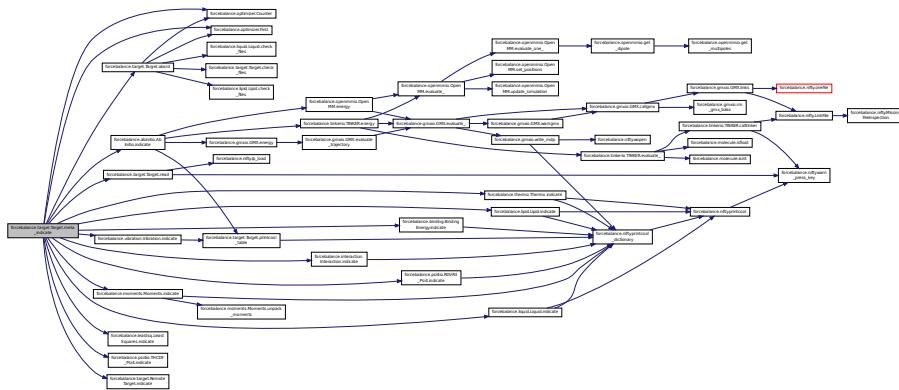
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

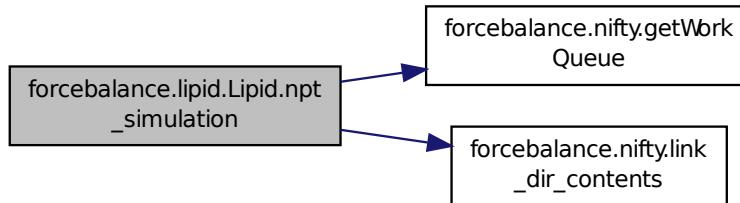
Here is the call graph for this function:



**def forcebalance.lipid.Lipid.npt\_simulation ( self, temperature, pressure, simnum )** Submit a NPT simulation to the Work Queue.

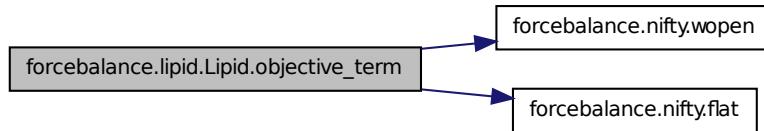
Definition at line 269 of file lipid.py.

Here is the call graph for this function:



**def forcebalance.lipid.Lipid.objective\_term ( self, points, expname, calc, err, grad, name = "Quantity", SubAverage = False )** Definition at line 330 of file lipid.py.

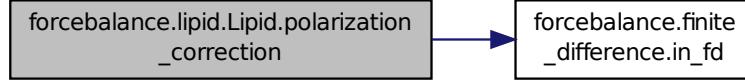
Here is the call graph for this function:



```
def forcebalance.lipid.Lipid.polarization_correction ( self, mvals )
```

Definition at line 287 of file lipid.py.

Here is the call graph for this function:

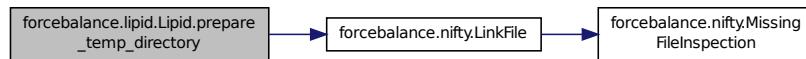


```
def forcebalance.lipid.Lipid.prepare_temp_directory ( self )
```

Prepare the temporary directory by copying in important files.

Definition at line 148 of file lipid.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table ( self, data = OrderedDict ([]), headings = [], banner = None, footnote = None, color = 0 )
```

[inherited] Print target information in an organized table format. Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

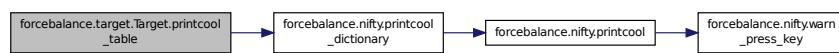
The input should be something like:

#### Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

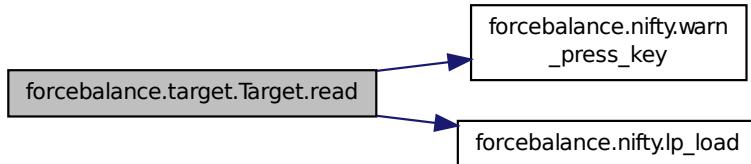
Here is the call graph for this function:



```
def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.
```

Definition at line 379 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.read_0grads ( self ) [inherited] Read a file from the target directory containing names of parameters that don't contribute to the gradient.
```

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

```
def forcebalance.lipid.Lipid.read_data ( self ) Definition at line 155 of file lipid.py.
```

```
def forcebalance.target.Target.refresh_temp_directory ( self ) [inherited] Back up the temporary directory if desired, delete it and then create a new one.
```

Definition at line 321 of file target.py.

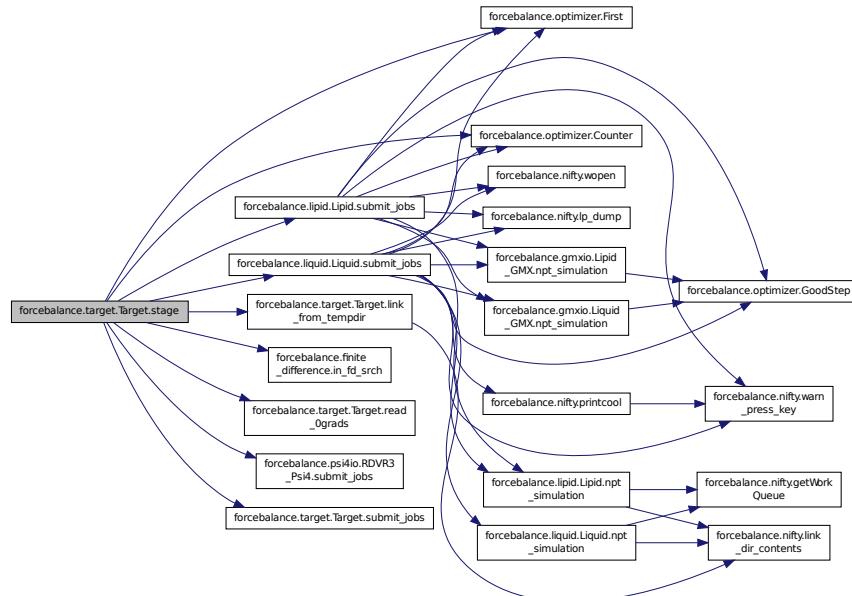
```
def forcebalance.BaseClass.set_option ( self, in_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

The 'get' method should not worry about the directory that it's running in.

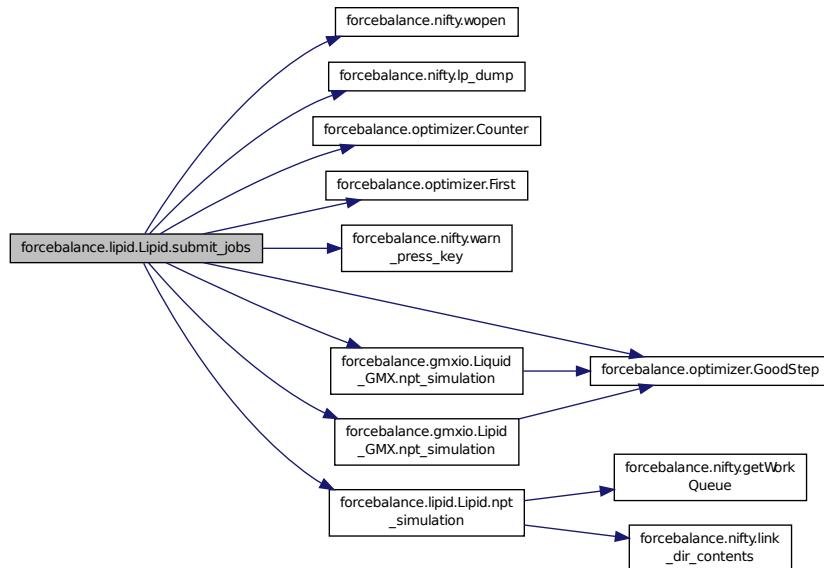
Definition at line 565 of file target.py.

Here is the call graph for this function:



**def forcebalance.lipid.Lipid.submit\_jobs ( self, mvals, AGrad = True, AHess = True )** Definition at line 414 of file lipid.py.

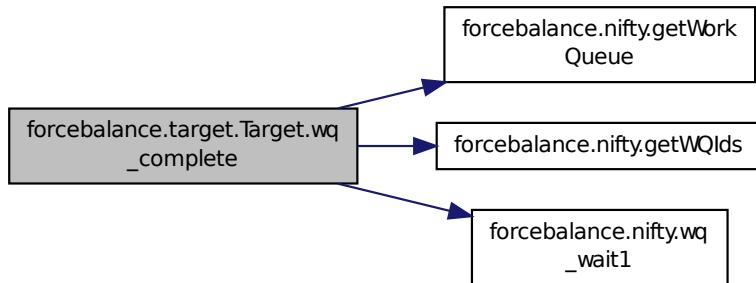
Here is the call graph for this function:



**def forcebalance.target.Target.wq\_complete ( self ) [inherited]** This method determines whether the Work Queue tasks for the current target have completed.

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.32.4 Member Data Documentation

**forcebalance.lipid.Lipid.do\_self.pol** Definition at line 96 of file lipid.py.

**forcebalance.lipid.Lipid.engname** Definition at line 276 of file lipid.py.

**forcebalance.lipid.Lipid.extra\_output** Definition at line 112 of file lipid.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.lipid.Lipid.gas\_engine** Read the reference data.

Definition at line 127 of file lipid.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.lipid.Lipid.Gp** Definition at line 724 of file lipid.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.lipid.Lipid.Labels** Definition at line 242 of file lipid.py.

**forcebalance.lipid.Lipid.last\_traj** Definition at line 110 of file lipid.py.

**forcebalance.lipid.Lipid.lipid\_mol** Definition at line 108 of file lipid.py.

**forcebalance.lipid.Lipid.MBarEnergy** Evaluated energies for all trajectories (i.e. all iterations and all temperatures), using all mvals  
Definition at line 144 of file lipid.py.

**forcebalance.lipid.Lipid.Objective** Definition at line 726 of file lipid.py.

**forcebalance.target.Target.pgrad** [inherited] Iteration where we turn on zero-gradient skipping.  
Dictionary of whether to call the derivatives.  
Definition at line 127 of file target.py.

**forcebalance.lipid.Lipid.PhasePoints** Definition at line 238 of file lipid.py.

**forcebalance.lipid.Lipid.Pp** Definition at line 721 of file lipid.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd** [inherited] Root directory of the whole project.  
Submit jobs to the Work Queue.  
Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.  
Definition at line 123 of file target.py.

**forcebalance.lipid.Lipid.read\_indicate** Definition at line 129 of file lipid.py.

**forcebalance.lipid.Lipid.read\_objective** Definition at line 132 of file lipid.py.

**forcebalance.lipid.Lipid.RefData** Definition at line 166 of file lipid.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.  
Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number  
The 'customdir' is customizable and can go below anything.  
Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.  
Definition at line 158 of file target.py.

**forcebalance.lipid.Lipid.SavedMVal** Saved force field mvals for all iterations.  
Definition at line 140 of file lipid.py.

**forcebalance.lipid.Lipid.SavedTraj** Saved trajectories for all iterations and all temperatures.  
Definition at line 142 of file lipid.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.  
Temporary (working) directory; it is temp/(target\_name) Used for storing temporary variables that don't change through the course of the optimization  
Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.lipid.Lipid.w\_al** Definition at line 698 of file lipid.py.

**forcebalance.lipid.Lipid.w\_alpha** Definition at line 694 of file lipid.py.

**forcebalance.lipid.Lipid.w\_cp** Definition at line 696 of file lipid.py.

**forcebalance.lipid.Lipid.w\_eps0** Definition at line 697 of file lipid.py.

**forcebalance.lipid.Lipid.w\_kappa** Definition at line 695 of file lipid.py.

**forcebalance.lipid.Lipid.w\_rho** Density.

Ignore enthalpy. Thermal expansion coefficient. Isothermal compressibility. Isobaric heat capacity. Static dielectric constant. Average area per lipid Deuterium order parameter Estimation of errors.

Definition at line 693 of file lipid.py.

**forcebalance.lipid.Lipid.w\_scd** Definition at line 699 of file lipid.py.

**forcebalance.lipid.Lipid.Wp** Definition at line 719 of file lipid.py.

**forcebalance.lipid.Lipid.write\_indicate** Definition at line 130 of file lipid.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

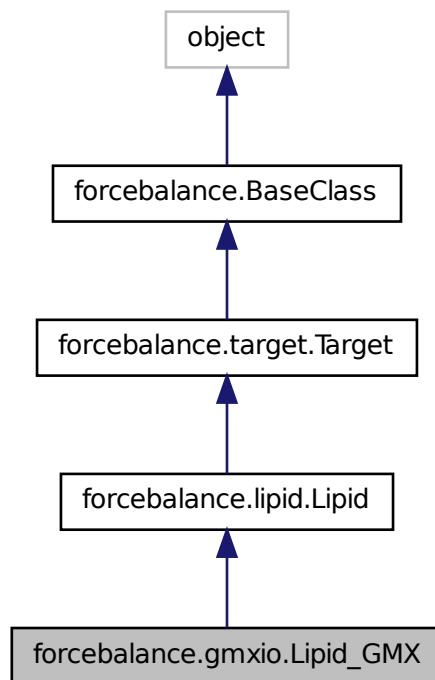
**forcebalance.lipid.Lipid.Xp** Definition at line 717 of file lipid.py.

The documentation for this class was generated from the following file:

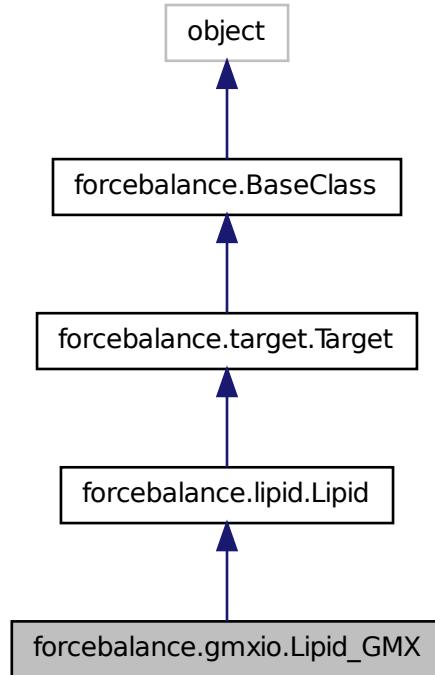
- [lipid.py](#)

### 8.33 forcebalance.gmxio.Lipid\_GMX Class Reference

Inheritance diagram for forcebalance.gmxio.Lipid\_GMX:



Collaboration diagram for forcebalance.gmxio.Lipid\_GMX:



### Public Member Functions

- def `__init__`
- def `npt_simulation`

*Submit a NPT simulation to the Work Queue.*
- def `prepare_temp_directory`

*Prepare the temporary directory by copying in important files.*
- def `read_data`
- def `check_files`
- def `polarization_correction`
- def `indicate`
- def `objective_term`
- def `submit_jobs`
- def `get`

*Fitting of lipid bulk properties.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_0grads`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_0grads`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*

- def [get\\_G](#)  
*Computes the objective function contribution and its gradient.*
- def [get\\_H](#)  
*Computes the objective function contribution and its gradient / Hessian.*
- def [link\\_from\\_tempdir](#)
- def [refresh\\_temp\\_directory](#)  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def [read](#)  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def [absrd](#)  
*Supply the correct directory specified by user's "read" option.*
- def [maxrd](#)  
*Supply the latest existing temp-directory containing valid data.*
- def [meta\\_indicate](#)  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def [meta\\_get](#)  
*Wrapper around the get function.*
- def [stage](#)  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def [wq\\_complete](#)  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def [printcool\\_table](#)  
*Print target information in an organized table format.*
- def [\\_\\_setattr\\_\\_](#)
- def [set\\_option](#)

## Public Attributes

- [engine\\_](#)
- [engname](#)
- [nptpx](#)
- [nptfiles](#)
- [scripts](#)
- [extra\\_output](#)
- [LfDict](#)
- [LfDict\\_New](#)
- [last\\_traj](#)
- [do\\_self\\_pol](#)
- [lipid\\_mol](#)
- [gas\\_engine](#)  
*Read the reference data.*
- [read\\_indicate](#)
- [write\\_indicate](#)
- [read\\_objective](#)
- [SavedMVal](#)  
*Saved force field mvals for all iterations.*
- [SavedTraj](#)  
*Saved trajectories for all iterations and all temperatures.*

- **MBarEnergy**  
*Evaluated energies for all trajectories (i.e.*
- **RefData**
- **PhasePoints**
- **Labels**
- **w\_rho**  
*Density.*
- **w\_alpha**
- **w\_kappa**
- **w\_cp**
- **w\_eps0**
- **w\_al**
- **w\_scd**
- **Xp**
- **Wp**
- **Pp**
- **Gp**
- **Objective**
- **rd**  
*Root directory of the whole project.*
- **pgrad**  
*Iteration where we turn on zero-gradient skipping.*
- **tempbase**  
*Relative directory of target.*
- **tempdir**
- **rundir**  
*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- **FF**  
*Need the forcefield (here for now)*
- **xct**  
*Counts how often the objective function was computed.*
- **gct**  
*Counts how often the gradient was computed.*
- **hct**  
*Counts how often the Hessian was computed.*
- **write\_objective**  
*Whether to write objective.p at every iteration (true for all but remote.)*
- **verbose\_options**
- **PrintOptionDict**

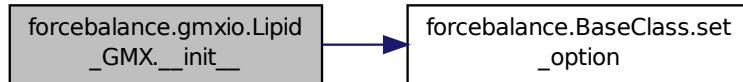
### 8.33.1 Detailed Description

Definition at line 1396 of file gmxio.py.

### 8.33.2 Constructor & Destructor Documentation

```
def forcebalance.gmxio.Lipid_GMX.__init__ ( self, options, tgt_opts, forcefield ) Definition at line 1397 of file gmxio.py.
```

Here is the call graph for this function:



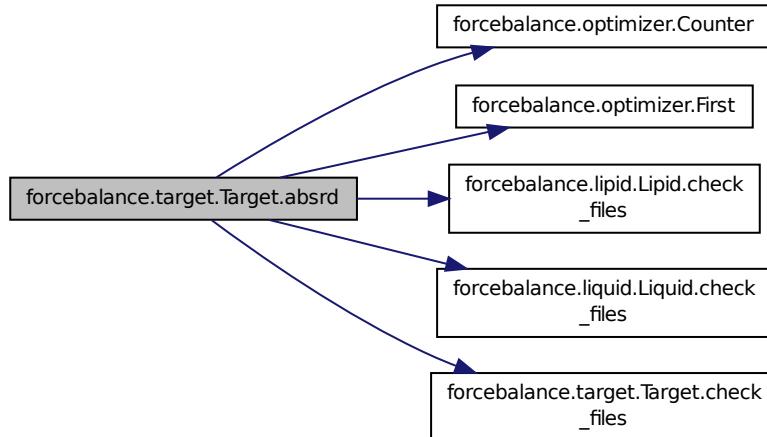
### 8.33.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.lipid.Lipid.check_files ( self, there ) [inherited] Definition at line 252 of file lipid.py.
```

```
def forcebalance.lipid.Lipid.get ( self, mvals, AGrad = True, AHess = True ) [inherited] Fitting of lipid bulk properties.
```

This is the current major direction of development for ForceBalance. Basically, fitting the QM energies / forces alone does not always give us the best simulation behavior. In many cases it makes more sense to try and reproduce some experimentally known data as well.

In order to reproduce experimentally known data, we need to run a simulation and compare the simulation result to experiment. The main challenge here is that the simulations are computationally intensive (i.e. they require energy and force evaluations), and furthermore the results are noisy. We need to run the simulations automatically and remotely (i.e. on clusters) and a good way to calculate the derivatives of the simulation results with respect to the parameter values.

This function contains some experimentally known values of the density and enthalpy of vaporization ( $H_{vap}$ ) of lipid water. It launches the density and  $H_{vap}$  calculations on the cluster, and gathers the results / derivatives. The actual calculation of results / derivatives is done in a separate file.

After the results come back, they are gathered together to form an objective function.

#### Parameters

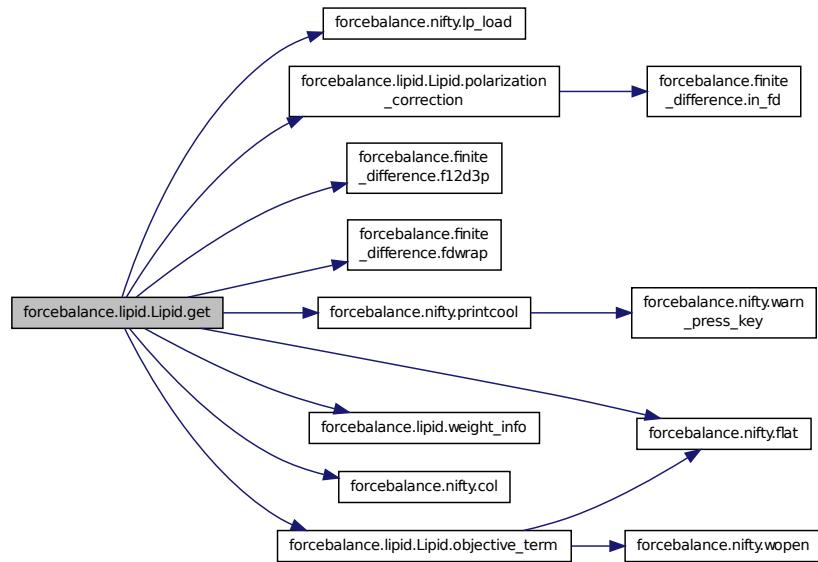
in	<i>mvals</i>	Mathematical parameter values
in	<i>AGrad</i>	Switch to turn on analytic gradient
in	<i>AHess</i>	Switch to turn on analytic Hessian

#### Returns

Answer Contribution to the objective function Fill in the weight matrix with MBAR weights where MBAR was run, and equal weights otherwise.

Definition at line 479 of file lipid.py.

Here is the call graph for this function:



**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

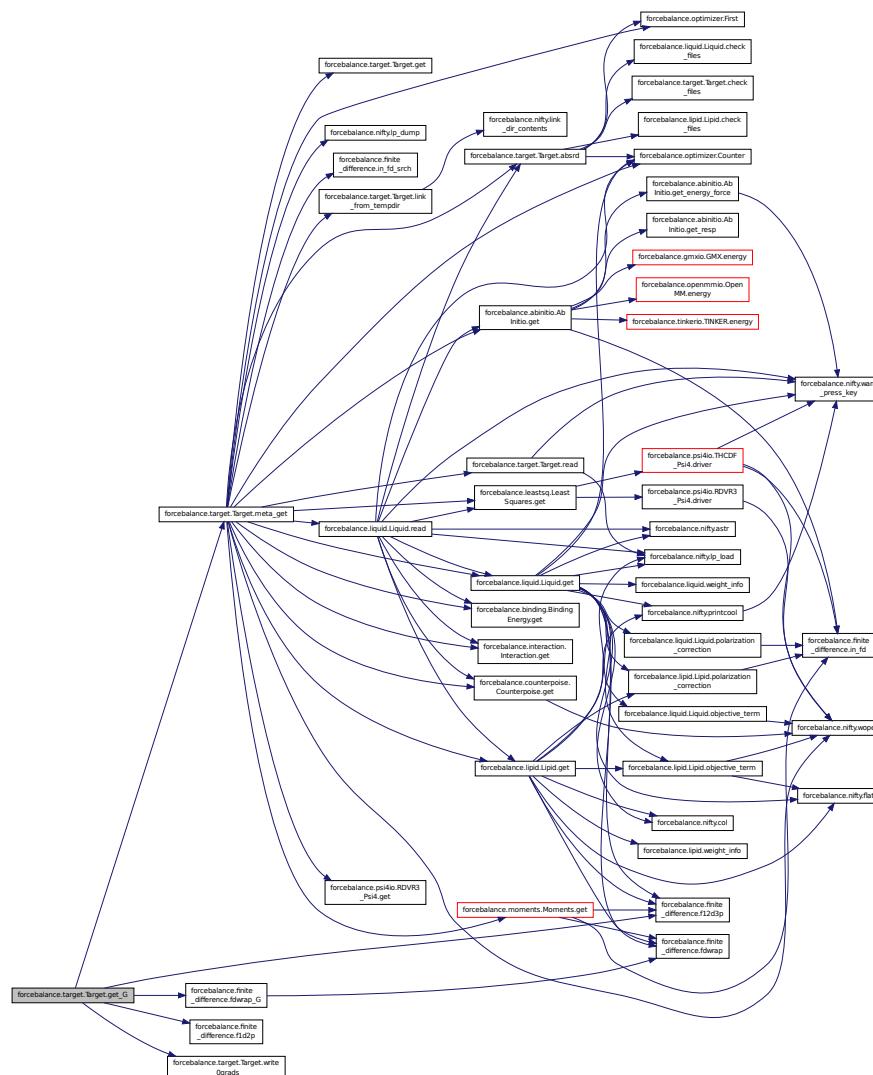
First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned

on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



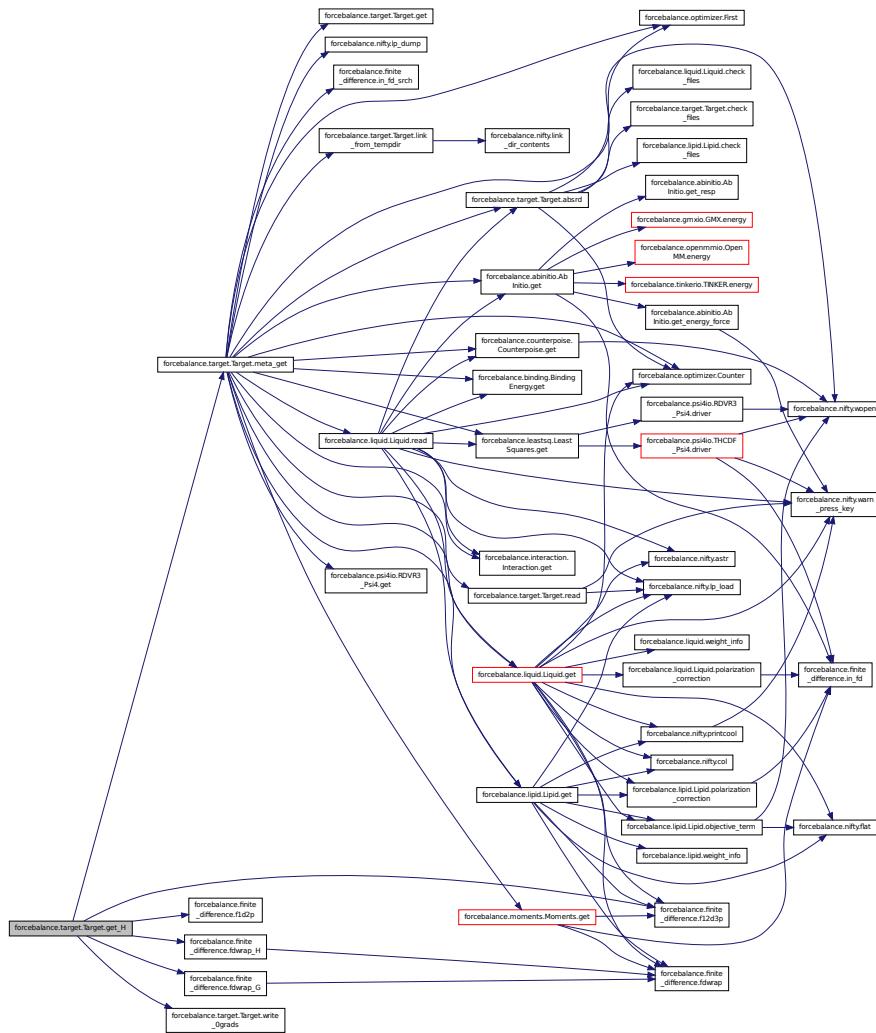
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

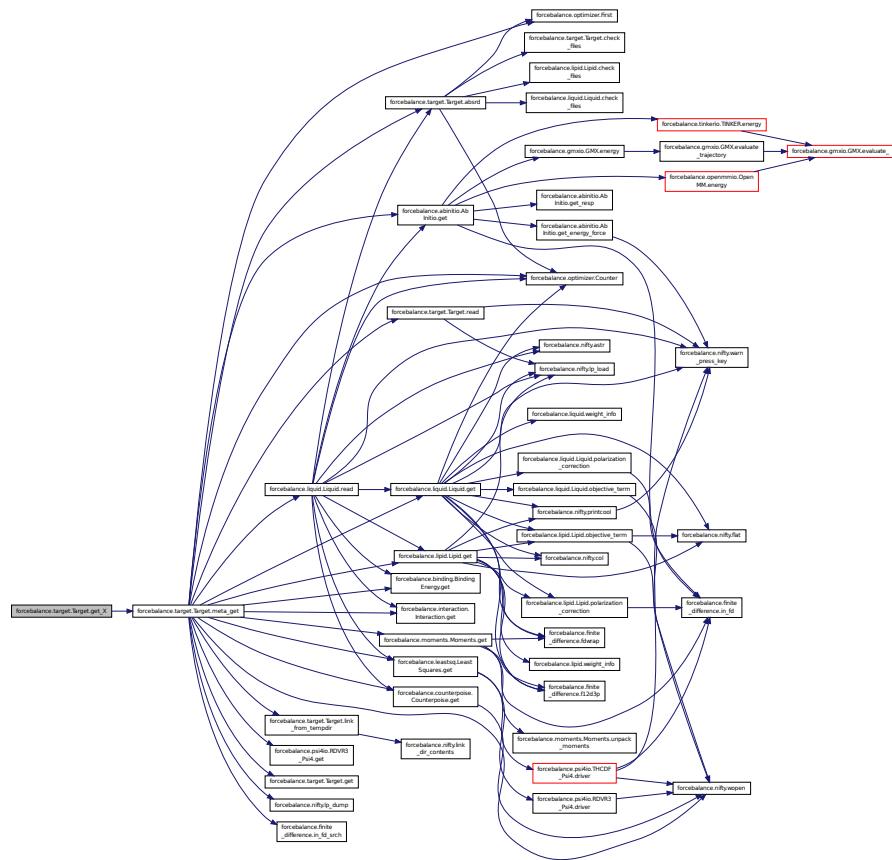
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

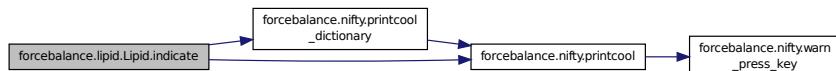
Definition at line 184 of file target.py.

Here is the call graph for this function:



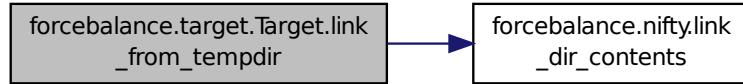
**def forcebalance.lipid.Lipid.indicate ( self ) [inherited]** Definition at line 303 of file lipid.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

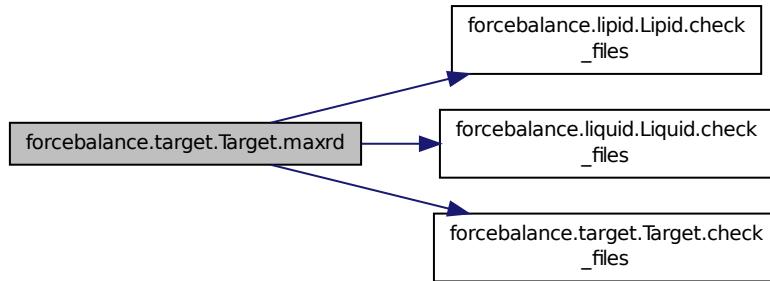
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

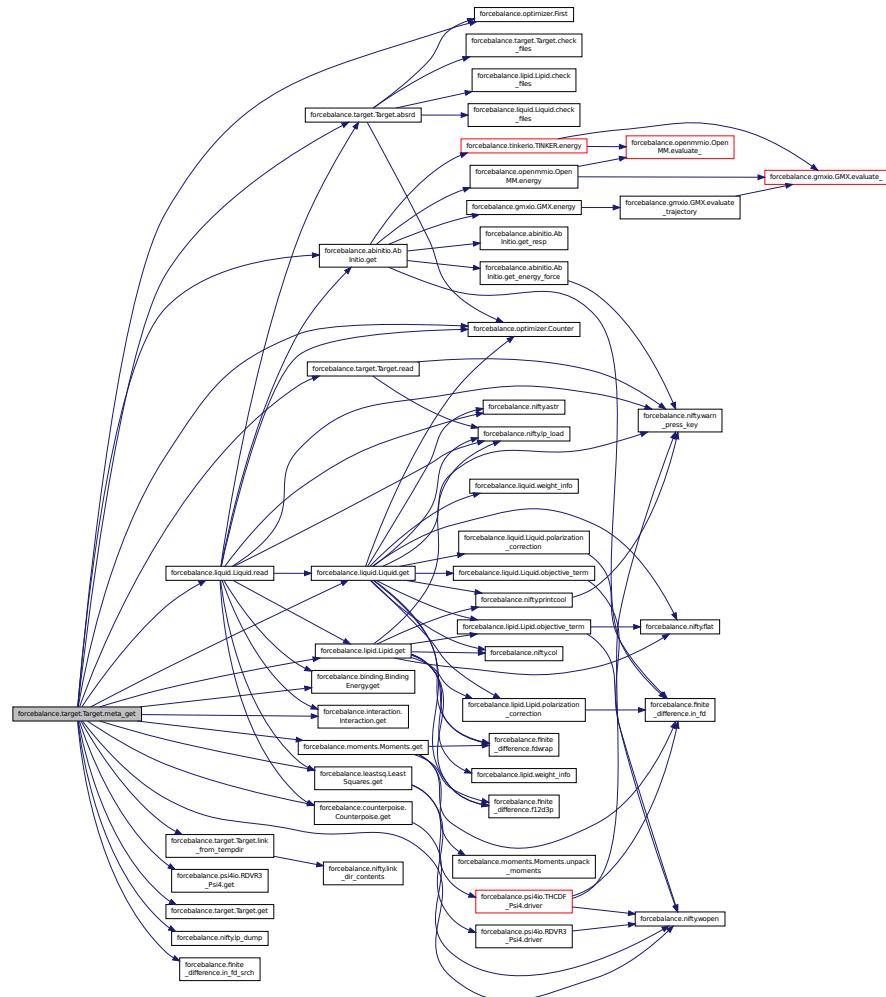


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

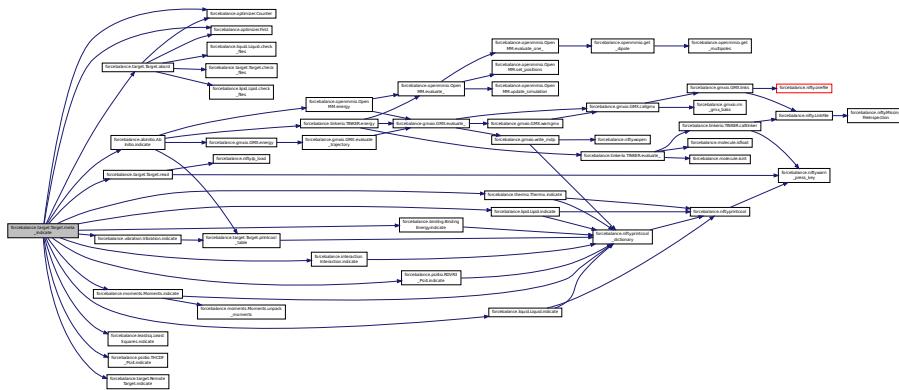
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

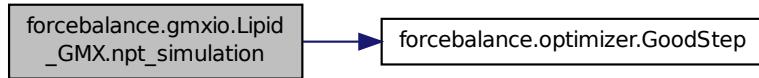
Here is the call graph for this function:



**def forcebalance.gmxio.Lipid\_GMX.npt\_simulation ( self, temperature, pressure, simnum )** Submit a NPT simulation to the Work Queue.

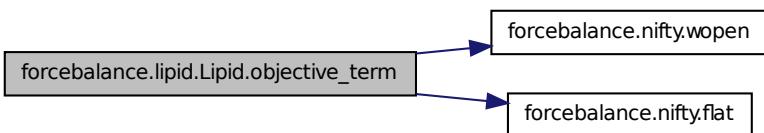
Definition at line 1434 of file gmxio.py.

Here is the call graph for this function:



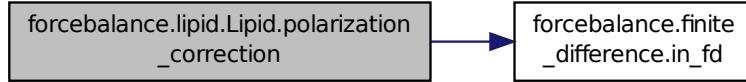
**def forcebalance.lipid.Lipid.objective\_term ( self, points, expname, calc, err, grad, name = "Quantity", SubAverage = False )** [inherited] Definition at line 330 of file lipid.py.

Here is the call graph for this function:



**def forcebalance.lipid.Lipid.polarization\_correction ( self, mvals )** [inherited] Definition at line 287 of file lipid.py.

Here is the call graph for this function:



**def forcebalance.lipid.Lipid.prepare\_temp\_directory ( self ) [inherited]** Prepare the temporary directory by copying in important files.

Definition at line 148 of file lipid.py.

Here is the call graph for this function:



**def forcebalance.target.Target.printcool\_table ( self, data = OrderedDict ([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited]** Print target information in an organized table format.

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

The input should be something like:

#### Parameters

<b>data</b>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<b>headings</b>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<b>banner</b>	Optional heading line, which will be printed at the top in the title.
<b>footnote</b>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

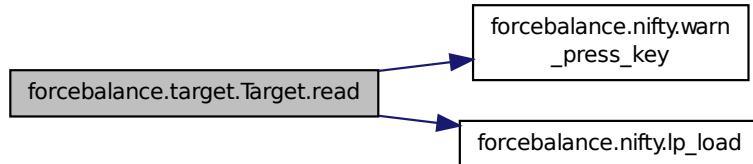
Here is the call graph for this function:



```
def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.
```

Definition at line 379 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.read_0grads ( self ) [inherited] Read a file from the target directory containing names of parameters that don't contribute to the gradient.
```

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

```
def forcebalance.lipid.Lipid.read_data ( self ) [inherited] Definition at line 155 of file lipid.py.
```

```
def forcebalance.target.Target.refresh_temp_directory ( self ) [inherited] Back up the temporary directory if desired, delete it and then create a new one.
```

Definition at line 321 of file target.py.

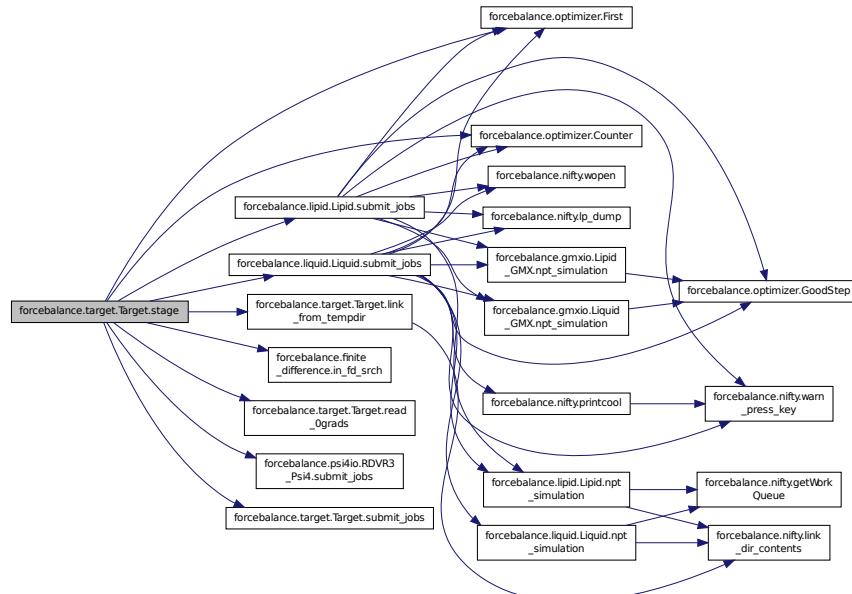
```
def forcebalance.BaseClass.set_option ( self, in_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

The 'get' method should not worry about the directory that it's running in.

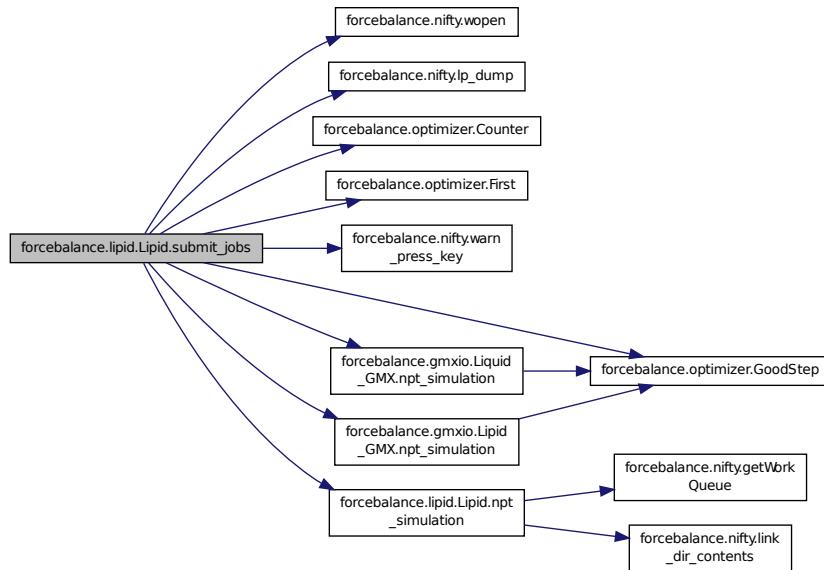
Definition at line 565 of file target.py.

Here is the call graph for this function:



**def forcebalance.lipid.Lipid.submit\_jobs ( self, mvals, AGrad = True, AHess = True ) [inherited]**  
Definition at line 414 of file lipid.py.

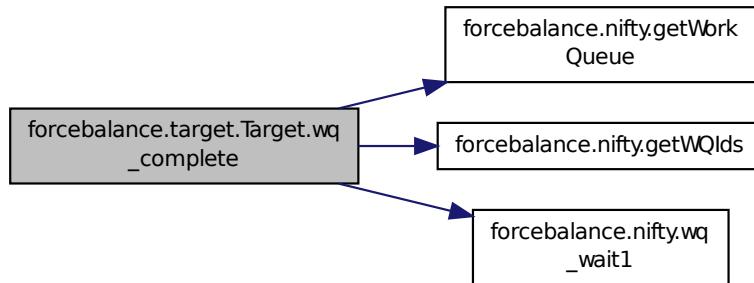
Here is the call graph for this function:



```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.write_0grads( self, Ans ) [inherited] Write a file to the target directory containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

#### 8.33.4 Member Data Documentation

`forcebalance.lipid.Lipid.do_self.pol` [inherited] Definition at line 96 of file lipid.py.

`forcebalance.gmxio.Lipid_GMX.engine` Definition at line 1407 of file gmxio.py.

`forcebalance.gmxio.Lipid_GMX.engname` Definition at line 1409 of file gmxio.py.

`forcebalance.gmxio.Lipid_GMX.extra_output` Definition at line 1424 of file gmxio.py.

`forcebalance.target.Target.FF` [inherited] Need the forcefield (here for now)

Definition at line 160 of file target.py.

`forcebalance.lipid.Lipid.gas.engine` [inherited] Read the reference data.

Definition at line 127 of file lipid.py.

`forcebalance.target.Target.gct` [inherited] Counts how often the gradient was computed.

Definition at line 164 of file target.py.

`forcebalance.lipid.Lipid.Gp` [inherited] Definition at line 724 of file lipid.py.

`forcebalance.target.Target.hct` [inherited] Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

`forcebalance.lipid.Lipid.Labels` [inherited] Definition at line 242 of file lipid.py.

**forcebalance.gmxio.Lipid\_GMX.last\_traj** Definition at line 1445 of file gmxio.py.

**forcebalance.gmxio.Lipid\_GMX.LfDict** Definition at line 1429 of file gmxio.py.

**forcebalance.gmxio.Lipid\_GMX.LfDict\_New** Definition at line 1430 of file gmxio.py.

**forcebalance.lipid.Lipid.lipid\_mol** [inherited] Definition at line 108 of file lipid.py.

**forcebalance.lipid.Lipid.MBarEnergy** [inherited] Evaluated energies for all trajectories (i.e. all iterations and all temperatures), using all mvals  
Definition at line 144 of file lipid.py.

**forcebalance.gmxio.Lipid\_GMX.nptfiles** Definition at line 1413 of file gmxio.py.

**forcebalance.gmxio.Lipid\_GMX.nptpx** Definition at line 1411 of file gmxio.py.

**forcebalance.lipid.Lipid.Objective** [inherited] Definition at line 726 of file lipid.py.

**forcebalance.target.Target.pgrad** [inherited] Iteration where we turn on zero-gradient skipping.  
Dictionary of whether to call the derivatives.  
Definition at line 127 of file target.py.

**forcebalance.lipid.Lipid.PhasePoints** [inherited] Definition at line 238 of file lipid.py.

**forcebalance.lipid.Lipid.Pp** [inherited] Definition at line 721 of file lipid.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd** [inherited] Root directory of the whole project.  
Submit jobs to the Work Queue.  
Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.  
Definition at line 123 of file target.py.

**forcebalance.lipid.Lipid.read\_indicate** [inherited] Definition at line 129 of file lipid.py.

**forcebalance.lipid.Lipid.read\_objective** [inherited] Definition at line 132 of file lipid.py.

**forcebalance.lipid.Lipid.RefData** [inherited] Definition at line 166 of file lipid.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number  
The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.lipid.Lipid.SavedMVal** [inherited] Saved force field mvals for all iterations.  
Definition at line 140 of file lipid.py.

**forcebalance.lipid.Lipid.SavedTraj** [inherited] Saved trajectories for all iterations and all temperatures.  
Definition at line 142 of file lipid.py.

**forcebalance.gmxio.Lipid\_GMX.scripts** Definition at line 1416 of file gmxio.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.  
Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization  
Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.lipid.Lipid.w\_al** [inherited] Definition at line 698 of file lipid.py.

**forcebalance.lipid.Lipid.w\_alpha** [inherited] Definition at line 694 of file lipid.py.

**forcebalance.lipid.Lipid.w\_cp** [inherited] Definition at line 696 of file lipid.py.

**forcebalance.lipid.Lipid.w\_eps0** [inherited] Definition at line 697 of file lipid.py.

**forcebalance.lipid.Lipid.w\_kappa** [inherited] Definition at line 695 of file lipid.py.

**forcebalance.lipid.Lipid.w\_rho** [inherited] Density.

Ignore enthalpy. Thermal expansion coefficient. Isothermal compressibility. Isobaric heat capacity. Static dielectric constant. Average area per lipid Deuterium order parameter Estimation of errors.  
Definition at line 693 of file lipid.py.

**forcebalance.lipid.Lipid.w\_scd** [inherited] Definition at line 699 of file lipid.py.

**forcebalance.lipid.Lipid.Wp** [inherited] Definition at line 719 of file lipid.py.

**forcebalance.lipid.Lipid.write\_indicate** [inherited] Definition at line 130 of file lipid.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)  
Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.  
Definition at line 162 of file target.py.

**forcebalance.lipid.Lipid.Xp** [inherited] Definition at line 717 of file lipid.py.

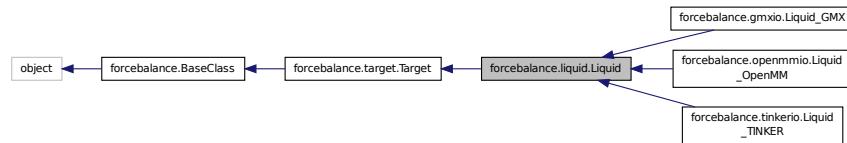
The documentation for this class was generated from the following file:

- [gmxio.py](#)

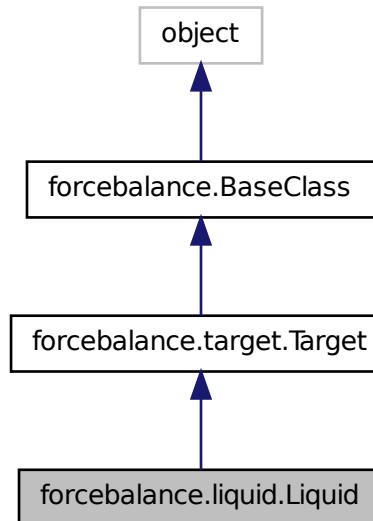
## 8.34 forcebalance.liquid.Liquid Class Reference

Subclass of Target for liquid property matching.

Inheritance diagram for forcebalance.liquid.Liquid:



Collaboration diagram for forcebalance.liquid.Liquid:



### Public Member Functions

- def `_init_`
- def `prepare_temp_directory`

*Prepare the temporary directory by copying in important files.*
- def `read_data`
- def `check_files`
- def `npt_simulation`

*Submit a NPT simulation to the Work Queue.*
- def `polarization_correction`
- def `indicate`
- def `objective_term`

- def `submit_jobs`
- def `read`  
*Read in time series for all previous iterations.*
- def `get`  
*Fitting of liquid bulk properties.*
- def `get_X`  
*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`  
*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`  
*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`  
*Computes the objective function contribution and its gradient.*
- def `get_H`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `do_self_pol`
- `liquid_mol`
- `gas_mol`
- `last_traj`
- `extra_output`
- `gas_engine`  
*Read the reference data.*
- `read_indicate`
- `write_indicate`
- `SavedTraj`  
*Saved trajectories for all iterations and all temperatures.*

- **MBarEnergy**  
*Evaluates energies for all trajectories (i.e.*
- **AllResults**  
*Saved results for all iterations self.SavedMVals = [].*
- **RefData**
- **PhasePoints**
- **Labels**
- **engname**
- **w\_rho**  
*Density.*
- **w\_hvap**
- **w\_alpha**
- **w\_kappa**
- **w\_cp**
- **w\_eps0**
- **Xp**
- **Wp**
- **Pp**
- **Gp**
- **Objective**
- **rd**  
*Root directory of the whole project.*
- **pgrad**  
*Iteration where we turn on zero-gradient skipping.*
- **tempbase**  
*Relative directory of target.*
- **tempdir**
- **rundir**  
*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- **FF**  
*Need the forcefield (here for now)*
- **xct**  
*Counts how often the objective function was computed.*
- **gct**  
*Counts how often the gradient was computed.*
- **hct**  
*Counts how often the Hessian was computed.*
- **read\_objective**  
*Whether to read objective.p from file when restarting an aborted run.*
- **write\_objective**  
*Whether to write objective.p at every iteration (true for all but remote.)*
- **verbose\_options**
- **PrintOptionDict**

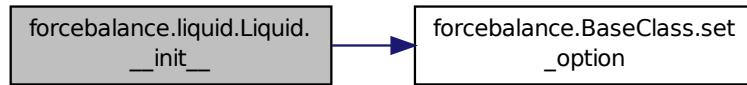
### 8.34.1 Detailed Description

Subclass of Target for liquid property matching.

Definition at line 51 of file liquid.py.

### 8.34.2 Constructor & Destructor Documentation

```
def forcebalance.liquid.Liquid.__init__( self, options, tgt_opts, forcefield ) Definition at line 54 of file liquid.py.  
Here is the call graph for this function:
```



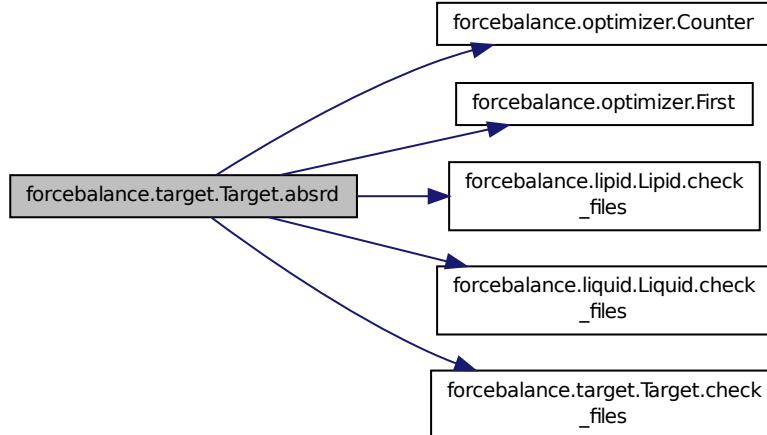
### 8.34.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.check_files( self, there ) Definition at line 253 of file liquid.py.
```

```
def forcebalance.liquid.Liquid.get( self, mvals, AGrad = True, AHess = True ) Fitting of liquid bulk properties.
```

This is the current major direction of development for ForceBalance. Basically, fitting the QM energies / forces alone does not always give us the best simulation behavior. In many cases it makes more sense to try and reproduce some experimentally known data as well.

In order to reproduce experimentally known data, we need to run a simulation and compare the simulation result to experiment. The main challenge here is that the simulations are computationally intensive (i.e. they require energy and force evaluations), and furthermore the results are noisy. We need to run the simulations automatically and remotely (i.e. on clusters) and a good way to calculate the derivatives of the simulation results with respect to the parameter values.

This function contains some experimentally known values of the density and enthalpy of vaporization (H<sub>vap</sub>) of liquid water. It launches the density and H<sub>vap</sub> calculations on the cluster, and gathers the results / derivatives. The actual calculation of results / derivatives is done in a separate file.

After the results come back, they are gathered together to form an objective function.

#### Parameters

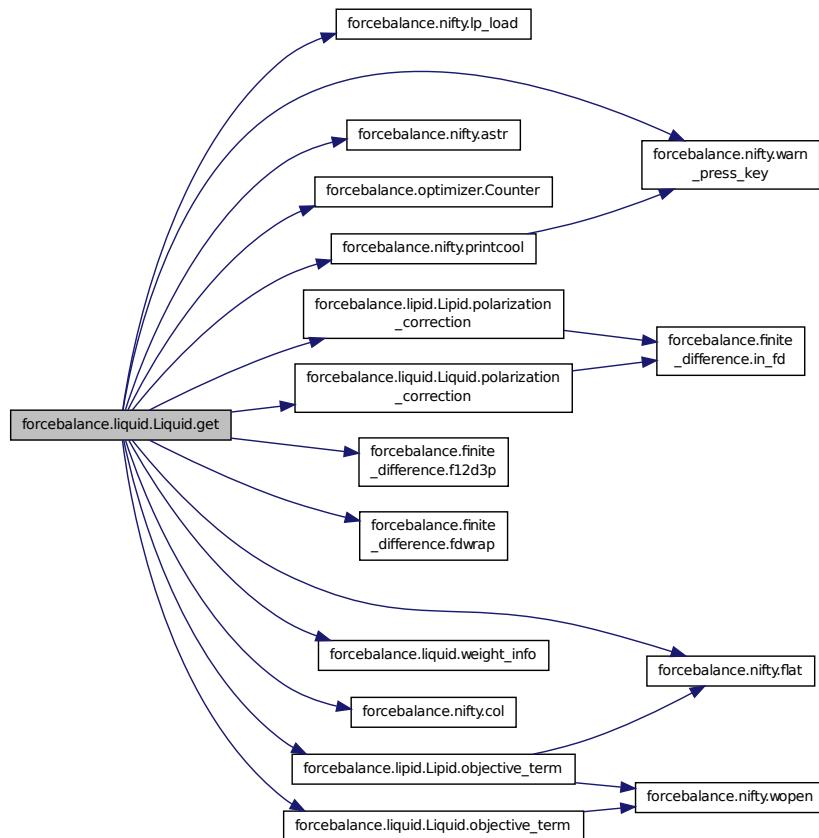
in	<i>mvals</i>	Mathematical parameter values
in	<i>AGrad</i>	Switch to turn on analytic gradient
in	<i>AHess</i>	Switch to turn on analytic Hessian

#### Returns

Answer Contribution to the objective function Fill in the weight matrix with MBAR weights where MBAR was run, and equal weights otherwise.

Definition at line 553 of file liquid.py.

Here is the call graph for this function:



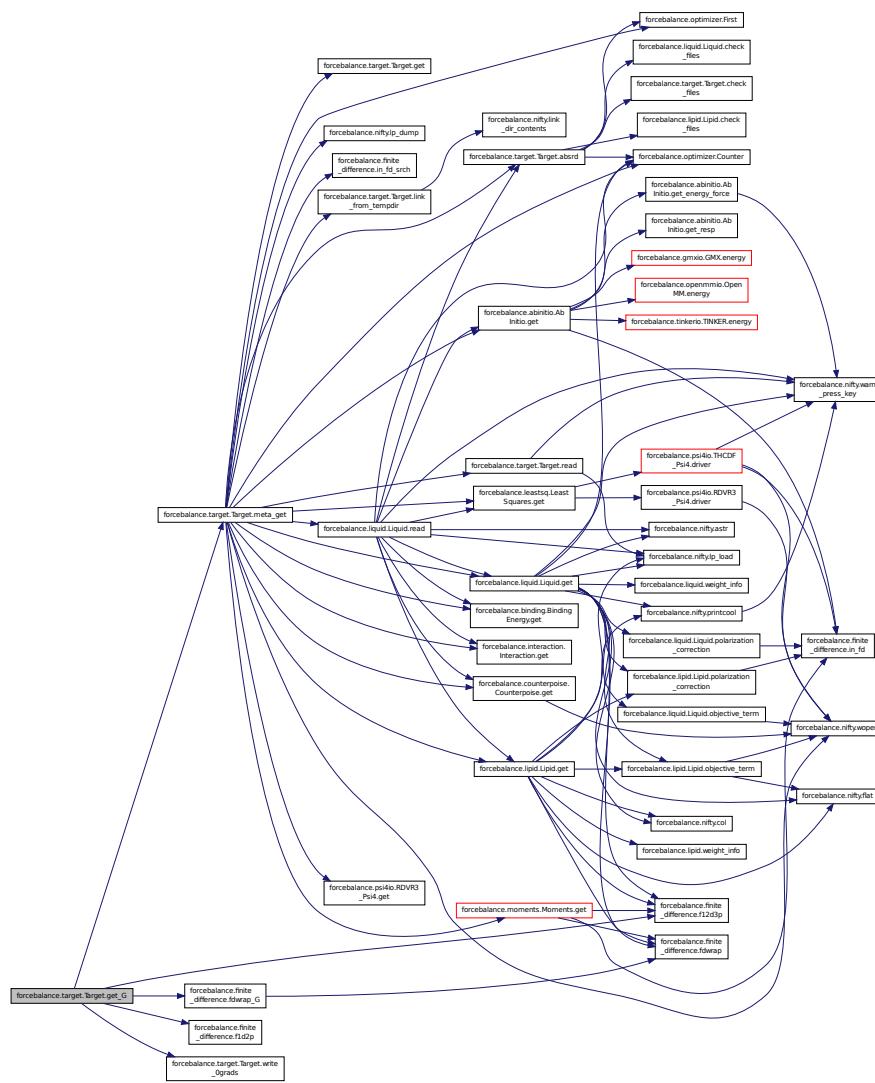
```
def forcebalance.target.Target.get_G ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.get_H ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient / Hessian.
```

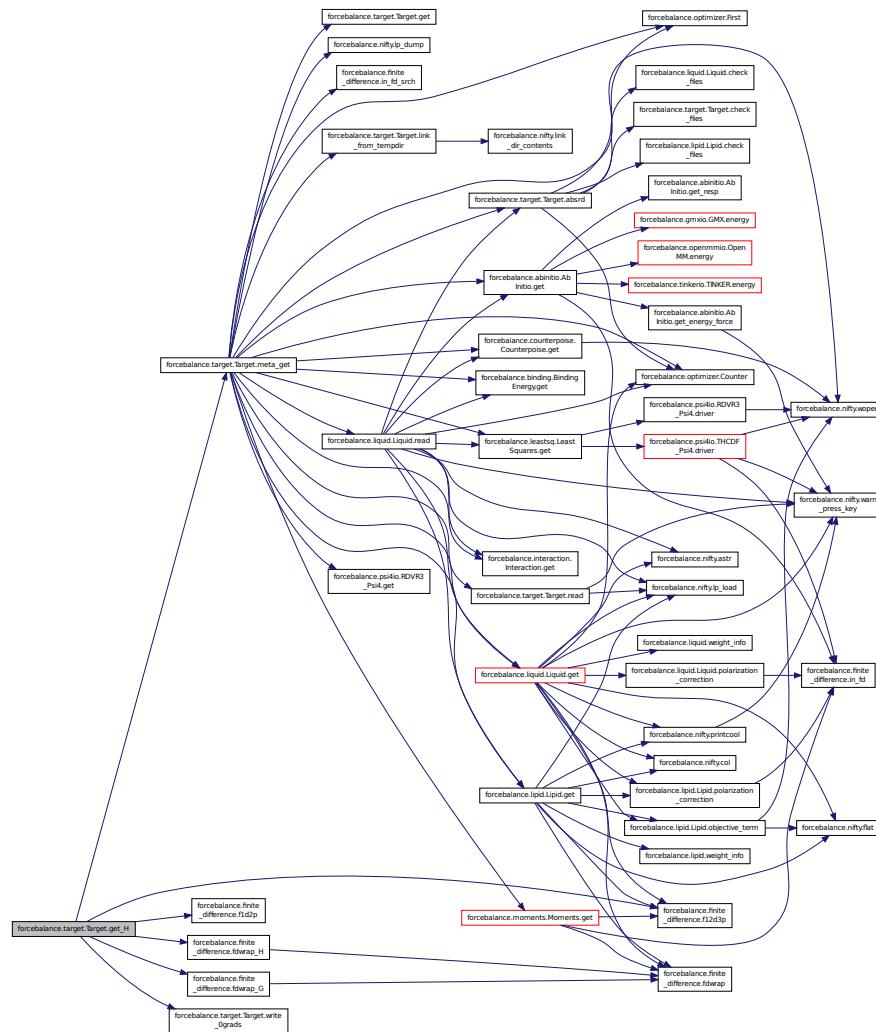
First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through

the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

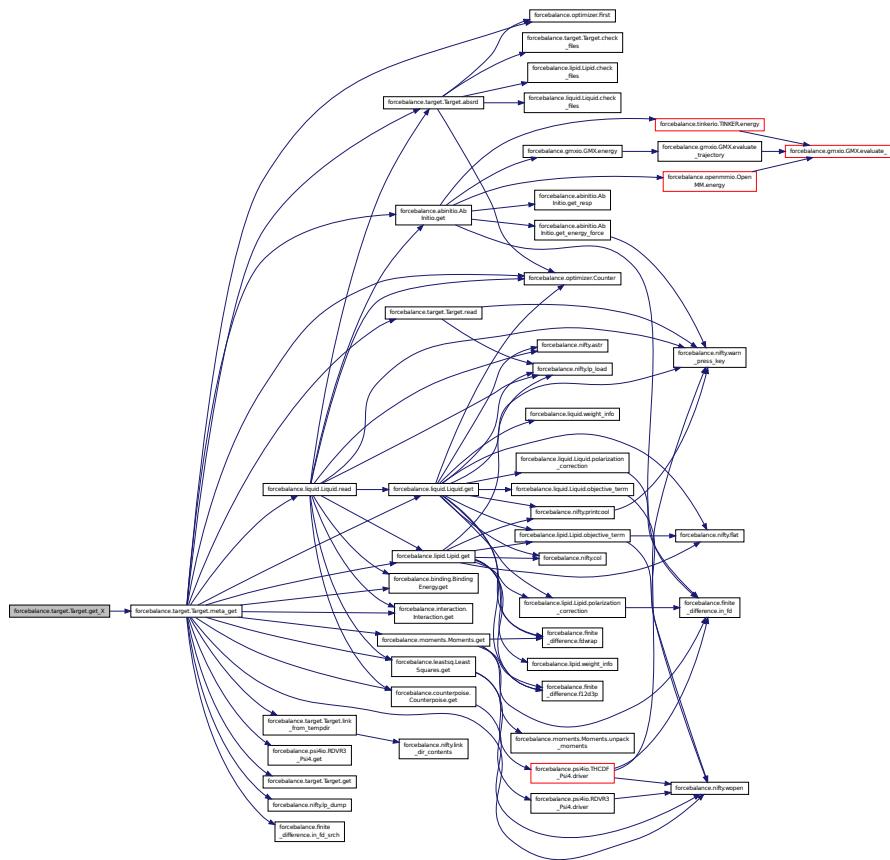
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

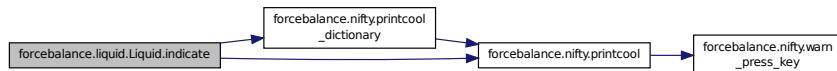
Definition at line 184 of file target.py.

Here is the call graph for this function:



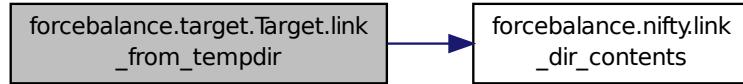
**def forcebalance.liquid.Liquid.indicate ( self )** Definition at line 304 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

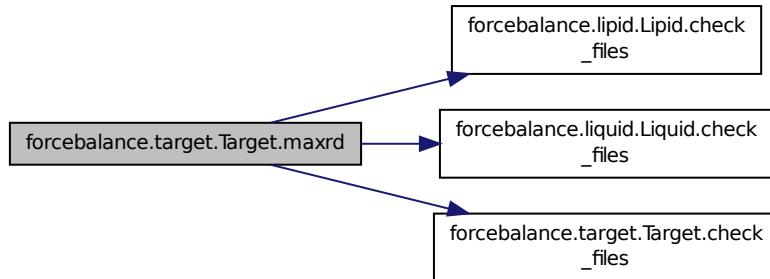
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

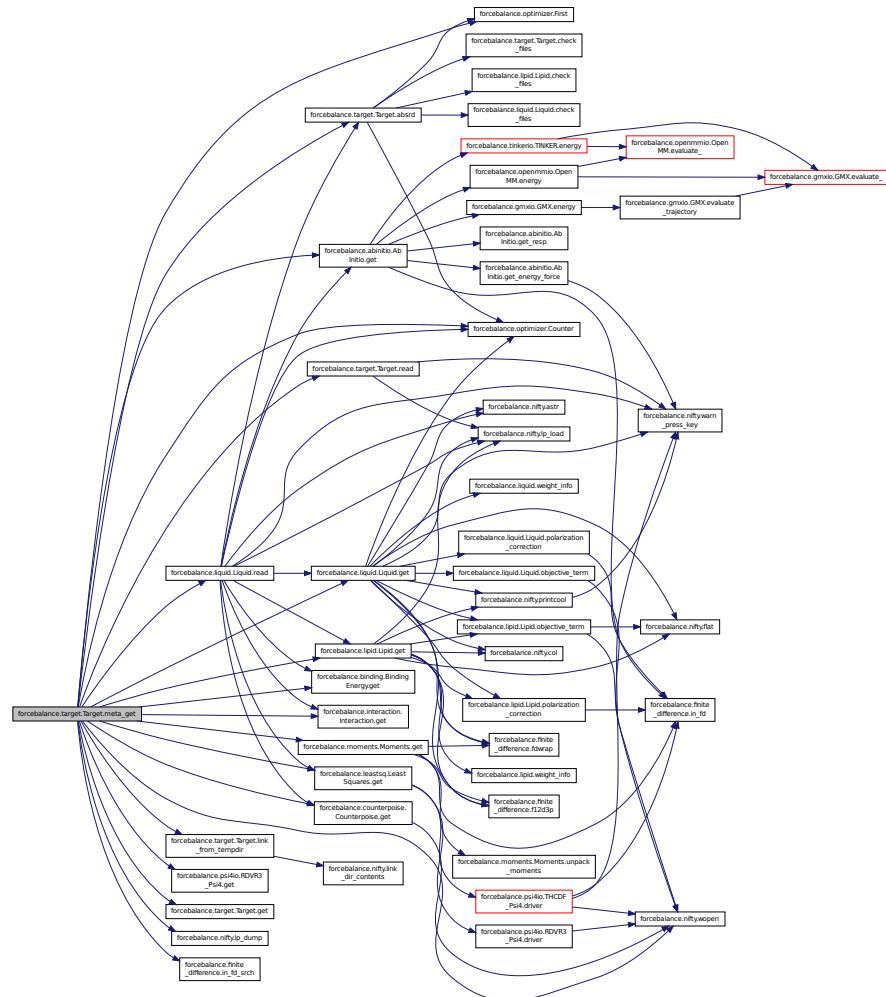


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

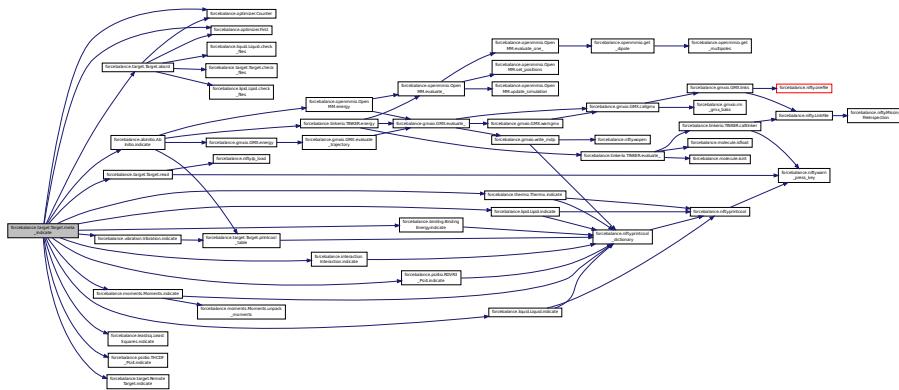
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

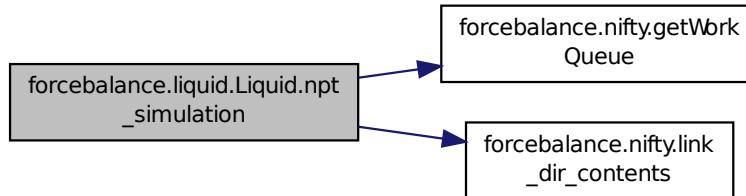
Here is the call graph for this function:



**def forcebalance.liquid.Liquid.npt\_simulation ( self, temperature, pressure, simnum )** Submit a NPT simulation to the Work Queue.

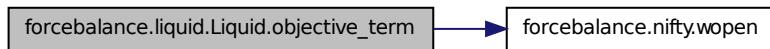
Definition at line 270 of file liquid.py.

Here is the call graph for this function:



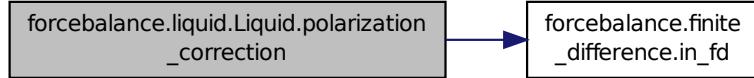
**def forcebalance.liquid.Liquid.objective\_term ( self, points, expname, calc, err, grad, name = "Quantity", SubAverage = False )** Definition at line 330 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.liquid.Liquid.polarization\_correction ( self, mvals )** Definition at line 286 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.liquid.Liquid.prepare\_temp\_directory ( self )** Prepare the temporary directory by copying in important files.

Definition at line 155 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.target.Target.printcool\_table ( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited]** Print target information in an organized table format.

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

The input should be something like:

Parameters

<b>data</b>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<b>headings</b>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<b>banner</b>	Optional heading line, which will be printed at the top in the title.
<b>footnote</b>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

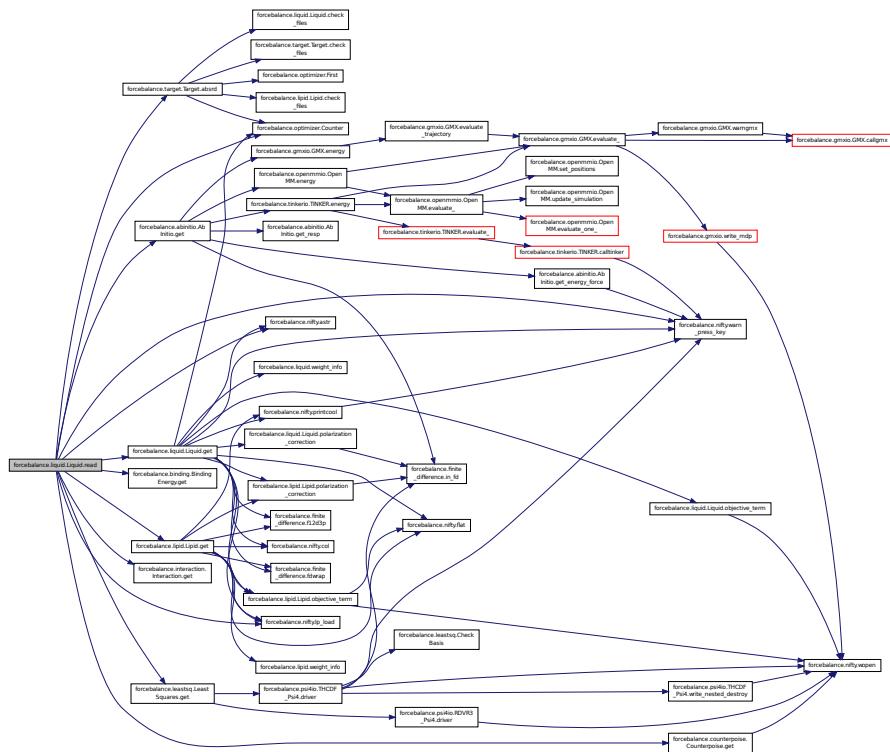
Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.read( self, mvals, AGrad = True, AHess = True ) Read in time series for all previous iterations.
```

Definition at line 447 of file liquid.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.read_0grads ( self ) [inherited] Read a file from the target directory containing names of parameters that don't contribute to the gradient.
```

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.liquid.Liquid.read\_data( self )** Definition at line 162 of file liquid.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

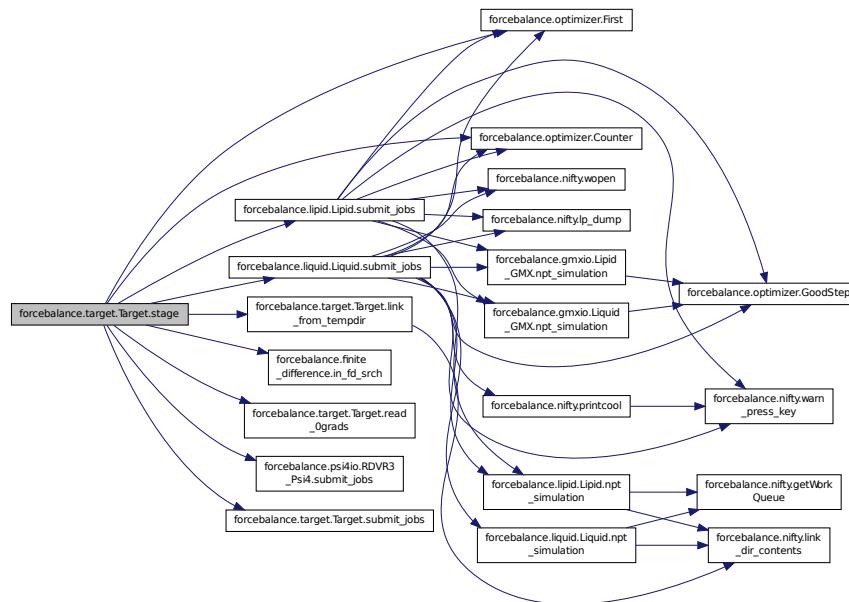
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

The 'get' method should not worry about the directory that it's running in.

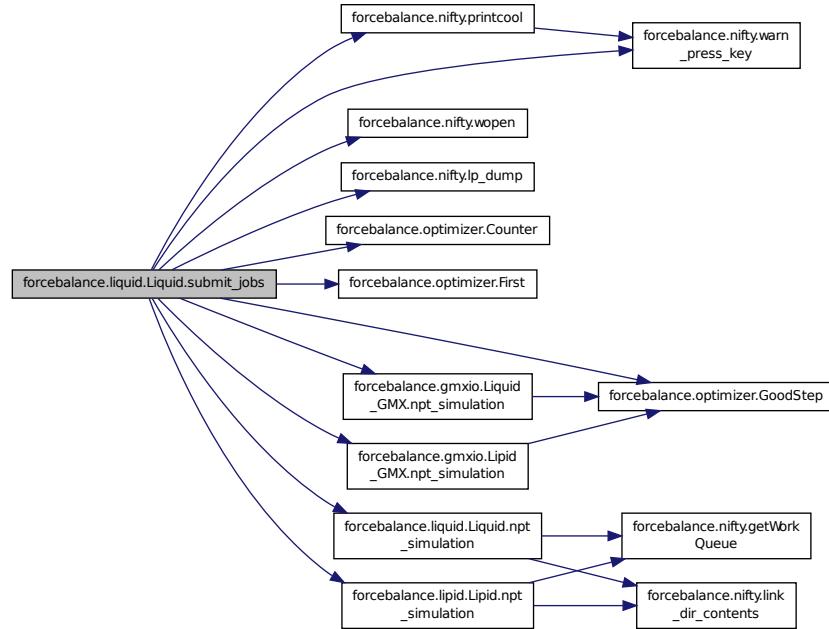
Definition at line 565 of file target.py.

Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.submit_jobs ( self, mvals, AGrad = True, AHess = True ) Definition at line
409 of file liquid.py.
```

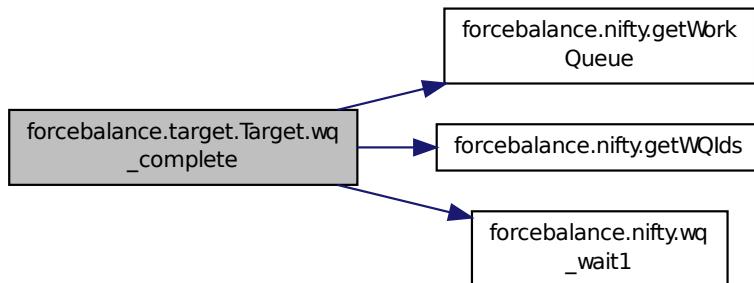
Here is the call graph for this function:



**def forcebalance.target.Target.wq\_complete( self ) [inherited]** This method determines whether the Work Queue tasks for the current target have completed.

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_Ograds( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.34.4 Member Data Documentation

**forcebalance.liquid.Liquid.AllResults** Saved results for all iterations self.SavedMVals = [].  
Definition at line 151 of file liquid.py.

**forcebalance.liquid.Liquid.do\_self\_pol** Definition at line 98 of file liquid.py.

**forcebalance.liquid.Liquid.engname** Definition at line 275 of file liquid.py.

**forcebalance.liquid.Liquid.extra\_output** Definition at line 118 of file liquid.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)  
Definition at line 160 of file target.py.

**forcebalance.liquid.Liquid.gas\_engine** Read the reference data.  
Definition at line 133 of file liquid.py.

**forcebalance.liquid.Liquid.gas\_mol** Definition at line 114 of file liquid.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.  
Definition at line 164 of file target.py.

**forcebalance.liquid.Liquid.Gp** Definition at line 895 of file liquid.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.liquid.Liquid.Labels** Definition at line 243 of file liquid.py.

**forcebalance.liquid.Liquid.last\_traj** Definition at line 116 of file liquid.py.

**forcebalance.liquid.Liquid.liquid\_mol** Definition at line 110 of file liquid.py.

**forcebalance.liquid.Liquid.MBarEnergy** Evaluated energies for all trajectories (i.e.  
all iterations and all temperatures), using all mvals  
Definition at line 148 of file liquid.py.

**forcebalance.liquid.Liquid.Objective** Definition at line 897 of file liquid.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.  
Dictionary of whether to call the derivatives.  
Definition at line 127 of file target.py.

**forcebalance.liquid.Liquid.PhasePoints** Definition at line 239 of file liquid.py.

**forcebalance.liquid.Liquid.Pp** Definition at line 892 of file liquid.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.liquid.Liquid.read\_indicate** Definition at line 135 of file liquid.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.liquid.Liquid.RefData** Definition at line 173 of file liquid.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.liquid.Liquid.SavedTraj** Saved trajectories for all iterations and all temperatures.

Definition at line 146 of file liquid.py.

**forcebalance.target.Target.tempbase [inherited]** Relative directory of target.

Temporary (working) directory; it is temp/(target\_name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir [inherited]** Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.liquid.Liquid.w\_alpha** Definition at line 868 of file liquid.py.

**forcebalance.liquid.Liquid.w\_cp** Definition at line 870 of file liquid.py.

**forcebalance.liquid.Liquid.w\_eps0** Definition at line 871 of file liquid.py.

**forcebalance.liquid.Liquid.w\_hvap** Definition at line 867 of file liquid.py.

**forcebalance.liquid.Liquid.w\_kappa** Definition at line 869 of file liquid.py.

**forcebalance.liquid.Liquid.w\_rho** Density.

Enthalpy of vaporization. Thermal expansion coefficient. Isothermal compressibility. Isobaric heat capacity. Static dielectric constant. Estimation of errors.

Definition at line 866 of file liquid.py.

**forcebalance.liquid.Liquid.Wp** Definition at line 890 of file liquid.py.

**forcebalance.liquid.Liquid.write\_indicate** Definition at line 136 of file liquid.py.

**forcebalance.target.Target.write\_objective [inherited]** Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 162 of file target.py.

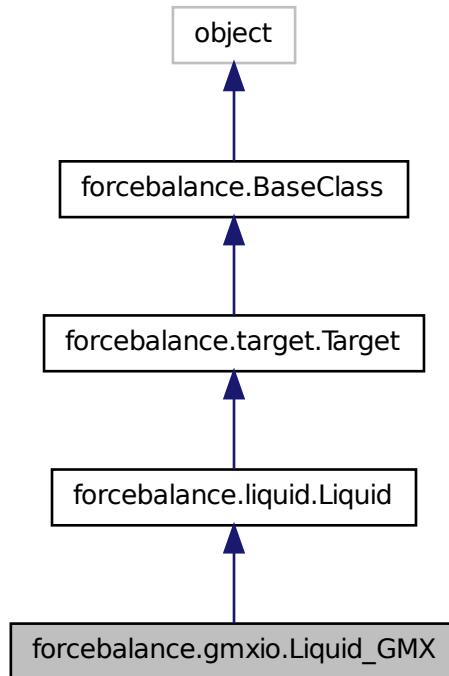
**forcebalance.liquid.Liquid.Xp** Definition at line 888 of file liquid.py.

The documentation for this class was generated from the following file:

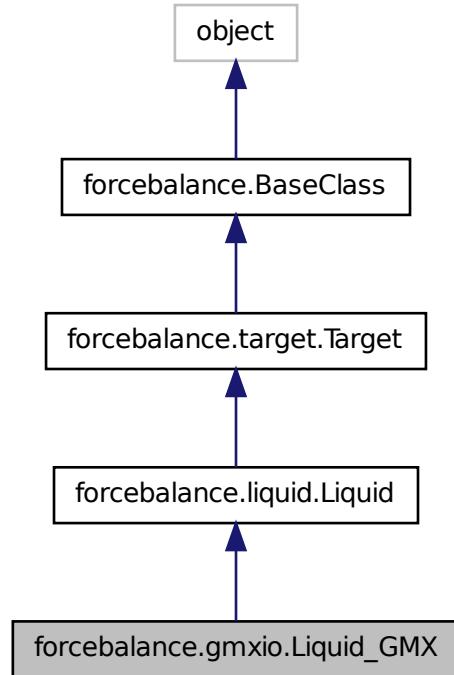
- [liquid.py](#)

## 8.35 forcebalance.gmxio.Liquid\_GMX Class Reference

Inheritance diagram for forcebalance.gmxio.Liquid\_GMX:



Collaboration diagram for forcebalance.gmxio.Liquid\_GMX:



### Public Member Functions

- def `__init__`
- def `npt_simulation`

*Submit a NPT simulation to the Work Queue.*
- def `prepare_temp_directory`

*Prepare the temporary directory by copying in important files.*
- def `read_data`
- def `check_files`
- def `polarization_correction`
- def `indicate`
- def `objective_term`
- def `submit_jobs`
- def `read`

*Read in time series for all previous iterations.*
- def `get`

*Fitting of liquid bulk properties.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_0grads`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*

- def [write\\_Ograds](#)

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*

- def [get\\_G](#)

*Computes the objective function contribution and its gradient.*

- def [get\\_H](#)

*Computes the objective function contribution and its gradient / Hessian.*

- def [link\\_from\\_tempdir](#)

- def [refresh\\_temp\\_directory](#)

*Back up the temporary directory if desired, delete it and then create a new one.*

- def [absrd](#)

*Supply the correct directory specified by user's "read" option.*

- def [maxrd](#)

*Supply the latest existing temp-directory containing valid data.*

- def [meta\\_indicate](#)

*Wrap around the indicate function, so it can print to screen and also to a file.*

- def [meta\\_get](#)

*Wrapper around the get function.*

- def [stage](#)

*Stages the directory for the target, and then launches Work Queue processes if any.*

- def [wq\\_complete](#)

*This method determines whether the Work Queue tasks for the current target have completed.*

- def [printcool\\_table](#)

*Print target information in an organized table format.*

- def [\\_\\_setattr\\_\\_](#)

- def [set\\_option](#)

## Public Attributes

- [engine\\_](#)
- [engname](#)
- [nptpx](#)
- [nptfiles](#)
- [gas\\_engine\\_args](#)
- [scripts](#)
- [extra\\_output](#)
- [LfDict](#)
- [LfDict\\_New](#)
- [last\\_traj](#)
- [do\\_self\\_pol](#)
- [liquid\\_mol](#)
- [gas\\_mol](#)
- [gas\\_engine](#)

*Read the reference data.*

- [read\\_indicate](#)

- [write\\_indicate](#)

- [SavedTraj](#)

*Saved trajectories for all iterations and all temperatures.*

- [MBarEnergy](#)

*Evaluates energies for all trajectories (i.e.*

- **AllResults**

*Saved results for all iterations self.SavedMVals = [].*

- **RefData**

- **PhasePoints**

- **Labels**

- **w\_rho**

*Density.*

- **w\_hvap**

- **w\_alpha**

- **w\_kappa**

- **w\_cp**

- **w\_eps0**

- **Xp**

- **Wp**

- **Pp**

- **Gp**

- **Objective**

- **rd**

*Root directory of the whole project.*

- **pgrad**

*Iteration where we turn on zero-gradient skipping.*

- **tempbase**

*Relative directory of target.*

- **tempdir**

- **rundir**

*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*

- **FF**

*Need the forcefield (here for now)*

- **xct**

*Counts how often the objective function was computed.*

- **gct**

*Counts how often the gradient was computed.*

- **hct**

*Counts how often the Hessian was computed.*

- **read\_objective**

*Whether to read objective.p from file when restarting an aborted run.*

- **write\_objective**

*Whether to write objective.p at every iteration (true for all but remote.)*

- **verbose\_options**

- **PrintOptionDict**

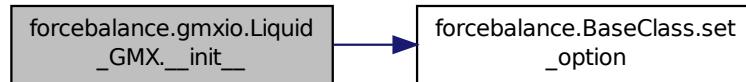
### 8.35.1 Detailed Description

Definition at line 1340 of file gmxio.py.

### 8.35.2 Constructor & Destructor Documentation

```
def forcebalance.gmxio.Liquid_GMX.__init__ ( self, options, tgt_opts, forcefield ) Definition at line 1341 of file gmxio.py.
```

Here is the call graph for this function:



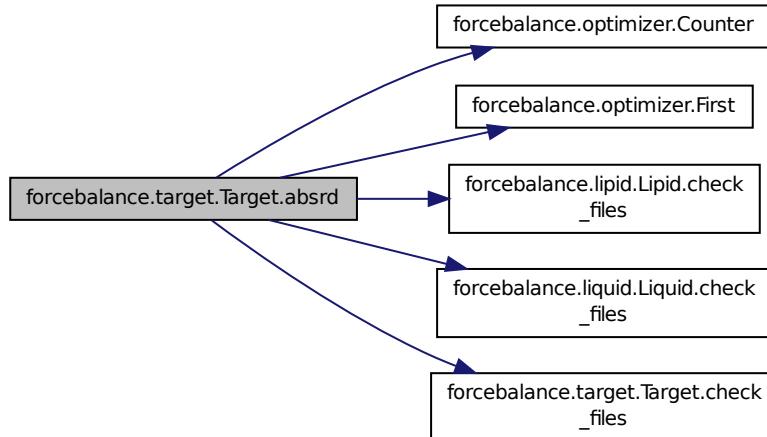
### 8.35.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.check_files ( self, there ) [inherited] Definition at line 253 of file liquid.py.
```

```
def forcebalance.liquid.Liquid.get ( self, mvals, AGrad = True, AHess = True ) [inherited] Fitting of liquid bulk properties.
```

This is the current major direction of development for ForceBalance. Basically, fitting the QM energies / forces alone does not always give us the best simulation behavior. In many cases it makes more sense to try and reproduce some experimentally known data as well.

In order to reproduce experimentally known data, we need to run a simulation and compare the simulation result to experiment. The main challenge here is that the simulations are computationally intensive (i.e. they require energy and force evaluations), and furthermore the results are noisy. We need to run the simulations automatically and remotely (i.e. on clusters) and a good way to calculate the derivatives of the simulation results with respect to the parameter values.

This function contains some experimentally known values of the density and enthalpy of vaporization (H<sub>vap</sub>) of liquid water. It launches the density and H<sub>vap</sub> calculations on the cluster, and gathers the results / derivatives. The actual calculation of results / derivatives is done in a separate file.

After the results come back, they are gathered together to form an objective function.

#### Parameters

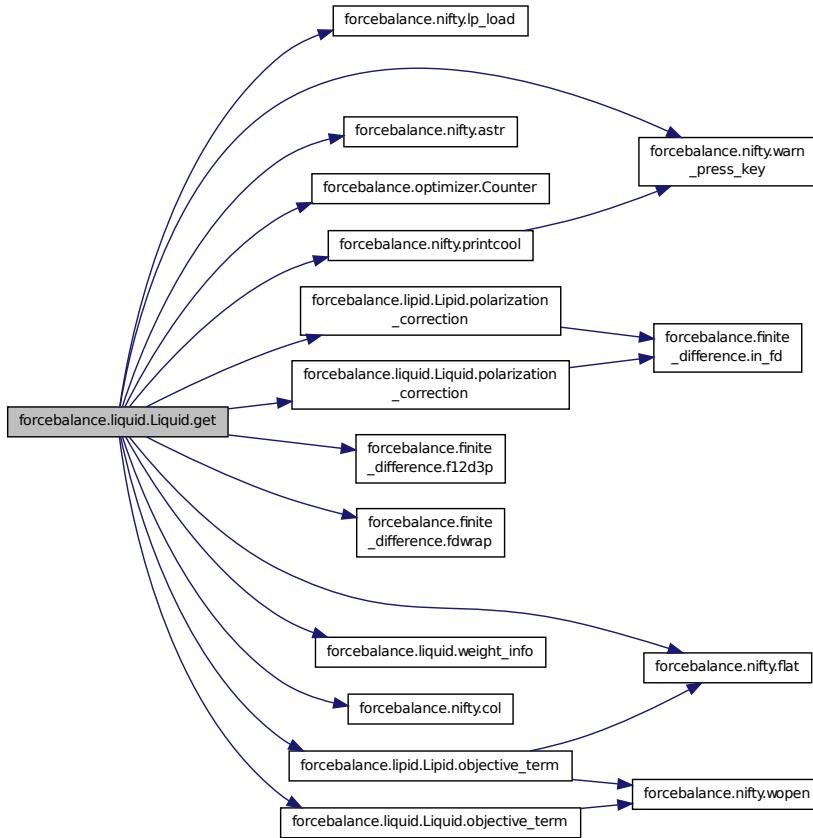
in	<i>mvals</i>	Mathematical parameter values
in	<i>AGrad</i>	Switch to turn on analytic gradient
in	<i>AHess</i>	Switch to turn on analytic Hessian

#### Returns

Answer Contribution to the objective function Fill in the weight matrix with MBAR weights where MBAR was run, and equal weights otherwise.

Definition at line 553 of file liquid.py.

Here is the call graph for this function:



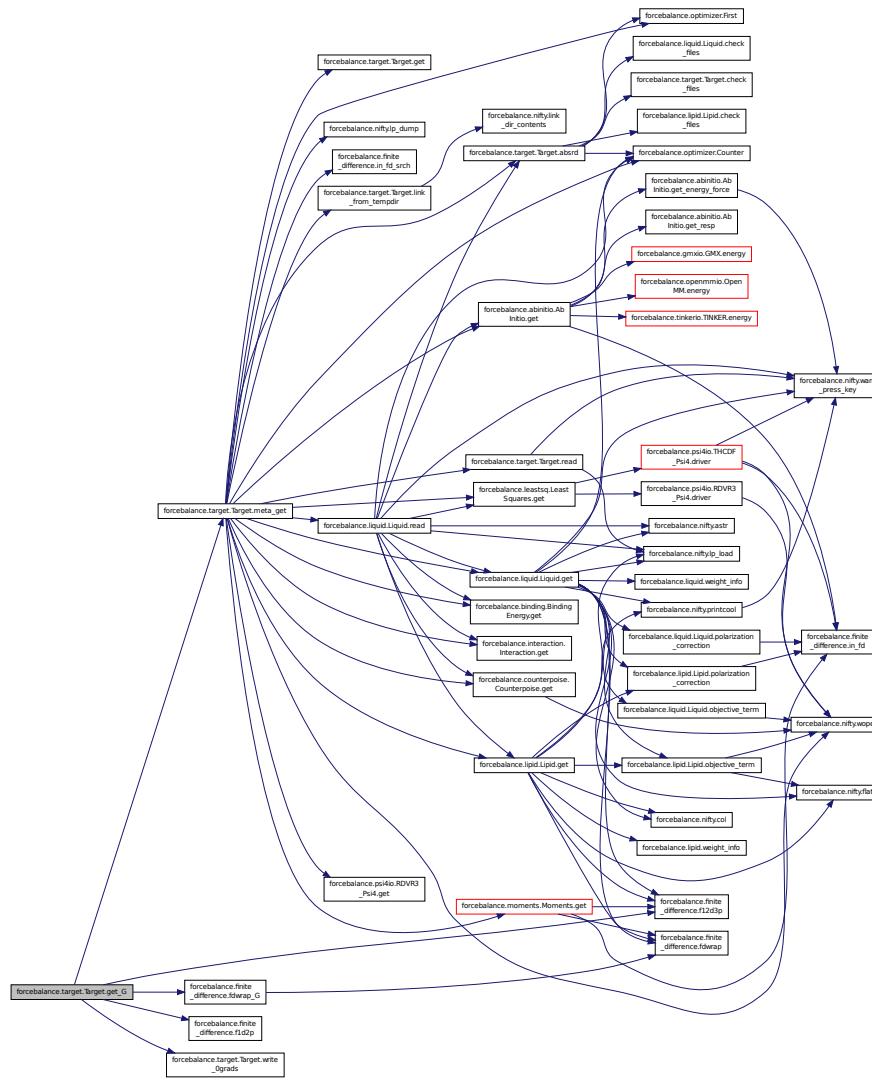
**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



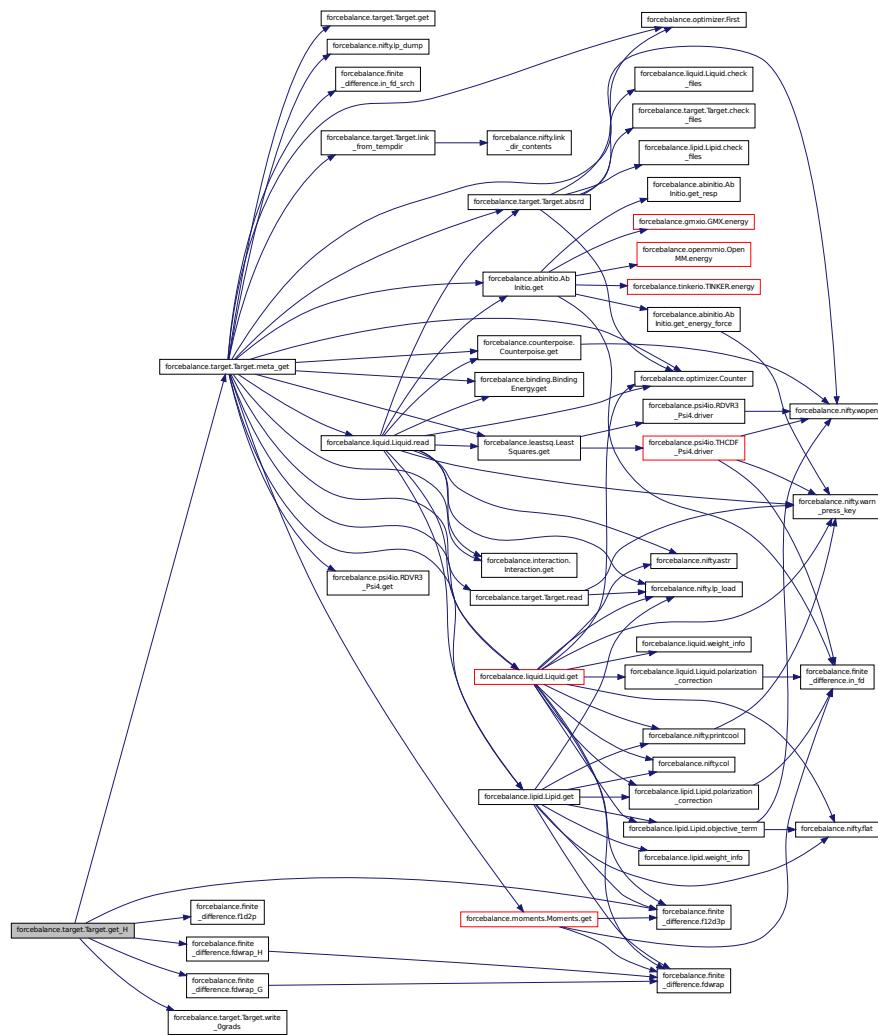
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

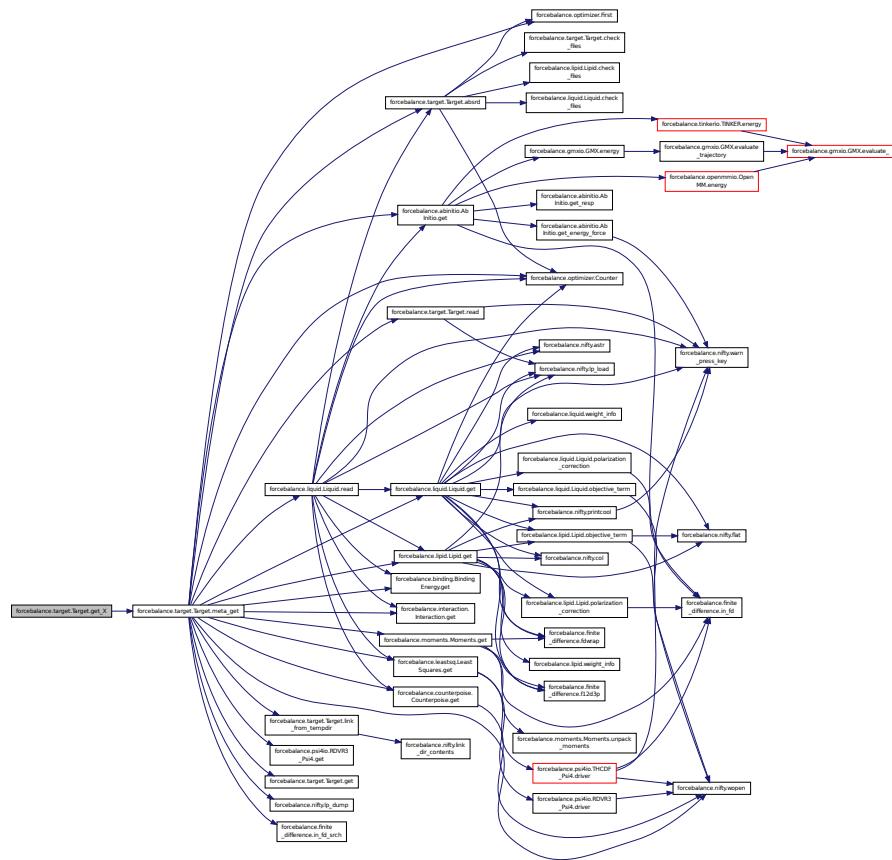
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

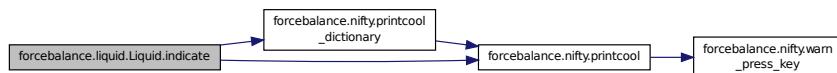
Definition at line 184 of file target.py.

Here is the call graph for this function:



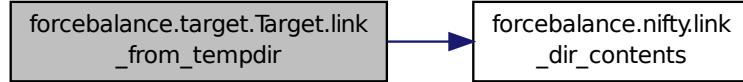
**def forcebalance.liquid.Liquid.indicate ( self ) [inherited]** Definition at line 304 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

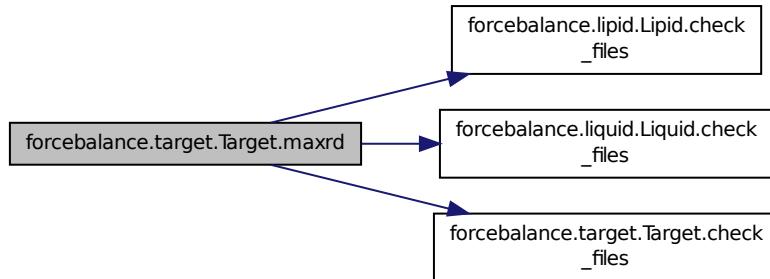
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

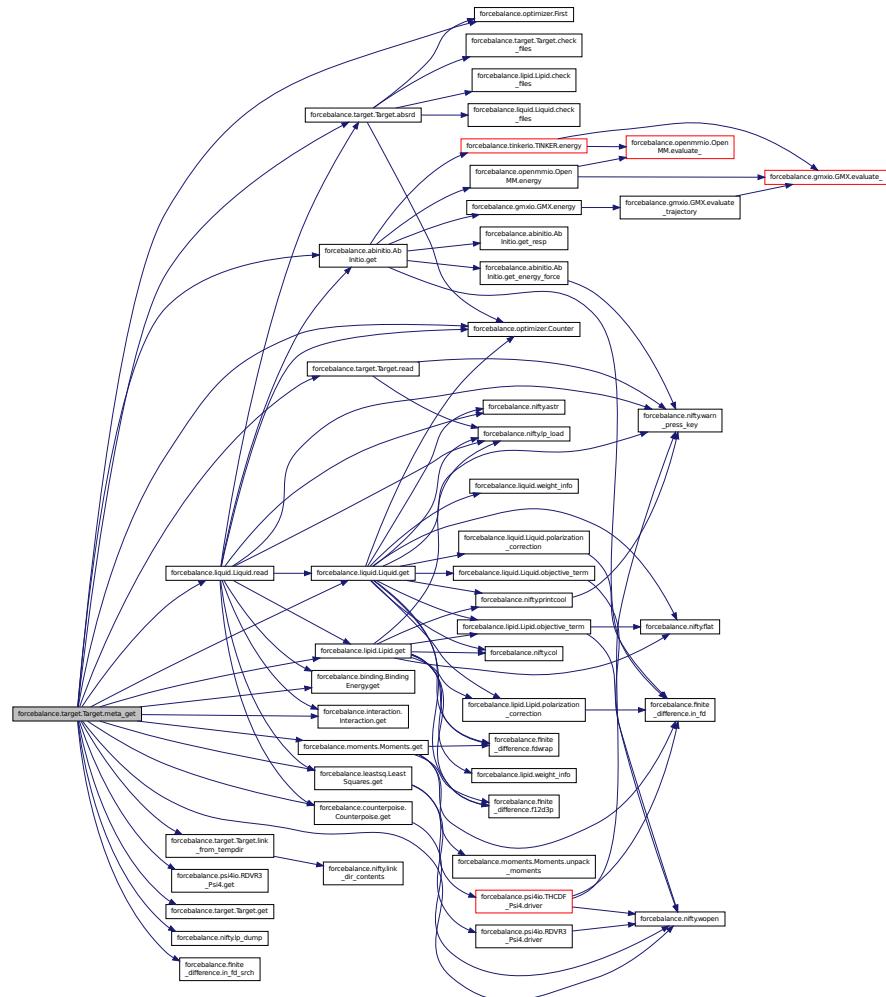


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

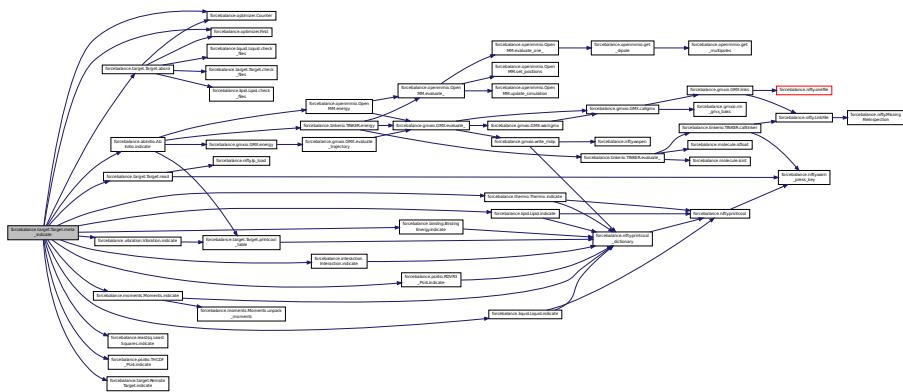
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

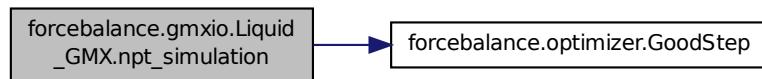
Here is the call graph for this function:



```
def forcebalance.gmxio.Liquid_GMX.npt_simulation( self, temperature, pressure, simnum ) Submit a NPT simulation to the Work Queue.
```

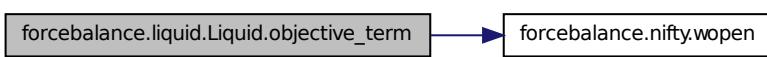
Definition at line 1383 of file gmxio.py.

Here is the call graph for this function:



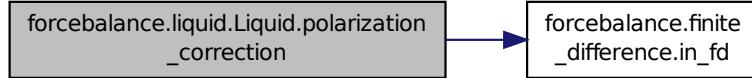
```
def forcebalance.liquid.Liquid.objective.term ( self, points, expname, calc, err, grad, name = "Quantity", SubAverage = False ) [inherited] Definition at line 330 of file liquid.py.
```

Here is the call graph for this function:



**def** forcebalance.liquid.Liquid.polarization\_correction ( *self*, *mvals* ) [inherited] Definition at line 286 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.liquid.Liquid.prepare\_temp\_directory ( self ) [inherited]** Prepare the temporary directory by copying in important files.

Definition at line 155 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.target.Target.printcool\_table ( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited]** Print target information in an organized table format.

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

The input should be something like:

#### Parameters

<b>data</b>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<b>headings</b>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<b>banner</b>	Optional heading line, which will be printed at the top in the title.
<b>footnote</b>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

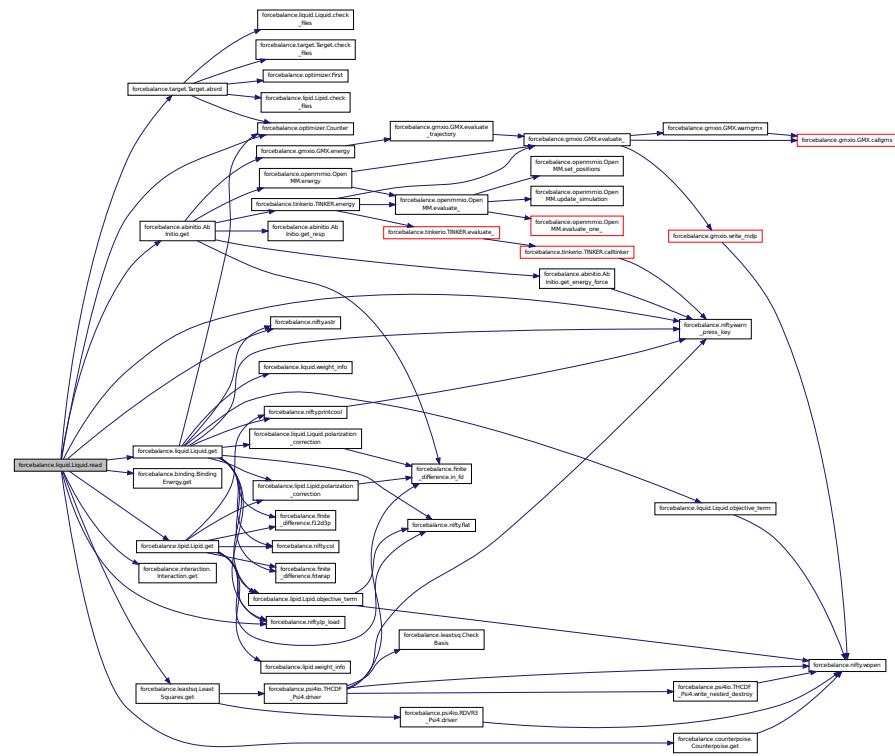
Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.read( self, mvals, AGrad = True, AHess = True ) [inherited] Read in time series for all previous iterations.
```

Definition at line 447 of file liquid.py.

Here is the call graph for this function:



Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

```
def forcebalance.liquid.Liquid.read_data( self ) [inherited] Definition at line 162 of file liquid.py.
```

```
def forcebalance.target.Target.refresh_temp_directory( self ) [inherited] Back up the temporary directory if desired, delete it and then create a new one.
```

Definition at line 321 of file target.py.

```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

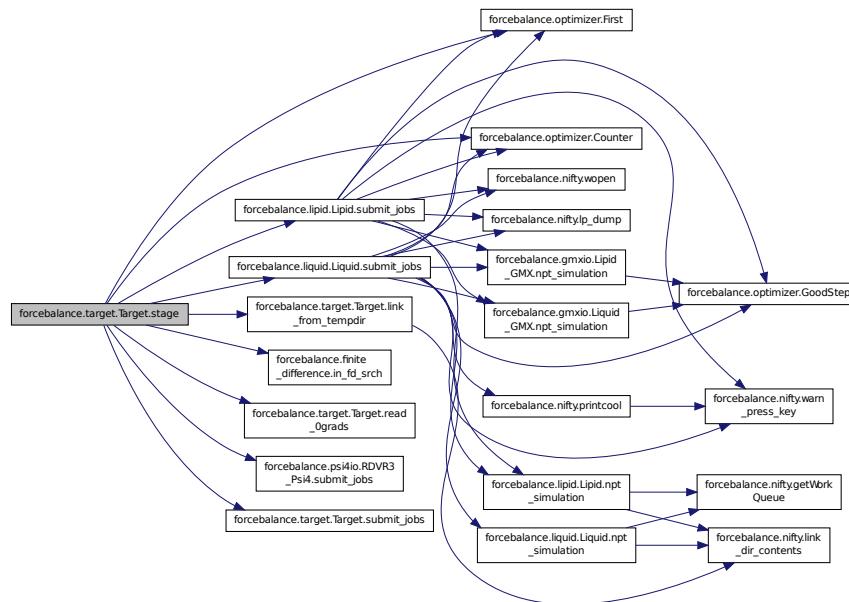
```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]
```

Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

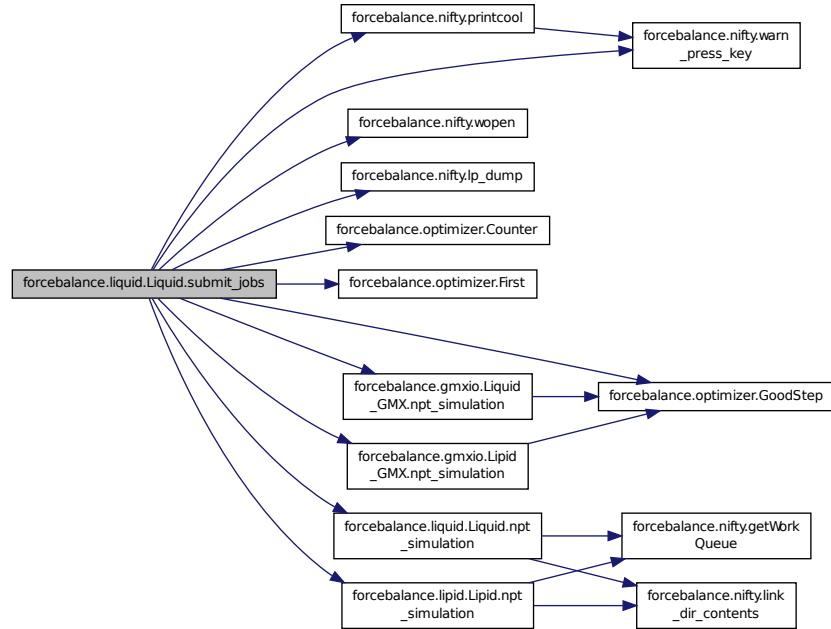
Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.submit_jobs ( self, mvals, AGrad = True, AHess = True ) [inherited]
```

Definition at line 409 of file liquid.py.

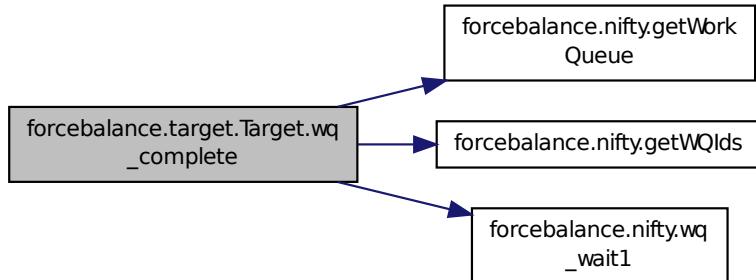
Here is the call graph for this function:



**def forcebalance.target.Target.wq\_complete( self ) [inherited]** This method determines whether the Work Queue tasks for the current target have completed.

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_Ograds( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.35.4 Member Data Documentation

**forcebalance.liquid.Liquid.AllResults [inherited]** Saved results for all iterations self.SavedMVals = [].  
Definition at line 151 of file liquid.py.

**forcebalance.liquid.Liquid.do\_self\_pol [inherited]** Definition at line 98 of file liquid.py.

**forcebalance.gmxio.Liquid\_GMX.engine\_** Definition at line 1353 of file gmxio.py.

**forcebalance.gmxio.Liquid\_GMX.engname** Definition at line 1355 of file gmxio.py.

**forcebalance.gmxio.Liquid\_GMX.extra\_output** Definition at line 1373 of file gmxio.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)  
Definition at line 160 of file target.py.

**forcebalance.liquid.Liquid.gas\_engine [inherited]** Read the reference data.  
Definition at line 133 of file liquid.py.

**forcebalance.gmxio.Liquid\_GMX.gas\_engine\_args** Definition at line 1362 of file gmxio.py.

**forcebalance.liquid.Liquid.gas\_mol [inherited]** Definition at line 114 of file liquid.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.  
Definition at line 164 of file target.py.

**forcebalance.liquid.Liquid.Gp [inherited]** Definition at line 895 of file liquid.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.liquid.Liquid.Labels [inherited]** Definition at line 243 of file liquid.py.

**forcebalance.gmxio.Liquid\_GMX.last\_traj** Definition at line 1394 of file gmxio.py.

**forcebalance.gmxio.Liquid\_GMX.LfDict** Definition at line 1378 of file gmxio.py.

**forcebalance.gmxio.Liquid\_GMX.LfDict\_New** Definition at line 1379 of file gmxio.py.

**forcebalance.liquid.Liquid.liquid\_mol [inherited]** Definition at line 110 of file liquid.py.

**forcebalance.liquid.Liquid.MBarEnergy [inherited]** Evaluated energies for all trajectories (i.e.  
all iterations and all temperatures), using all mvals  
Definition at line 148 of file liquid.py.

**forcebalance.gmxio.Liquid\_GMX.nptfiles** Definition at line 1359 of file gmxio.py.

**forcebalance.gmxio.Liquid\_GMX.nptpx** Definition at line 1357 of file gmxio.py.

**forcebalance.liquid.Liquid.Objective** [inherited] Definition at line 897 of file liquid.py.

**forcebalance.target.Target.pgrad** [inherited] Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.liquid.Liquid.PhasePoints** [inherited] Definition at line 239 of file liquid.py.

**forcebalance.liquid.Liquid.Pp** [inherited] Definition at line 892 of file liquid.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd** [inherited] Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.liquid.Liquid.read\_indicate** [inherited] Definition at line 135 of file liquid.py.

**forcebalance.target.Target.read\_objective** [inherited] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.liquid.Liquid.RefData** [inherited] Definition at line 173 of file liquid.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.liquid.Liquid.SavedTraj** [inherited] Saved trajectories for all iterations and all temperatures.

Definition at line 146 of file liquid.py.

**forcebalance.gmxio.Liquid\_GMX.scripts** Definition at line 1365 of file gmxio.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.liquid.Liquid.w\_alpha** [inherited] Definition at line 868 of file liquid.py.

**forcebalance.liquid.Liquid.w\_cp** [inherited] Definition at line 870 of file liquid.py.

**forcebalance.liquid.Liquid.w\_eps0** [inherited] Definition at line 871 of file liquid.py.

**forcebalance.liquid.Liquid.w\_hvap** [inherited] Definition at line 867 of file liquid.py.

**forcebalance.liquid.Liquid.w\_kappa** [inherited] Definition at line 869 of file liquid.py.

**forcebalance.liquid.Liquid.w\_rho** [inherited] Density.

Enthalpy of vaporization. Thermal expansion coefficient. Isothermal compressibility. Isobaric heat capacity. Static dielectric constant. Estimation of errors.

Definition at line 866 of file liquid.py.

**forcebalance.liquid.Liquid.Wp** [inherited] Definition at line 890 of file liquid.py.

**forcebalance.liquid.Liquid.write\_indicate** [inherited] Definition at line 136 of file liquid.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

**forcebalance.liquid.Liquid.Xp** [inherited] Definition at line 888 of file liquid.py.

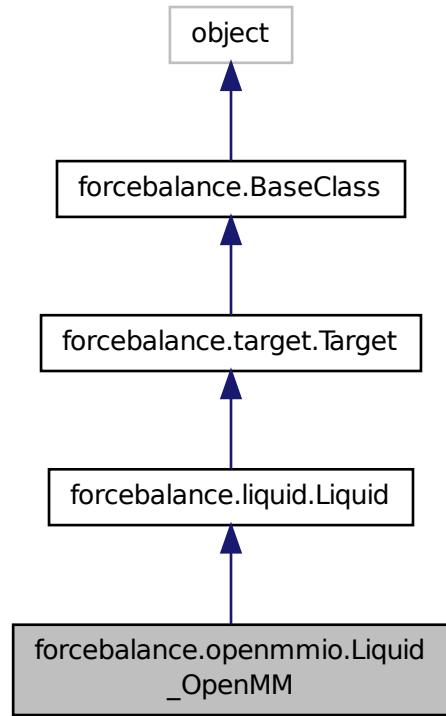
The documentation for this class was generated from the following file:

- [gmxio.py](#)

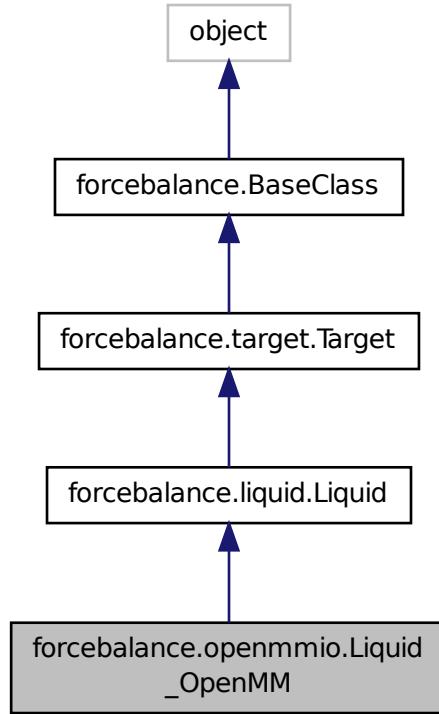
## 8.36 forcebalance.openmmio.Liquid\_OpenMM Class Reference

Condensed phase property matching using [OpenMM](#).

Inheritance diagram for forcebalance.openmmio.Liquid\_OpenMM:



Collaboration diagram for forcebalance.openmmio.Liquid\_OpenMM:



### Public Member Functions

- def `__init__`
- def `prepare_temp_directory`

*Prepare the temporary directory by copying in important files.*
- def `read_data`
- def `check_files`
- def `npt_simulation`

*Submit a NPT simulation to the Work Queue.*
- def `polarization_correction`
- def `indicate`
- def `objective_term`
- def `submit_jobs`
- def `read`

*Read in time series for all previous iterations.*
- def `get`

*Fitting of liquid bulk properties.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*

- def [read\\_0grads](#)  
*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def [write\\_0grads](#)  
*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def [get\\_G](#)  
*Computes the objective function contribution and its gradient.*
- def [get\\_H](#)  
*Computes the objective function contribution and its gradient / Hessian.*
- def [link\\_from\\_tempdir](#)
- def [refresh\\_temp\\_directory](#)  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def [absrd](#)  
*Supply the correct directory specified by user's "read" option.*
- def [maxrd](#)  
*Supply the latest existing temp-directory containing valid data.*
- def [meta\\_indicate](#)  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def [meta\\_get](#)  
*Wrapper around the get function.*
- def [stage](#)  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def [wq\\_complete](#)  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def [printcool\\_table](#)  
*Print target information in an organized table format.*
- def [\\_\\_setattr\\_\\_](#)
- def [set\\_option](#)

## Public Attributes

- [engine\\_](#)
- [engname](#)
- [nptpx](#)
- [nptfiles](#)
- [gas\\_engine\\_args](#)
- [scripts](#)
- [extra\\_output](#)
- [do\\_self\\_pol](#)
- [liquid\\_mol](#)
- [gas\\_mol](#)
- [last\\_traj](#)
- [gas\\_engine](#)  
*Read the reference data.*
- [read\\_indicate](#)
- [write\\_indicate](#)
- [SavedTraj](#)  
*Saved trajectories for all iterations and all temperatures.*
- [MBarEnergy](#)

- **AllResults**  
*Evaluates energies for all trajectories (i.e. Saved results for all iterations self.SavedMVals = []).*
- **RefData**
- **PhasePoints**
- **Labels**
- **w\_rho**  
*Density.*
  - **w\_hvap**
  - **w\_alpha**
  - **w\_kappa**
  - **w\_cp**
  - **w\_eps0**
  - **Xp**
  - **Wp**
  - **Pp**
  - **Gp**
  - **Objective**
  - **rd**
- **pgrad**  
*Root directory of the whole project.*
- **tempbase**  
*Iteration where we turn on zero-gradient skipping.*
- **tempdir**  
*Relative directory of target.*
- **rundir**  
`self.tempdir = os.path.join('temp',self.name)` *The directory in which the simulation is running - this can be updated.*
- **FF**  
*Need the forcefield (here for now)*
- **xct**  
*Counts how often the objective function was computed.*
- **gct**  
*Counts how often the gradient was computed.*
- **hct**  
*Counts how often the Hessian was computed.*
- **read\_objective**  
*Whether to read objective.p from file when restarting an aborted run.*
- **write\_objective**  
*Whether to write objective.p at every iteration (true for all but remote.)*
- **verbose\_options**
- **PrintOptionDict**

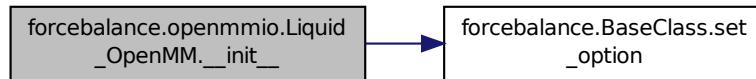
### 8.36.1 Detailed Description

Condensed phase property matching using OpenMM.  
 Definition at line 1128 of file openmmio.py.

### 8.36.2 Constructor & Destructor Documentation

```
def forcebalance.openmmio.Liquid_OpenMM.__init__ ( self, options, tgt_opts, forcefield ) Definition at line  
1129 of file openmmio.py.
```

Here is the call graph for this function:



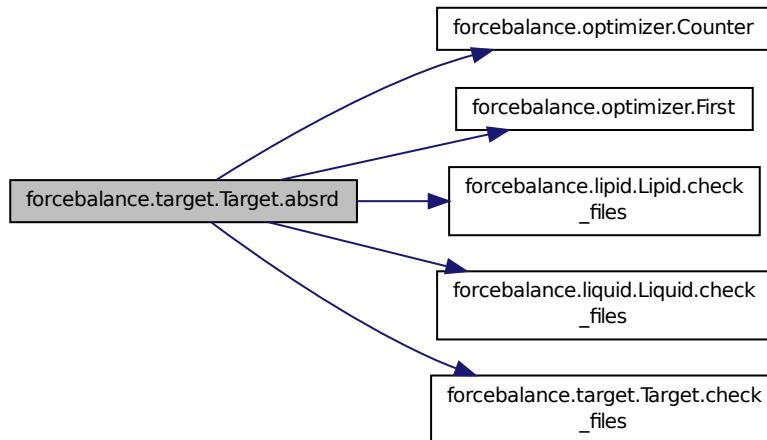
### 8.36.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory  
specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.check_files ( self, there ) [inherited] Definition at line 253 of file liquid.py.
```

```
def forcebalance.liquid.Liquid.get ( self, mvals, AGrad = True, AHess = True ) [inherited] Fitting  
of liquid bulk properties.
```

This is the current major direction of development for ForceBalance. Basically, fitting the QM energies / forces alone does not always give us the best simulation behavior. In many cases it makes more sense to try and reproduce some experimentally known data as well.

In order to reproduce experimentally known data, we need to run a simulation and compare the simulation result to experiment. The main challenge here is that the simulations are computationally intensive (i.e. they require energy and force evaluations), and furthermore the results are noisy. We need to run the simulations automatically and remotely (i.e. on clusters) and a good way to calculate the derivatives of the simulation results with respect to the parameter values.

This function contains some experimentally known values of the density and enthalpy of vaporization (H<sub>vap</sub>) of liquid water. It launches the density and H<sub>vap</sub> calculations on the cluster, and gathers the results / derivatives. The actual calculation of results / derivatives is done in a separate file.

After the results come back, they are gathered together to form an objective function.

#### Parameters

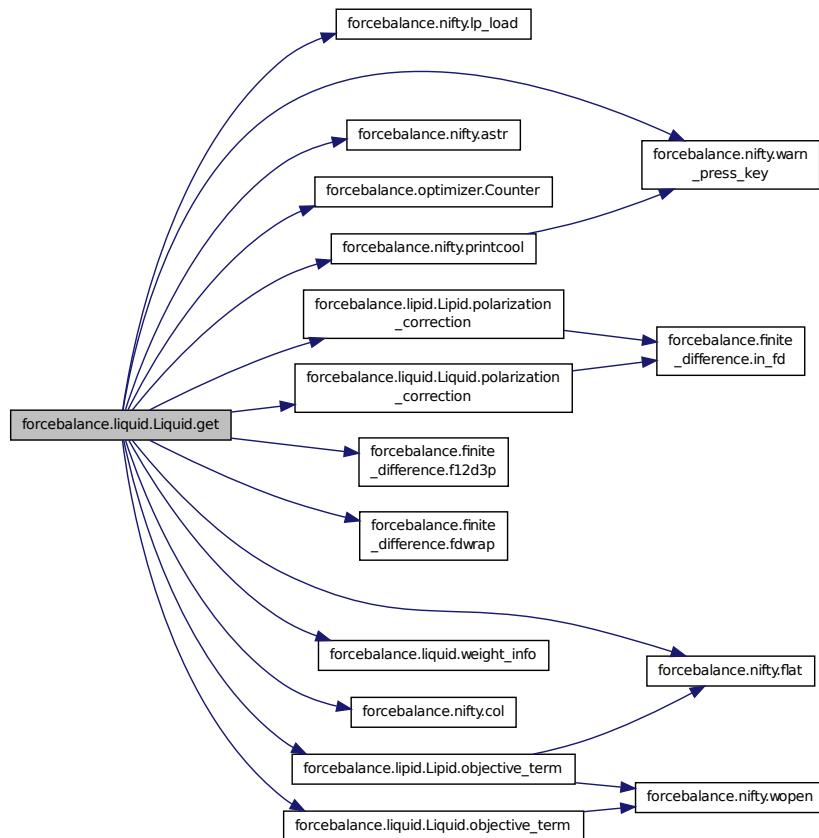
in	<i>mvals</i>	Mathematical parameter values
in	<i>AGrad</i>	Switch to turn on analytic gradient
in	<i>AHess</i>	Switch to turn on analytic Hessian

#### Returns

Answer Contribution to the objective function Fill in the weight matrix with MBAR weights where MBAR was run, and equal weights otherwise.

Definition at line 553 of file liquid.py.

Here is the call graph for this function:



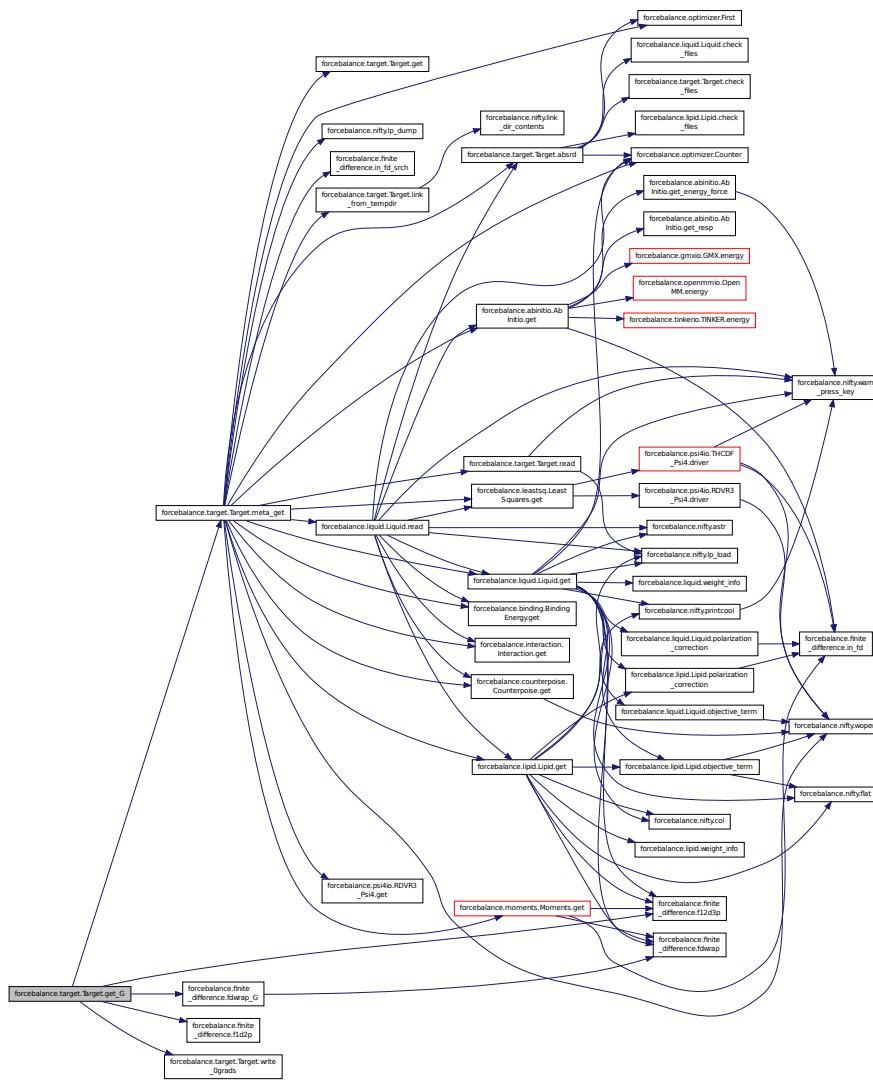
```
def forcebalance.target.Target.get_G ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.get_H ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient / Hessian.
```

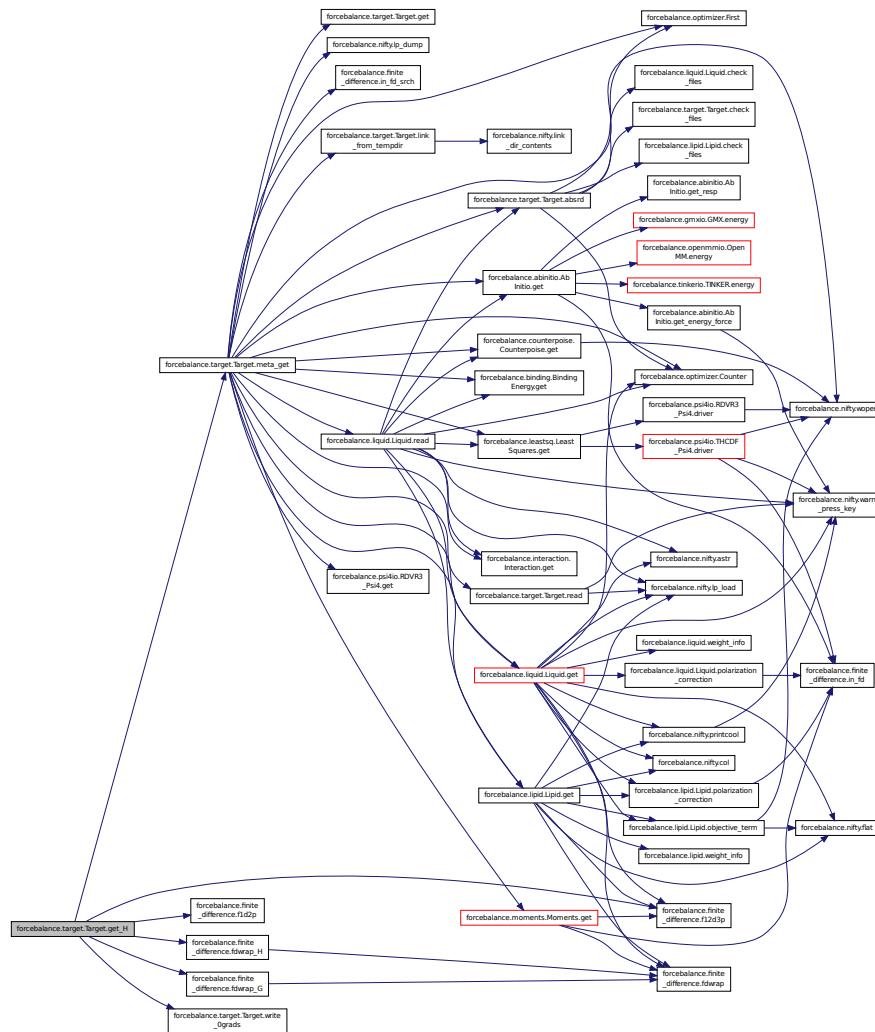
First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through

the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

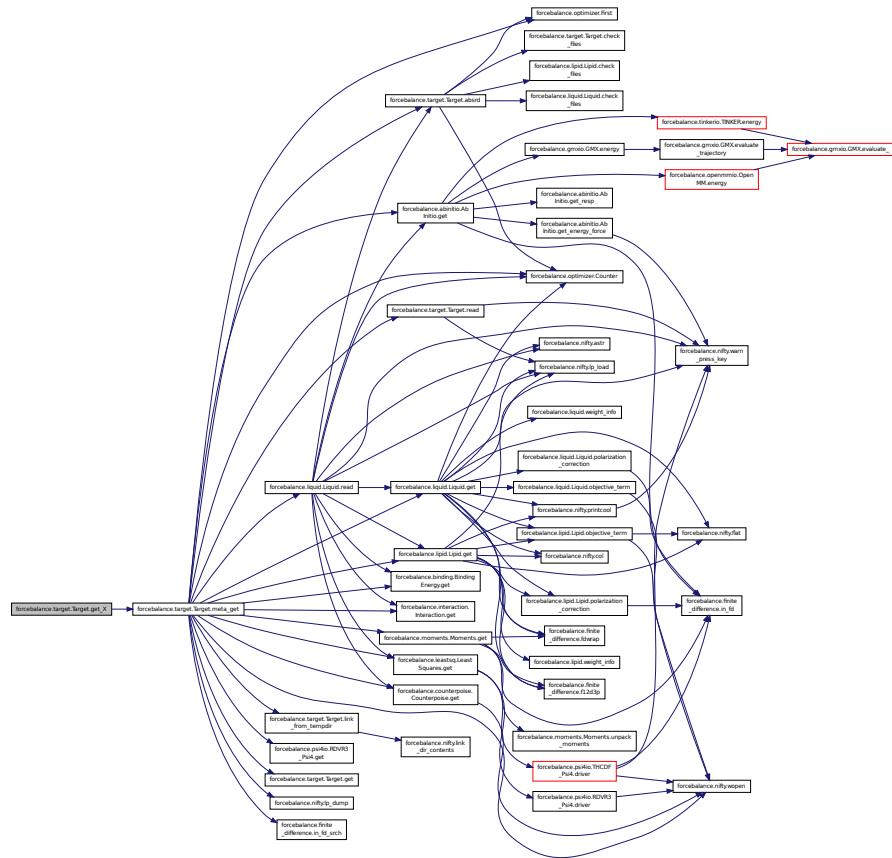
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

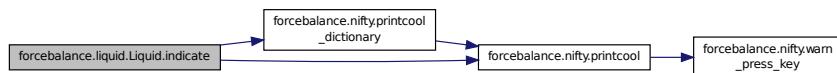
Definition at line 184 of file target.py.

Here is the call graph for this function:



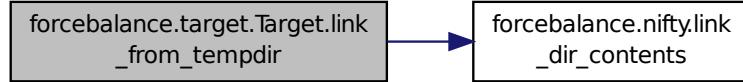
**def forcebalance.liquid.Liquid.indicate ( self ) [inherited]** Definition at line 304 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

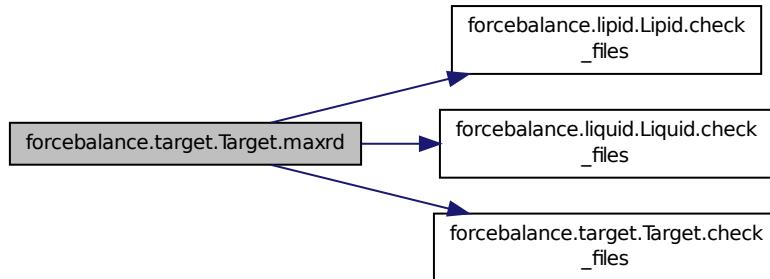
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

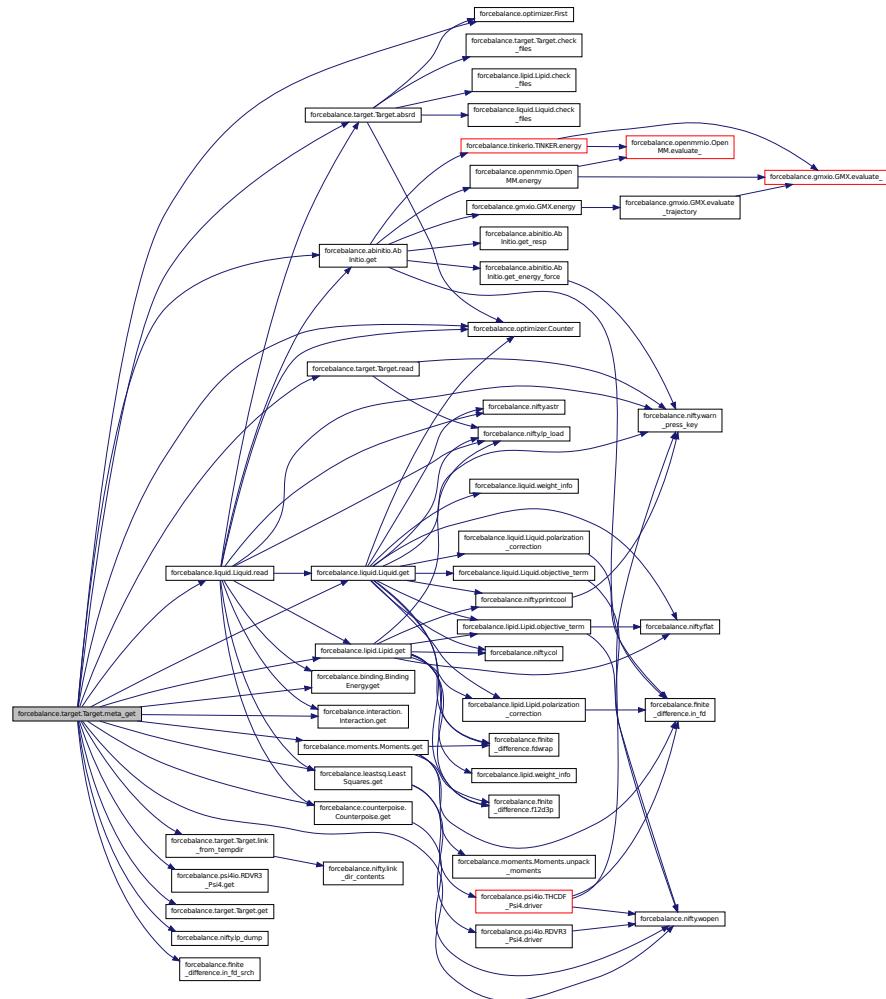


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

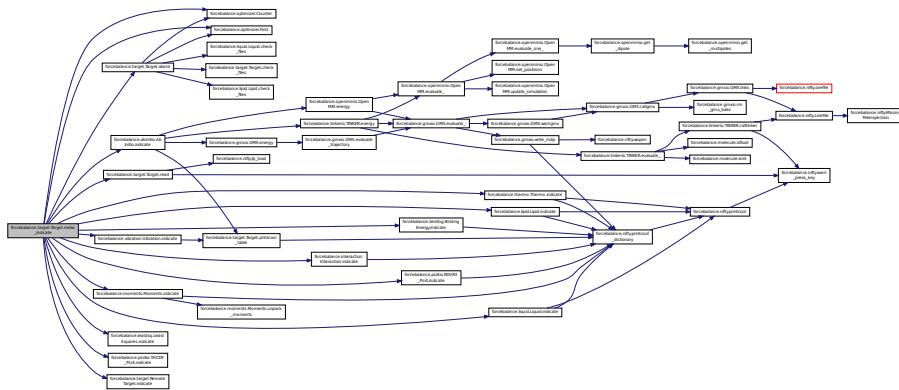
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:

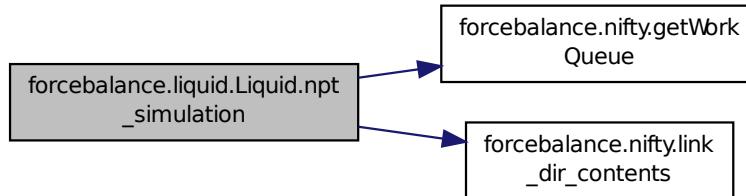


```
def forcebalance.liquid.Liquid.npt_simulation ( self, temperature, pressure, simnum ) [inherited]
```

Submit a NPT simulation to the Work Queue.

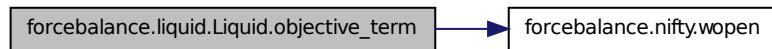
Definition at line 270 of file liquid.py.

Here is the call graph for this function:



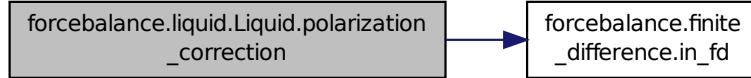
```
def forcebalance.liquid.Liquid.objective_term ( self, points, expname, calc, err, grad, name = "Quantity", SubAverage = False ) [inherited] Definition at line 330 of file liquid.py.
```

Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.polarization_correction ( self, mvals ) [inherited] Definition at line 286 of file liquid.py.
```

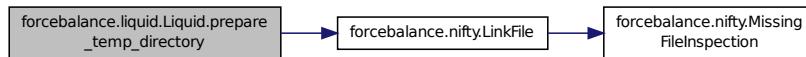
Here is the call graph for this function:



**def forcebalance.liquid.Liquid.prepare\_temp\_directory ( self ) [inherited]** Prepare the temporary directory by copying in important files.

Definition at line 155 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.target.Target.printcool\_table ( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited]** Print target information in an organized table format.

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

The input should be something like:

#### Parameters

<b>data</b>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<b>headings</b>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<b>banner</b>	Optional heading line, which will be printed at the top in the title.
<b>footnote</b>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

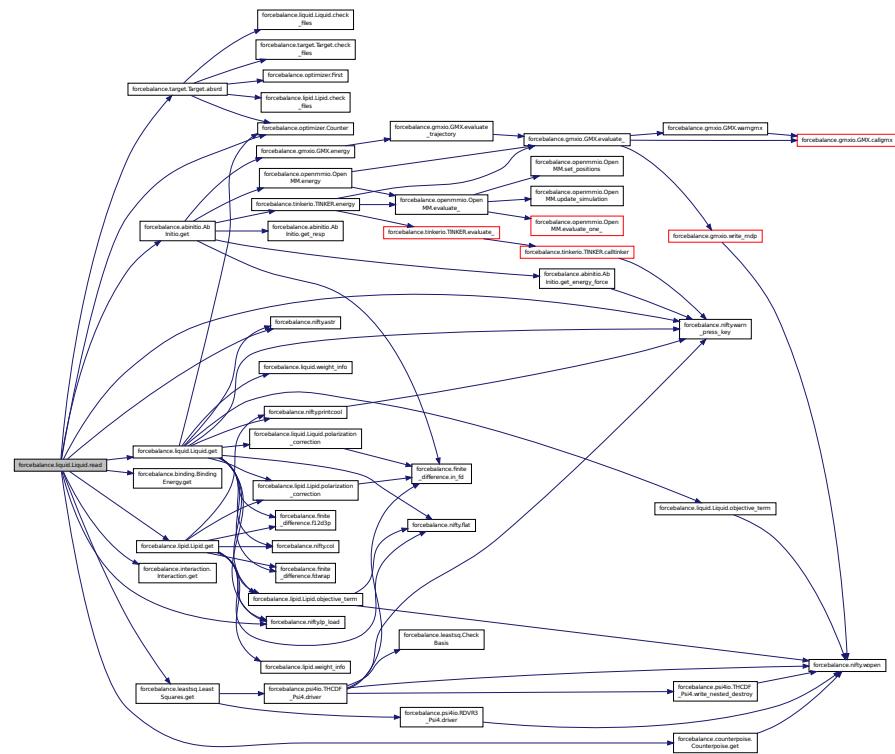
Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.read( self, mvals, AGrad = True, AHess = True ) [inherited] Read in time series for all previous iterations.
```

Definition at line 447 of file liquid.py.

Here is the call graph for this function:



Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

```
def forcebalance.liquid.Liquid.read_data( self ) [inherited] Definition at line 162 of file liquid.py.
```

```
def forcebalance.target.Target.refresh_temp_directory( self ) [inherited] Back up the temporary directory if desired, delete it and then create a new one.
```

Definition at line 321 of file target.py.

```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

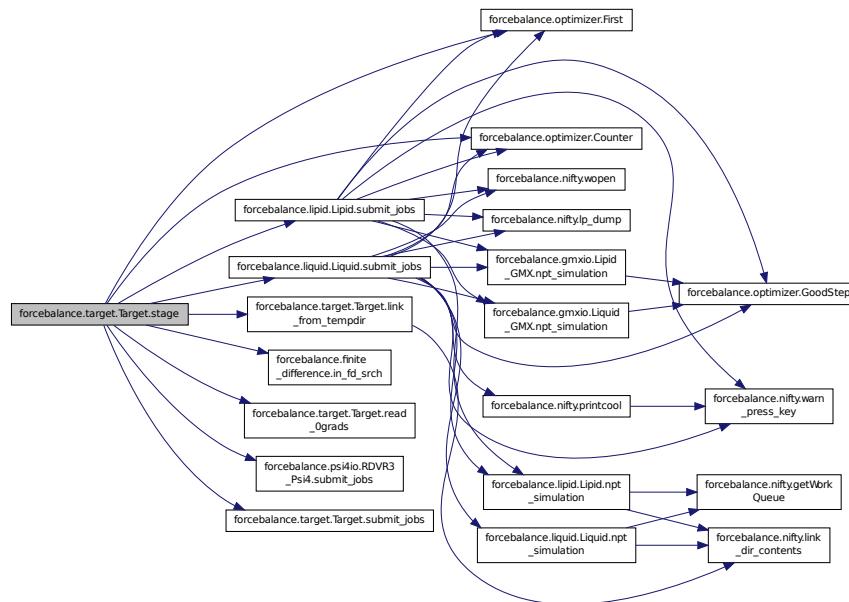
```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]
```

Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

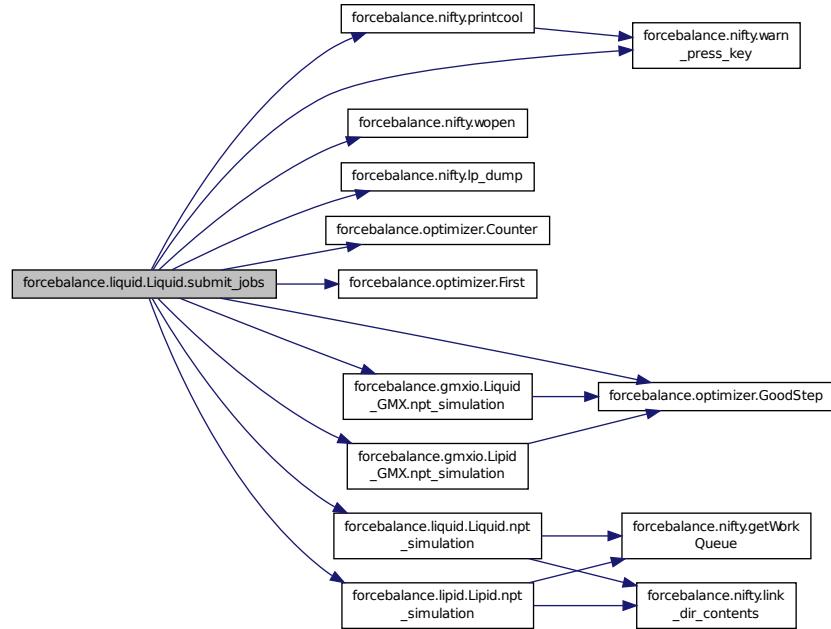
Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.submit_jobs ( self, mvals, AGrad = True, AHess = True ) [inherited]
```

Definition at line 409 of file liquid.py.

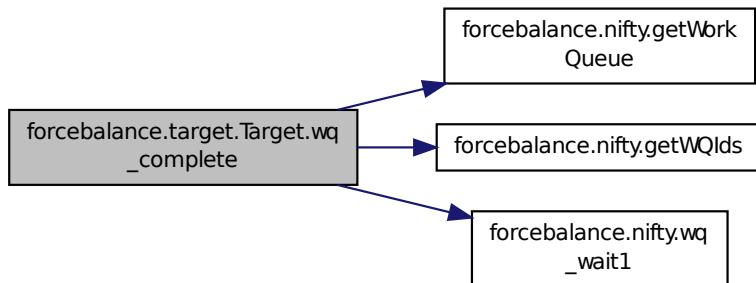
Here is the call graph for this function:



**def forcebalance.target.Target.wq\_complete( self ) [inherited]** This method determines whether the Work Queue tasks for the current target have completed.

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_Ograds( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.36.4 Member Data Documentation

**forcebalance.liquid.Liquid.AllResults [inherited]** Saved results for all iterations self.SavedMVals = [].  
Definition at line 151 of file liquid.py.

**forcebalance.liquid.Liquid.do\_self\_pol [inherited]** Definition at line 98 of file liquid.py.

**forcebalance.openmmio.Liquid\_OpenMM.engine** Definition at line 1147 of file openmmio.py.

**forcebalance.openmmio.Liquid\_OpenMM.engname** Definition at line 1149 of file openmmio.py.

**forcebalance.openmmio.Liquid\_OpenMM.extra\_output** Definition at line 1162 of file openmmio.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)  
Definition at line 160 of file target.py.

**forcebalance.liquid.Liquid.gas\_engine [inherited]** Read the reference data.  
Definition at line 133 of file liquid.py.

**forcebalance.openmmio.Liquid\_OpenMM.gas\_engine\_args** Definition at line 1155 of file openmmio.py.

**forcebalance.liquid.Liquid.gas\_mol [inherited]** Definition at line 114 of file liquid.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.  
Definition at line 164 of file target.py.

**forcebalance.liquid.Liquid.Gp [inherited]** Definition at line 895 of file liquid.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.liquid.Liquid.Labels [inherited]** Definition at line 243 of file liquid.py.

**forcebalance.liquid.Liquid.last\_traj [inherited]** Definition at line 116 of file liquid.py.

**forcebalance.liquid.Liquid.liquid\_mol [inherited]** Definition at line 110 of file liquid.py.

**forcebalance.liquid.Liquid.MBarEnergy [inherited]** Evaluated energies for all trajectories (i.e. all iterations and all temperatures), using all mvals  
Definition at line 148 of file liquid.py.

**forcebalance.openmmio.Liquid\_OpenMM.nptfiles** Definition at line 1153 of file openmmio.py.

**forcebalance.openmmio.Liquid\_OpenMM.nptpfx** Definition at line 1151 of file openmmio.py.

**forcebalance.liquid.Liquid.Objective [inherited]** Definition at line 897 of file liquid.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.  
Dictionary of whether to call the derivatives.  
Definition at line 127 of file target.py.

**forcebalance.liquid.Liquid.PhasePoints** [inherited] Definition at line 239 of file liquid.py.

**forcebalance.liquid.Liquid.Pp** [inherited] Definition at line 892 of file liquid.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd** [inherited] Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.liquid.Liquid.read\_indicate** [inherited] Definition at line 135 of file liquid.py.

**forcebalance.target.Target.read\_objective** [inherited] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.liquid.Liquid.RefData** [inherited] Definition at line 173 of file liquid.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.liquid.Liquid.SavedTraj** [inherited] Saved trajectories for all iterations and all temperatures.

Definition at line 146 of file liquid.py.

**forcebalance.openmmio.Liquid\_OpenMM.scripts** Definition at line 1157 of file openmmio.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.liquid.Liquid.w\_alpha** [inherited] Definition at line 868 of file liquid.py.

**forcebalance.liquid.Liquid.w\_cp** [inherited] Definition at line 870 of file liquid.py.

**forcebalance.liquid.Liquid.w\_eps0** [inherited] Definition at line 871 of file liquid.py.

**forcebalance.liquid.Liquid.w\_hvap** [inherited] Definition at line 867 of file liquid.py.

**forcebalance.liquid.Liquid.w\_kappa** [inherited] Definition at line 869 of file liquid.py.

**forcebalance.liquid.Liquid.w\_rho** [inherited] Density.

Enthalpy of vaporization. Thermal expansion coefficient. Isothermal compressibility. Isobaric heat capacity. Static dielectric constant. Estimation of errors.

Definition at line 866 of file liquid.py.

**forcebalance.liquid.Liquid.Wp** [inherited] Definition at line 890 of file liquid.py.

**forcebalance.liquid.Liquid.write\_indicate** [inherited] Definition at line 136 of file liquid.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

**forcebalance.liquid.Liquid.Xp** [inherited] Definition at line 888 of file liquid.py.

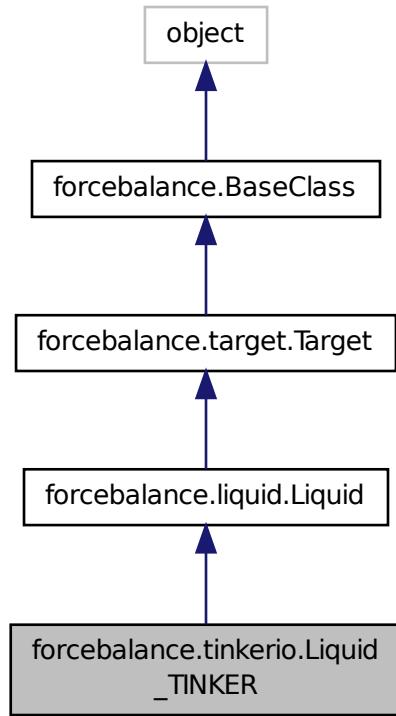
The documentation for this class was generated from the following file:

- [openmmio.py](#)

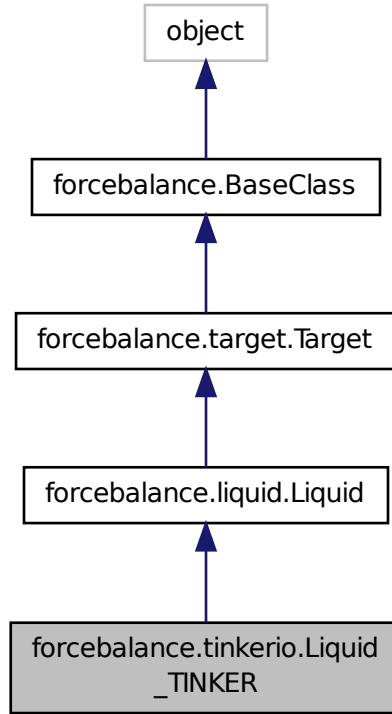
## 8.37 forcebalance.tinkerio.Liquid\_TINKER Class Reference

Condensed phase property matching using [TINKER](#).

Inheritance diagram for forcebalance.tinkerio.Liquid\_TINKER:



Collaboration diagram for forcebalance.tinkerio.Liquid\_TINKER:



### Public Member Functions

- def `__init__`
- def `npt_simulation`

*Submit a NPT simulation to the Work Queue.*
- def `prepare_temp_directory`

*Prepare the temporary directory by copying in important files.*
- def `read_data`
- def `check_files`
- def `polarization_correction`
- def `indicate`
- def `objective_term`
- def `submit_jobs`
- def `read`

*Read in time series for all previous iterations.*
- def `get`

*Fitting of liquid bulk properties.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*

- def [read\\_0grads](#)  
*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def [write\\_0grads](#)  
*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def [get\\_G](#)  
*Computes the objective function contribution and its gradient.*
- def [get\\_H](#)  
*Computes the objective function contribution and its gradient / Hessian.*
- def [link\\_from\\_tempdir](#)
- def [refresh\\_temp\\_directory](#)  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def [absrd](#)  
*Supply the correct directory specified by user's "read" option.*
- def [maxrd](#)  
*Supply the latest existing temp-directory containing valid data.*
- def [meta\\_indicate](#)  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def [meta\\_get](#)  
*Wrapper around the get function.*
- def [stage](#)  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def [wq\\_complete](#)  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def [printcool\\_table](#)  
*Print target information in an organized table format.*
- def [\\_\\_setattr\\_\\_](#)
- def [set\\_option](#)

## Public Attributes

- [engine\\_](#)
- [engname](#)
- [nptpx](#)
- [nptfiles](#)
- [gas\\_engine\\_args](#)
- [scripts](#)
- [extra\\_output](#)
- [DynDict](#)
- [DynDict\\_New](#)
- [last\\_traj](#)
- [do\\_self\\_pol](#)
- [liquid\\_mol](#)
- [gas\\_mol](#)
- [gas\\_engine](#)  
*Read the reference data.*
- [read\\_indicate](#)
- [write\\_indicate](#)
- [SavedTraj](#)  
*Saved trajectories for all iterations and all temperatures.*

- **MBarEnergy**  
*Evaluates energies for all trajectories (i.e.*
- **AllResults**  
*Saved results for all iterations self.SavedMVals = [].*
- **RefData**
- **PhasePoints**
- **Labels**
- **w\_rho**  
*Density.*
- **w\_hvap**
- **w\_alpha**
- **w\_kappa**
- **w\_cp**
- **w\_eps0**
- **Xp**
- **Wp**
- **Pp**
- **Gp**
- **Objective**
- **rd**  
*Root directory of the whole project.*
- **pgrad**  
*Iteration where we turn on zero-gradient skipping.*
- **tempbase**  
*Relative directory of target.*
- **tempdir**
- **rundir**  
*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- **FF**  
*Need the forcefield (here for now)*
- **xct**  
*Counts how often the objective function was computed.*
- **gct**  
*Counts how often the gradient was computed.*
- **hct**  
*Counts how often the Hessian was computed.*
- **read\_objective**  
*Whether to read objective.p from file when restarting an aborted run.*
- **write\_objective**  
*Whether to write objective.p at every iteration (true for all but remote.)*
- **verbose\_options**
- **PrintOptionDict**

### 8.37.1 Detailed Description

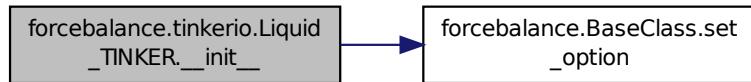
Condensed phase property matching using [TINKER](#).

Definition at line 1014 of file `tinkerio.py`.

### 8.37.2 Constructor & Destructor Documentation

```
def forcebalance.tinkerio.Liquid_TINKER.__init__ ( self, options, tgt_opts, forcefield ) Definition at line 1015  
of file tinkerio.py.
```

Here is the call graph for this function:



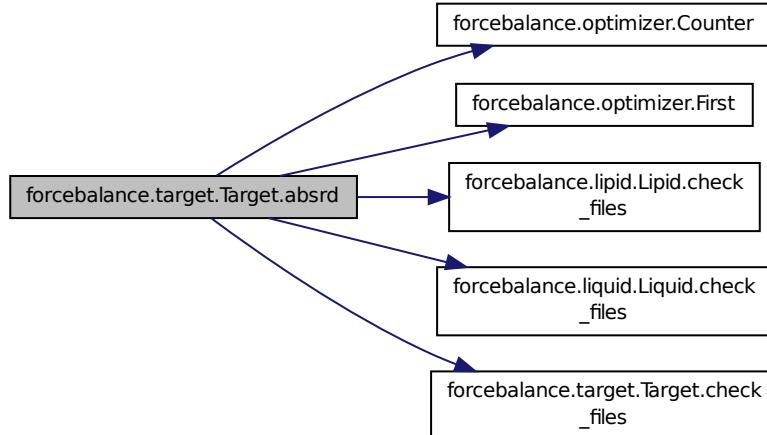
### 8.37.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory  
specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.check_files ( self, there ) [inherited] Definition at line 253 of file liquid.py.
```

```
def forcebalance.liquid.Liquid.get ( self, mvals, AGrad = True, AHess = True ) [inherited] Fitting  
of liquid bulk properties.
```

This is the current major direction of development for ForceBalance. Basically, fitting the QM energies / forces alone does not always give us the best simulation behavior. In many cases it makes more sense to try and reproduce some experimentally known data as well.

In order to reproduce experimentally known data, we need to run a simulation and compare the simulation result to experiment. The main challenge here is that the simulations are computationally intensive (i.e. they require energy and force evaluations), and furthermore the results are noisy. We need to run the simulations automatically and remotely (i.e. on clusters) and a good way to calculate the derivatives of the simulation results with respect to the parameter values.

This function contains some experimentally known values of the density and enthalpy of vaporization (H<sub>vap</sub>) of liquid water. It launches the density and H<sub>vap</sub> calculations on the cluster, and gathers the results / derivatives. The actual calculation of results / derivatives is done in a separate file.

After the results come back, they are gathered together to form an objective function.

#### Parameters

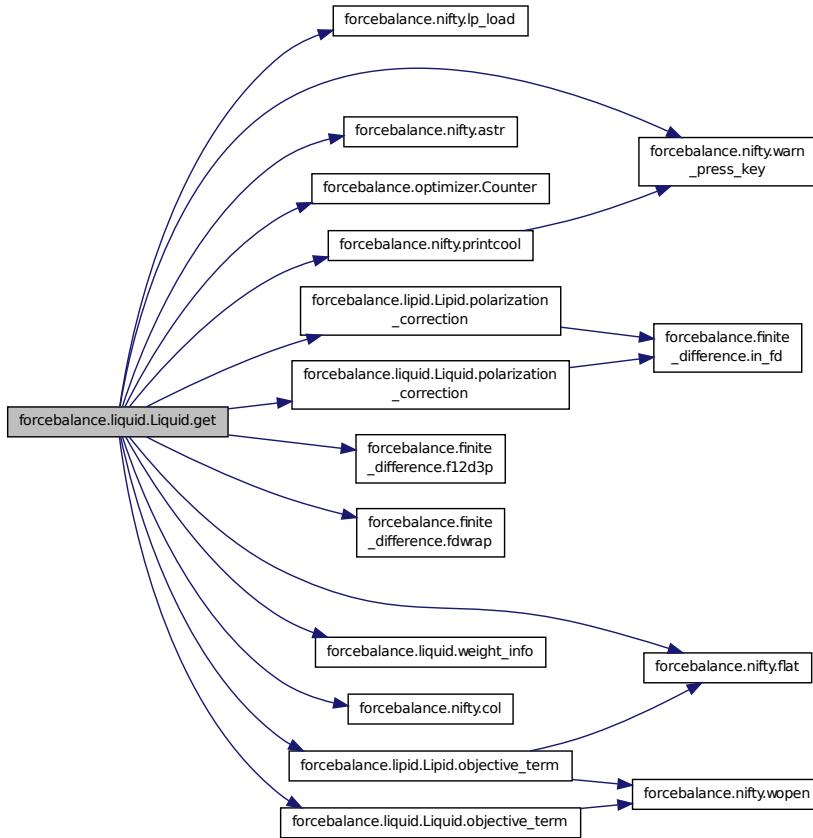
in	<i>mvals</i>	Mathematical parameter values
in	<i>AGrad</i>	Switch to turn on analytic gradient
in	<i>AHess</i>	Switch to turn on analytic Hessian

#### Returns

Answer Contribution to the objective function Fill in the weight matrix with MBAR weights where MBAR was run, and equal weights otherwise.

Definition at line 553 of file liquid.py.

Here is the call graph for this function:



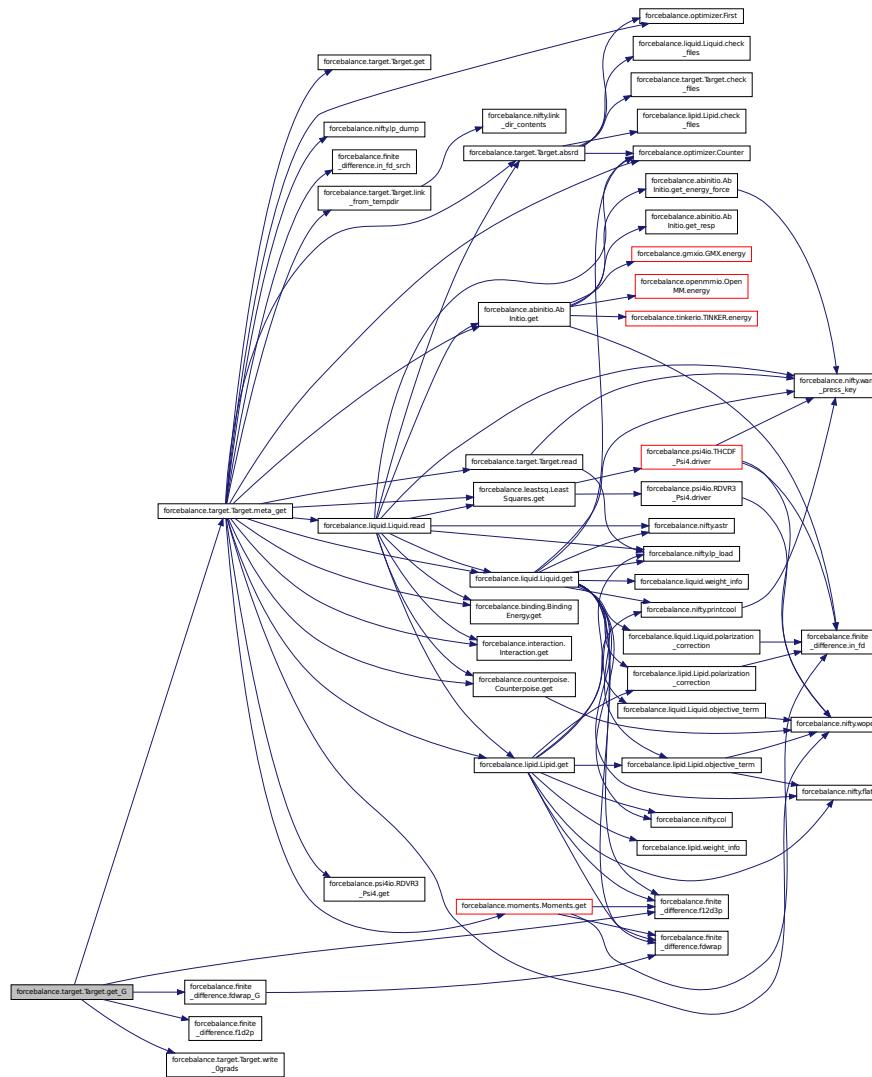
```
def forcebalance.target.Target.get_G( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



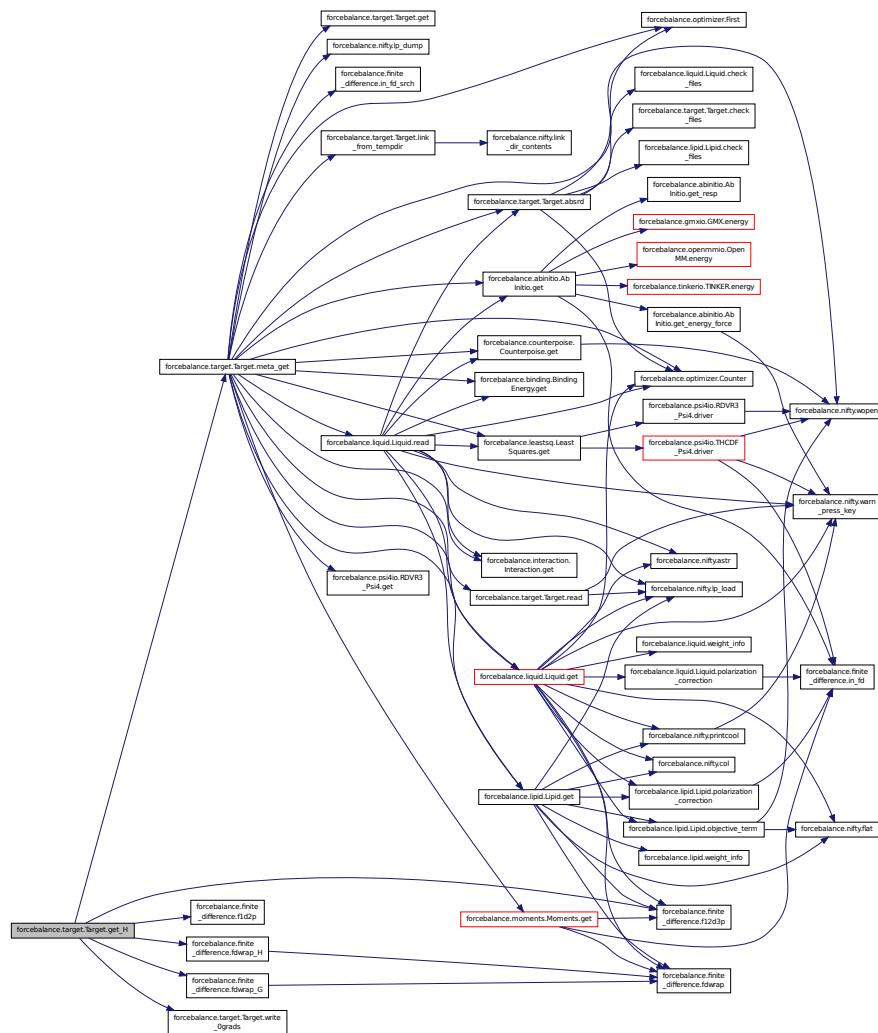
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

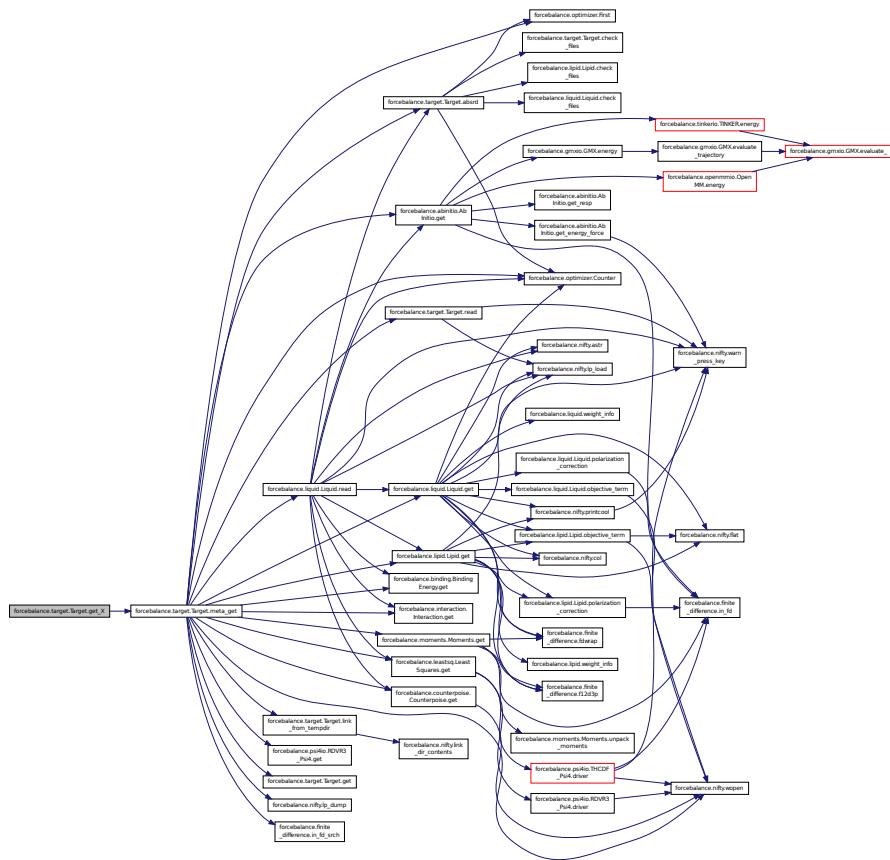
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

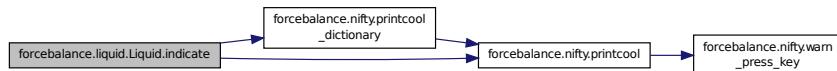
Definition at line 184 of file target.py.

Here is the call graph for this function:



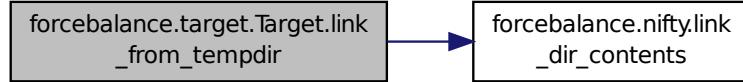
**def forcebalance.liquid.Liquid.indicate ( self ) [inherited]** Definition at line 304 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

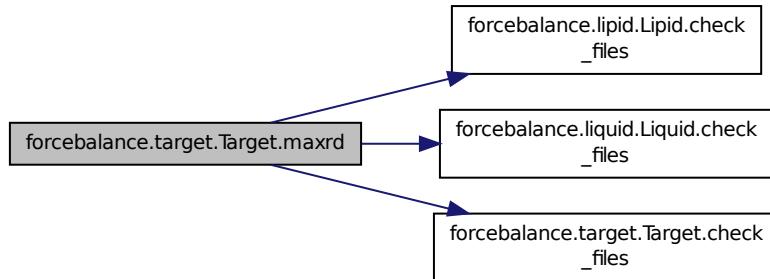
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

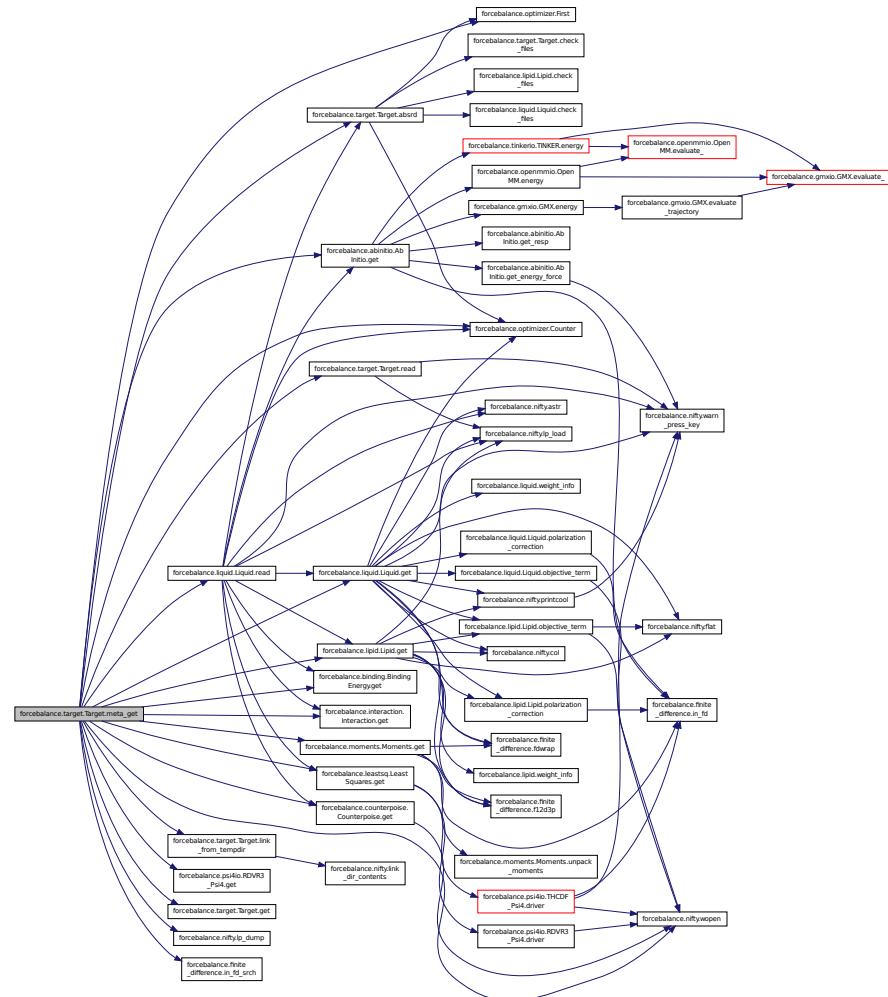


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

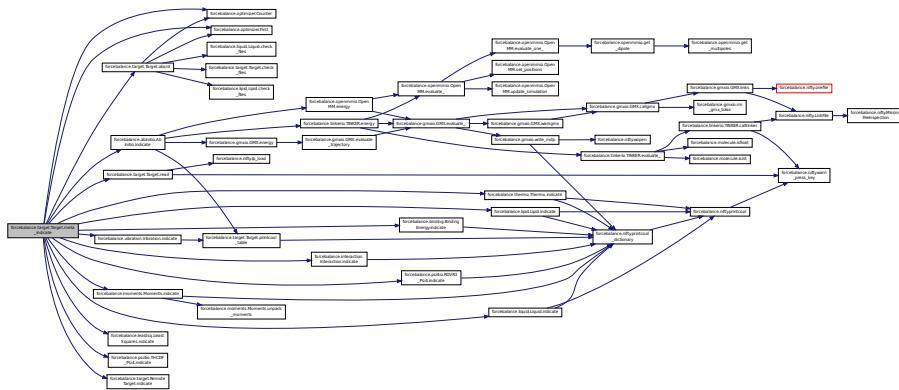
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

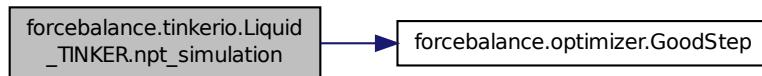
Here is the call graph for this function:



**def forcebalance.tinkerio.Liquid\_TINKER.npt\_simulation ( self, temperature, pressure, simnum )** Submit a NPT simulation to the Work Queue.

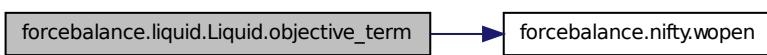
Definition at line 1052 of file tinkerio.py.

Here is the call graph for this function:



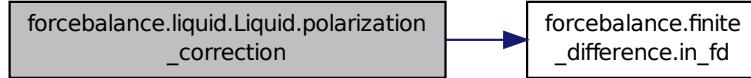
**def forcebalance.liquid.Liquid.objective\_term ( self, points, expname, calc, err, grad, name = "Quantity", SubAverage = False )** [inherited] Definition at line 330 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.liquid.Liquid.polarization.correction ( self, mvals )** [inherited] Definition at line 286 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.liquid.Liquid.prepare\_temp\_directory ( self ) [inherited]** Prepare the temporary directory by copying in important files.

Definition at line 155 of file liquid.py.

Here is the call graph for this function:



**def forcebalance.target.Target.printcool\_table ( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited]** Print target information in an organized table format.

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

The input should be something like:

#### Parameters

<b>data</b>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<b>headings</b>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<b>banner</b>	Optional heading line, which will be printed at the top in the title.
<b>footnote</b>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

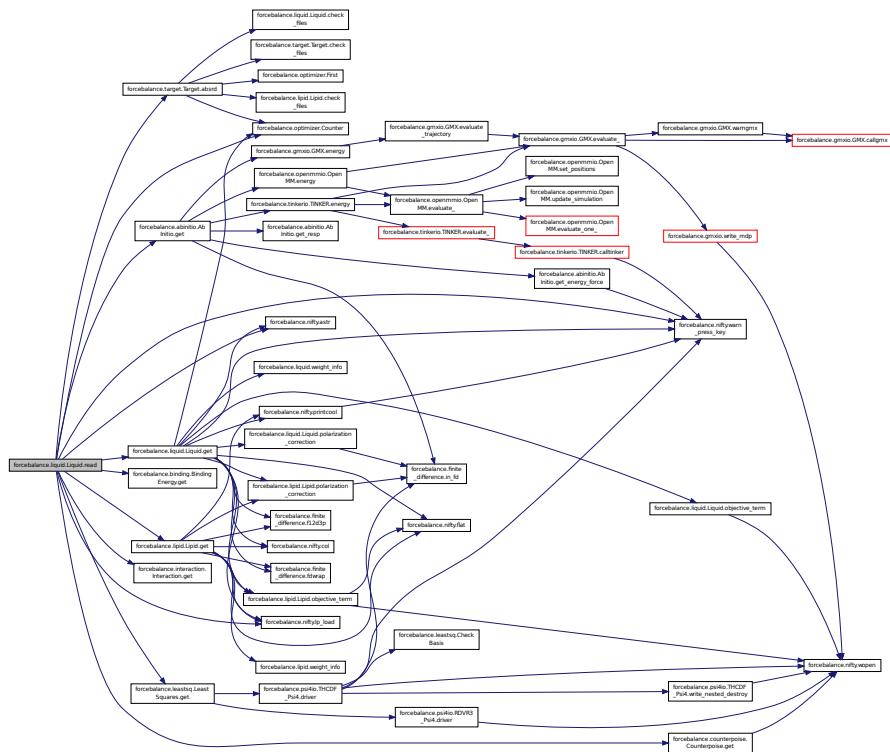
Here is the call graph for this function:



**def** forcebalance.liquid.Liquid.read ( self, mvals, AGrad = True, AHess = True ) [inherited] Read in time series for all previous iterations.

Definition at line 447 of file liquid.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.read_0grads ( self ) [inherited] Read a file from the target directory containing names of parameters that don't contribute to the gradient.
```

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.liquid.Liquid.read\_data ( self ) [inherited]** Definition at line 162 of file liquid.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

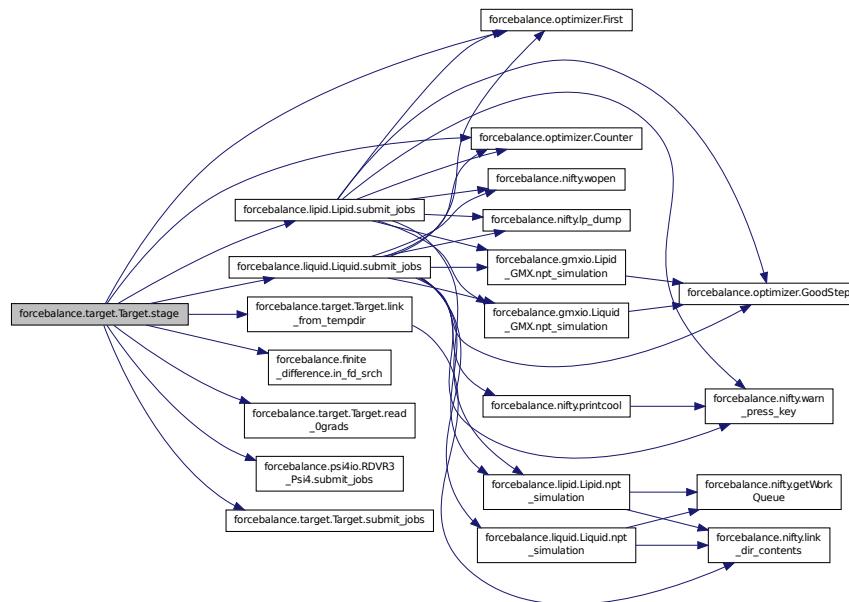
```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]
```

Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

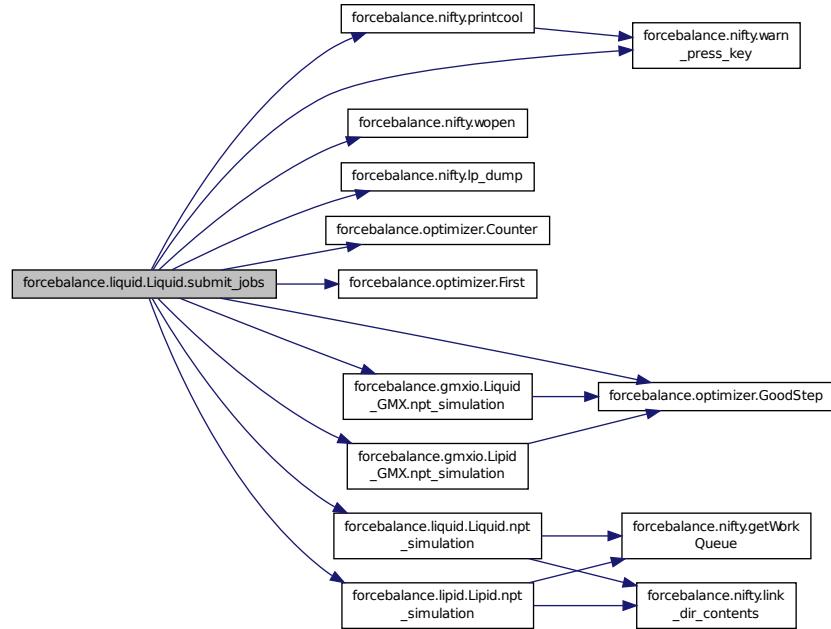
Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.submit_jobs ( self, mvals, AGrad = True, AHess = True ) [inherited]
```

Definition at line 409 of file liquid.py.

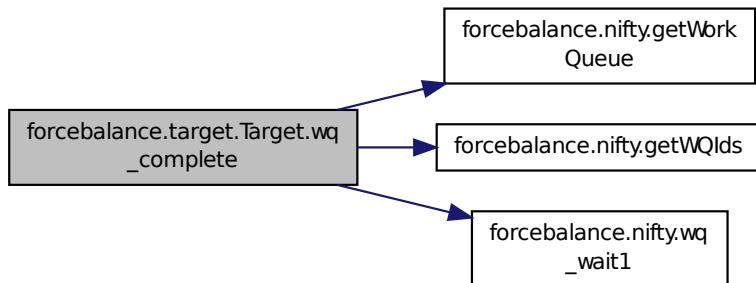
Here is the call graph for this function:



**def forcebalance.target.Target.wq\_complete( self ) [inherited]** This method determines whether the Work Queue tasks for the current target have completed.

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_Ograds( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.37.4 Member Data Documentation

**forcebalance.liquid.Liquid.AllResults [inherited]** Saved results for all iterations self.SavedMVals = [].  
Definition at line 151 of file liquid.py.

**forcebalance.liquid.Liquid.do\_self\_pol [inherited]** Definition at line 98 of file liquid.py.

**forcebalance.tinkerio.Liquid\_TINKER.DynDict** Definition at line 1047 of file tinkerio.py.

**forcebalance.tinkerio.Liquid\_TINKER.DynDict\_New** Definition at line 1048 of file tinkerio.py.

**forcebalance.tinkerio.Liquid\_TINKER.engine\_** Definition at line 1025 of file tinkerio.py.

**forcebalance.tinkerio.Liquid\_TINKER.engname** Definition at line 1027 of file tinkerio.py.

**forcebalance.tinkerio.Liquid\_TINKER.extra\_output** Definition at line 1043 of file tinkerio.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)  
Definition at line 160 of file target.py.

**forcebalance.liquid.Liquid.gas\_engine [inherited]** Read the reference data.  
Definition at line 133 of file liquid.py.

**forcebalance.tinkerio.Liquid\_TINKER.gas\_engine\_args** Definition at line 1033 of file tinkerio.py.

**forcebalance.liquid.Liquid.gas\_mol [inherited]** Definition at line 114 of file liquid.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.  
Definition at line 164 of file target.py.

**forcebalance.liquid.Liquid.Gp [inherited]** Definition at line 895 of file liquid.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.liquid.Liquid.Labels [inherited]** Definition at line 243 of file liquid.py.

**forcebalance.tinkerio.Liquid\_TINKER.last\_traj** Definition at line 1063 of file tinkerio.py.

**forcebalance.liquid.Liquid.liquid\_mol [inherited]** Definition at line 110 of file liquid.py.

**forcebalance.liquid.Liquid.MBarEnergy [inherited]** Evaluated energies for all trajectories (i.e.  
all iterations and all temperatures), using all mvals  
Definition at line 148 of file liquid.py.

**forcebalance.tinkerio.Liquid\_TINKER.nptfiles** Definition at line 1031 of file tinkerio.py.

**forcebalance.tinkerio.Liquid\_TINKER.nptpfx** Definition at line 1029 of file tinkerio.py.

**forcebalance.liquid.Liquid.Objective** [inherited] Definition at line 897 of file liquid.py.

**forcebalance.target.Target.pgrad** [inherited] Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.liquid.Liquid.PhasePoints** [inherited] Definition at line 239 of file liquid.py.

**forcebalance.liquid.Liquid.Pp** [inherited] Definition at line 892 of file liquid.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd** [inherited] Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.liquid.Liquid.read\_indicate** [inherited] Definition at line 135 of file liquid.py.

**forcebalance.target.Target.read\_objective** [inherited] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.liquid.Liquid.RefData** [inherited] Definition at line 173 of file liquid.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.liquid.Liquid.SavedTraj** [inherited] Saved trajectories for all iterations and all temperatures.

Definition at line 146 of file liquid.py.

**forcebalance.tinkerio.Liquid\_TINKER.scripts** Definition at line 1035 of file tinkerio.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.liquid.Liquid.w\_alpha** [inherited] Definition at line 868 of file liquid.py.

**forcebalance.liquid.Liquid.w\_cp** [inherited] Definition at line 870 of file liquid.py.

**forcebalance.liquid.Liquid.w\_eps0** [inherited] Definition at line 871 of file liquid.py.

**forcebalance.liquid.Liquid.w\_hvap** [inherited] Definition at line 867 of file liquid.py.

**forcebalance.liquid.Liquid.w\_kappa** [inherited] Definition at line 869 of file liquid.py.

**forcebalance.liquid.Liquid.w\_rho** [inherited] Density.  
Enthalpy of vaporization. Thermal expansion coefficient. Isothermal compressibility. Isobaric heat capacity. Static dielectric constant. Estimation of errors.  
Definition at line 866 of file liquid.py.

**forcebalance.liquid.Liquid.Wp** [inherited] Definition at line 890 of file liquid.py.

**forcebalance.liquid.Liquid.write\_indicate** [inherited] Definition at line 136 of file liquid.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)  
Definition at line 174 of file target.py.

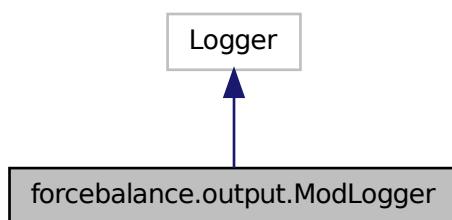
**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.  
Definition at line 162 of file target.py.

**forcebalance.liquid.Liquid.Xp** [inherited] Definition at line 888 of file liquid.py.  
The documentation for this class was generated from the following file:

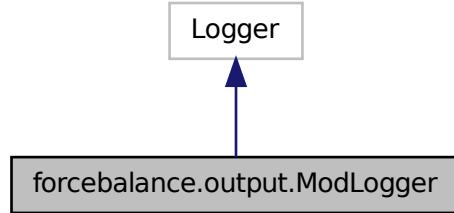
- [tinkerio.py](#)

## 8.38 forcebalance.output.ModLogger Class Reference

Inheritance diagram for forcebalance.output.ModLogger:



Collaboration diagram for forcebalance.output.ModLogger:



#### Public Member Functions

- def [error](#)

##### 8.38.1 Detailed Description

Definition at line 82 of file output.py.

##### 8.38.2 Member Function Documentation

**def forcebalance.output.ModLogger.error ( self, msg, args, kwargs )** Definition at line 83 of file output.py.

The documentation for this class was generated from the following file:

- [output.py](#)

#### 8.39 forcebalance.Mol2.mol2 Class Reference

This is to manage one [mol2](#) series of lines on the form:

#### Public Member Functions

- def [\\_\\_init\\_\\_](#)
- def [\\_\\_repr\\_\\_](#)
- def [out](#)
- def [set\\_mol\\_name](#)  
    *bond identifier (integer, starting from 1)*
- def [set\\_num\\_atoms](#)  
    *number of atoms (integer)*
- def [set\\_num\\_bonds](#)  
    *number of bonds (integer)*
- def [set\\_num\\_subst](#)  
    *number of substructures (integer)*
- def [set\\_num\\_feat](#)  
    *number of features (integer)*
- def [set\\_num\\_sets](#)

- def `set_mol_type`  
`bond identifier (integer, starting from 1)`
- def `set_charge_type`  
`bond identifier (integer, starting from 1)`
- def `parse`  
`Parse a series of text lines, and setup compound information.`
- def `get_atom`  
`return the atom instance given its atom identifier`
- def `get_bonded_atoms`  
`return a dictionary of atom instances bonded to the atom, and their types`
- def `set_donor_acceptor_atoms`  
`modify atom types to specify donor, acceptor, or both`

## Public Attributes

- `mol_name`
- `num_atoms`
- `num_bonds`
- `num_subst`
- `num_feat`
- `num_sets`
- `mol_type`
- `charge_type`
- `comments`
- `atoms`
- `bonds`

### 8.39.1 Detailed Description

This is to manage one `mol2` series of lines on the form:

```
@<TRIPOS>MOLECULE
CDK2.xray.inhl.1E9H
34 37 0 0 0
SMALL
GASTEIGER
Energy = 0

@<TRIPOS>ATOM
 1 C1          5.4790   42.2880   49.5910 C.ar     1  <1>      0.0424
 2 C2          4.4740   42.6430   50.5070 C.ar     1  <1>      0.0447
@<TRIPOS>BOND
 1    1    2    ar
 2    1    6    ar
```

Definition at line 288 of file Mol2.py.

### 8.39.2 Constructor & Destructor Documentation

`def forcebalance.Mol2.mol2.__init__ ( self, data )` Definition at line 289 of file Mol2.py.

### 8.39.3 Member Function Documentation

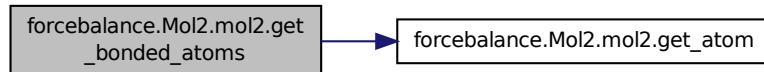
`def forcebalance.Mol2.mol2.__repr__ ( self )` Definition at line 305 of file Mol2.py.

```
def forcebalance.Mol2.mol2.get_atom( self, id )  return the atom instance given its atom identifier  
Definition at line 461 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2.get_bonded_atoms( self, id )  return a dictionary of atom instances bonded to the  
atom, and their types
```

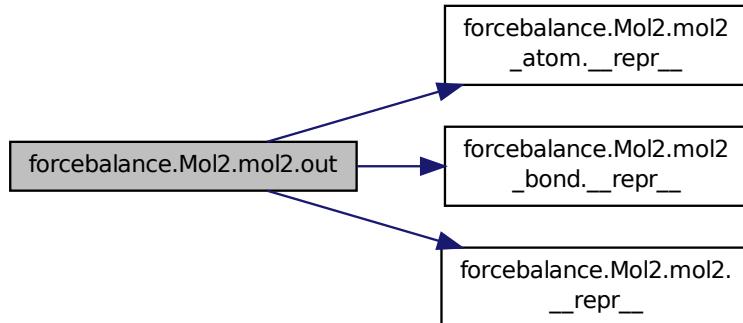
Definition at line 475 of file Mol2.py.

Here is the call graph for this function:



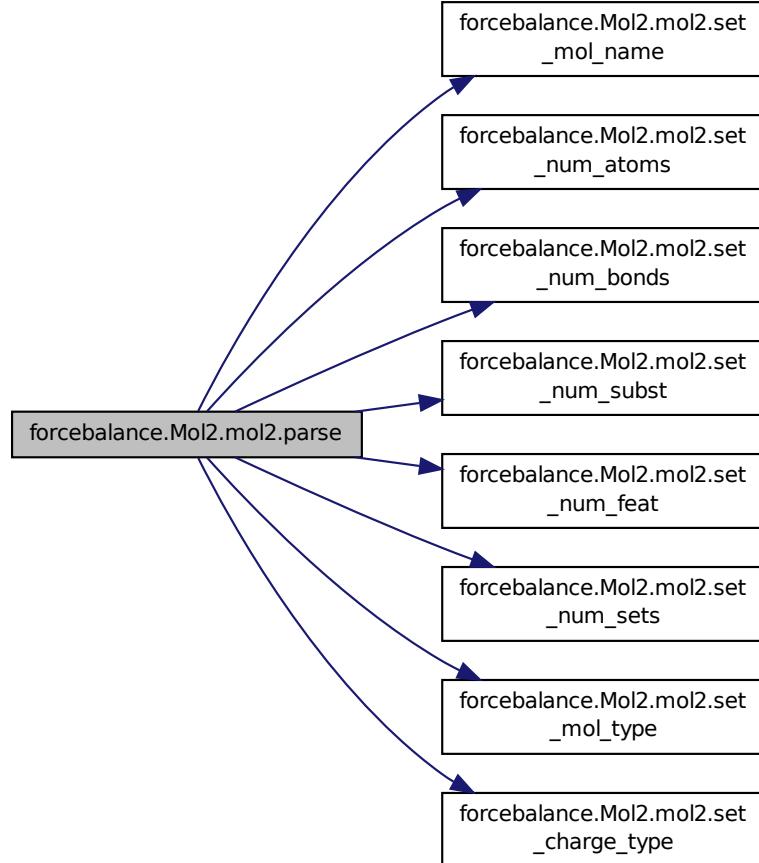
```
def forcebalance.Mol2.mol2.out( self, f=sys.stdout )  Definition at line 325 of file Mol2.py.
```

Here is the call graph for this function:



```
def forcebalance.Mol2.mol2.parse( self, data )  Parse a series of text lines, and setup compound information.  
Definition at line 405 of file Mol2.py.
```

Here is the call graph for this function:



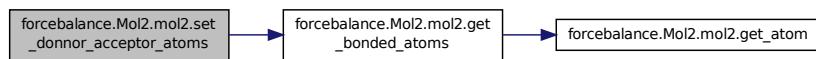
```
def forcebalance.Mol2.mol2.set_charge_type ( self, charge_type = None ) bond identifier (integer, starting from 1)
```

Definition at line 395 of file Mol2.py.

```
def forcebalance.Mol2.mol2.set_donor_acceptor_atoms ( self, verbose = 0 ) modify atom types to specify donor, acceptor, or both
```

Definition at line 490 of file Mol2.py.

Here is the call graph for this function:



```
def forcebalance.Mol2.mol2.set.mol_name ( self, mol_name = None ) bond identifier (integer, starting from 1)
Definition at line 332 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2.set.mol_type ( self, mol_type = None ) bond identifier (integer, starting from 1)
Definition at line 386 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2.set.num_atoms ( self, num_atoms = None ) number of atoms (integer)
Definition at line 341 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2.set.num_bonds ( self, num_bonds = None ) number of bonds (integer)
Definition at line 350 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2.set.num_feat ( self, num_feat = None ) number of features (integer)
Definition at line 368 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2.set.num_sets ( self, num_sets = None ) number of sets (integer)
Definition at line 377 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2.set.num_subst ( self, num_subst = None ) number of substructures (integer)
Definition at line 359 of file Mol2.py.
```

#### 8.39.4 Member Data Documentation

**forcebalance.Mol2.mol2.atoms** Definition at line 300 of file Mol2.py.

**forcebalance.Mol2.mol2.bonds** Definition at line 301 of file Mol2.py.

**forcebalance.Mol2.mol2.charge\_type** Definition at line 297 of file Mol2.py.

**forcebalance.Mol2.mol2.comments** Definition at line 298 of file Mol2.py.

**forcebalance.Mol2.mol2.mol\_name** Definition at line 290 of file Mol2.py.

**forcebalance.Mol2.mol2.mol\_type** Definition at line 296 of file Mol2.py.

**forcebalance.Mol2.mol2.num\_atoms** Definition at line 291 of file Mol2.py.

**forcebalance.Mol2.mol2.num\_bonds** Definition at line 292 of file Mol2.py.

**forcebalance.Mol2.mol2.num\_feat** Definition at line 294 of file Mol2.py.

**forcebalance.Mol2.mol2.num\_sets** Definition at line 295 of file Mol2.py.

**forcebalance.Mol2.mol2.num\_subst** Definition at line 293 of file Mol2.py.  
The documentation for this class was generated from the following file:

- [Mol2.py](#)

#### 8.40 forcebalance.Mol2.mol2\_atom Class Reference

This is to manage [mol2](#) atomic lines on the form: 1 C1 5.4790 42.2880 49.5910 C.ar 1 <1> 0.0424.

## Public Member Functions

- def `_init_`  
*if data is passed, it will be installed*
- def `parse`  
*split the text line into a series of properties*
- def `_repr_`  
*assemble the properties as a text line, and return it*
- def `set_atom_id`  
*atom identifier (integer, starting from 1)*
- def `set_atom_name`  
*The name of the atom (string)*
- def `set_crds`  
*the coordinates of the atom*
- def `set_atom_type`  
*The mol2 type of the atom.*
- def `set_subst_id`  
*substructure identifier*
- def `set_subst_name`  
*substructure name*
- def `set_charge`  
*atomic charge*
- def `set_status_bit`  
*Never to use (in theory)*

## Public Attributes

- `atom_id`
- `atom_name`
- `x`
- `y`
- `z`
- `atom_type`
- `subst_id`
- `subst_name`
- `charge`
- `status_bit`

### 8.40.1 Detailed Description

This is to manage mol2 atomic lines on the form: 1 C1 5.4790 42.2880 49.5910 C.ar 1 <1> 0.0424.  
Definition at line 32 of file Mol2.py.

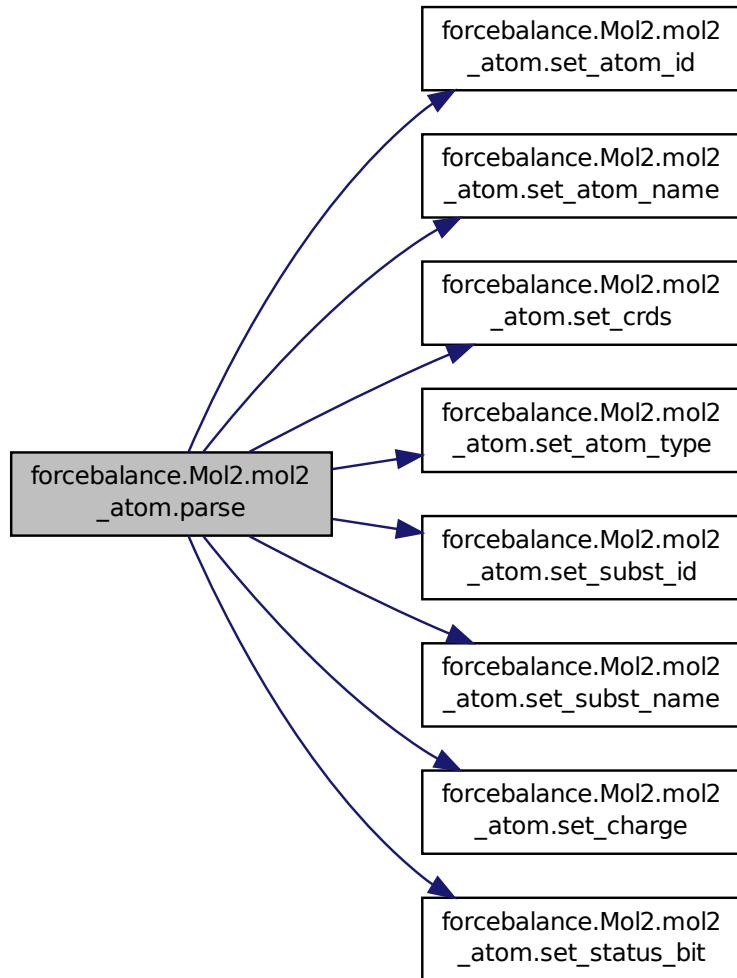
### 8.40.2 Constructor & Destructor Documentation

`def forcebalance.Mol2.mol2_atom.__init__( self, data = None )` if data is passed, it will be installed  
Definition at line 37 of file Mol2.py.

### 8.40.3 Member Function Documentation

`def forcebalance.Mol2.mol2_atom.__repr__( self )` assemble the properties as a text line, and return it  
Definition at line 78 of file Mol2.py.

```
def forcebalance.Mol2.mol2_atom.parse( self, data ) split the text line into a series of properties
Definition at line 56 of file Mol2.py.
Here is the call graph for this function:
```



```
def forcebalance.Mol2.mol2_atom.set_atom_id( self, atom_id = None ) atom identifier (integer, starting from 1)
Definition at line 90 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2_atom.set_atom_name( self, atom_name = None ) The name of the atom (string)
Definition at line 99 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2_atom.set_atom_type( self, atom_type = None ) The mol2 type of the atom.
Definition at line 119 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2_atom.set_charge( self, charge = None ) atomic charge
Definition at line 146 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2_atom.set_crds( self, x = None, y = None, z = None ) the coordinates of the
atom
Definition at line 108 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2_atom.set_status_bit( self, status_bit = None ) Never to use (in theory)
Definition at line 155 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2_atom.set_subst_id( self, subst_id = None ) substructure identifier
Definition at line 128 of file Mol2.py.
```

```
def forcebalance.Mol2.mol2_atom.set_subst_name( self, subst_name = None ) substructure name
Definition at line 137 of file Mol2.py.
```

#### 8.40.4 Member Data Documentation

**forcebalance.Mol2.mol2\_atom.atom\_id** Definition at line 38 of file Mol2.py.

**forcebalance.Mol2.mol2\_atom.atom\_name** Definition at line 39 of file Mol2.py.

**forcebalance.Mol2.mol2\_atom.atom\_type** Definition at line 43 of file Mol2.py.

**forcebalance.Mol2.mol2\_atom.charge** Definition at line 46 of file Mol2.py.

**forcebalance.Mol2.mol2\_atom.status\_bit** Definition at line 47 of file Mol2.py.

**forcebalance.Mol2.mol2\_atom.subst\_id** Definition at line 44 of file Mol2.py.

**forcebalance.Mol2.mol2\_atom.subst\_name** Definition at line 45 of file Mol2.py.

**forcebalance.Mol2.mol2\_atom.x** Definition at line 40 of file Mol2.py.

**forcebalance.Mol2.mol2\_atom.y** Definition at line 41 of file Mol2.py.

**forcebalance.Mol2.mol2\_atom.z** Definition at line 42 of file Mol2.py.

The documentation for this class was generated from the following file:

- [Mol2.py](#)

#### 8.41 forcebalance.Mol2.mol2\_bond Class Reference

This is to manage mol2 bond lines on the form: 1 1 2 ar.

## Public Member Functions

- def `__init__`  
*if data is passed, it will be installed*
- def `__repr__`
- def `parse`  
*split the text line into a series of properties*
- def `set_bond_id`  
*bond identifier (integer, starting from 1)*
- def `set_origin_atom_id`  
*the origin atom identifier (integer)*
- def `set_target_atom_id`  
*the target atom identifier (integer)*
- def `set_bond_type`  
*bond type (string) one of: 1 = single 2 = double 3 = triple am = amide ar = aromatic du = dummy un = unknown nc = not connected*
- def `set_status_bit`  
*Never to use (in theory)*

## Public Attributes

- `bond_id`
- `origin_atom_id`
- `target_atom_id`
- `bond_type`
- `status_bit`

### 8.41.1 Detailed Description

This is to manage mol2 bond lines on the form: 1 1 2 ar.

Definition at line 172 of file Mol2.py.

### 8.41.2 Constructor & Destructor Documentation

`def forcebalance.Mol2.mol2_bond.__init__( self, data = None )` if data is passed, it will be installed

Definition at line 177 of file Mol2.py.

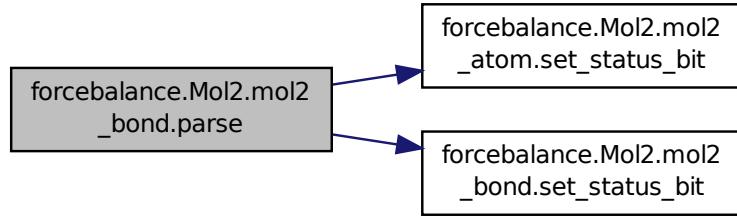
### 8.41.3 Member Function Documentation

`def forcebalance.Mol2.mol2_bond.__repr__( self )` Definition at line 186 of file Mol2.py.

`def forcebalance.Mol2.mol2_bond.parse( self, data )` split the text line into a series of properties

Definition at line 197 of file Mol2.py.

Here is the call graph for this function:



**def forcebalance.Mol2.mol2.bond.set\_bond\_id ( self, bond\_id = None )** bond identifier (integer, starting from 1)  
Definition at line 214 of file Mol2.py.

**def forcebalance.Mol2.mol2.bond.set\_bond\_type ( self, bond\_type = None )** bond type (string) one of: 1 = single 2 = double 3 = triple am = amide ar = aromatic du = dummy un = unknown nc = not connected  
Definition at line 250 of file Mol2.py.

**def forcebalance.Mol2.mol2.bond.set\_origin\_atom\_id ( self, origin\_atom\_id = None )** the origin atom identifier (integer)  
Definition at line 223 of file Mol2.py.

**def forcebalance.Mol2.mol2.bond.set\_status\_bit ( self, status\_bit = None )** Never to use (in theory)  
Definition at line 259 of file Mol2.py.

**def forcebalance.Mol2.mol2.bond.set\_target\_atom\_id ( self, target\_atom\_id = None )** the target atom identifier (integer)  
Definition at line 232 of file Mol2.py.

#### 8.41.4 Member Data Documentation

**forcebalance.Mol2.mol2.bond.bond\_id** Definition at line 178 of file Mol2.py.

**forcebalance.Mol2.mol2.bond.bond\_type** Definition at line 181 of file Mol2.py.

**forcebalance.Mol2.mol2.bond.origin\_atom\_id** Definition at line 179 of file Mol2.py.

**forcebalance.Mol2.mol2.bond.status\_bit** Definition at line 206 of file Mol2.py.

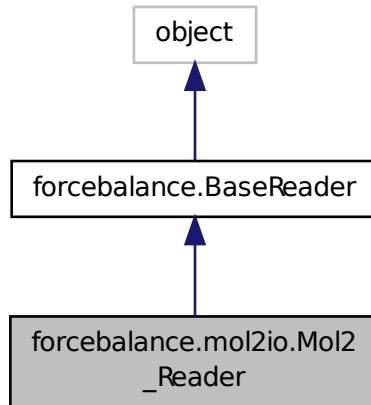
**forcebalance.Mol2.mol2.bond.target\_atom\_id** Definition at line 180 of file Mol2.py.  
The documentation for this class was generated from the following file:

- [Mol2.py](#)

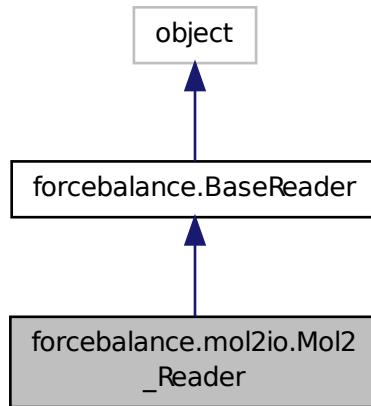
## 8.42 forcebalance.mol2io.Mol2\_Reader Class Reference

Finite state machine for parsing Mol2 force field file.

Inheritance diagram for forcebalance.mol2io.Mol2\_Reader:



Collaboration diagram for forcebalance.mol2io.Mol2\_Reader:



### Public Member Functions

- def `__init__`
- def `feed`
- def `Split`

- def [Whites](#)
- def [build\\_pid](#)  
*Returns the parameter type (e.g.*

## Public Attributes

- [pdict](#)  
*The parameter dictionary (defined in this file)*
- [atom](#)  
*The atom numbers in the interaction (stored in the parser)*
- [itype](#)
- [suffix](#)
- [ln](#)
- [adict](#)  
*The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.*
- [molatom](#)  
*The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.*
- [Molecules](#)
- [AtomTypes](#)

### 8.42.1 Detailed Description

Finite state machine for parsing [Mol2](#) force field file.  
 (just for parameterizing the charges)  
 Definition at line 22 of file mol2io.py.

### 8.42.2 Constructor & Destructor Documentation

**def forcebalance.mol2io.Mol2\_Reader.\_\_init\_\_ ( self, fnm )** Definition at line 24 of file mol2io.py.

### 8.42.3 Member Function Documentation

**def forcebalance.BaseReader.build\_pid ( self, pfid ) [inherited]** Returns the parameter type (e.g. K in BONDSK) based on the current interaction type.  
 Both the 'pdict' dictionary (see [gmxio.pdict](#)) and the interaction type 'state' (here, BONDS) are needed to get the parameter type.  
 If, however, 'pdict' does not contain the ptype value, a suitable substitute is simply the field number.  
 Note that if the interaction type state is not set, then it defaults to the file name, so a generic parameter ID is 'filename.line\_num.field\_num'  
 Definition at line 124 of file \_\_init\_\_.py.

**def forcebalance.mol2io.Mol2\_Reader.feed ( self, line )** Definition at line 32 of file mol2io.py.

**def forcebalance.BaseReader.Split ( self, line ) [inherited]** Definition at line 99 of file \_\_init\_\_.py.

**def forcebalance.BaseReader.Whites ( self, line ) [inherited]** Definition at line 102 of file \_\_init\_\_.py.

### 8.42.4 Member Data Documentation

**forcebalance.BaseReader.adict [inherited]** The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.  
 Definition at line 89 of file \_\_init\_\_.py.

**forcebalance.mol2io.Mol2\_Reader.atom** The atom numbers in the interaction (stored in the parser)  
Definition at line 30 of file mol2io.py.

**forcebalance.BaseReader.AtomTypes [inherited]** Definition at line 97 of file \_\_init\_\_.py.

**forcebalance.mol2io.Mol2\_Reader.itype** Definition at line 36 of file mol2io.py.

**forcebalance.BaseReader.In [inherited]** Definition at line 84 of file \_\_init\_\_.py.

**forcebalance.BaseReader.molatom [inherited]** The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.

self.molecdict = OrderedDict() The listing of 'RES:ATOMNAMES' for atom names in the line This is obviously a placeholder.

Definition at line 94 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules [inherited]** Definition at line 96 of file \_\_init\_\_.py.

**forcebalance.mol2io.Mol2\_Reader.pdict** The parameter dictionary (defined in this file)

Definition at line 28 of file mol2io.py.

**forcebalance.mol2io.Mol2\_Reader.suffix** Definition at line 44 of file mol2io.py.

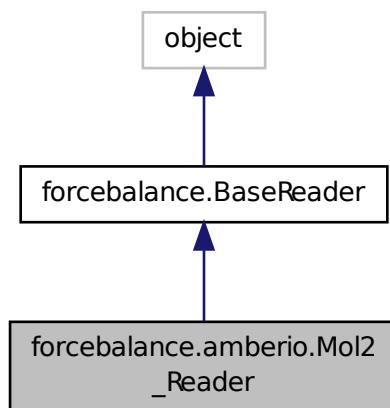
The documentation for this class was generated from the following file:

- [mol2io.py](#)

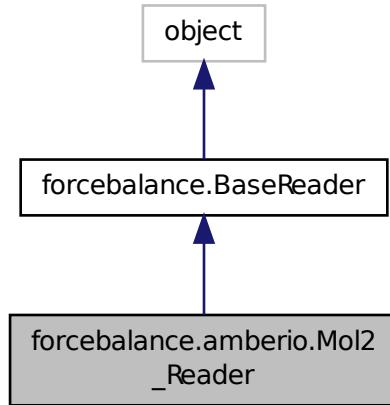
## 8.43 forcebalance.amberio.Mol2\_Reader Class Reference

Finite state machine for parsing Mol2 force field file.

Inheritance diagram for forcebalance.amberio.Mol2\_Reader:



Collaboration diagram for forcebalance.amberio.Mol2\_Reader:



## Public Member Functions

- def `_init_`
- def `feed`
- def `Split`
- def `Whites`
- def `build_pid`

*Returns the parameter type (e.g.*

## Public Attributes

- `pdict`  
*The parameter dictionary (defined in this file)*
- `atom`  
*The atom numbers in the interaction (stored in the parser)*
- `atomnames`  
*The mol2 file provides a list of atom names.*
- `section`  
*The section that we're in.*
- `mol`
- `itype`
- `suffix`
- `molatom`
- `In`
- `adict`  
*The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.*
- `Molecules`
- `AtomTypes`

### 8.43.1 Detailed Description

Finite state machine for parsing Mol2 force field file.  
(just for parameterizing the charges)  
Definition at line 43 of file amberio.py.

### 8.43.2 Constructor & Destructor Documentation

**def forcebalance.amberio.Mol2\_Reader.\_\_init\_\_ ( self, fnm )** Definition at line 45 of file amberio.py.

### 8.43.3 Member Function Documentation

**def forcebalance.BaseReader.build\_pid ( self, pfd ) [inherited]** Returns the parameter type (e.g. K in BONDSK) based on the current interaction type.  
Both the 'pdict' dictionary (see gmlio.pdict) and the interaction type 'state' (here, BONDS) are needed to get the parameter type.  
If, however, 'pdict' does not contain the ptype value, a suitable substitute is simply the field number.  
Note that if the interaction type state is not set, then it defaults to the file name, so a generic parameter ID is 'filename.line.num.field.num'  
Definition at line 124 of file \_\_init\_\_.py.

**def forcebalance.amberio.Mol2\_Reader.feed ( self, line )** Definition at line 59 of file amberio.py.

**def forcebalance.BaseReader.Split ( self, line ) [inherited]** Definition at line 99 of file \_\_init\_\_.py.

**def forcebalance.BaseReader.Whites ( self, line ) [inherited]** Definition at line 102 of file \_\_init\_\_.py.

### 8.43.4 Member Data Documentation

**forcebalance.BaseReader.adict [inherited]** The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.  
Definition at line 89 of file \_\_init\_\_.py.

**forcebalance.amberio.Mol2\_Reader.atom** The atom numbers in the interaction (stored in the parser)  
Definition at line 51 of file amberio.py.

**forcebalance.amberio.Mol2\_Reader.atomnames** The mol2 file provides a list of atom names.  
Definition at line 53 of file amberio.py.

**forcebalance.BaseReader.AtomTypes [inherited]** Definition at line 97 of file \_\_init\_\_.py.

**forcebalance.amberio.Mol2\_Reader.itype** Definition at line 64 of file amberio.py.

**forcebalance.BaseReader.In [inherited]** Definition at line 84 of file \_\_init\_\_.py.

**forcebalance.amberio.Mol2\_Reader.mol** Definition at line 57 of file amberio.py.

**forcebalance.amberio.Mol2\_Reader.molatom** Definition at line 95 of file amberio.py.

**forcebalance.BaseReader.Molecules [inherited]** Definition at line 96 of file \_\_init\_\_.py.

**forcebalance.amberio.Mol2.Reader.pdict** The parameter dictionary (defined in this file)  
Definition at line 49 of file amberio.py.

**forcebalance.amberio.Mol2.Reader.section** The section that we're in.  
Definition at line 55 of file amberio.py.

**forcebalance.amberio.Mol2.Reader.suffix** Definition at line 93 of file amberio.py.  
The documentation for this class was generated from the following file:

- [amberio.py](#)

## 8.44 forcebalance.Mol2.mol2\_set Class Reference

### Public Member Functions

- def [\\_\\_init\\_\\_](#)

*A collection is organized as a dictionnary of compounds self.num\_compounds : the number of compounds self.compounds : the dictionnary of compounds data : the data to setup the set subset: it is possible to specify a subset of the compounds to load, based on their mol.name identifiers.*

- def [parse](#)

*parse a list of lines, detect compounds, load them only load the subset if specified.*

### Public Attributes

- [num\\_compounds](#)
- [comments](#)
- [compounds](#)

#### 8.44.1 Detailed Description

Definition at line 568 of file Mol2.py.

#### 8.44.2 Constructor & Destructor Documentation

**def forcebalance.Mol2.mol2\_set.\_\_init\_\_ ( self, data = None, subset = None )** A collection is organized as a dictionnary of compounds self.num\_compounds : the number of compounds self.compounds : the dictionnary of compounds data : the data to setup the set subset: it is possible to specify a subset of the compounds to load, based on their mol.name identifiers.

Definition at line 577 of file Mol2.py.

#### 8.44.3 Member Function Documentation

**def forcebalance.Mol2.mol2\_set.parse ( self, data, subset = None )** parse a list of lines, detect compounds, load them only load the subset if specified.

Definition at line 621 of file Mol2.py.

#### 8.44.4 Member Data Documentation

**forcebalance.Mol2.mol2\_set.comments** Definition at line 579 of file Mol2.py.

**forcebalance.Mol2.mol2\_set.compounds** Definition at line 580 of file Mol2.py.

**forcebalance.Mol2.mol2.set.num\_compounds** Definition at line 578 of file Mol2.py.

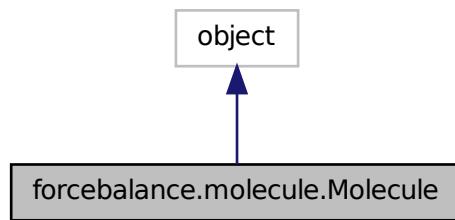
The documentation for this class was generated from the following file:

- [Mol2.py](#)

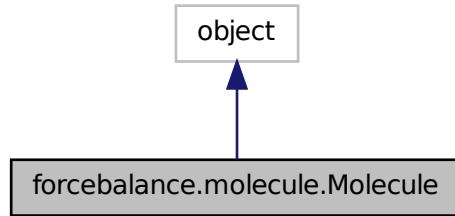
## 8.45 forcebalance.molecule.Molecule Class Reference

Lee-Ping's general file format conversion class.

Inheritance diagram for forcebalance.molecule.Molecule:



Collaboration diagram for forcebalance.molecule.Molecule:



### Public Member Functions

- def [\\_\\_len\\_\\_](#)  
*Return the number of frames in the trajectory.*
- def [\\_\\_getattr\\_\\_](#)  
*Whenever we try to get a class attribute, it first tries to get the attribute from the Data dictionary.*
- def [\\_\\_setattr\\_\\_](#)  
*Whenever we try to get a class attribute, it first tries to get the attribute from the Data dictionary.*
- def [\\_\\_getitem\\_\\_](#)  
*The [Molecule](#) class has list-like behavior, so we can get slices of it.*

- def `__delitem__`  
`Similarly, in order to delete a frame, we simply perform item deletion on framewise variables.`
- def `__iter__`  
`List-like behavior for looping over trajectories.`
- def `_add__`  
`Add method for Molecule objects.`
- def `_iadd__`  
`Add method for Molecule objects.`
- def `repair`  
`Attempt to repair trivial issues that would otherwise break the object.`
- def `append`
- def `__init__`  
`To create the Molecule object, we simply define the table of file reading/writing functions and read in a file if it is provided.`
- def `require`
- def `write`
- def `center_of_mass`
- def `radius_of_gyration`
- def `rigid_water`  
`If one atom is oxygen and the next two are hydrogen, make the water molecule rigid.`
- def `load_frames`
- def `edit_qcrems`  
`Edit Q-Chem rem variables with a dictionary.`
- def `add_quantum`
- def `add_virtual_site`  
`Add a virtual site to the system.`
- def `replace_peratom`  
`Replace all of the data for a certain attribute in the system from orig to want.`
- def `replace_peratom_conditional`  
`Replace all of the data for a attribute key2 from orig to want, contingent on key1 being equal to cond.`
- def `atom_select`  
`Return a copy of the object with certain atoms selected.`
- def `atom_stack`  
`Return a copy of the object with another molecule object appended.`
- def `align_by_moments`  
`Align molecules using the moment of inertia.`
- def `align`  
`Align molecules.`
- def `build_topology`  
`A bare-bones implementation of the bond graph capability in the nanoreactor code.`
- def `find_angles`  
`Return a list of 3-tuples corresponding to all of the angles in the system.`
- def `find_dihedrals`  
`Return a list of 4-tuples corresponding to all of the dihedral angles in the system.`
- def `measure_dihedrals`  
`Return a series of dihedral angles, given four atom indices numbered from zero.`
- def `all_pairwise_rmsd`  
`Find pairwise RMSD (super slow, not like the one in MSMBuilder.)`

- def `pathwise_rmsd`  
*Find RMSD between frames along path.*
- def `ref_rmsd`  
*Find RMSD to a reference frame.*
- def `align_center`
- def `openmm_positions`  
*Returns the Cartesian coordinates in the `Molecule` object in a list of OpenMM-compatible positions, so it is possible to type `simulation.context.setPositions(Mol.openmm_positions()[0])` or something like that.*
- def `openmm_boxes`  
*Returns the periodic box vectors in the `Molecule` object in a list of OpenMM-compatible boxes, so it is possible to type `simulation.context.setPeriodicBoxVectors(Mol.openmm_boxes()[0])` or something like that.*
- def `split`  
*Split the molecule object into a number of separate files (chunks), either by specifying the number of frames per chunk or the number of chunks.*
- def `read_xyz`  
*.xyz files can be TINKER formatted which is why we have the try/except here.*
- def `read_xyz0`  
*Parse a .xyz file which contains several xyz coordinates, and return their elements.*
- def `read_mdcrd`  
*Parse an AMBER .mdcrd file.*
- def `read_qdata`
- def `read_mol2`
- def `read_dcd`
- def `read_com`  
*Parse a Gaussian .com file and return a SINGLE-ELEMENT list of xyz coordinates (no multiple file support)*
- def `read_arc`  
*Read a TINKER .arc file.*
- def `read_gro`  
*Read a GROMACS .gro file.*
- def `read_charmm`  
*Read a CHARMM .cor (or .crd) file.*
- def `read_qcin`  
*Read a Q-Chem input file.*
- def `read_pdb`  
*Loads a PDB and returns a dictionary containing its data.*
- def `read_qcesp`
- def `read_qcout`  
*Q-Chem output file reader, adapted for our parser.*
- def `write_qcin`
- def `write_xyz`
- def `write_molproq`
- def `write_mdcrd`
- def `write_arc`
- def `write_gro`
- def `write_dcd`
- def `write_pdb`  
*Save to a PDB.*
- def `write_qdata`

*Text quantum data format.*

- def [require\\_resid](#)
- def [require\\_resname](#)
- def [require\\_boxes](#)

## Public Attributes

- [Read\\_Tab](#)

*The table of file readers.*

- [Write\\_Tab](#)

*The table of file writers.*

- [Funnel](#)

*A funnel dictionary that takes redundant file types and maps them down to a few.*

- [positive\\_resid](#)

*Creates entries like 'gromacs' : 'gromacs' and 'xyz' : 'xyz' in the Funnel.*

- [built\\_bonds](#)

- [Data](#)

- [comms](#)

*Read in stuff if we passed in a file name, otherwise return an empty instance.*

- [topology](#)

*Make sure the comment line isn't too long for i in range(len(self.comms)): self.comms[i] = self.comms[i][:100] if len(self.comms[i]) > 100 else self.comms[i]* Attempt to build the topology for small systems.

- [molecules](#)

- [fout](#)

*Fill in comments.*

- [resid](#)

- [resname](#)

- [boxes](#)

### 8.45.1 Detailed Description

Lee-Ping's general file format conversion class.

The purpose of this class is to read and write chemical file formats in a way that is convenient for research. There are highly general file format converters out there (e.g. catdcd, openbabel) but I find that writing my own class can be very helpful for specific purposes. Here are some things this class can do:

- Convert a .gro file to a .xyz file, or a .pdb file to a .dcd file. Data is stored internally, so any readable file can be converted into any writable file as long as there is sufficient information to write that file.
- Accumulate information from different files. For example, we may read A.gro to get a list of coordinates, add quantum settings from a B.in file, and write A.in (this gives us a file that we can use to run QM calculations)
- Concatenate two trajectories together as long as they're compatible. This is done by creating two [Molecule](#) objects and then simply adding them. Addition means two things: (1) Information fields missing from each class, but present in the other, are added to the sum, and (2) Appendable or per-frame fields (i.e. coordinates) are concatenated together.
- Slice trajectories using reasonable Python language. That is to say, MyMolecule[1:10] returns a new [Molecule](#) object that contains frames 1 through 9 (inclusive and numbered starting from zero.)

Next step: Read in Q-Chem output data using this too!

Special variables: These variables cannot be set manually because there is a special method associated with getting them.

na = The number of atoms. You'll get this if you use MyMol.na or MyMol['na']. na = The number of snapshots. You'll get this if you use MyMol.ns or MyMol['ns'].

Unit system: Angstroms.

Definition at line 691 of file molecule.py.

## 8.45.2 Constructor & Destructor Documentation

```
def forcebalance.molecule.Molecule.__init__( self, fnm = None, ftype = None, positive_resid = True, build_topology = True, kwargs )
```

To create the [Molecule](#) object, we simply define the table of file reading/writing functions and read in a file if it is provided.

Definition at line 896 of file molecule.py.

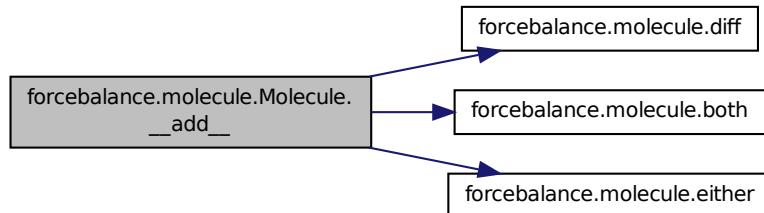
## 8.45.3 Member Function Documentation

```
def forcebalance.molecule.Molecule.__add__( self, other )
```

Add method for [Molecule](#) objects.

Definition at line 800 of file molecule.py.

Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.__delitem__( self, key )
```

Similarly, in order to delete a frame, we simply perform item deletion on framewise variables.

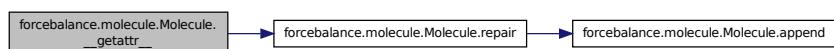
Definition at line 780 of file molecule.py.

```
def forcebalance.molecule.Molecule.__getattr__( self, key )
```

Whenever we try to get a class attribute, it first tries to get the attribute from the Data dictionary.

Definition at line 711 of file molecule.py.

Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.__getitem__ ( self, key )
```

The `Molecule` class has list-like behavior, so we can get slices of it.

If we say `MyMolecule[0:10]`, then we'll return a copy of `MyMolecule` with frames 0 through 9.

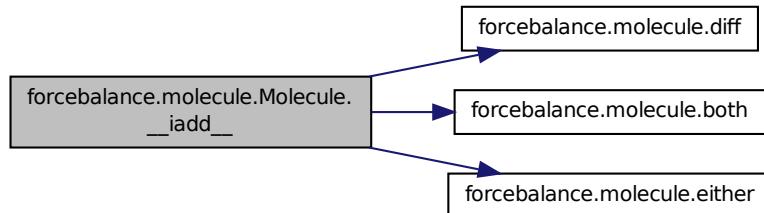
Definition at line 759 of file `molecule.py`.

```
def forcebalance.molecule.Molecule.__iadd__ ( self, other )
```

Add method for `Molecule` objects.

Definition at line 839 of file `molecule.py`.

Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.__iter__ ( self )
```

List-like behavior for looping over trajectories.

Note that these values are returned by reference. Note that this is intended to be more efficient than `getitem`, so when we loop over a trajectory, it's best to go "for m in M" instead of "for i in range(len(M)): m = M[i]"

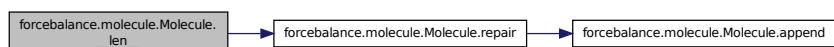
Definition at line 789 of file `molecule.py`.

```
def forcebalance.molecule.Molecule.__len__ ( self )
```

Return the number of frames in the trajectory.

Definition at line 695 of file `molecule.py`.

Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.__setattr__ ( self, key, value )
```

Whenever we try to get a class attribute, it first tries to get the attribute from the Data dictionary.

Definition at line 747 of file `molecule.py`.

```
def forcebalance.molecule.Molecule.add_quantum ( self, other )
```

Definition at line 1125 of file `molecule.py`.

```
def forcebalance.molecule.Molecule.add_virtual_site ( self, idx, kwargs )
```

Add a virtual site to the system.

This does NOT set the position of the virtual site; it sits at the origin.

Definition at line 1137 of file `molecule.py`.

```
def forcebalance.molecule.Molecule.align( self, smooth = False, center = True, center_mass = False, select = None ) Align molecules.
```

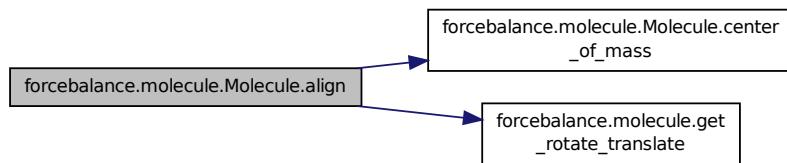
Has the option to create smooth trajectories (align each frame to the previous one) or to align each frame to the first one.

Also has the option to remove the center of mass.

Provide a list of atom indices to align along selected atoms.

Definition at line 1280 of file molecule.py.

Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.align_by_moments( self ) Align molecules using the moment of inertia.
```

Departs from MSMBuilder convention of using arithmetic mean for mass.

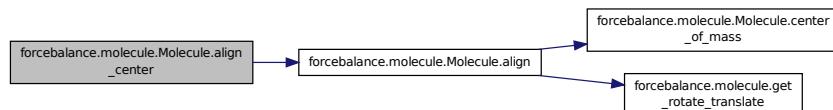
Definition at line 1258 of file molecule.py.

Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.align_center( self ) Definition at line 1583 of file molecule.py.
```

Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.all_pairwise_rmsd( self ) Find pairwise RMSD (super slow, not like the one in MSMBuilder.)
```

Definition at line 1534 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.append ( self, other )** Definition at line 889 of file molecule.py.

**def forcebalance.molecule.Molecule.atom\_select ( self, atomslice )** Return a copy of the object with certain atoms selected.

Takes an integer, list or array as argument.

Definition at line 1175 of file molecule.py.

**def forcebalance.molecule.Molecule.atom\_stack ( self, other )** Return a copy of the object with another molecule object appended.

WARNING: This function may invalidate stuff like QM energies.

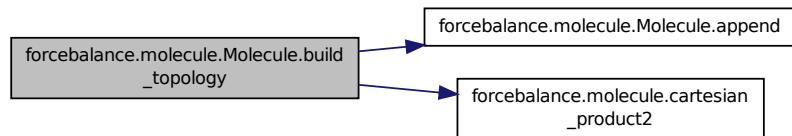
Definition at line 1214 of file molecule.py.

**def forcebalance.molecule.Molecule.build\_topology ( self, sn = None, Fac = 1.2 )** A bare-bones implementation of the bond graph capability in the nanoreactor code.

Returns a NetworkX graph that depicts the molecular topology, which might be useful for stuff. Provide, optionally, the frame number used to compute the topology.

Definition at line 1310 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.center\_of\_mass ( self )** Definition at line 1042 of file molecule.py.

**def forcebalance.molecule.Molecule.edit\_qcrems ( self, in\_dict, subcalc = None )** Edit Q-Chem rem variables with a dictionary.

Pass a value of None to delete a rem variable.

Definition at line 1110 of file molecule.py.

**def forcebalance.molecule.Molecule.find\_angles( self )** Return a list of 3-tuples corresponding to all of the angles in the system.

Verified for lysine and tryptophan dipeptide when comparing to TINKER's analyze program.

Definition at line 1458 of file molecule.py.

**def forcebalance.molecule.Molecule.find\_dihedrals( self )** Return a list of 4-tuples corresponding to all of the dihedral angles in the system.

Verified for alanine and tryptophan dipeptide when comparing to TINKER's analyze program.

Definition at line 1484 of file molecule.py.

**def forcebalance.molecule.Molecule.load\_frames( self, fnm )** Definition at line 1101 of file molecule.py.

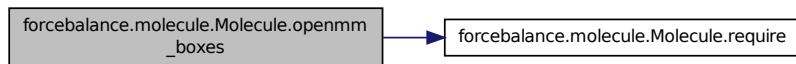
**def forcebalance.molecule.Molecule.measure\_dihedrals( self, i, j, k, l )** Return a series of dihedral angles, given four atom indices numbered from zero.

Definition at line 1507 of file molecule.py.

**def forcebalance.molecule.Molecule.openmm\_boxes( self )** Returns the periodic box vectors in the [Molecule](#) object in a list of OpenMM-compatible boxes, so it is possible to type simulation.context.setPeriodicBoxVectors(Mol.-openmm\_boxes()[0]) or something like that.

Definition at line 1609 of file molecule.py.

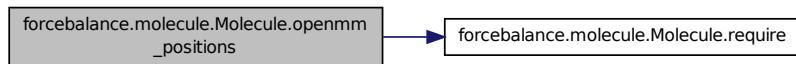
Here is the call graph for this function:



**def forcebalance.molecule.Molecule.openmm\_positions( self )** Returns the Cartesian coordinates in the [Molecule](#) object in a list of OpenMM-compatible positions, so it is possible to type simulation.context.setPositions(Mol.-openmm\_positions()[0]) or something like that.

Definition at line 1592 of file molecule.py.

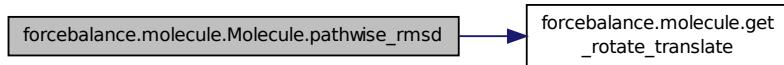
Here is the call graph for this function:



**def forcebalance.molecule.Molecule.pathwise\_rmsd( self )** Find RMSD between frames along path.

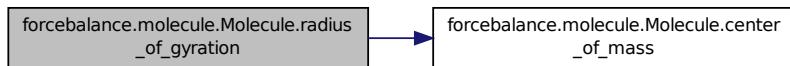
Definition at line 1552 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.radius\_of\_gyration ( self )** Definition at line 1046 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_arc ( self, fnm, kwargs )** Read a TINKER .arc file.

Parameters

in	fnm	The input file name
----	-----	---------------------

Returns

xyzs A list for the XYZ coordinates.

boxes A list of periodic boxes (newer .arc files have these)

resid The residue ID numbers. These are not easy to get!

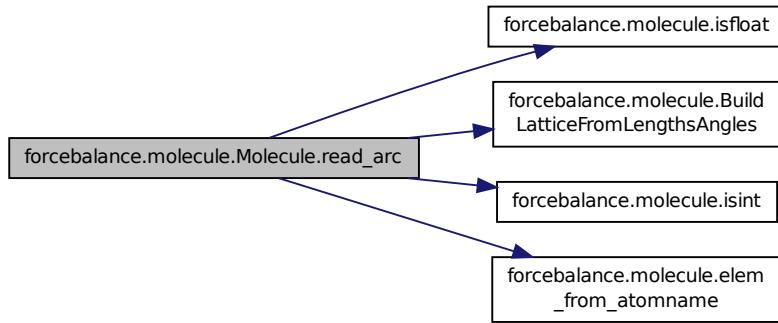
elem A list of chemical elements in the XYZ file

comms A single-element list for the comment.

tinkersuf The suffix that comes after lines in the XYZ coordinates; this is usually topology info

Definition at line 1898 of file molecule.py.

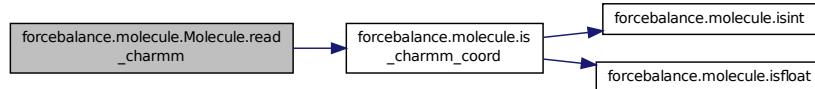
Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_charmm ( *self*, *fnm*, *kwargs* )** Read a CHARMM .cor (or .crd) file.

Definition at line 2051 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_com ( *self*, *fnm*, *kwargs* )** Parse a Gaussian .com file and return a SINGLE-ELEMENT list of xyz coordinates (no multiple file support)

Parameters

in	<i>fnm</i>	The input file name
----	------------	---------------------

Returns

elem A list of chemical elements in the XYZ file

comms A single-element list for the comment.

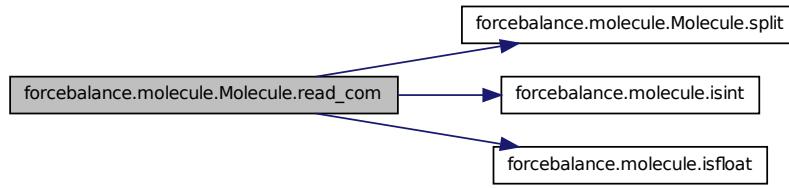
xyzs A single-element list for the XYZ coordinates.

charge The total charge of the system.

mult The spin multiplicity of the system.

Definition at line 1853 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_dcd ( self, fnm, kwargs )** Definition at line 1814 of file molecule.py.

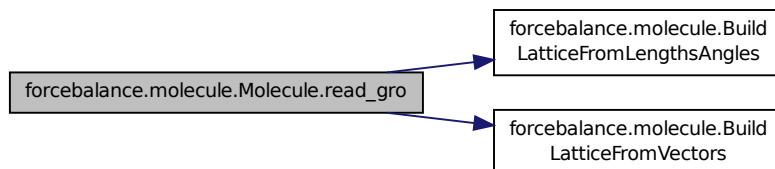
Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_gro ( self, fnm, kwargs )** Read a GROMACS .gro file.

Definition at line 1971 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_mdcrd ( self, fnm, kwargs )** Parse an AMBER .mdcrd file.

This requires at least the number of atoms. This will FAIL for monatomic trajectories (but who the heck makes those?)

## Parameters

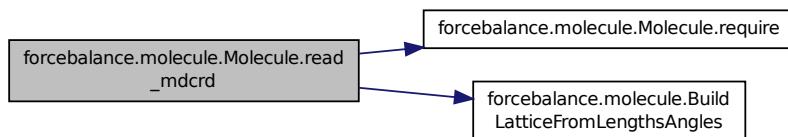
in	<i>fnm</i>	The input file name
----	------------	---------------------

## Returns

*xyzs* A list of XYZ coordinates (number of snapshots times number of atoms)  
*boxes* Boxes (if present.)

Definition at line 1704 of file molecule.py.

Here is the call graph for this function:

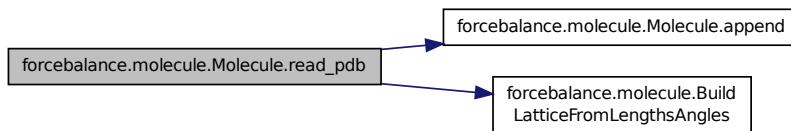


**def forcebalance.molecule.Molecule.read\_mol2 ( self, fnm, kwargs )** Definition at line 1765 of file molecule.py.

**def forcebalance.molecule.Molecule.read\_pdb ( self, fnm, kwargs )** Loads a PDB and returns a dictionary containing its data.

Definition at line 2237 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_qcesp ( self, fnm, kwargs )** Definition at line 2312 of file molecule.py.

Here is the call graph for this function:



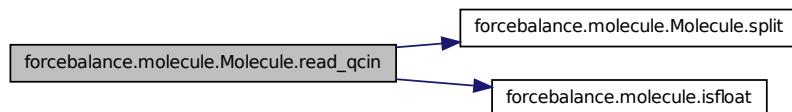
```
def forcebalance.molecule.Molecule.read_qcin( self, fnm, kwargs ) Read a Q-Chem input file.
```

These files can be very complicated, and I can't write a completely general parser for them. It is important to keep our goal in mind:

- 1) The main goal is to convert a trajectory to Q-Chem input files with identical calculation settings.
- 2) When we print the Q-Chem file, we should preserve the line ordering of the 'rem' section, but also be able to add 'rem' options at the end.
- 3) We should accommodate the use case that the Q-Chem file may have follow-up calculations delimited by '@@'.
- 4) We can read in all of the xyz's as a trajectory, but only the Q-Chem settings belonging to the first xyz will be saved.

Definition at line 2120 of file molecule.py.

Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.read_qcout( self, fnm, errok = [], kwargs ) Q-Chem output file reader, adapted for our parser.
```

Q-Chem output files are very flexible and there's no way I can account for all of them. Here's what I am able to account for:

A list of:

- Coordinates
- Energies
- Forces

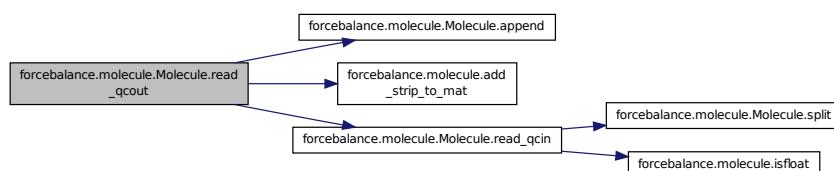
Calling with `errok` will proceed with reading file even if the specified error messages are encountered.

Note that each step in a geometry optimization counts as a frame.

As with all Q-Chem output files, note that successive calculations can have different numbers of atoms.

Definition at line 2344 of file molecule.py.

Here is the call graph for this function:

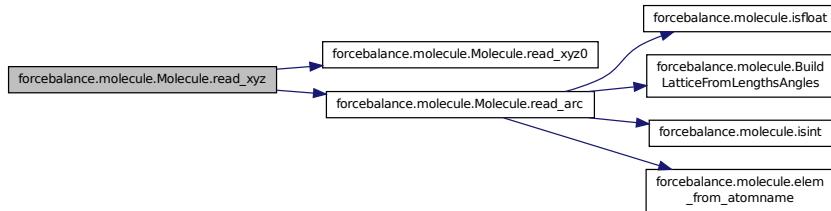


```
def forcebalance.molecule.Molecule.read_qdata( self, fnm, kwargs ) Definition at line 1729 of file molecule.py.
```

```
def forcebalance.molecule.Molecule.read_xyz( self, fnm, kwargs ) .xyz files can be TINKER formatted which  
is why we have the try/except here.
```

Definition at line 1642 of file molecule.py.

Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.read_xyz0( self, fnm, kwargs ) Parse a .xyz file which contains several  
xyz coordinates, and return their elements.
```

Parameters

in	fnm	The input file name
----	-----	---------------------

Returns

elem A list of chemical elements in the XYZ file

comms A list of comments.

xyzs A list of XYZ coordinates (number of snapshots times number of atoms)

Definition at line 1657 of file molecule.py.

```
def forcebalance.molecule.Molecule.ref_rmsd( self, i ) Find RMSD to a reference frame.
```

Definition at line 1569 of file molecule.py.

Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.repair( self, key, klast ) Attempt to repair trivial issues that would  
otherwise break the object.
```

Definition at line 874 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.replace\_peratom ( self, key, orig, want )** Replace all of the data for a certain attribute in the system from orig to want.

Definition at line 1154 of file molecule.py.

**def forcebalance.molecule.Molecule.replace\_peratom\_conditional ( self, key1, cond, key2, orig, want )**

Replace all of the data for a attribute key2 from orig to want, contingent on key1 being equal to cond.

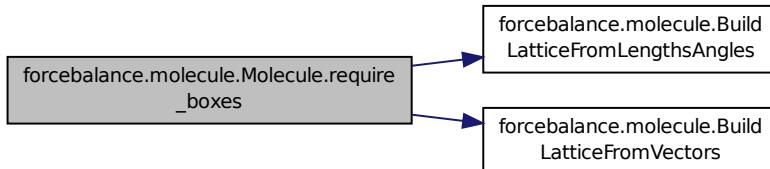
For instance: replace H1 with H2 if resname is SOL.

Definition at line 1165 of file molecule.py.

**def forcebalance.molecule.Molecule.require ( self, args )** Definition at line 987 of file molecule.py.

**def forcebalance.molecule.Molecule.require\_boxes ( self )** Definition at line 2956 of file molecule.py.

Here is the call graph for this function:



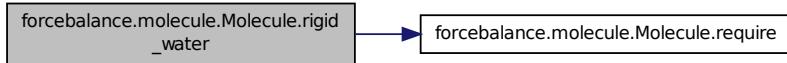
**def forcebalance.molecule.Molecule.require\_resid ( self )** Definition at line 2943 of file molecule.py.

**def forcebalance.molecule.Molecule.require\_resname ( self )** Definition at line 2951 of file molecule.py.

**def forcebalance.molecule.Molecule.rigid\_water ( self )** If one atom is oxygen and the next two are hydrogen, make the water molecule rigid.

Definition at line 1058 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.split ( self, fnm = None, ftype = None, method = "chunks", num = None )** Split the molecule object into a number of separate files (chunks), either by specifying the number of frames per chunk or the number of chunks.

Only relevant for "trajectories". The type of file may be specified; if they aren't specified then the original file type is used.

The output file names are [name].[numbers].[extension] where [name] can be specified by passing 'fnm' or taken from the object's 'fnm' attribute by default. [numbers] are integers ranging from the lowest to the highest chunk number, prepended by zeros.

If the number of chunks / frames is not specified, then one file is written for each frame.

Returns

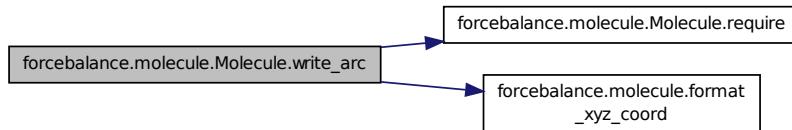
fnms A list of the file names that were written.

Definition at line 1632 of file molecule.py.

**def forcebalance.molecule.Molecule.write ( self, fnm = None, ftype = None, append = False, select = None, kwargs )** Definition at line 1002 of file molecule.py.

**def forcebalance.molecule.Molecule.write\_arc ( self, select, kwargs )** Definition at line 2710 of file molecule.py.

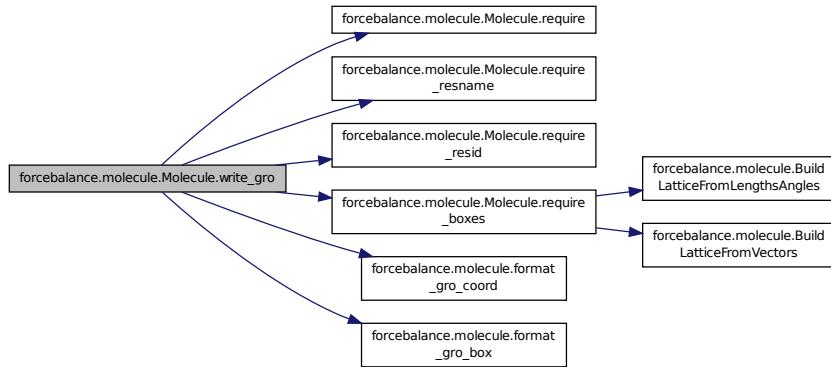
Here is the call graph for this function:



**def forcebalance.molecule.Molecule.write\_dcd ( self, select, kwargs )** Definition at line 2760 of file molecule.py.

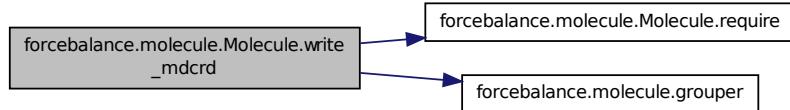
**def forcebalance.molecule.Molecule.write\_gro ( self, select, kwargs )** Definition at line 2725 of file molecule.py.

Here is the call graph for this function:



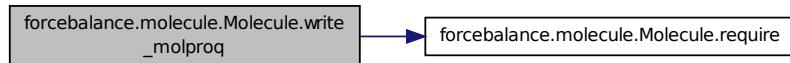
**def forcebalance.molecule.Molecule.write\_mdcrd ( self, select, kwargs )** Definition at line 2699 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.write\_molproq ( self, select, kwargs )** Definition at line 2687 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.write\_pdb ( self, select, kwargs )** Save to a PDB.

Copied wholesale from MSMBuilder.

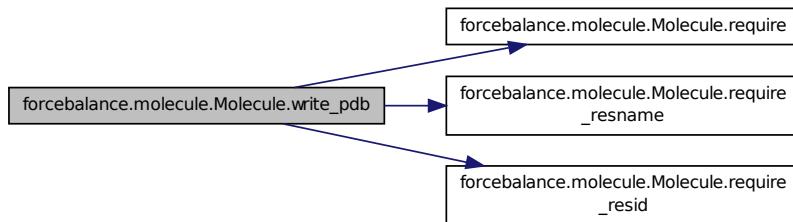
**COLUMNS TYPE FIELD DEFINITION** 7-11 int serial Atom serial number. 13-16 string name Atom name. 17 string altLoc Alternate location indicator. 18-20 (17-21 KAB) string resName Residue name. 22 string chainID Chain identifier. 23-26 int resSeq Residue sequence number. 27 string iCode Code for insertion of residues. 31-38 float x Orthogonal coordinates for X in Angstroms. 39-46 float y Orthogonal coordinates for Y in Angstroms. 47-54 float z Orthogonal coordinates for Z in Angstroms. 55-60 float occupancy Occupancy. 61-66 float tempFactor Temperature factor. 73-76 string segID Segment identifier, left-justified. 77-78 string element Element symbol, right-justified. 79-80 string charge Charge on the atom.

CRYST1 line, added by Lee-Ping

**COLUMNS TYPE FIELD DEFINITION** 7-15 float a a (Angstroms). 16-24 float b b (Angstroms). 25-33 float c c (Angstroms). 34-40 float alpha alpha (degrees). 41-47 float beta beta (degrees). 48-54 float gamma gamma (degrees). 56-66 string sGroup Space group. 67-70 int z Z value.

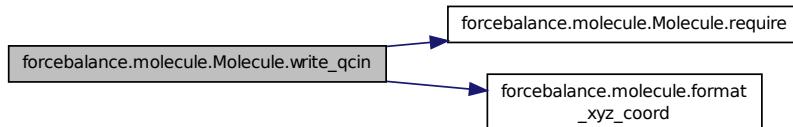
Definition at line 2816 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.write\_qcin ( self, select, kwargs )** Definition at line 2611 of file molecule.py.

Here is the call graph for this function:



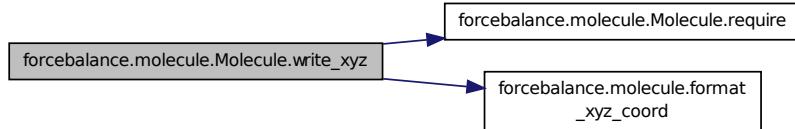
**def forcebalance.molecule.Molecule.write\_qdata ( self, select, kwargs )** Text quantum data format. Definition at line 2922 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.write\_xyz( self, select, kwargs )** Definition at line 2676 of file molecule.py.

Here is the call graph for this function:



#### 8.45.4 Member Data Documentation

**forcebalance.molecule.Molecule.boxes** Definition at line 3004 of file molecule.py.

**forcebalance.molecule.Molecule.built\_bonds** Definition at line 944 of file molecule.py.

**forcebalance.molecule.Molecule.comms** Read in stuff if we passed in a file name, otherwise return an empty instance.

Try to determine from the file name using the extension. Actually read the file. Set member variables. Create a list of comment lines if we don't already have them from reading the file.

Definition at line 965 of file molecule.py.

**forcebalance.molecule.Molecule.Data** Definition at line 948 of file molecule.py.

**forcebalance.molecule.Molecule.fout** Fill in comments.

I needed to add in this line because the DCD writer requires the file name, but the other methods don't.

Definition at line 1013 of file molecule.py.

**forcebalance.molecule.Molecule.Funnel** A funnel dictionary that takes redundant file types and maps them down to a few.

Definition at line 928 of file molecule.py.

**forcebalance.molecule.Molecule.molecules** Definition at line 976 of file molecule.py.

**forcebalance.molecule.Molecule.positive\_resid** Creates entries like 'gromacs' : 'gromacs' and 'xyz' : 'xyz' in the Funnel.

Definition at line 943 of file molecule.py.

**forcebalance.molecule.Molecule.Read\_Tab** The table of file readers.

Definition at line 903 of file molecule.py.

**forcebalance.molecule.Molecule.resid** Definition at line 2947 of file molecule.py.

**forcebalance.molecule.Molecule.resname** Definition at line 2954 of file molecule.py.

**forcebalance.molecule.Molecule.topology** Make sure the comment line isn't too long for i in range(len(self.comms)): self.comms[i] = self.comms[i][:100] if len(self.comms[i]) > 100 else self.comms[i] Attempt to build the topology for small systems.

:)

Definition at line 975 of file molecule.py.

**forcebalance.molecule.Molecule.Write\_Tab** The table of file writers.

Definition at line 917 of file molecule.py.

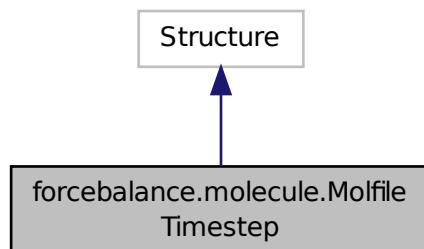
The documentation for this class was generated from the following file:

- [molecule.py](#)

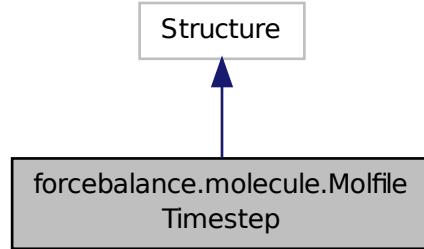
## 8.46 forcebalance.molecule.MolfileTimestep Class Reference

Wrapper for the timestep C structure used in molfile plugins.

Inheritance diagram for forcebalance.molecule.MolfileTimestep:



Collaboration diagram for forcebalance.molecule.MolfileTimestep:



#### 8.46.1 Detailed Description

Wrapper for the timestep C structure used in molfile plugins.

Definition at line 499 of file molecule.py.

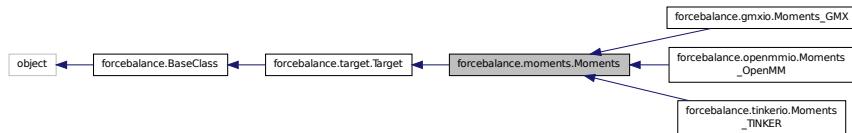
The documentation for this class was generated from the following file:

- [molecule.py](#)

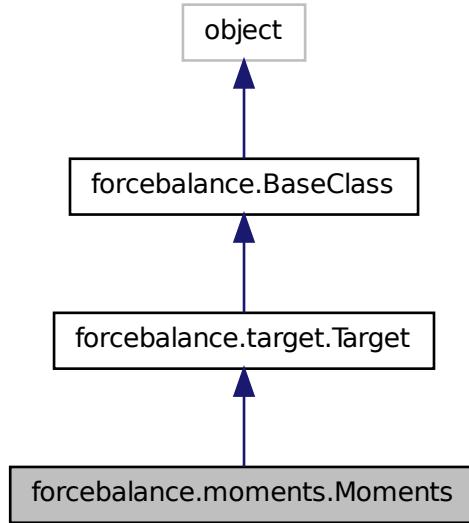
### 8.47 forcebalance.moments.Moments Class Reference

Subclass of Target for fitting force fields to multipole moments (from experiment or theory).

Inheritance diagram for forcebalance.moments.Moments:



Collaboration diagram for forcebalance.moments.Moments:



## Public Member Functions

- def `__init__`  
*Initialization.*
- def `read_reference_data`  
*Read the reference data from a file.*
- def `indicate`  
*Print qualitative indicator.*
- def `unpack_moments`
- def `get`  
*Evaluate objective function.*
- def `get_X`  
*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`  
*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`  
*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`  
*Computes the objective function contribution and its gradient.*
- def `get_H`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*

- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `denoms`
- `mfnm`  
*The mdata.txt file that contains the moments.*
- `ref_moments`  
*Dictionary of reference multipole moments.*
- `engine`  
*Read in the reference data.*
- `na`  
*Number of atoms.*
- `ref_eigvals`
- `ref_eigvecs`
- `calc_moments`
- `objective`
- `rd`  
*Root directory of the whole project.*
- `pgrad`  
*Iteration where we turn on zero-gradient skipping.*
- `tempbase`  
*Relative directory of target.*
- `tempdir`
- `rundir`  
*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- `FF`  
*Need the forcefield (here for now)*

- **xct**  
Counts how often the objective function was computed.
- **gct**  
Counts how often the gradient was computed.
- **hct**  
Counts how often the Hessian was computed.
- **read\_indicate**  
Whether to read indicate.log from file when restarting an aborted run.
- **write\_indicate**  
Whether to write indicate.log at every iteration (true for all but remote.)
- **read\_objective**  
Whether to read objective.p from file when restarting an aborted run.
- **write\_objective**  
Whether to write objective.p at every iteration (true for all but remote.)
- **verbose\_options**
- **PrintOptionDict**

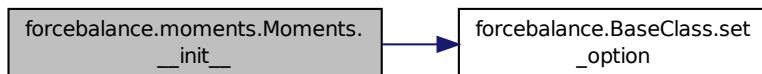
#### 8.47.1 Detailed Description

Subclass of Target for fitting force fields to multipole moments (from experiment or theory).  
Currently Tinker is supported.  
Definition at line 30 of file moments.py.

#### 8.47.2 Constructor & Destructor Documentation

**def forcebalance.moments.Moments.\_\_init\_\_( self, options, tgt\_opts, forcefield )** Initialization.

Definition at line 35 of file moments.py.  
Here is the call graph for this function:



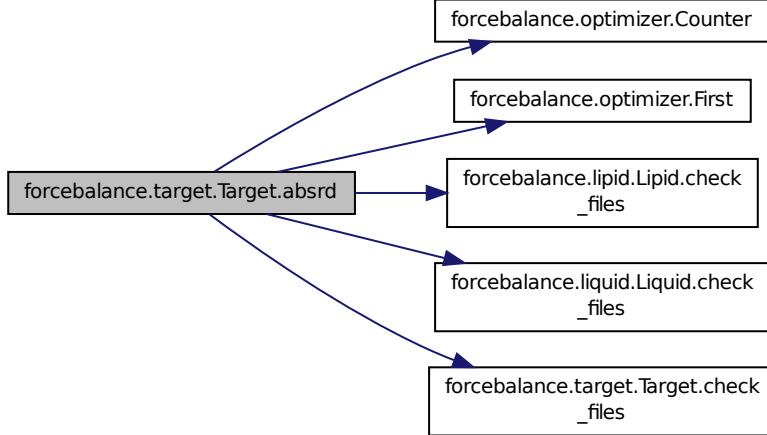
#### 8.47.3 Member Function Documentation

**def forcebalance.BaseClass.\_\_setattr\_\_( self, key, value ) [inherited]** Definition at line 28 of file \_\_init\_\_.py.

**def forcebalance.target.Target.absrd( self, inum = None ) [inherited]** Supply the correct directory specified by user's "read" option.

Definition at line 393 of file target.py.

Here is the call graph for this function:



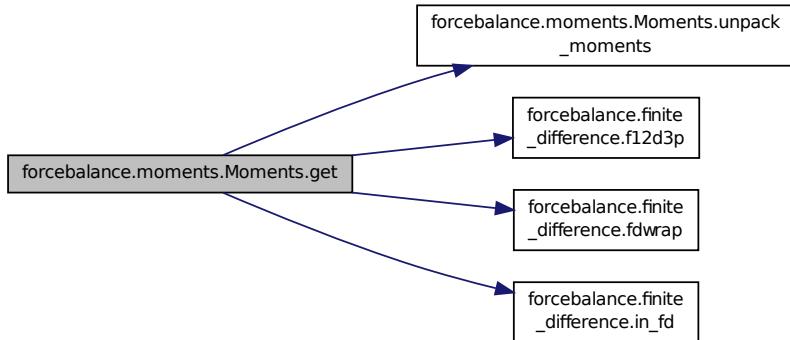
**def forcebalance.target.Target.check\_files( self, there ) [inherited]** Check this directory for the presence of readable files when the 'read' option is set.

Definition at line 364 of file target.py.

**def forcebalance.moments.Moments.get( self, mvals, AGrad = False, AHess = False )** Evaluate objective function.

Definition at line 171 of file moments.py.

Here is the call graph for this function:



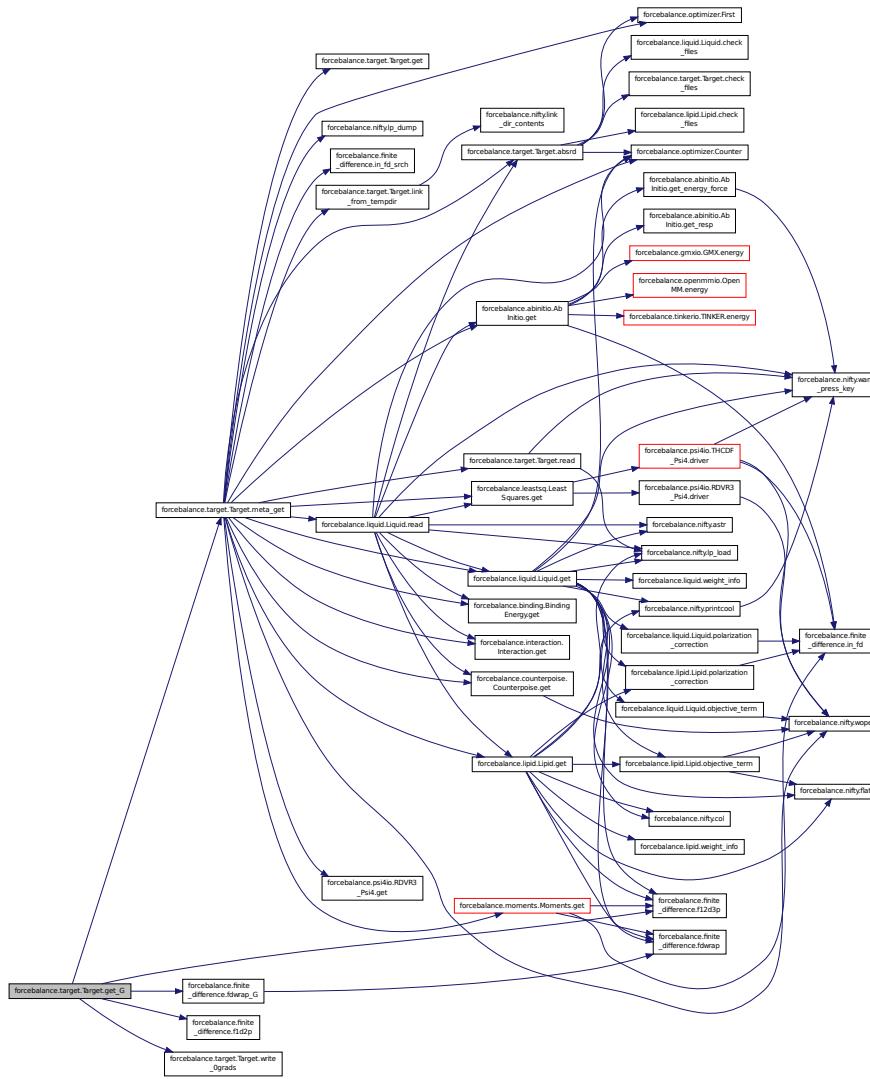
**def forcebalance.target.Target.get\_G( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



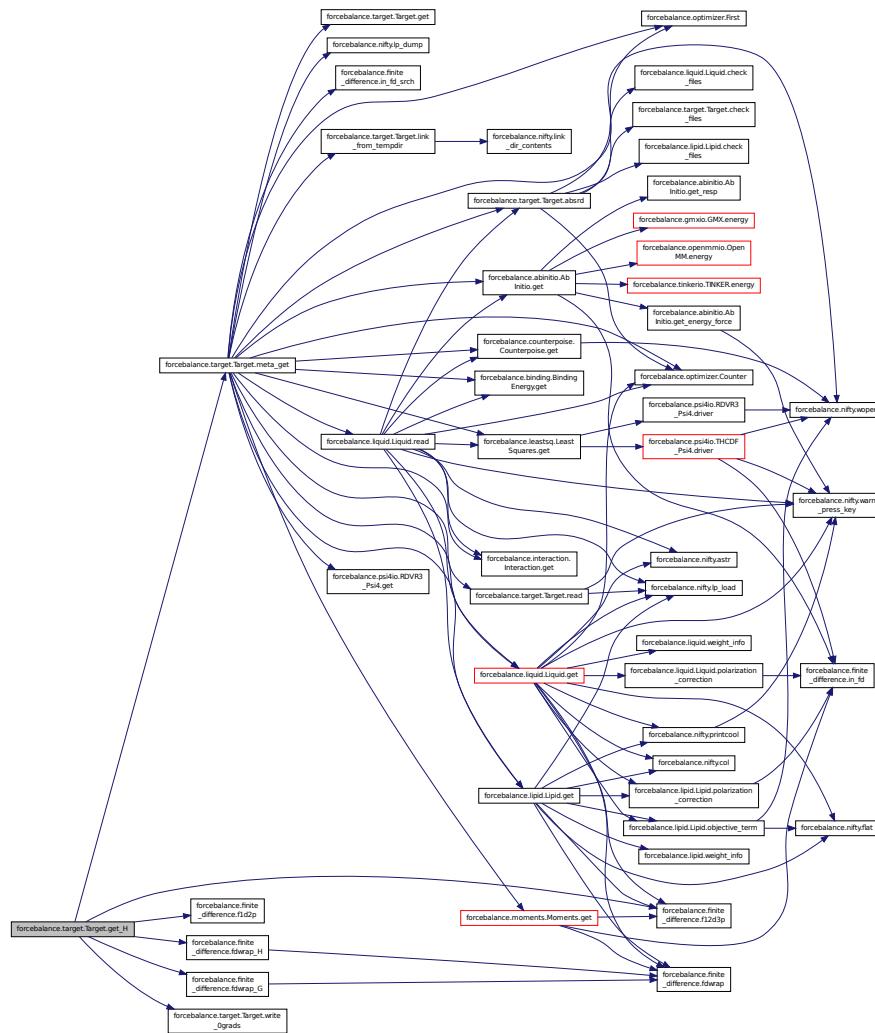
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

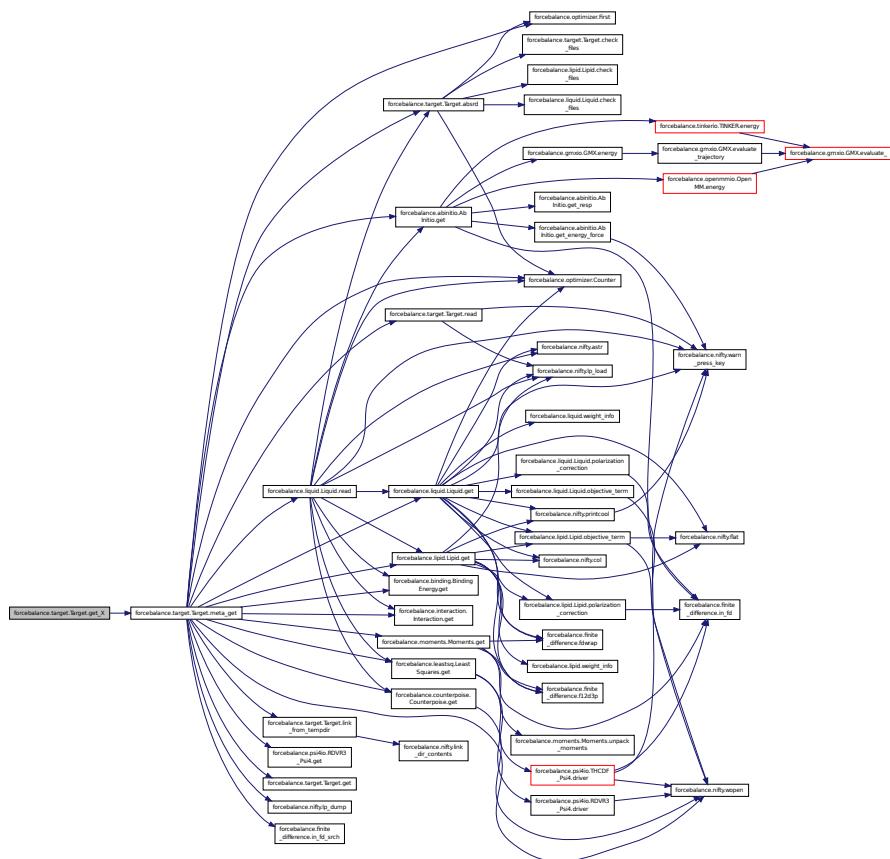
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

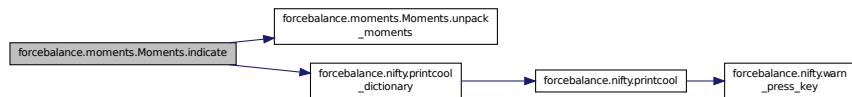
Here is the call graph for this function:



**def forcebalance.moments.Moments.indicate ( self )** Print qualitative indicator.

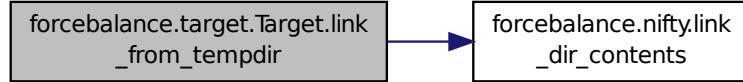
Definition at line 139 of file moments.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

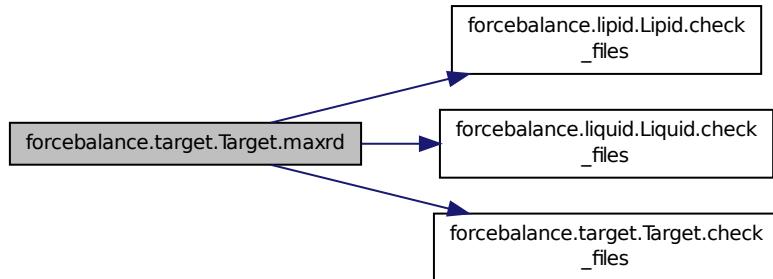
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

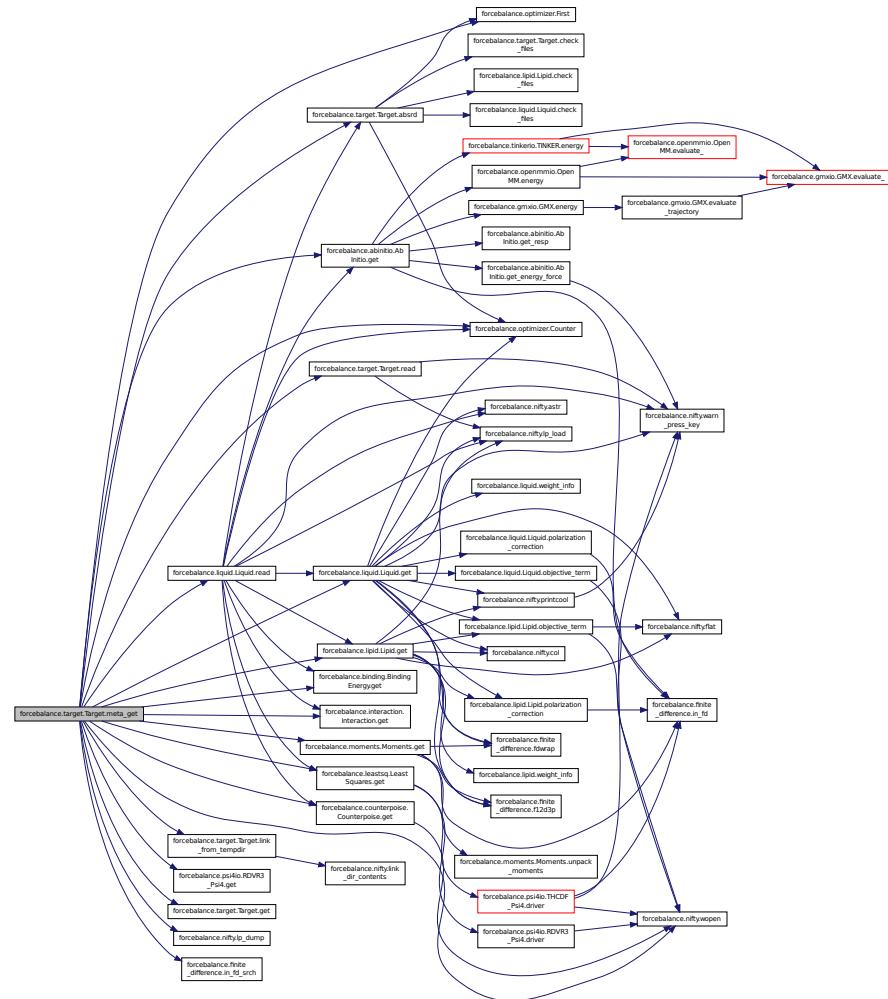


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

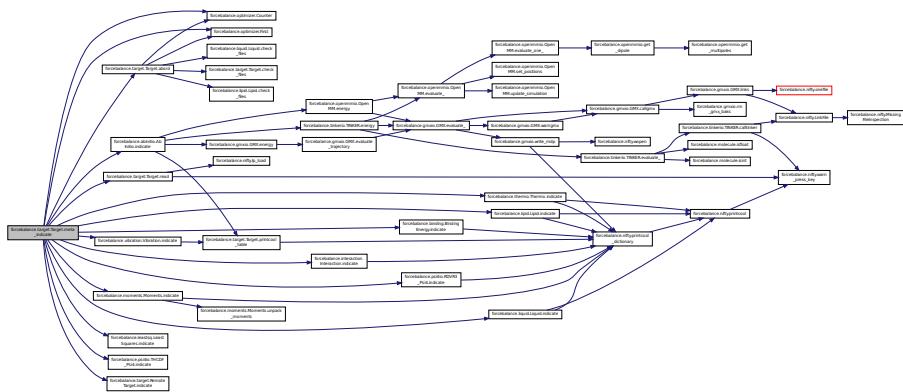
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

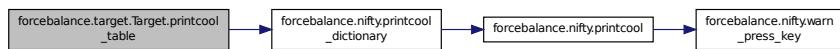
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

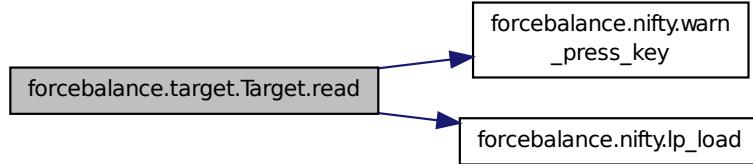
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.moments.Moments.read\_reference\_data ( self )** Read the reference data from a file.

Definition at line 70 of file moments.py.

**def forcebalance.target.Target.refresh\_temp\_directory ( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

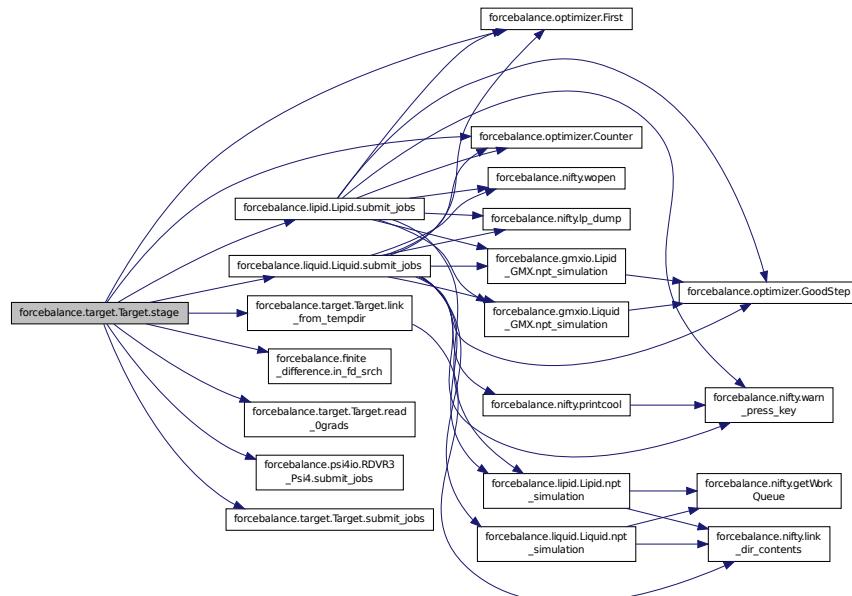
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



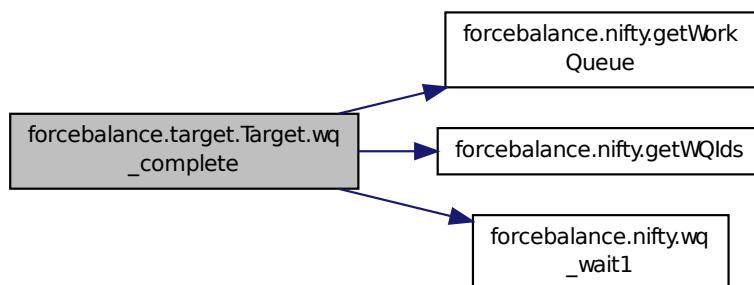
```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.moments.Moments.unpack_moments ( self, moment_dict ) Definition at line 165 of file moments.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.47.4 Member Data Documentation

**forcebalance.moments.Moments.calc\_moments** Definition at line 198 of file moments.py.

**forcebalance.moments.Moments.denoms** Definition at line 48 of file moments.py.

**forcebalance.moments.Moments.engine** Read in the reference data.

Build keyword dictionaries to pass to engine. Create engine object.

Definition at line 66 of file moments.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.moments.Moments.mfnm** The mdata.txt file that contains the moments.

Definition at line 57 of file moments.py.

**forcebalance.moments.Moments.na** Number of atoms.

Definition at line 72 of file moments.py.

**forcebalance.moments.Moments.objective** Definition at line 199 of file moments.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [**inherited**] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.moments.Moments.ref\_eigvals** Definition at line 73 of file moments.py.

**forcebalance.moments.Moments.ref\_eigvecs** Definition at line 74 of file moments.py.

**forcebalance.moments.Moments.ref\_moments** Dictionary of reference multipole moments.

Definition at line 59 of file moments.py.

**forcebalance.target.Target.rundir** [**inherited**] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number  
The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [**inherited**] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [**inherited**] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate** [**inherited**] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

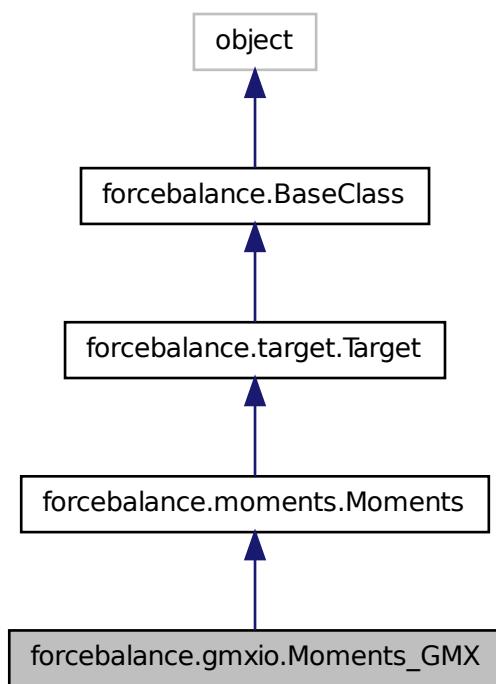
The documentation for this class was generated from the following file:

- [moments.py](#)

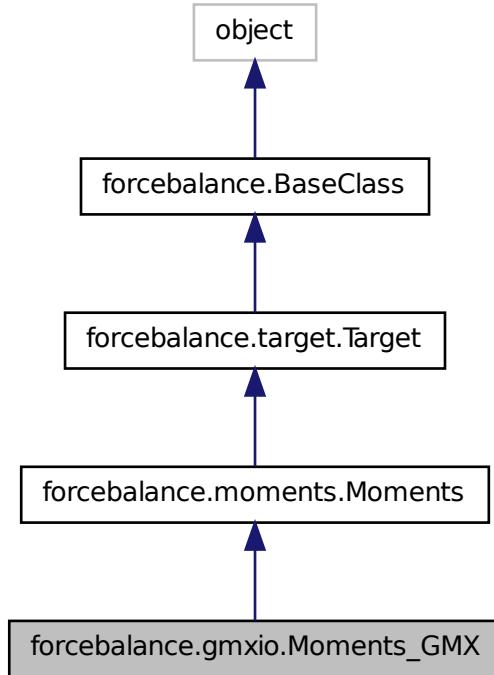
## 8.48 forcebalance.gmxio.Moments\_GMX Class Reference

Multipole moment matching using GROMACS.

Inheritance diagram for forcebalance.gmxio.Moments\_GMX:



Collaboration diagram for forcebalance.gmxio.Moments\_GMX:



## Public Member Functions

- def `__init__`
- def `read_reference_data`

*Read the reference data from a file.*
- def `indicate`

*Print qualitative indicator.*
- def `unpack_moments`
- def `get`

*Evaluate objective function.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*
- def `get_H`

*Computes the objective function contribution and its gradient / Hessian.*

- def `link_from_tempdir`  
 Back up the temporary directory if desired, delete it and then create a new one.
- def `refresh_temp_directory`  
 Check this directory for the presence of readable files when the 'read' option is set.
- def `read`  
 Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.
- def `absrd`  
 Supply the correct directory specified by user's "read" option.
- def `maxrd`  
 Supply the latest existing temp-directory containing valid data.
- def `meta_indicate`  
 Wrap around the indicate function, so it can print to screen and also to a file.
- def `meta_get`  
 Wrapper around the get function.
- def `submit_jobs`
- def `stage`  
 Stages the directory for the target, and then launches Work Queue processes if any.
- def `wq_complete`  
 This method determines whether the Work Queue tasks for the current target have completed.
- def `printcool_table`  
 Print target information in an organized table format.
- def `__setattr__`
- def `set_option`

## Public Attributes

- `engine_`  
 Default file names for coordinates and key file.
- `denoms`
- `mfnm`  
 The mdata.txt file that contains the moments.
- `ref_moments`  
 Dictionary of reference multipole moments.
- `engine`  
 Read in the reference data.
- `na`  
 Number of atoms.
- `ref_eigvals`
- `ref_eigvecs`
- `calc_moments`
- `objective`
- `rd`  
 Root directory of the whole project.
- `pgrad`  
 Iteration where we turn on zero-gradient skipping.
- `tempbase`  
 Relative directory of target.

- `tempdir`  
`self.tempdir = os.path.join('temp',self.name)` The directory in which the simulation is running - this can be updated.
- `FF`  
`Need the forcefield (here for now)`
- `xct`  
`Counts how often the objective function was computed.`
- `gct`  
`Counts how often the gradient was computed.`
- `hct`  
`Counts how often the Hessian was computed.`
- `read_indicate`  
`Whether to read indicate.log from file when restarting an aborted run.`
- `write_indicate`  
`Whether to write indicate.log at every iteration (true for all but remote.)`
- `read_objective`  
`Whether to read objective.p from file when restarting an aborted run.`
- `write_objective`  
`Whether to write objective.p at every iteration (true for all but remote.)`
- `verbose_options`
- `PrintOptionDict`

#### 8.48.1 Detailed Description

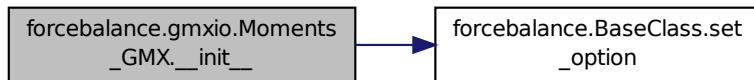
Multipole moment matching using GROMACS.

Definition at line 1481 of file gmxio.py.

#### 8.48.2 Constructor & Destructor Documentation

`def forcebalance.gmxio.Moments_GMX.__init__ ( self, options, tgt_opts, forcefield )` Definition at line 1482 of file gmxio.py.

Here is the call graph for this function:



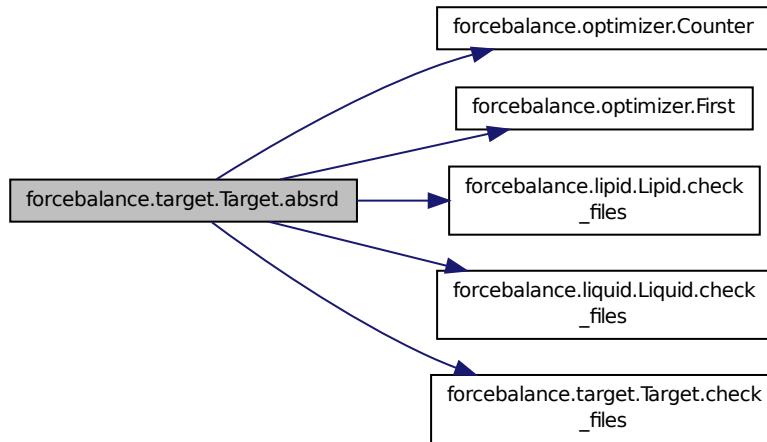
#### 8.48.3 Member Function Documentation

`def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited]` Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory  
specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



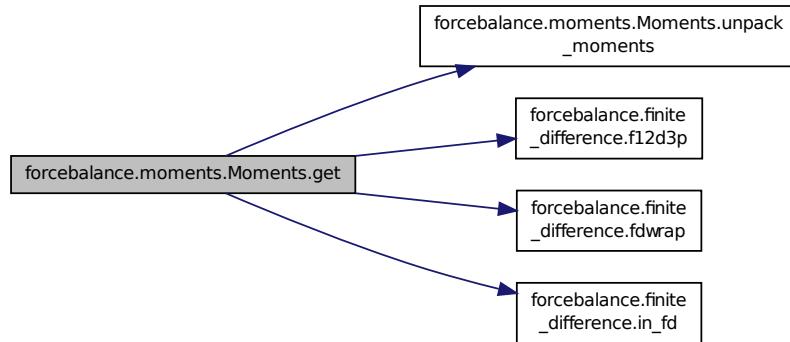
```
def forcebalance.target.Target.check_files ( self, there ) [inherited] Check this directory for the presence  
of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.moments.Moments.get ( self, mvals, AGrad=False, AHess=False ) [inherited]  
Evaluate objective function.
```

Definition at line 171 of file moments.py.

Here is the call graph for this function:



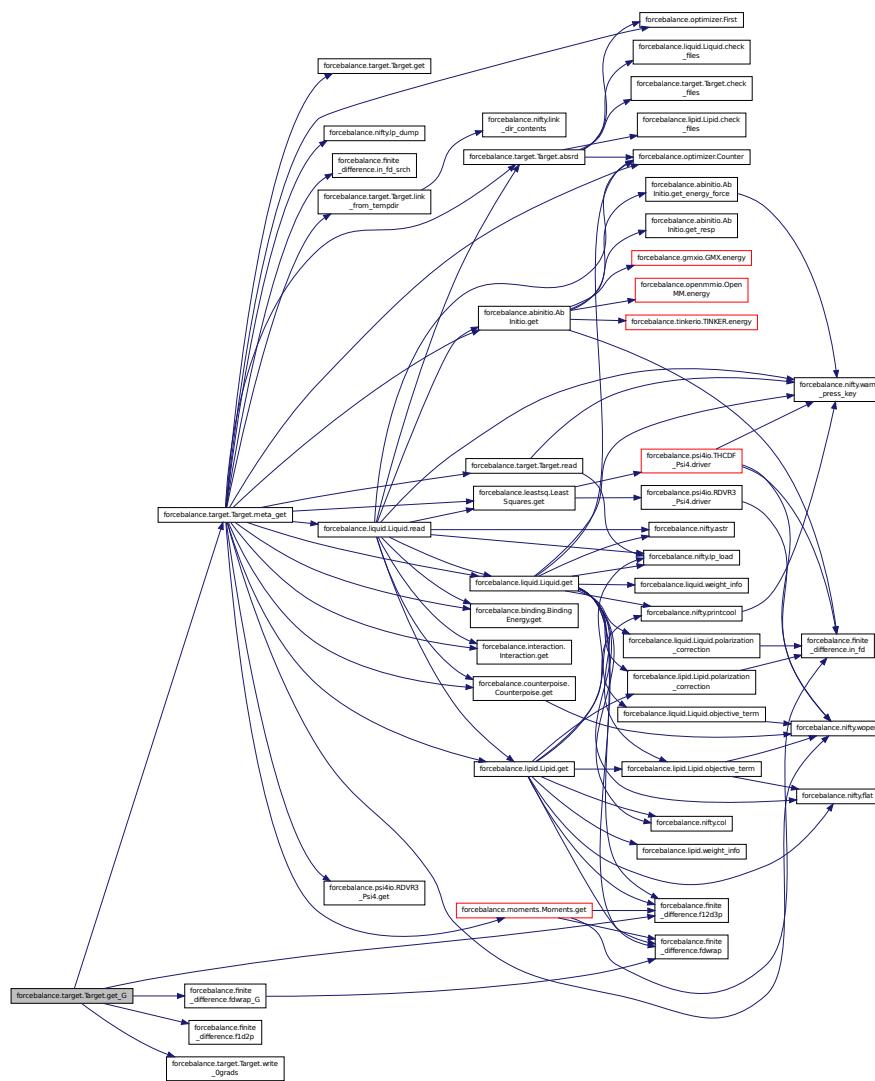
```
def forcebalance.target.Target.get_G ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.get_H ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient / Hessian.
```

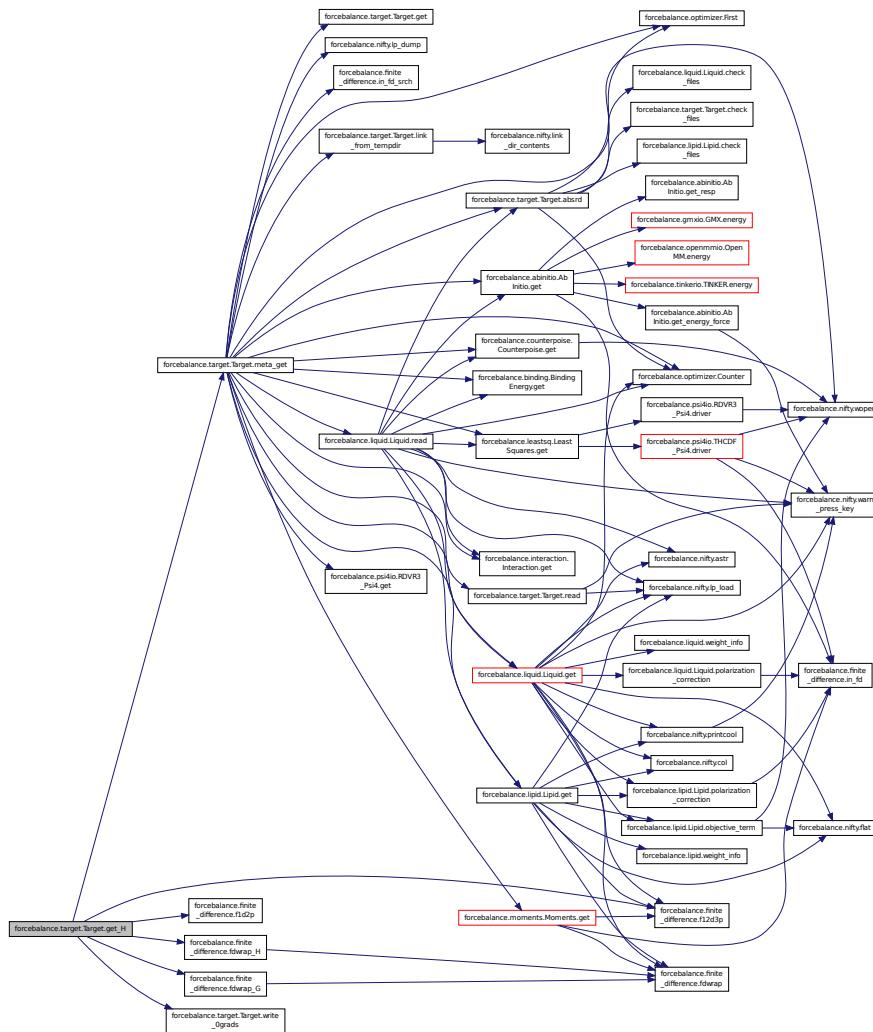
First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through

the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

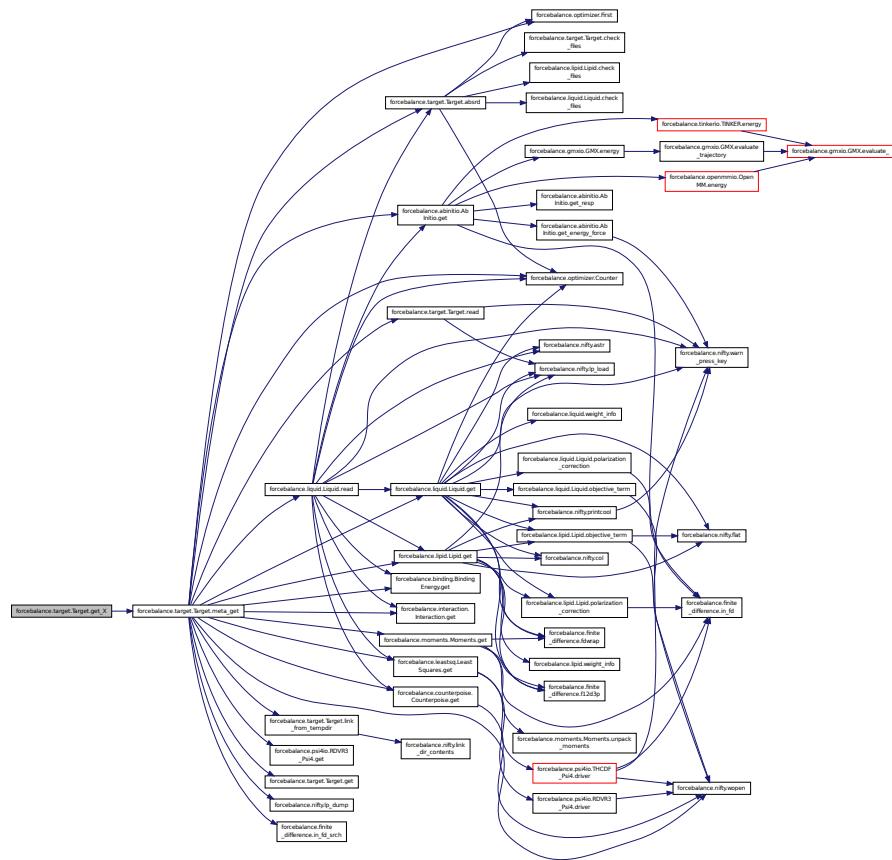
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

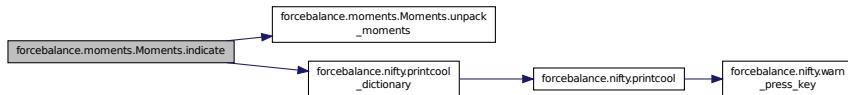
Here is the call graph for this function:



**def forcebalance.moments.Moments.indicate ( self ) [inherited]** Print qualitative indicator.

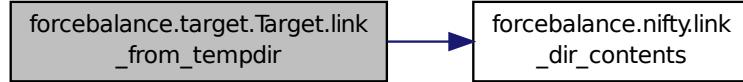
Definition at line 139 of file moments.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

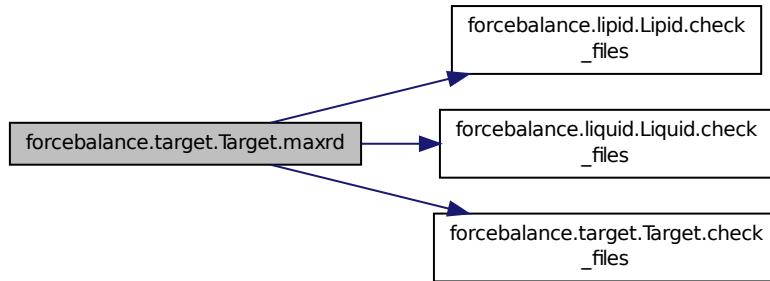
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

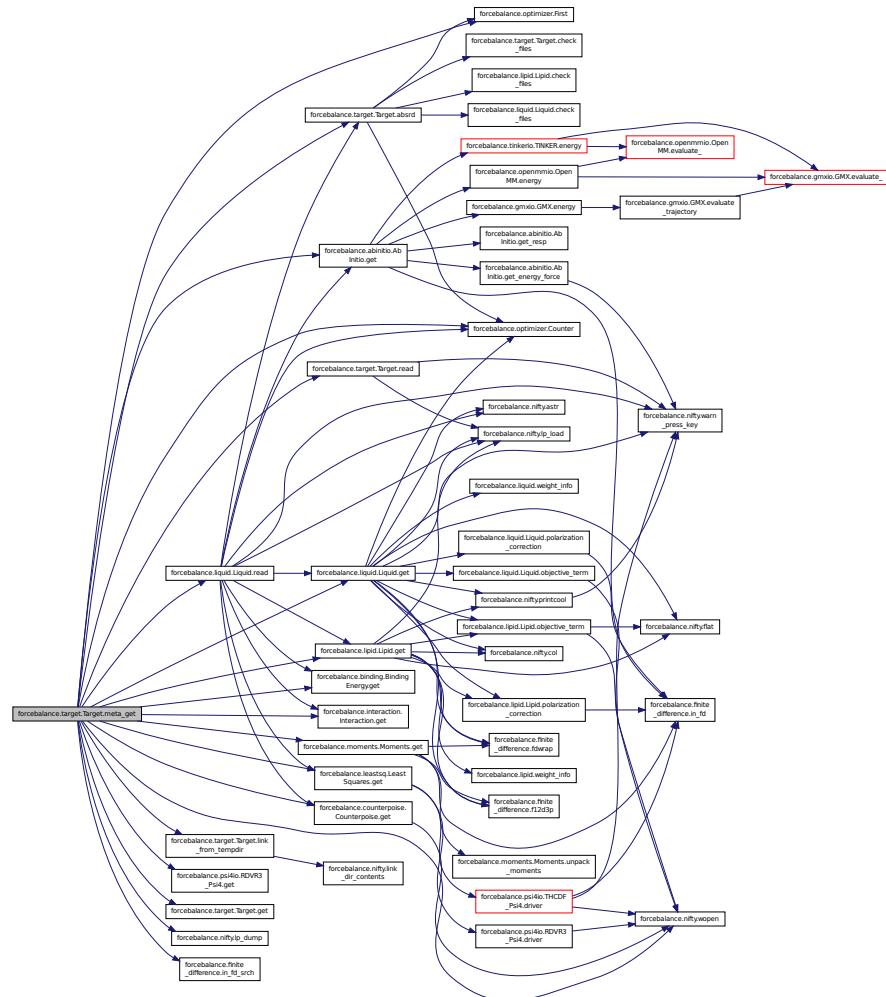


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

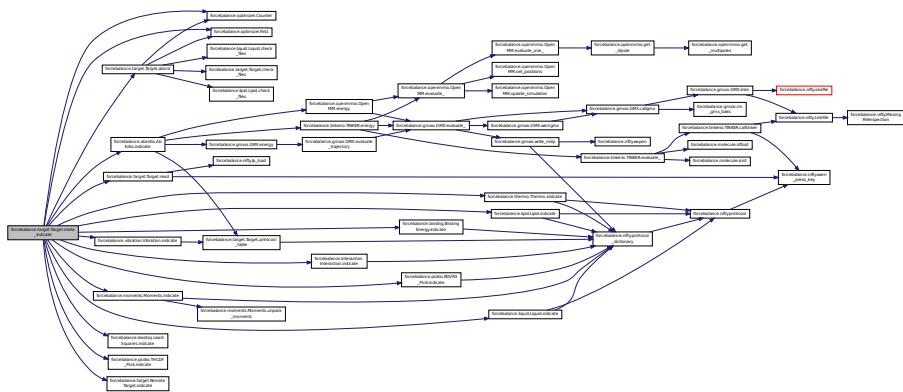
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict ([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

The input should be something like:

### Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

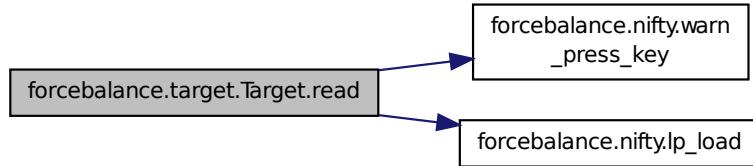
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.moments.Moments.read\_reference\_data ( self ) [inherited]** Read the reference data from a file.

Definition at line 70 of file moments.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

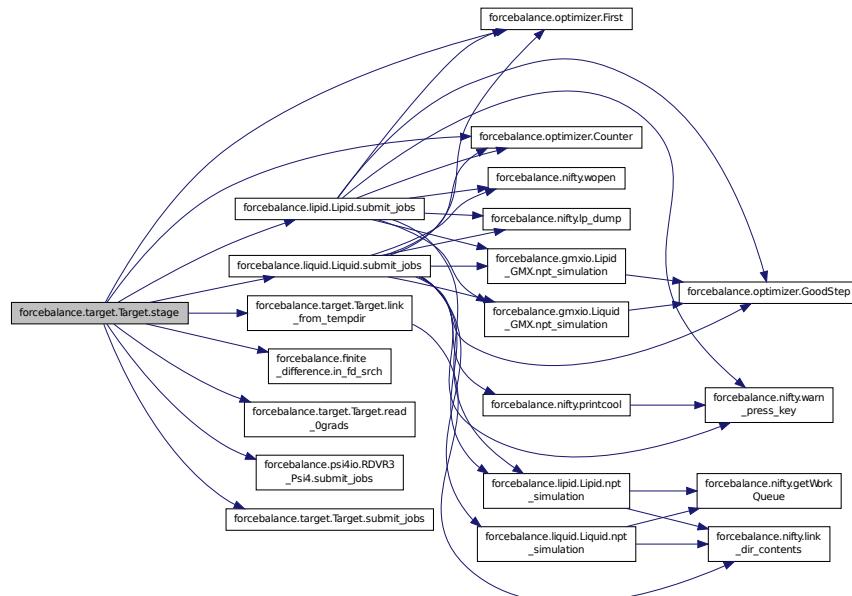
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src\_key, dest\_key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



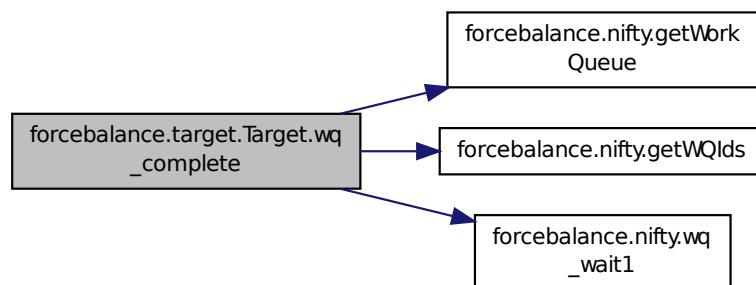
```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.moments.Moments.unpack_moments ( self, moment_dict ) [inherited] Definition at
line 165 of file moments.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work
Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.48.4 Member Data Documentation

**forcebalance.moments.Moments.calc\_moments [inherited]** Definition at line 198 of file moments.py.

**forcebalance.moments.Moments.denoms [inherited]** Definition at line 48 of file moments.py.

**forcebalance.moments.Moments.engine [inherited]** Read in the reference data.

Build keyword dictionaries to pass to engine. Create engine object.

Definition at line 66 of file moments.py.

**forcebalance.gmxio.Moments\_GMX.engine** Default file names for coordinates and key file.

Definition at line 1487 of file gmxio.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.moments.Moments.mfnm [inherited]** The mdata.txt file that contains the moments.

Definition at line 57 of file moments.py.

**forcebalance.moments.Moments.na [inherited]** Number of atoms.

Definition at line 72 of file moments.py.

**forcebalance.moments.Moments.objective [inherited]** Definition at line 199 of file moments.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [inherited] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.moments.Moments.ref\_eigvals** [inherited] Definition at line 73 of file moments.py.

**forcebalance.moments.Moments.ref\_eigvecs** [inherited] Definition at line 74 of file moments.py.

**forcebalance.moments.Moments.ref\_moments** [inherited] Dictionary of reference multipole moments.

Definition at line 59 of file moments.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number  
The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate** [inherited] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

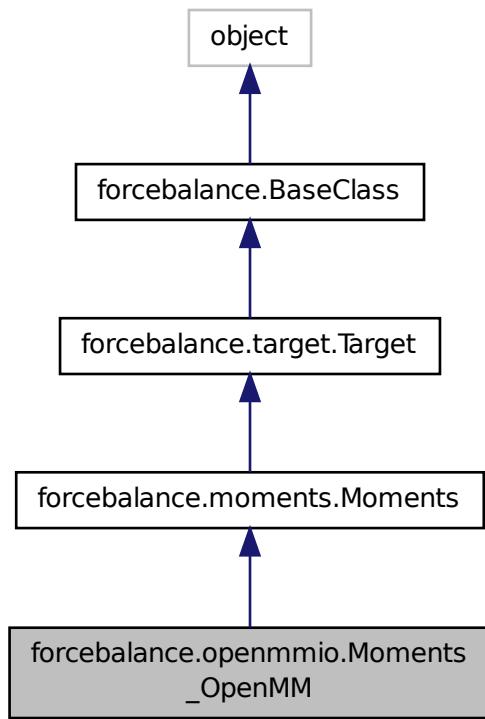
The documentation for this class was generated from the following file:

- [gmxio.py](#)

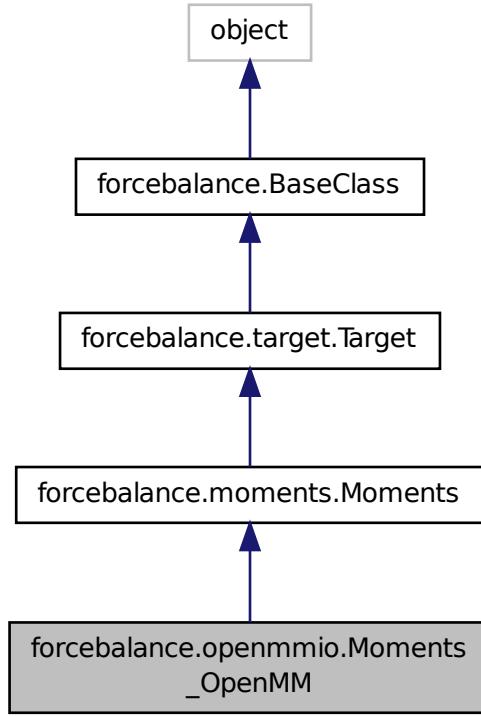
## 8.49 forcebalance.openmmio.Moments\_OpenMM Class Reference

Multipole moment matching using [OpenMM](#).

Inheritance diagram for forcebalance.openmmio.Moments\_OpenMM:



Collaboration diagram for forcebalance.openmmio.Moments\_OpenMM:



### Public Member Functions

- def `_init_`
- def `read_reference_data`

*Read the reference data from a file.*
- def `indicate`

*Print qualitative indicator.*
- def `unpack_moments`
- def `get`

*Evaluate objective function.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*
- def `get_H`

- def `link_from_tempdir`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool.table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `engine_`  
*Default file names for coordinates and key file.*
- `denoms`
- `mfnm`  
*The mdata.txt file that contains the moments.*
- `ref_moments`  
*Dictionary of reference multipole moments.*
- `engine`  
*Read in the reference data.*
- `na`  
*Number of atoms.*
- `ref_eigvals`
- `ref_eigvecs`
- `calc_moments`
- `objective`
- `rd`  
*Root directory of the whole project.*
- `pgrad`  
*Iteration where we turn on zero-gradient skipping.*
- `tempbase`

- Relative directory of target.*
- `tempdir`  
`self.tempdir = os.path.join('temp',self.name)` *The directory in which the simulation is running - this can be updated.*
  - `FF`  
*Need the forcefield (here for now)*
  - `xct`  
*Counts how often the objective function was computed.*
  - `gct`  
*Counts how often the gradient was computed.*
  - `hct`  
*Counts how often the Hessian was computed.*
  - `read_indicate`  
*Whether to read indicate.log from file when restarting an aborted run.*
  - `write_indicate`  
*Whether to write indicate.log at every iteration (true for all but remote.)*
  - `read_objective`  
*Whether to read objective.p from file when restarting an aborted run.*
  - `write_objective`  
*Whether to write objective.p at every iteration (true for all but remote.)*
  - `verbose_options`
  - `PrintOptionDict`

#### 8.49.1 Detailed Description

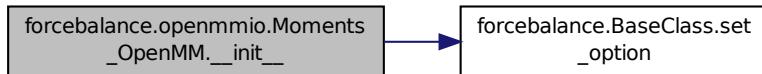
Multipole moment matching using [OpenMM](#).

Definition at line 1202 of file openmmio.py.

#### 8.49.2 Constructor & Destructor Documentation

**def forcebalance.openmmio.Moments\_OpenMM.\_\_init\_\_ ( `self`, `options`, `tgt_opts`, `forcefield` )** Definition at line 1203 of file openmmio.py.

Here is the call graph for this function:



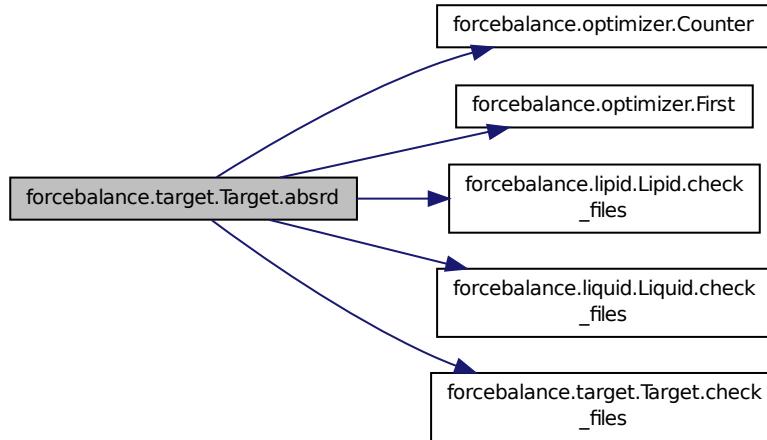
#### 8.49.3 Member Function Documentation

**def forcebalance.BaseClass.\_\_setattr\_\_ ( `self`, `key`, `value` ) [inherited]** Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory  
specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



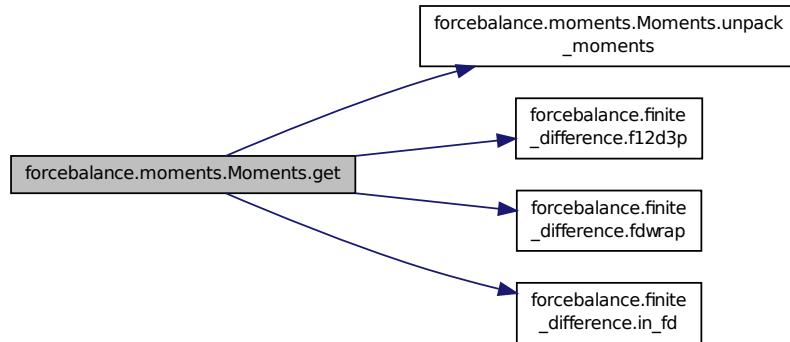
```
def forcebalance.target.Target.check_files ( self, there ) [inherited] Check this directory for the presence  
of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.moments.Moments.get ( self, mvals, AGrad=False, AHess=False ) [inherited]  
Evaluate objective function.
```

Definition at line 171 of file moments.py.

Here is the call graph for this function:



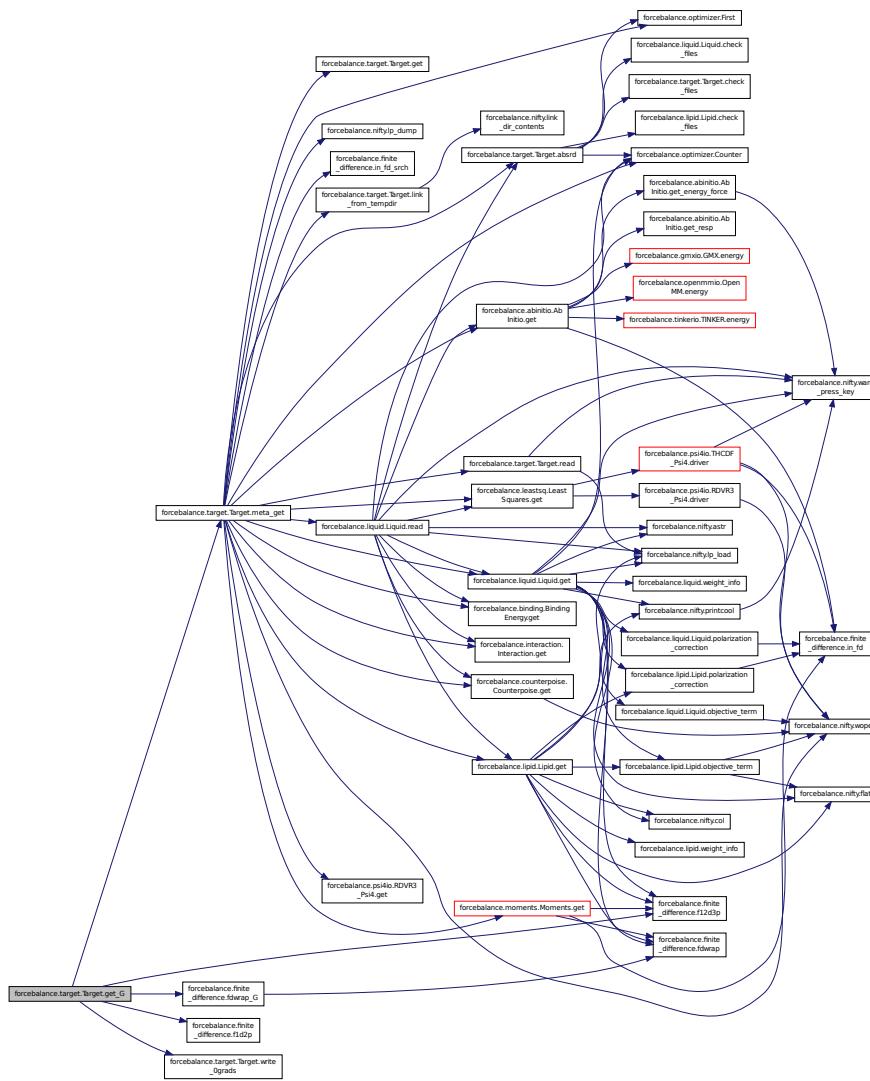
```
def forcebalance.target.Target.get_G ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.get_H ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient / Hessian.
```

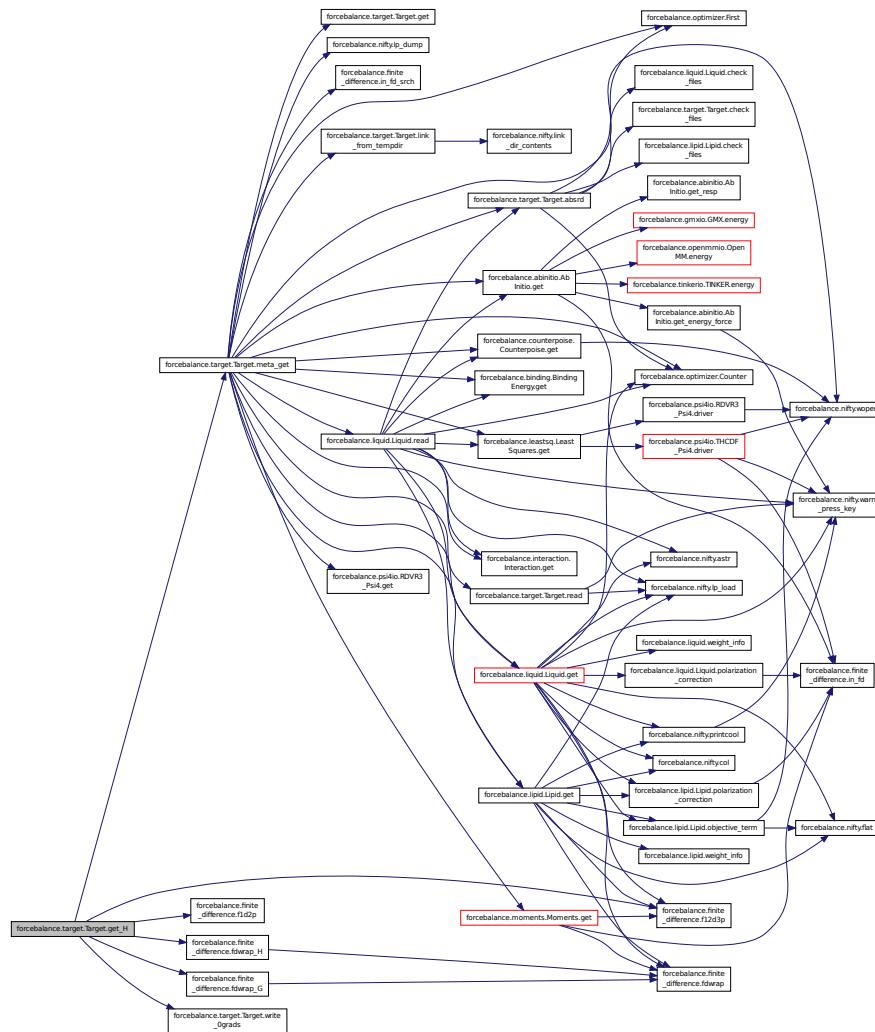
First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through

the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

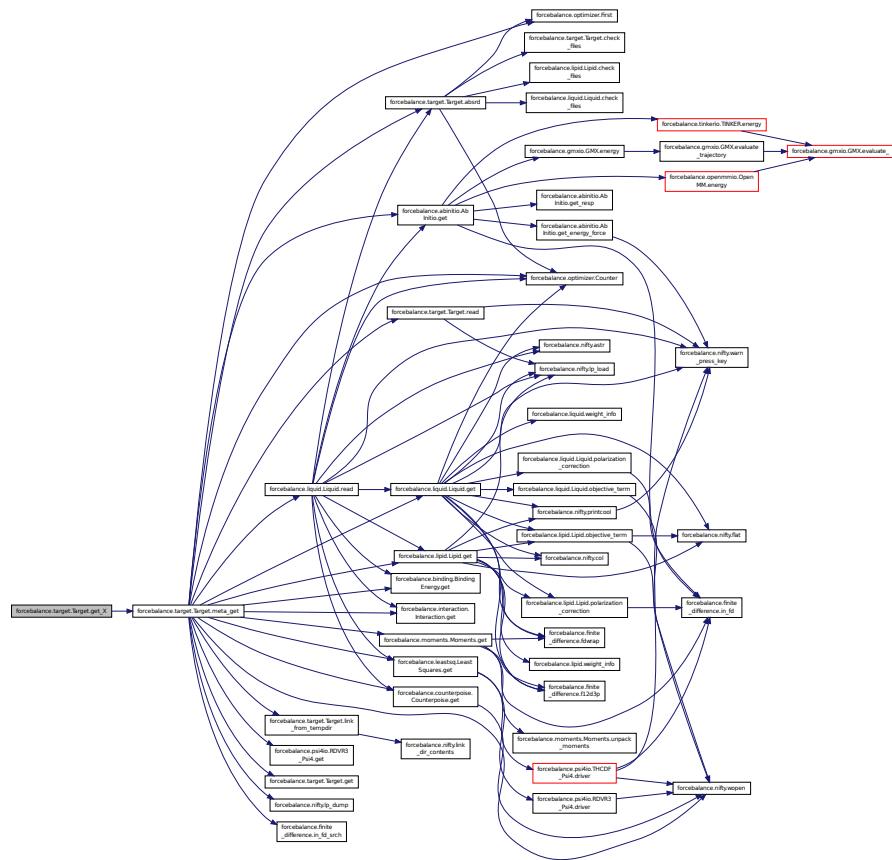
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

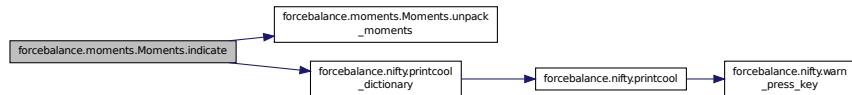
Here is the call graph for this function:



**def forcebalance.moments.Moments.indicate ( self ) [inherited]** Print qualitative indicator.

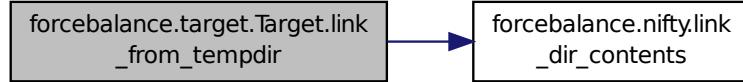
Definition at line 139 of file moments.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

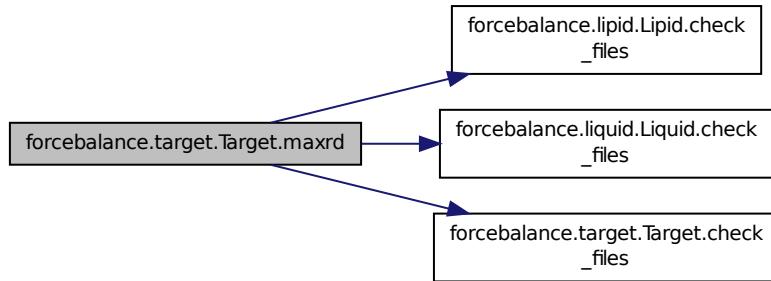
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

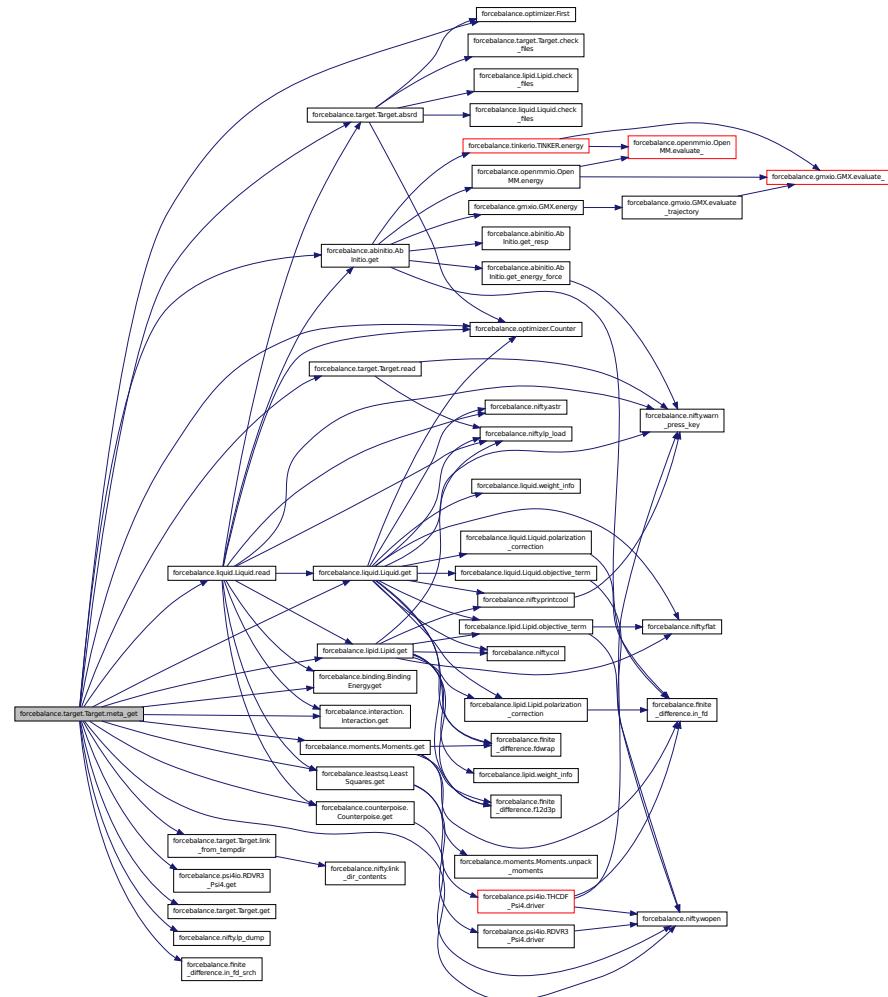


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

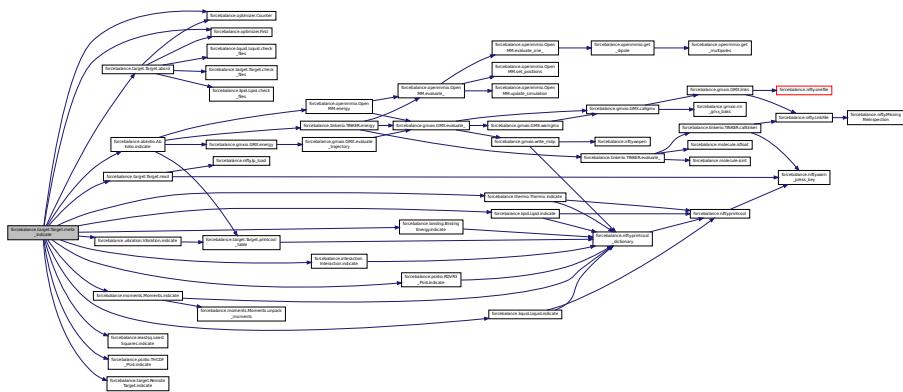
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

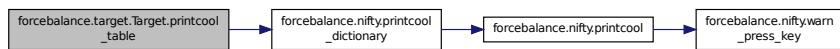
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

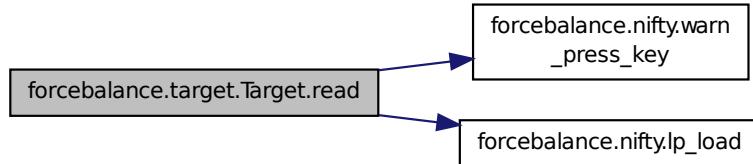
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.moments.Moments.read\_reference\_data ( self ) [inherited]** Read the reference data from a file.

Definition at line 70 of file moments.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

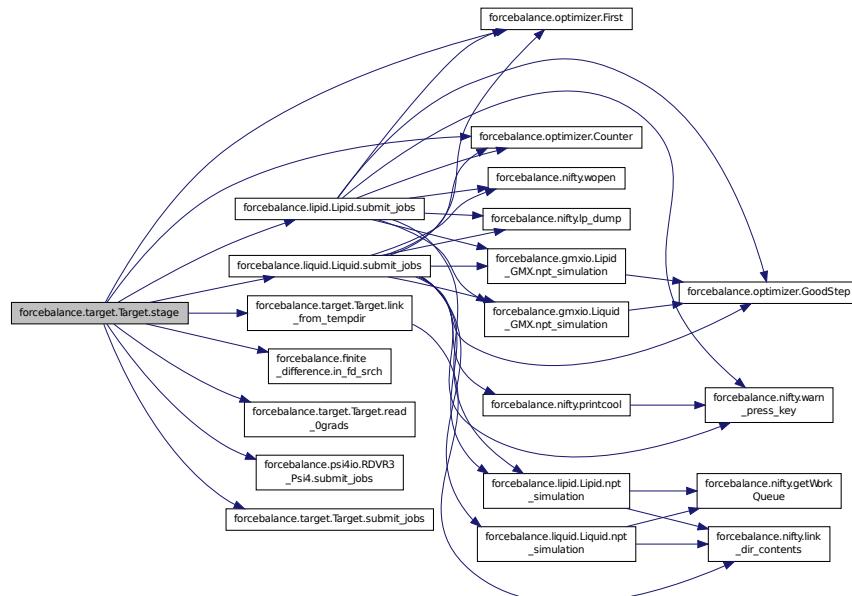
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src\_key, dest\_key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



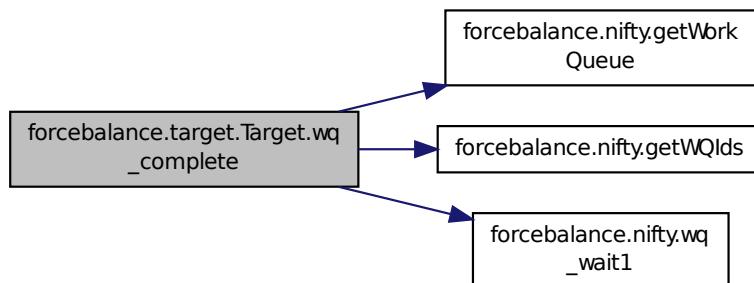
```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.moments.Moments.unpack_moments ( self, moment_dict ) [inherited] Definition at
line 165 of file moments.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work
Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.49.4 Member Data Documentation

**forcebalance.moments.Moments.calc\_moments [inherited]** Definition at line 198 of file moments.py.

**forcebalance.moments.Moments.denoms [inherited]** Definition at line 48 of file moments.py.

**forcebalance.moments.Moments.engine [inherited]** Read in the reference data.

Build keyword dictionaries to pass to engine. Create engine object.

Definition at line 66 of file moments.py.

**forcebalance.openmmio.Moments\_OpenMM.engine** Default file names for coordinates and key file.

Definition at line 1208 of file openmmio.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.moments.Moments.mfnm [inherited]** The mdata.txt file that contains the moments.

Definition at line 57 of file moments.py.

**forcebalance.moments.Moments.na [inherited]** Number of atoms.

Definition at line 72 of file moments.py.

**forcebalance.moments.Moments.objective [inherited]** Definition at line 199 of file moments.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [inherited] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.moments.Moments.ref\_eigvals** [inherited] Definition at line 73 of file moments.py.

**forcebalance.moments.Moments.ref\_eigvecs** [inherited] Definition at line 74 of file moments.py.

**forcebalance.moments.Moments.ref\_moments** [inherited] Dictionary of reference multipole moments.

Definition at line 59 of file moments.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number  
The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate** [inherited] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

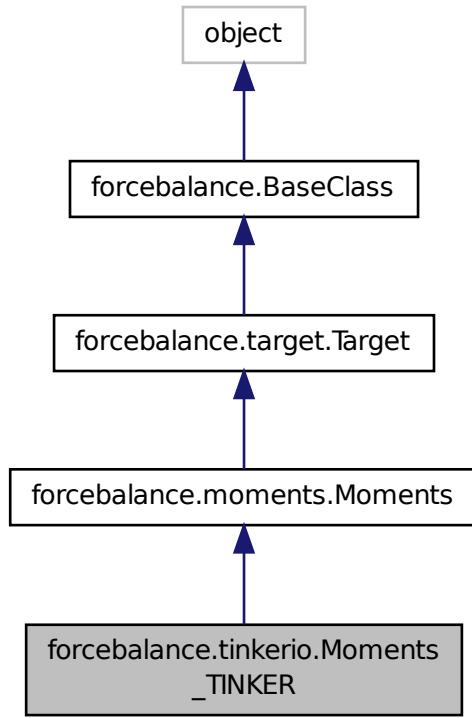
The documentation for this class was generated from the following file:

- [openmmio.py](#)

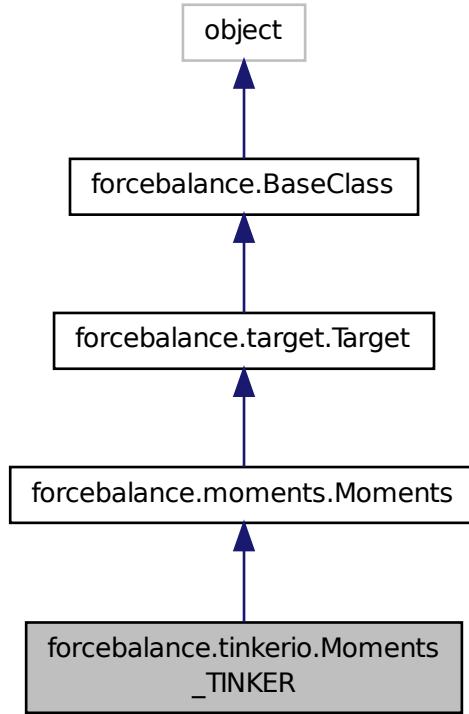
## 8.50 forcebalance.tinkerio.Moments\_TINKER Class Reference

Subclass of Target for multipole moment matching using TINKER.

Inheritance diagram for forcebalance.tinkerio.Moments\_TINKER:



Collaboration diagram for forcebalance.tinkerio.Moments\_TINKER:



## Public Member Functions

- def `_init_`
- def `read_reference_data`

*Read the reference data from a file.*
- def `indicate`

*Print qualitative indicator.*
- def `unpack_moments`
- def `get`

*Evaluate objective function.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*
- def `get_H`

- def `link_from_tempdir`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool.table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `engine_`  
*Default file names for coordinates and key file.*
- `denoms`
- `mfnm`  
*The mdata.txt file that contains the moments.*
- `ref_moments`  
*Dictionary of reference multipole moments.*
- `engine`  
*Read in the reference data.*
- `na`  
*Number of atoms.*
- `ref_eigvals`
- `ref_eigvecs`
- `calc_moments`
- `objective`
- `rd`  
*Root directory of the whole project.*
- `pgrad`  
*Iteration where we turn on zero-gradient skipping.*
- `tempbase`

- Relative directory of target.*
- **tempdir**
  - **rundir**

*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
  - **FF**

*Need the forcefield (here for now)*
  - **xct**

*Counts how often the objective function was computed.*
  - **gct**

*Counts how often the gradient was computed.*
  - **hct**

*Counts how often the Hessian was computed.*
  - **read\_indicate**

*Whether to read indicate.log from file when restarting an aborted run.*
  - **write\_indicate**

*Whether to write indicate.log at every iteration (true for all but remote.)*
  - **read\_objective**

*Whether to read objective.p from file when restarting an aborted run.*
  - **write\_objective**

*Whether to write objective.p at every iteration (true for all but remote.)*
  - **verbose\_options**
  - **PrintOptionDict**

### 8.50.1 Detailed Description

Subclass of Target for multipole moment matching using TINKER.  
Definition at line 1097 of file tinkerio.py.

### 8.50.2 Constructor & Destructor Documentation

**def forcebalance.tinkerio.Moments.TINKER.\_\_init\_\_ ( self, options, tgt\_opts, forcefield )** Definition at line 1098 of file tinkerio.py.

Here is the call graph for this function:



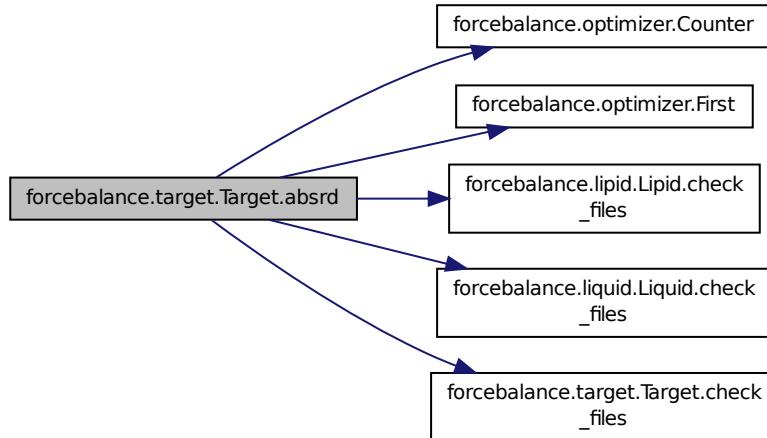
### 8.50.3 Member Function Documentation

**def forcebalance.BaseClass.\_\_setattr\_\_ ( self, key, value ) [inherited]** Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory  
specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



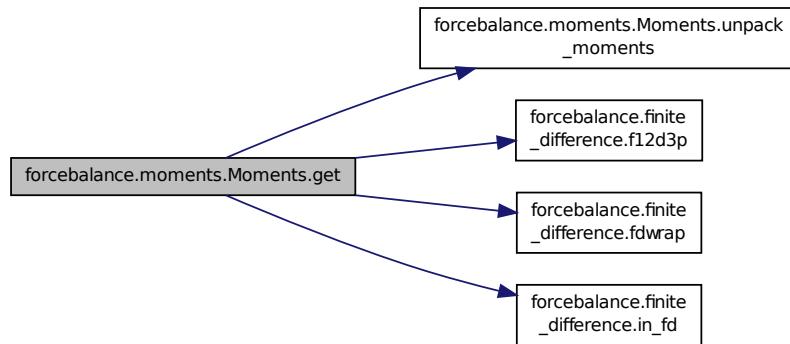
```
def forcebalance.target.Target.check_files ( self, there ) [inherited] Check this directory for the presence  
of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.moments.Moments.get ( self, mvals, AGrad=False, AHess=False ) [inherited]  
Evaluate objective function.
```

Definition at line 171 of file moments.py.

Here is the call graph for this function:



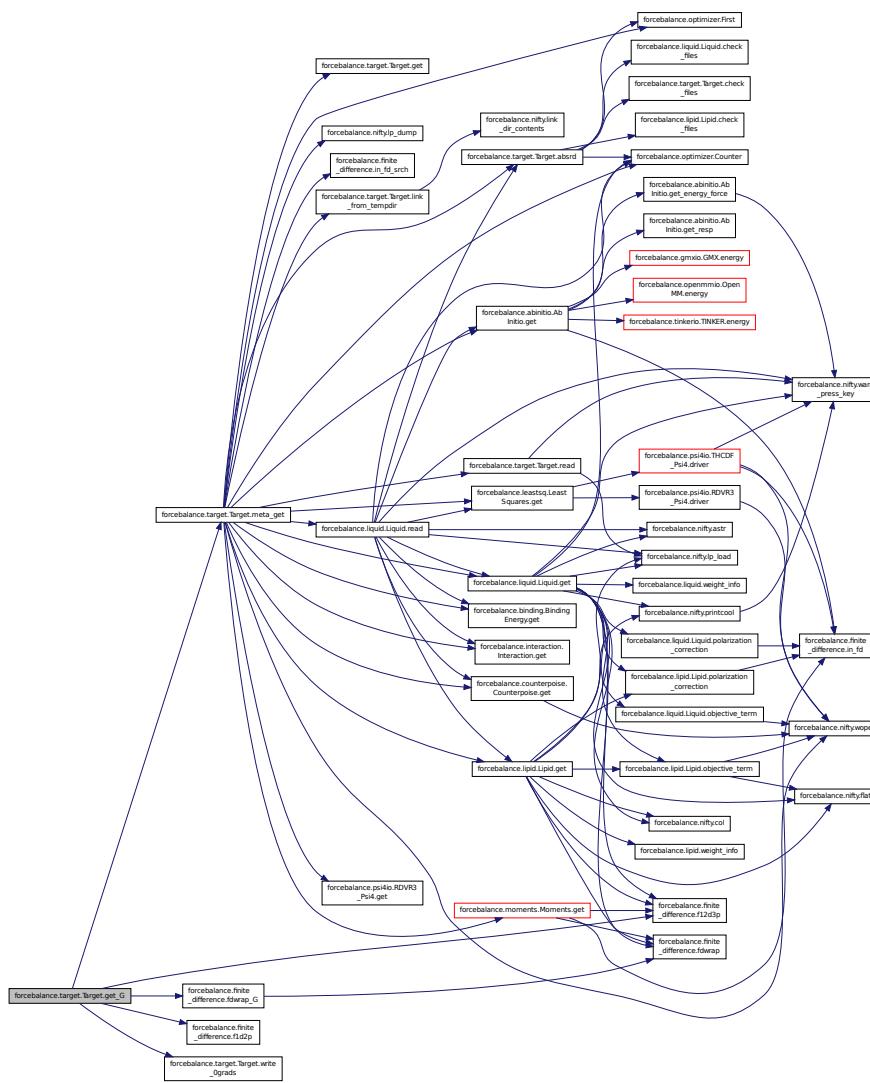
**def forcebalance.target.Target.get\_G( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the `fd1_pids` and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

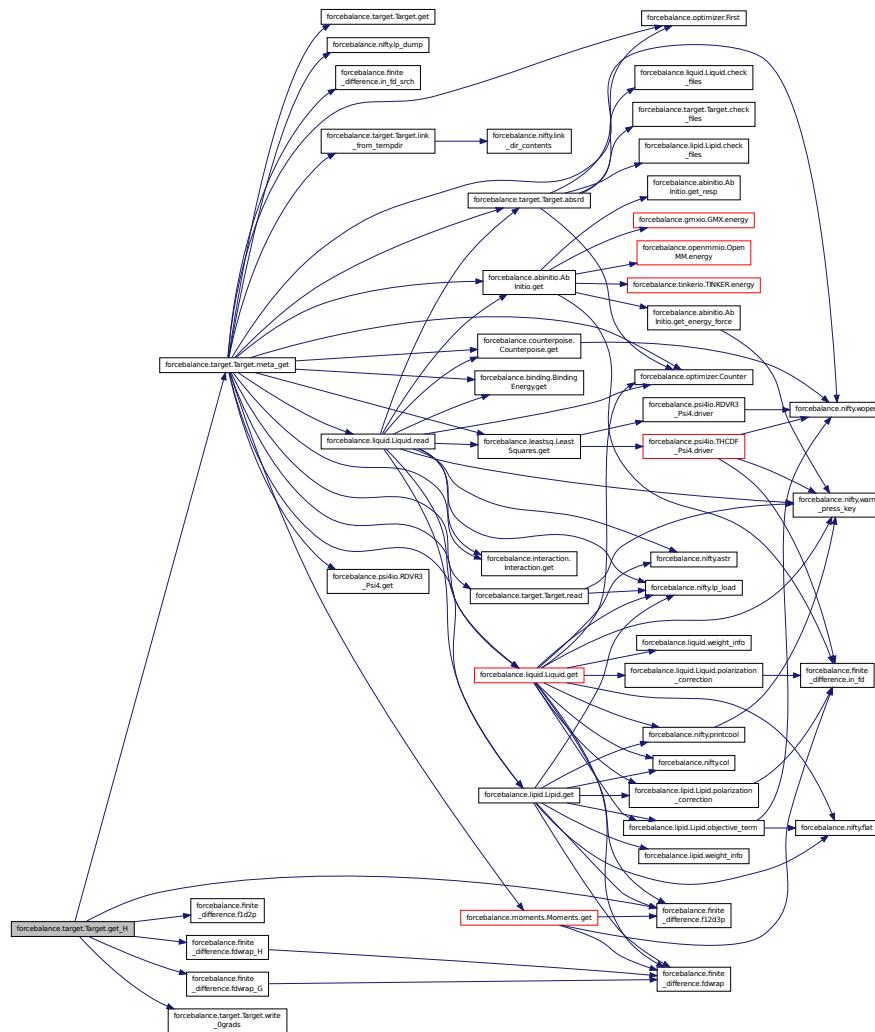
First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through

the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

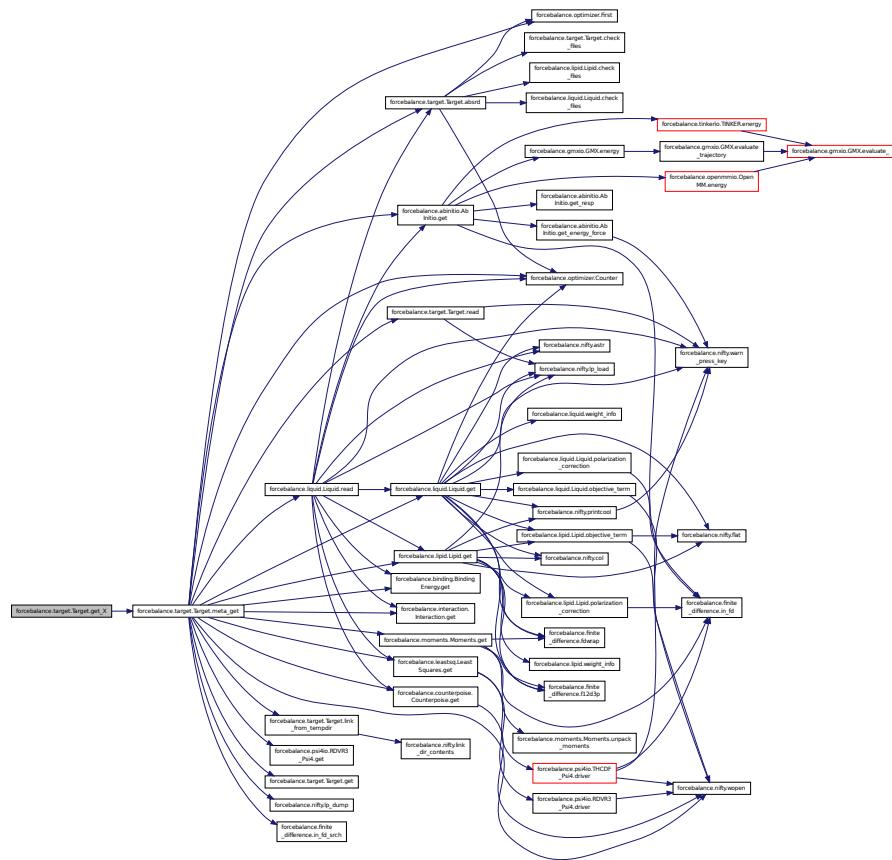
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

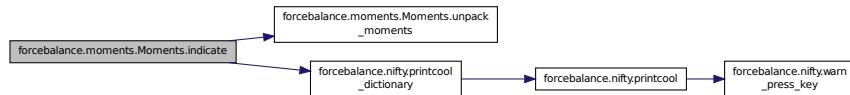
Here is the call graph for this function:



**def forcebalance.moments.Moments.indicate ( self ) [inherited]** Print qualitative indicator.

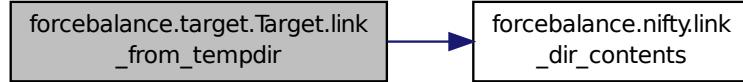
Definition at line 139 of file moments.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

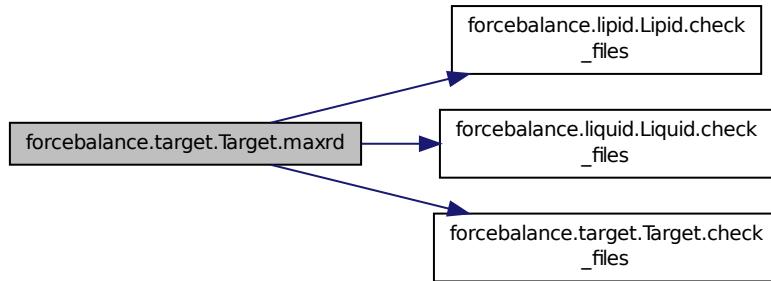
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

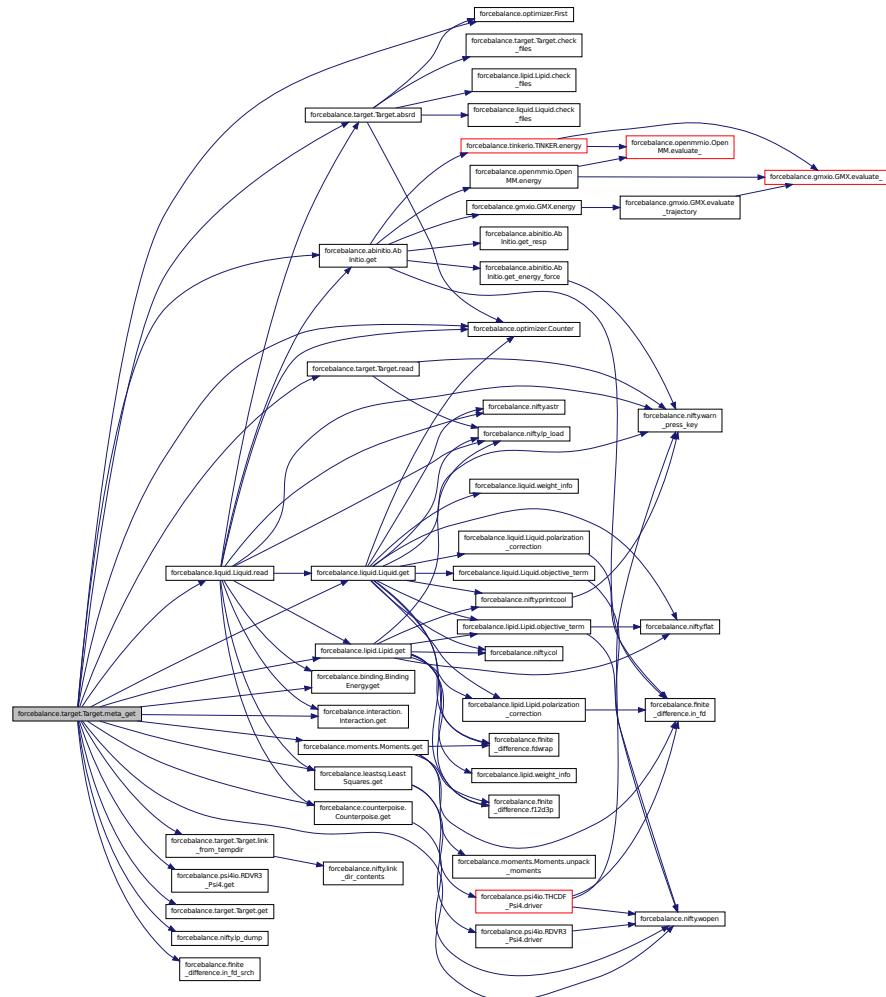


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

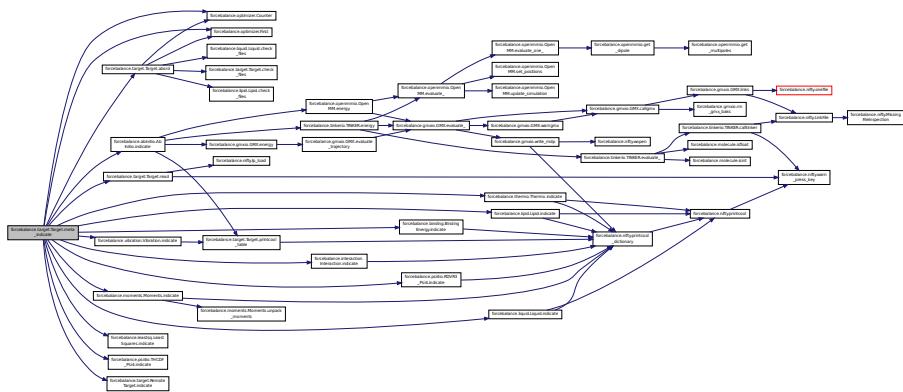
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

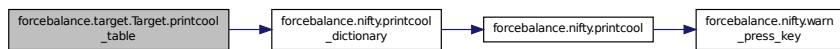
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

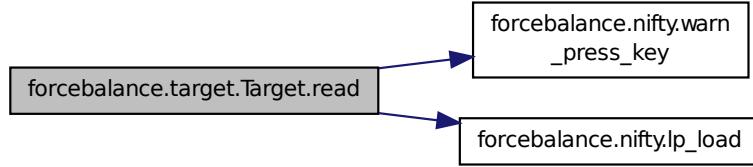
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.moments.Moments.read\_reference\_data ( self ) [inherited]** Read the reference data from a file.

Definition at line 70 of file moments.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

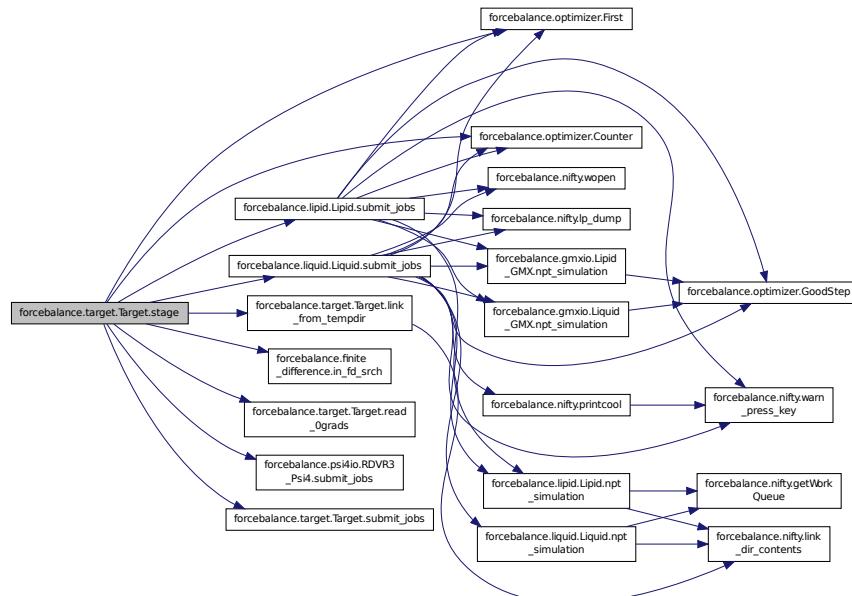
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src\_key, dest\_key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



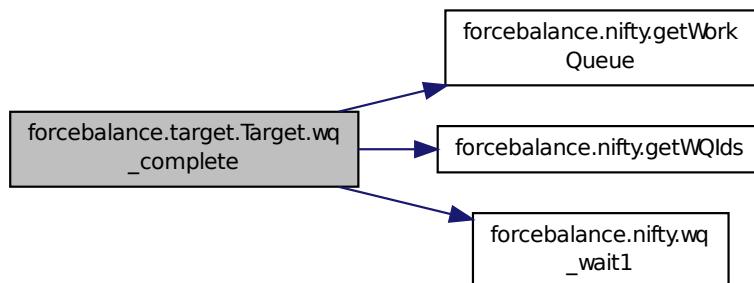
```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.moments.Moments.unpack_moments ( self, moment_dict ) [inherited] Definition at
line 165 of file moments.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work
Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.50.4 Member Data Documentation

**forcebalance.moments.Moments.calc\_moments [inherited]** Definition at line 198 of file moments.py.

**forcebalance.moments.Moments.denoms [inherited]** Definition at line 48 of file moments.py.

**forcebalance.moments.Moments.engine [inherited]** Read in the reference data.

Build keyword dictionaries to pass to engine. Create engine object.

Definition at line 66 of file moments.py.

**forcebalance.tinkerio.Moments.TINKER.engine\_** Default file names for coordinates and key file.

Definition at line 1102 of file tinkerio.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.moments.Moments.mfnm [inherited]** The mdata.txt file that contains the moments.

Definition at line 57 of file moments.py.

**forcebalance.moments.Moments.na [inherited]** Number of atoms.

Definition at line 72 of file moments.py.

**forcebalance.moments.Moments.objective [inherited]** Definition at line 199 of file moments.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [inherited] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.moments.Moments.ref\_eigvals** [inherited] Definition at line 73 of file moments.py.

**forcebalance.moments.Moments.ref\_eigvecs** [inherited] Definition at line 74 of file moments.py.

**forcebalance.moments.Moments.ref\_moments** [inherited] Dictionary of reference multipole moments.

Definition at line 59 of file moments.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number  
The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate** [inherited] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

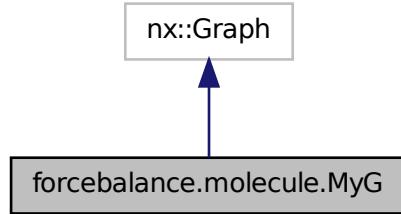
Definition at line 162 of file target.py.

The documentation for this class was generated from the following file:

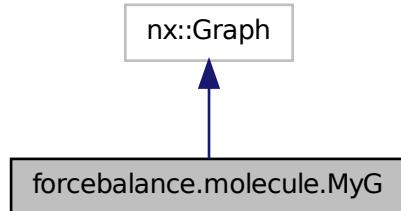
- [tinkerio.py](#)

## 8.51 forcebalance.molecule.MyG Class Reference

Inheritance diagram for forcebalance.molecule.MyG:



Collaboration diagram for forcebalance.molecule.MyG:



### Public Member Functions

- def `_init_`
- def `_eq_`
- def `_hash_`

*The hash function is something we can use to discard two things that are obviously not equal.*

- def `L`

*Return a list of the sorted atom numbers in this graph.*

- def `AStr`

*Return a string of atoms, which serves as a rudimentary 'fingerprint' : '99,100,103,151'.*

- def `e`

*Return an array of the elements.*

- def `ef`

*Create an Empirical Formula.*

- def `x`

*Get a list of the coordinates.*

## Public Attributes

- Alive

### 8.51.1 Detailed Description

Definition at line 324 of file molecule.py.

### 8.51.2 Constructor & Destructor Documentation

**def forcebalance.molecule.MyG.\_\_init\_\_ ( self )** Definition at line 325 of file molecule.py.

### 8.51.3 Member Function Documentation

**def forcebalance.molecule.MyG.\_\_eq\_\_ ( self, other )** Definition at line 328 of file molecule.py.

**def forcebalance.molecule.MyG.\_\_hash\_\_ ( self )** The hash function is something we can use to discard two things that are obviously not equal.

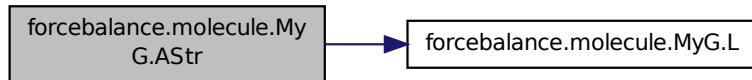
Here we neglect the hash.

Definition at line 337 of file molecule.py.

**def forcebalance.molecule.MyG.AStr ( self )** Return a string of atoms, which serves as a rudimentary 'fingerprint' : '99,100,103,151' .

Definition at line 345 of file molecule.py.

Here is the call graph for this function:

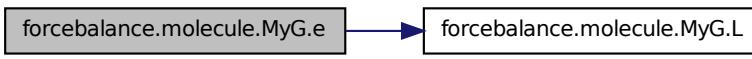


**def forcebalance.molecule.MyG.e ( self )** Return an array of the elements.

For instance ['H' 'C' 'C' 'H'].

Definition at line 349 of file molecule.py.

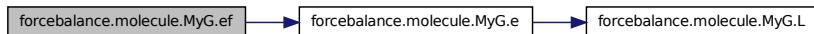
Here is the call graph for this function:



**def forcebalance.molecule.MyG.ef ( self )** Create an Empirical Formula.

Definition at line 354 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.MyG.L ( self )** Return a list of the sorted atom numbers in this graph.

Definition at line 341 of file molecule.py.

**def forcebalance.molecule.MyG.x ( self )** Get a list of the coordinates.

Definition at line 359 of file molecule.py.

Here is the call graph for this function:



#### 8.51.4 Member Data Documentation

**forcebalance.molecule.MyG.Alive** Definition at line 327 of file molecule.py.

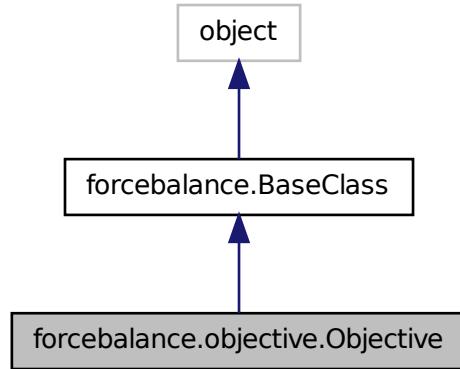
The documentation for this class was generated from the following file:

- [molecule.py](#)

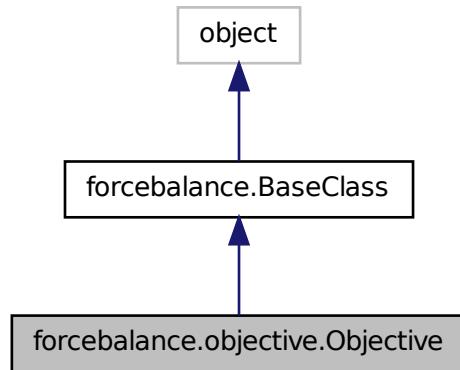
### 8.52 forcebalance.objective.Objective Class Reference

[Objective](#) function.

Inheritance diagram for forcebalance.objective.Objective:



Collaboration diagram for forcebalance.objective.Objective:



## Public Member Functions

- def `_init_`
- def `Target_Terms`
- def `Indicate`  
    *Print objective function contributions.*
- def `Full`
- def `__setattr__`
- def `set_option`

## Public Attributes

- [Targets](#)  
*Work Queue Port (The specific target itself may or may not actually use this.)*
- [FF](#)  
*The force field (it seems to be everywhere)*
- [Penalty](#)  
*Initialize the penalty function.*
- [WTot](#)  
*Obtain the denominator.*
- [ObjDict](#)
- [ObjDict.Last](#)
- [verbose\\_options](#)
- [PrintOptionDict](#)

### 8.52.1 Detailed Description

[Objective](#) function.

The objective function is a combination of contributions from the different fitting targets. Basically, it loops through the targets, gets their contributions to the objective function and then sums all of them (although more elaborate schemes are conceivable). The return value is the same data type as calling the target itself: a dictionary containing the objective function, the gradient and the Hessian.

The penalty function is also computed here; it keeps the parameters from straying too far from their initial values.  
Parameters

in	<i>mvals</i>	The mathematical parameters that enter into computing the objective function
in	<i>Order</i>	The requested order of differentiation

Definition at line 115 of file objective.py.

### 8.52.2 Constructor & Destructor Documentation

**def forcebalance.objective.Objective.\_\_init\_\_ ( *self*, *options*, *tgt\_opts*, *forcefield* )** Definition at line 116 of file objective.py.

Here is the call graph for this function:

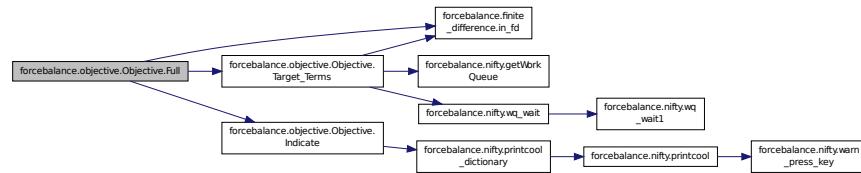


### 8.52.3 Member Function Documentation

**def forcebalance.BaseClass.\_\_setattr\_\_ ( *self*, *key*, *value* ) [inherited]** Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.objective.Objective.Full ( self, mvals, Order = 0, verbose = False ) Definition at line 261  
of file objective.py.
```

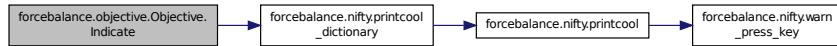
Here is the call graph for this function:



```
def forcebalance.objective.Objective.Indicate ( self ) Print objective function contributions.
```

Definition at line 223 of file objective.py.

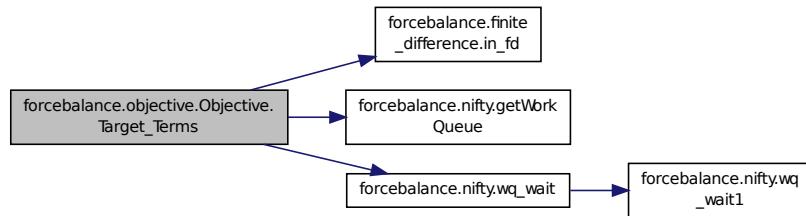
Here is the call graph for this function:



```
def forcebalance.BaseClass.set_option ( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

```
def forcebalance.objective.Objective.Target_Terms ( self, mvals, Order = 0, verbose = False ) Definition at line 163 of file objective.py.
```

Here is the call graph for this function:



#### 8.52.4 Member Data Documentation

**forcebalance.objective.Objective.FF** The force field (it seems to be everywhere)  
Definition at line 142 of file objective.py.

**forcebalance.objective.Objective.ObjDict** Definition at line 152 of file objective.py.

**forcebalance.objective.Objective.ObjDict\_Last** Definition at line 153 of file objective.py.

**forcebalance.objective.Objective.Penalty** Initialize the penalty function.

Definition at line 144 of file objective.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.objective.Objective.Targets** Work Queue Port (The specific target itself may or may not actually use this.)

Asynchronous objective function evaluation (i.e. execute Work Queue and local objective concurrently.) The list of fitting targets

Definition at line 131 of file objective.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.objective.Objective.WTot** Obtain the denominator.

Definition at line 149 of file objective.py.

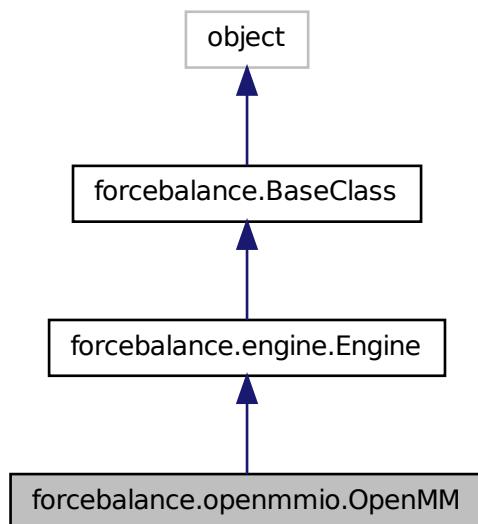
The documentation for this class was generated from the following file:

- [objective.py](#)

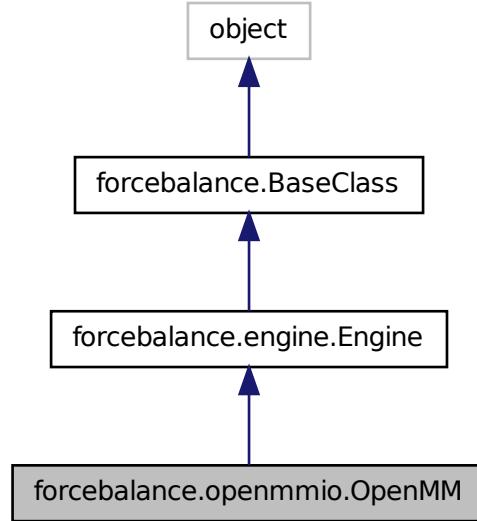
## 8.53 forcebalance.openmmio.OpenMM Class Reference

Derived from Engine object for carrying out general purpose [OpenMM](#) calculations.

Inheritance diagram for forcebalance.openmmio.OpenMM:



Collaboration diagram for forcebalance.openmmio.OpenMM:



## Public Member Functions

- def [`\_init\_`](#)
- def [`setopts`](#)

*Called by `init` ; Set OpenMM-specific options.*
- def [`readsrc`](#)

*Called by `init` ; read files from the source directory.*
- def [`prepare`](#)

*Prepare the calculation.*
- def [`create\_simulation`](#)

*Create simulation object.*
- def [`update\_simulation`](#)

*Create the simulation object, or update the force field parameters in the existing simulation object.*
- def [`set\_positions`](#)

*Set the positions and periodic box vectors to one of the stored coordinates.*
- def [`compute\_volume`](#)

*Compute the total volume of an `OpenMM` system.*
- def [`compute\_mass`](#)

*Compute the total mass of an `OpenMM` system.*
- def [`evaluate\_one`](#)
- def [`evaluate\_`](#)

*Utility function for computing energy, and (optionally) forces and dipoles using `OpenMM`.*
- def [`energy\_one`](#)
- def [`energy\_force\_one`](#)

- def `energy`
- def `energy_force`

*Loop through the snapshots and compute the energies and forces using OpenMM.*
- def `energy_dipole`

*Loop through the snapshots and compute the energies and forces using OpenMM.*
- def `normal_modes`
- def `optimize`

*Optimize the geometry and align the optimized geometry to the starting geometry, and return the RMSD.*
- def `multipole_moments`

*Return the multipole moments of the i-th snapshot in Debye and Buckingham units.*
- def `energy_rmsd`

*Calculate energy of the 1st structure (optionally minimize and return the minimized energy and RMSD).*
- def `interaction_energy`

*Calculate the interaction energy for two fragments.*
- def `molecular_dynamics`

*Method for running a molecular dynamics simulation.*
- def `setopts`
- def `prepare`
- def `__setattr__`
- def `set_option`

## Public Attributes

- `valkwds`
- `platname`

*Target settings override.*
- `precision`
- `platform`

*Set the simulation platform.*
- `simkargs`
- `mol`
- `pdb`

*Create the OpenMM PDB object.*
- `ffxml`

*Create the OpenMM ForceField object.*
- `forcefield`
- `mmopts`

*OpenMM options for setting up the System.*
- `AMOEBA`

*Are we using AMOEBA?*
- `pbc`

*Set system options from ForceBalance force field options.*
- `xyz_omm`

*Generate OpenMM-compatible positions.*
- `AtomMask`

*Build a topology and atom lists.*
- `AtomLists`
- `tdiv`

*Determine the integrator.*

- [simulation](#)  
*If no temperature control, default to the Verlet integrator.*

- [mod](#)
- [system](#)
- [vsinfo](#)
- [nbcharges](#)
- [vsprm](#)
- [name](#)
- [A](#)
- [B](#)
- [mass](#)
- [ndof](#)
- [verbose](#)
- [target](#)

*Engines can get properties from the Target that creates them.*

- [root](#)
- [srcdir](#)
- [tempdir](#)
- [FF](#)
- [verbose\\_options](#)
- [PrintOptionDict](#)

### 8.53.1 Detailed Description

Derived from Engine object for carrying out general purpose [OpenMM](#) calculations.

Definition at line 467 of file openmmio.py.

### 8.53.2 Constructor & Destructor Documentation

**def forcebalance.openmmio.OpenMM.\_\_init\_\_( self, name = "openmm", kwargs )** Definition at line 470 of file openmmio.py.

### 8.53.3 Member Function Documentation

**def forcebalance.BaseClass.\_\_setattr\_\_( self, key, value ) [inherited]** Definition at line 28 of file \_\_init\_\_.py.

**def forcebalance.openmmio.OpenMM.compute\_mass( self, system )** Compute the total mass of an [OpenMM](#) system.

Definition at line 773 of file openmmio.py.

**def forcebalance.openmmio.OpenMM.compute\_volume( self, box\_vectors )** Compute the total volume of an [OpenMM](#) system.

Definition at line 764 of file openmmio.py.

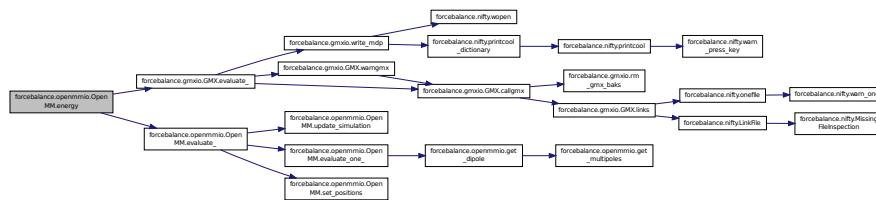
**def forcebalance.openmmio.OpenMM.create\_simulation( self, timestep = 1.0, faststep = 0.25, temperature = None, pressure = None, anisotropic = False, mts = False, collision = 1.0, nbarostat = 25, rmpd\_beads = 0, kwargs )** Create simulation object.

Note that this also takes in some options pertinent to system setup, including the type of MD integrator and type of pressure control.

Definition at line 644 of file openmmio.py.

**def forcebalance.openmmio.OpenMM.energy ( self )** Definition at line 835 of file openmmio.py.

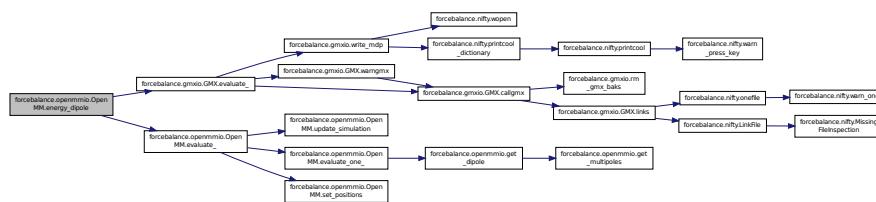
Here is the call graph for this function:



**def forcebalance.openmmio.OpenMM.energy\_dipole ( self )** Loop through the snapshots and compute the energies and forces using OpenMM.

Definition at line 848 of file openmmio.py.

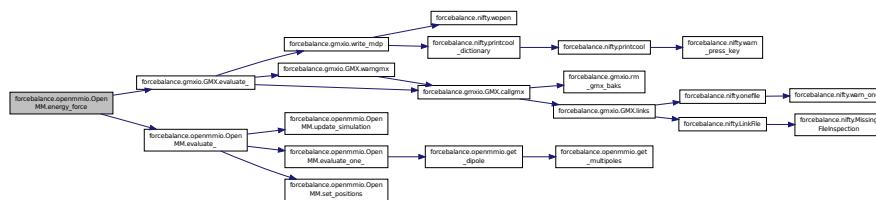
Here is the call graph for this function:



**def forcebalance.openmmio.OpenMM.energy\_force ( self )** Loop through the snapshots and compute the energies and forces using OpenMM.

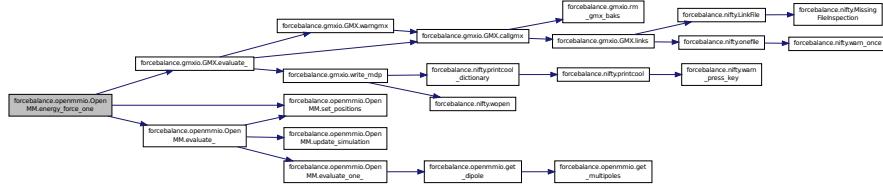
Definition at line 840 of file openmmio.py.

Here is the call graph for this function:



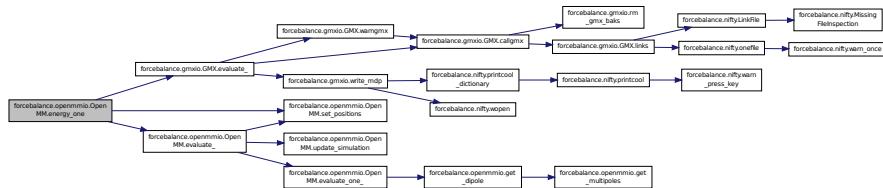
**def forcebalance.openmmio.OpenMM.energy\_force\_one ( self )** Definition at line 830 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.OpenMM.energy\_one ( self, shot )** Definition at line 826 of file openmmio.py.

Here is the call graph for this function:

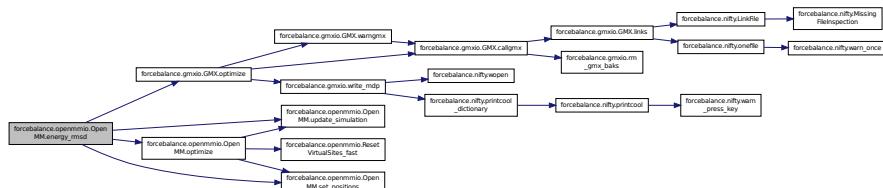


**def forcebalance.openmmio.OpenMM.energy\_rmsd ( self, shot = 0, optimize = True )** Calculate energy of the 1st structure (optionally minimize and return the minimized energy and RMSD).

In kcal/mol.

Definition at line 909 of file openmmio.py.

Here is the call graph for this function:



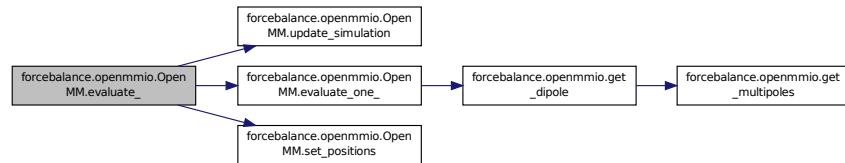
**def forcebalance.openmmio.OpenMM.evaluate\_ ( self, force = False, dipole = False, traj = False )** Utility function for computing energy, and (optionally) forces and dipoles using OpenMM.

Inputs: force: Switch for calculating the force. dipole: Switch for calculating the dipole. traj: Trajectory (listing of coordinate and box 2-tuples). If provide, will loop over these snapshots. Otherwise will do a single point evaluation at the current geometry.

Outputs: Result: Dictionary containing energies, forces and/or dipoles.

Definition at line 804 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.OpenMM.evaluate\_one\_( self, force = False, dipole = False )** Definition at line 779 of file openmmio.py.

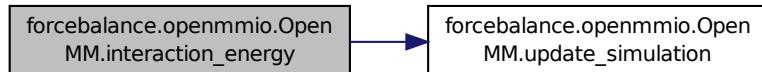
Here is the call graph for this function:



**def forcebalance.openmmio.OpenMM.interaction\_energy( self, fraga, fragb )** Calculate the interaction energy for two fragments.

Definition at line 930 of file openmmio.py.

Here is the call graph for this function:



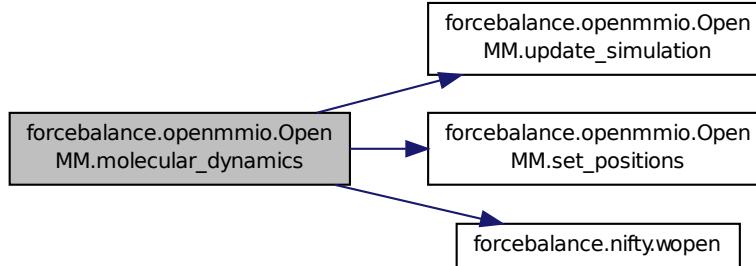
**def forcebalance.openmmio.OpenMM.molecular\_dynamics( self, nsteps, timestep, temperature = None, pressure = None, nequil = 0, nsave = 1000, minimize = True, anisotropic = False, save\_traj = False, verbose = False, kwargs )** Method for running a molecular dynamics simulation.

Required arguments: nsteps = (int) Number of total time steps timestep = (float) Time step in FEMTOSECOND-S temperature = (float) Temperature control (Kelvin) pressure = (float) Pressure control (atmospheres) nequil = (int) Number of additional time steps at the beginning for equilibration nsave = (int) Step interval for saving and printing data minimize = (bool) Perform an energy minimization prior to dynamics

Returns simulation data: Rhos = (array) Density in kilogram m^-3 Potentials = (array) Potential energies Kinetics = (array) Kinetic energies Volumes = (array) Box volumes Dips = (3xN array) Dipole moments EComps = (dict) Energy components

Definition at line 980 of file openmmio.py.

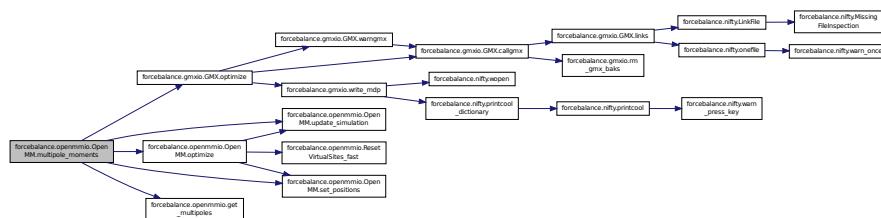
Here is the call graph for this function:



**def forcebalance.openmmio.OpenMM.multipole\_moments ( self, shot = 0, optimize = True, polarizability = False )** Return the multipole moments of the i-th snapshot in Debye and Buckingham units.

Definition at line 887 of file openmmio.py.

Here is the call graph for this function:

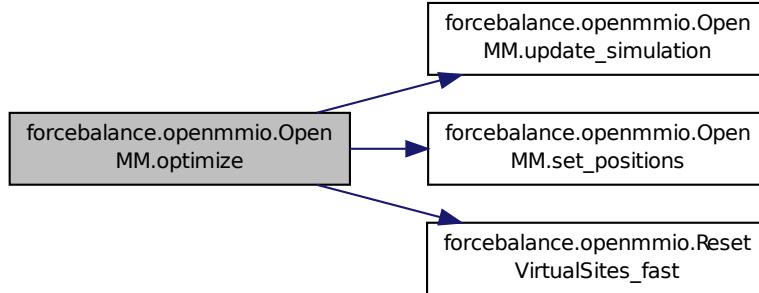


**def forcebalance.openmmio.OpenMM.normal\_modes ( self, shot = 0, optimize = True )** Definition at line 852 of file openmmio.py.

**def forcebalance.openmmio.OpenMM.optimize ( self, shot = 0, crit = 1e-4 )** Optimize the geometry and align the optimized geometry to the starting geometry, and return the RMSD.

Definition at line 857 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.engine.Engine.prepare( self, kwargs ) [inherited]** Definition at line 95 of file engine.py.

**def forcebalance.openmmio.OpenMM.prepare( self, pbc = False, mmopts = {}, kwargs )** Prepare the calculation.

Note that we don't create the Simulation object yet, because that may depend on MD integrator parameters, thermostat, barostat etc.

Definition at line 545 of file openmmio.py.

**def forcebalance.openmmio.OpenMM.readsrc( self, kwargs )** Called by `init` ; read files from the source directory.

Provide a molecule object or a coordinate file. Add an optional PDB file for residues, atom names etc.

Definition at line 517 of file openmmio.py.

**def forcebalance.BaseClass.set\_option( self, in\_dict, src\_key, dest\_key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.openmmio.OpenMM.set\_positions( self, shot = 0, traj = None )** Set the positions and periodic box vectors to one of the stored coordinates.

\*\*\* NOTE: If you run a MD simulation, then the coordinates are overwritten by the MD trajectory. \*\*\*

Definition at line 743 of file openmmio.py.

**def forcebalance.engine.Engine.setopts( self, kwargs ) [inherited]** Definition at line 89 of file engine.py.

**def forcebalance.openmmio.OpenMM.setopts( self, platname = "CUDA", precision = "single", kwargs )** Called by `init` ; Set OpenMM-specific options.

Definition at line 476 of file openmmio.py.

**def forcebalance.openmmio.OpenMM.update\_simulation( self, kwargs )** Create the simulation object, or update the force field parameters in the existing simulation object.

This should be run when we write a new force field XML file.

Definition at line 705 of file openmmio.py.

#### 8.53.4 Member Data Documentation

**forcebalance.openmmio.OpenMM.A** Definition at line 941 of file openmmio.py.

**forcebalance.openmmio.OpenMM.AMOEBA** Are we using AMOEBA?

Definition at line 571 of file openmmio.py.

**forcebalance.openmmio.OpenMM.AtomLists** Definition at line 632 of file openmmio.py.

**forcebalance.openmmio.OpenMM.AtomMask** Build a topology and atom lists.

Definition at line 631 of file openmmio.py.

**forcebalance.openmmio.OpenMM.B** Definition at line 943 of file openmmio.py.

**forcebalance.engine.Engine.FF** [inherited] Definition at line 65 of file engine.py.

**forcebalance.openmmio.OpenMM.ffxml** Create the [OpenMM](#) ForceField object.

Definition at line 556 of file openmmio.py.

**forcebalance.openmmio.OpenMM.forcefield** Definition at line 557 of file openmmio.py.

**forcebalance.openmmio.OpenMM.mass** Definition at line 1016 of file openmmio.py.

**forcebalance.openmmio.OpenMM.mmopts** [OpenMM](#) options for setting up the System.

Definition at line 568 of file openmmio.py.

**forcebalance.openmmio.OpenMM.mod** Definition at line 710 of file openmmio.py.

**forcebalance.openmmio.OpenMM.mol** Definition at line 528 of file openmmio.py.

**forcebalance.openmmio.OpenMM.name** Definition at line 935 of file openmmio.py.

**forcebalance.openmmio.OpenMM.nbcharges** Definition at line 715 of file openmmio.py.

**forcebalance.openmmio.OpenMM.ndof** Definition at line 1019 of file openmmio.py.

**forcebalance.openmmio.OpenMM.pbc** Set system options from ForceBalance force field options.

Set system options from periodic boundary conditions.

Definition at line 587 of file openmmio.py.

**forcebalance.openmmio.OpenMM.pdb** Create the [OpenMM](#) PDB object.

Definition at line 550 of file openmmio.py.

**forcebalance.openmmio.OpenMM.platform** Set the simulation platform.

Definition at line 497 of file openmmio.py.

**forcebalance.openmmio.OpenMM.platname** Target settings override.

Set the device to the environment variable or zero otherwise.

Definition at line 481 of file openmmio.py.

**forcebalance.openmmio.OpenMM.precision** Definition at line 482 of file openmmio.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.engine.Engine.root** [inherited] Definition at line 56 of file engine.py.

**forcebalance.openmmio.OpenMM.simkwargs** Definition at line 511 of file openmmio.py.

**forcebalance.openmmio.OpenMM.simulation** If no temperature control, default to the Verlet integrator.

Add the barostat. Set up for energy component analysis. If virtual particles are used with AMOEBA... Finally create the simulation object.

Definition at line 691 of file openmmio.py.

**forcebalance.engine.Engine.srcdir** [inherited] Definition at line 57 of file engine.py.

**forcebalance.openmmio.OpenMM.system** Definition at line 713 of file openmmio.py.

**forcebalance.engine.Engine.target** [inherited] Engines can get properties from the Target that creates them.

Definition at line 55 of file engine.py.

**forcebalance.openmmio.OpenMM.tdiv** Determine the integrator.

If temperature control is turned on, then run Langevin dynamics.

Definition at line 648 of file openmmio.py.

**forcebalance.engine.Engine.tempdir** [inherited] Definition at line 58 of file engine.py.

**forcebalance.openmmio.OpenMM.valkwd** Definition at line 471 of file openmmio.py.

**forcebalance.engine.Engine.verbose** [inherited] Definition at line 50 of file engine.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.openmmio.OpenMM.vsinfo** Definition at line 714 of file openmmio.py.

**forcebalance.openmmio.OpenMM.vsprm** Definition at line 728 of file openmmio.py.

**forcebalance.openmmio.OpenMM.xyz\_omms** Generate OpenMM-compatible positions.

Definition at line 605 of file openmmio.py.

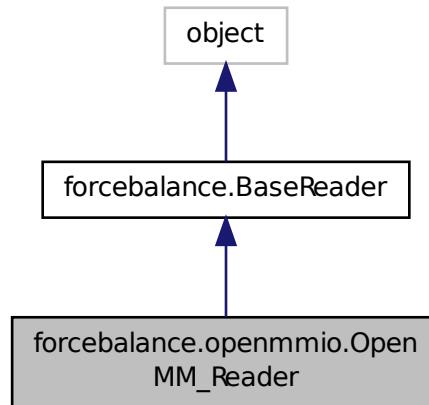
The documentation for this class was generated from the following file:

- [openmmio.py](#)

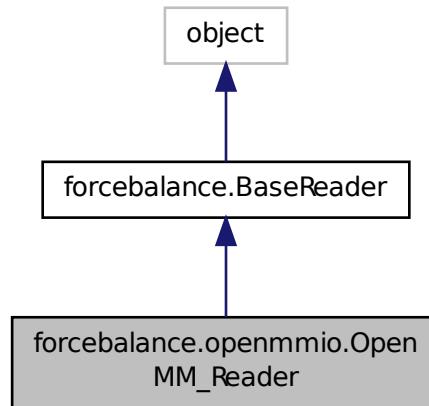
## 8.54 forcebalance.openmmio.OpenMM\_Reader Class Reference

Class for parsing [OpenMM](#) force field files.

Inheritance diagram for forcebalance.openmmio.OpenMM\_Reader:



Collaboration diagram for forcebalance.openmmio.OpenMM\_Reader:



### Public Member Functions

- def [\\_\\_init\\_\\_](#)
- def [build.pid](#)

*Build the parameter identifier (see link for an example)*

- def [Split](#)
- def [Whites](#)
- def [feed](#)
- def [build\\_pid](#)

*Returns the parameter type (e.g.*

## Public Attributes

- [pdict](#)  
*Initialize the superclass.*
- [In](#)
- [itype](#)
- [suffix](#)
- [adict](#)  
*The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.*
- [molatom](#)  
*The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.*
- [Molecules](#)
- [AtomTypes](#)

### 8.54.1 Detailed Description

Class for parsing [OpenMM](#) force field files.

Definition at line 440 of file openmmio.py.

### 8.54.2 Constructor & Destructor Documentation

**def forcebalance.openmmio.OpenMM\_Reader.\_\_init\_\_ ( self, fnm )** Definition at line 441 of file openmmio.py.

### 8.54.3 Member Function Documentation

**def forcebalance.BaseReader.build\_pid ( self, pfd ) [inherited]** Returns the parameter type (e.g. K in BONDSK) based on the current interaction type.

Both the 'pdict' dictionary (see [gmxio.pdict](#)) and the interaction type 'state' (here, BONDS) are needed to get the parameter type.

If, however, 'pdict' does not contain the ptype value, a suitable substitute is simply the field number.

Note that if the interaction type state is not set, then it defaults to the file name, so a generic parameter ID is 'filename.line.num.field.num'

Definition at line 124 of file \_\_init\_\_.py.

**def forcebalance.openmmio.OpenMM\_Reader.build\_pid ( self, element, parameter )** Build the parameter identifier (see [link](#) for an example)

**Todo** Add a link here

Definition at line 450 of file openmmio.py.

**def forcebalance.BaseReader.feed ( self, line ) [inherited]** Definition at line 105 of file \_\_init\_\_.py.

**def forcebalance.BaseReader.Split ( self, line ) [inherited]** Definition at line 99 of file \_\_init\_\_.py.

**def forcebalance.BaseReader.Whites ( self, line ) [inherited]** Definition at line 102 of file \_\_init\_\_.py.

#### 8.54.4 Member Data Documentation

**forcebalance.BaseReader.adict [inherited]** The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.

Definition at line 89 of file \_\_init\_\_.py.

**forcebalance.BaseReader.AtomTypes [inherited]** Definition at line 97 of file \_\_init\_\_.py.

**forcebalance.BaseReader.itype [inherited]** Definition at line 85 of file \_\_init\_\_.py.

**forcebalance.BaseReader.In [inherited]** Definition at line 84 of file \_\_init\_\_.py.

**forcebalance.BaseReader.molatom [inherited]** The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.

self.moleculerdict = OrderedDict() The listing of 'RES:ATOMNAMES' for atom names in the line This is obviously a placeholder.

Definition at line 94 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules [inherited]** Definition at line 96 of file \_\_init\_\_.py.

**forcebalance.openmmio.OpenMM\_Reader.pdict** Initialize the superclass.

:) The parameter dictionary (defined in this file)

Definition at line 445 of file openmmio.py.

**forcebalance.BaseReader.suffix [inherited]** Definition at line 86 of file \_\_init\_\_.py.

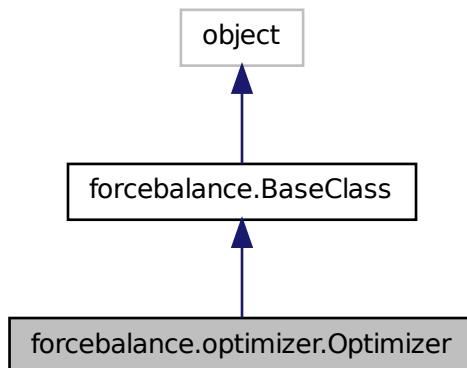
The documentation for this class was generated from the following file:

- [openmmio.py](#)

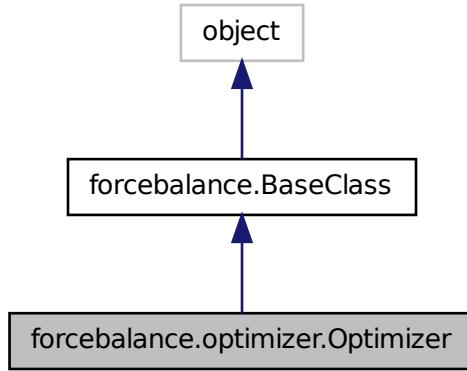
#### 8.55 forcebalance.optimizer.Optimizer Class Reference

[Optimizer](#) class.

Inheritance diagram for forcebalance.optimizer.Optimizer:



Collaboration diagram for forcebalance.optimizer.Optimizer:



## Public Member Functions

- def [`\_init\_`](#)  
*Create an [Optimizer](#) object.*
- def [`recover`](#)
- def [`save\_mvals\_to\_input`](#)  
*Write a new input file (`s_save.in`) containing the current mathematical parameters.*
- def [`Run`](#)  
*Call the appropriate optimizer.*
- def [`adjh`](#)
- def [`MainOptimizer`](#)  
*The main ForceBalance adaptive trust-radius pseudo-Newton optimizer.*
- def [`step`](#)  
*Computes the next step in the parameter space.*
- def [`NewtonRaphson`](#)  
*Optimize the force field parameters using the Newton-Raphson method .*
- def [`BFGS`](#)  
*Optimize the force field parameters using the BFGS method; currently the recommended choice .*
- def [`ScipyOptimizer`](#)  
*Driver for SciPy optimizations.*
- def [`GeneticAlgorithm`](#)  
*Genetic algorithm, under development.*
- def [`Simplex`](#)  
*Use SciPy's built-in simplex algorithm to optimize the parameters.*
- def [`Powell`](#)  
*Use SciPy's built-in Powell direction-set algorithm to optimize the parameters.*
- def [`Anneal`](#)  
*Use SciPy's built-in simulated annealing algorithm to optimize the parameters.*

- def **ConjugateGradient**  
*Use SciPy's built-in conjugate gradient algorithm to optimize the parameters.*
- def **Scipy\_BFGS**  
*Use SciPy's built-in BFGS algorithm to optimize the parameters.*
- def **BasinHopping**  
*Use SciPy's built-in basin hopping algorithm to optimize the parameters.*
- def **TruncatedNewton**  
*Use SciPy's built-in truncated Newton (fmin\_tnc) algorithm to optimize the parameters.*
- def **NewtonCG**  
*Use SciPy's built-in Newton-CG (fmin\_ncg) algorithm to optimize the parameters.*
- def **Scan\_Values**  
*Scan through parameter values.*
- def **ScanMVals**  
*Scan through the mathematical parameter space.*
- def **ScanPVals**  
*Scan through the physical parameter space.*
- def **SinglePoint**  
*A single-point objective function computation.*
- def **Gradient**  
*A single-point gradient computation.*
- def **Hessian**  
*A single-point Hessian computation.*
- def **FDCheckG**  
*Finite-difference checker for the objective function gradient.*
- def **FDCheckH**  
*Finite-difference checker for the objective function Hessian.*
- def **readchk**  
*Read the checkpoint file for the main optimizer.*
- def **writechk**  
*Write the checkpoint file for the main optimizer.*
- def **\_\_setattr\_\_**
- def **set\_option**

## Public Attributes

- **OptTab**  
*A list of all the things we can ask the optimizer to do.*
- **mvals\_bak**  
*The root directory.*
- **failmsg**  
*Print a special message on failure.*
- **Objective**  
*The objective function (needs to pass in when I instantiate)*
- **bhyp**  
*Whether the penalty function is hyperbolic.*
- **FF**  
*The force field itself.*
- **uncert**

*Target types which introduce uncertainty into the objective function.*

- `bakdir`
- `resdir`
- `excision`

*The indices to be excluded from the Hessian update.*

- `np`

*Number of parameters.*

- `mvals0`

*The original parameter values.*

- `h`
- `chk`
- `H`
- `dx`
- `Val`
- `Grad`
- `Hess`
- `Penalty`
- `prev_bad`
- `xk_prev`
- `x_prev`
- `x_best`
- `verbose_options`
- `PrintOptionDict`

### 8.55.1 Detailed Description

`Optimizer` class.

Contains several methods for numerical optimization.

For various reasons, the optimizer depends on the force field and fitting targets (i.e. we cannot treat it as a fully independent numerical optimizer). The dependency is rather weak which suggests that I can remove it someday.

Definition at line 51 of file optimizer.py.

### 8.55.2 Constructor & Destructor Documentation

`def forcebalance.optimizer.Optimizer.__init__ ( self, options, Objective, FF )` Create an `Optimizer` object.

The optimizer depends on both the FF and the fitting targets so there is a chain of dependencies: FF → FitSim → `Optimizer`, and FF → `Optimizer`

Here's what we do:

- Take options from the parser
- Pass in the objective function, force field, all fitting targets

Definition at line 64 of file optimizer.py.

### 8.55.3 Member Function Documentation

`def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited]` Definition at line 28 of file \_\_init\_\_.py.

`def forcebalance.optimizer.Optimizer.adjh ( self, trust )` Definition at line 334 of file optimizer.py.

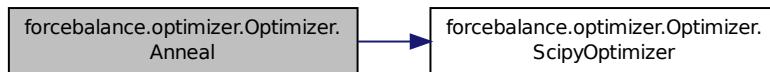
```
def forcebalance.optimizer.Optimizer.Anneal ( self ) Use SciPy's built-in simulated annealing algorithm to optimize the parameters.
```

See Also

[Optimizer::ScipyOptimizer](#)

Definition at line 1098 of file optimizer.py.

Here is the call graph for this function:



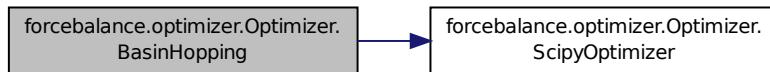
```
def forcebalance.optimizer.Optimizer.BasinHopping ( self ) Use SciPy's built-in basin hopping algorithm to optimize the parameters.
```

See Also

[Optimizer::ScipyOptimizer](#)

Definition at line 1113 of file optimizer.py.

Here is the call graph for this function:



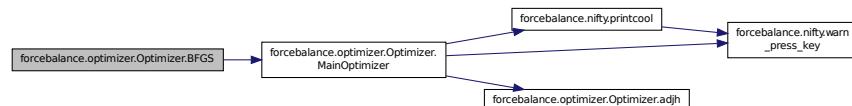
```
def forcebalance.optimizer.Optimizer.BFGS ( self ) Optimize the force field parameters using the BFGS method; currently the recommended choice () .
```

See Also

[MainOptimizer](#))

Definition at line 846 of file optimizer.py.

Here is the call graph for this function:



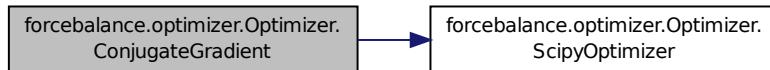
```
def forcebalance.optimizer.Optimizer.ConjugateGradient ( self ) Use SciPy's built-in conjugate gradient algorithm to optimize the parameters.
```

See Also

[Optimizer::ScipyOptimizer](#)

Definition at line 1103 of file optimizer.py.

Here is the call graph for this function:

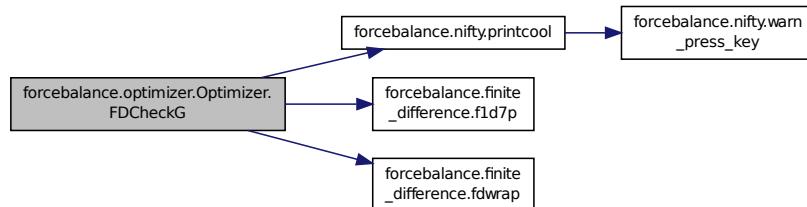


```
def forcebalance.optimizer.Optimizer.FDCheckG ( self ) Finite-difference checker for the objective function gradient.
```

For each element in the gradient, use a five-point finite difference stencil to compute a finite-difference derivative, and compare it to the analytic result.

Definition at line 1222 of file optimizer.py.

Here is the call graph for this function:



```
def forcebalance.optimizer.Optimizer.FDCheckH ( self ) Finite-difference checker for the objective function Hessian.
```

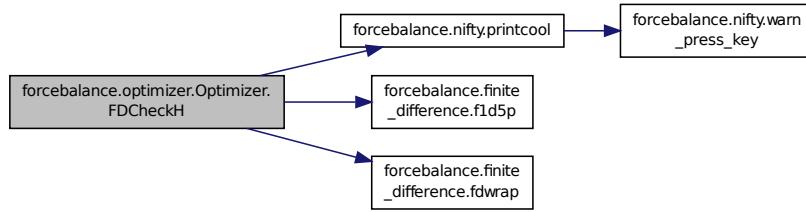
For each element in the Hessian, use a five-point stencil in both parameter indices to compute a finite-difference derivative, and compare it to the analytic result.

This is meant to be a foolproof checker, so it is pretty slow. We could write a faster checker if we assumed we had accurate first derivatives, but it's better to not make that assumption.

The second derivative is computed by double-wrapping the objective function via the 'wrap2' function.

Definition at line 1254 of file optimizer.py.

Here is the call graph for this function:



**def forcebalance.optimizer.Optimizer.GeneticAlgorithm ( self )** Genetic algorithm, under development.  
It currently works but a genetic algorithm is more like a concept; i.e. there is no single way to implement it.

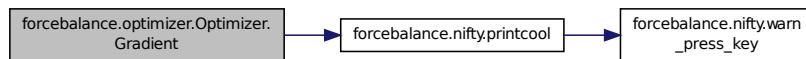
**Todo** Massive parallelization hasn't been implemented yet

Definition at line 995 of file optimizer.py.

**def forcebalance.optimizer.Optimizer.Gradient ( self )** A single-point gradient computation.

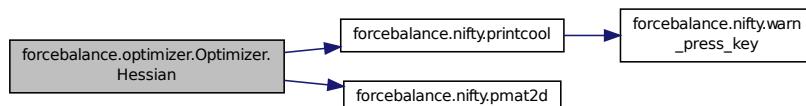
Definition at line 1197 of file optimizer.py.

Here is the call graph for this function:



**def forcebalance.optimizer.Optimizer.Hessian ( self )** A single-point Hessian computation.  
Definition at line 1205 of file optimizer.py.

Here is the call graph for this function:



**def forcebalance.optimizer.Optimizer.MainOptimizer ( self, b\_BFGS = 0 )** The main ForceBalance adaptive trust-radius pseudo-Newton optimizer.  
Tried and true in many situations. :)

Usually this function is called with the BFGS or NewtonRaphson method. The NewtonRaphson method is consistently the best method I have, because I always provide at least an approximate Hessian to the objective function. The BFGS method works well, but if gradients are cheap the SciPy\_BFGS method also works nicely.

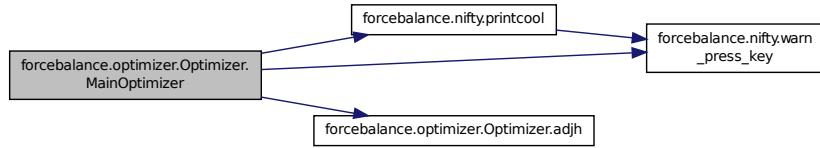
The method adaptively changes the step size. If the step is sufficiently good (i.e. the objective function goes down by a large fraction of the predicted decrease), then the step size is increased; if the step is bad, then it rejects the step and tries again.

The optimization is terminated after either a function value or step size tolerance is reached.

```
@param[in] b_BFGS Switch to use BFGS (True) or Newton-Raphson (False)
```

Definition at line 368 of file optimizer.py.

Here is the call graph for this function:



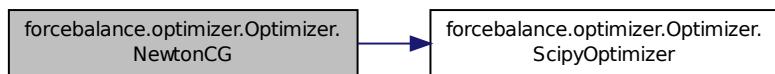
**def forcebalance.optimizer.Optimizer.NewtonCG ( self )** Use SciPy's built-in Newton-CG (fmin\_ncg) algorithm to optimize the parameters.

See Also

[Optimizer::ScipyOptimizer](#)

Definition at line 1123 of file optimizer.py.

Here is the call graph for this function:



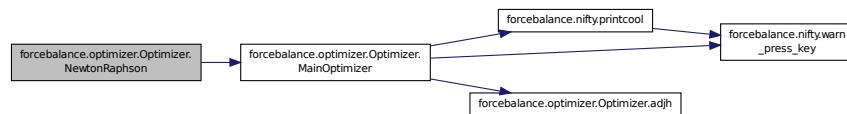
**def forcebalance.optimizer.Optimizer.NewtonRaphson ( self )** Optimize the force field parameters using the Newton-Raphson method (.

See Also

[MainOptimizer](#))

Definition at line 841 of file optimizer.py.

Here is the call graph for this function:



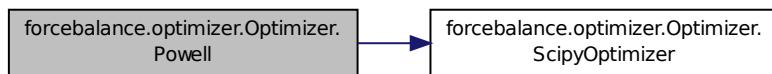
**def forcebalance.optimizer.Optimizer.Powell ( self )** Use SciPy's built-in Powell direction-set algorithm to optimize the parameters.

See Also

[Optimizer::ScipyOptimizer](#)

Definition at line 1093 of file optimizer.py.

Here is the call graph for this function:

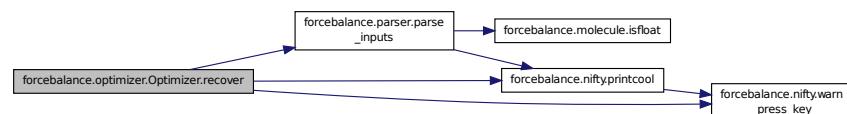


**def forcebalance.optimizer.Optimizer.readchk ( self )** Read the checkpoint file for the main optimizer.

Definition at line 1282 of file optimizer.py.

**def forcebalance.optimizer.Optimizer.recover ( self )** Definition at line 206 of file optimizer.py.

Here is the call graph for this function:

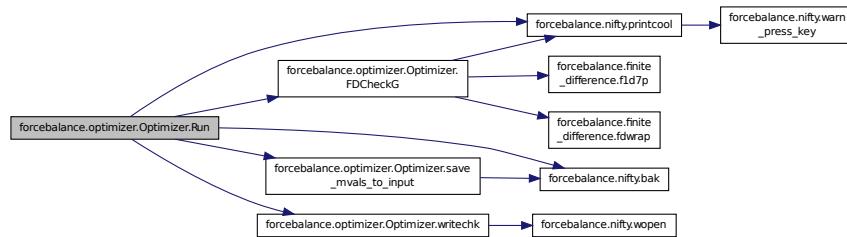


**def forcebalance.optimizer.Optimizer.Run ( self )** Call the appropriate optimizer.

This is the method we might want to call from an executable.

Definition at line 284 of file optimizer.py.

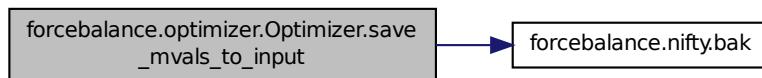
Here is the call graph for this function:



**def forcebalance.optimizer.Optimizer.save\_mvals\_to\_input ( self, mvals )** Write a new input file (s\_save.in) containing the current mathematical parameters.

Definition at line 243 of file optimizer.py.

Here is the call graph for this function:



**def forcebalance.optimizer.Optimizer.Scan\_Values ( self, MathPhys = 1 )** Scan through parameter values.

This option is activated using the inputs:

```
1 scan[mp]vals
2 scan_vals low:hi:nsteps
3 scan_idxnum (number) -or-
4 scan_idxname (name)
```

This method goes to the specified parameter indices and scans through the supplied values, evaluating the objective function at every step.

I hope this method will be useful for people who just want to look at changing one or two parameters and seeing how it affects the force field performance.

**Todo** Maybe a multidimensional grid can be done.

#### Parameters

in	MathPhys	Switch to use mathematical (True) or physical (False) parameters.
----	----------	---

Definition at line 1149 of file optimizer.py.

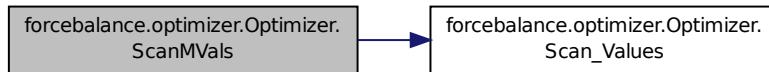
```
def forcebalance.optimizer.Optimizer.ScanMVals ( self ) Scan through the mathematical parameter space.
```

See Also

[Optimizer::ScanValues](#)

Definition at line 1181 of file optimizer.py.

Here is the call graph for this function:



```
def forcebalance.optimizer.Optimizer.ScanPVals ( self ) Scan through the physical parameter space.
```

See Also

[Optimizer::ScanValues](#)

Definition at line 1186 of file optimizer.py.

Here is the call graph for this function:



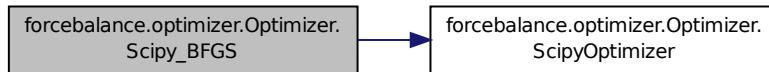
```
def forcebalance.optimizer.Optimizer.Scipy_BFGS ( self ) Use SciPy's built-in BFGS algorithm to optimize the\nparameters.
```

See Also

[Optimizer::ScipyOptimizer](#)

Definition at line 1108 of file optimizer.py.

Here is the call graph for this function:



```
def forcebalance.optimizer.Optimizer.ScipyOptimizer ( self, Algorithm = "None" ) Driver for SciPy optimizations.
```

Using any of the SciPy optimizers requires that SciPy is installed. This method first defines several wrappers around the objective function that the SciPy optimizers can use. Then it calls the algorithm itself.

Parameters

in	Algorithm	The optimization algorithm to use, for example 'powell', 'simplex' or 'anneal'
----	-----------	--

Definition at line 859 of file optimizer.py.

```
def forcebalance.BaseClass.set_option ( self, in_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

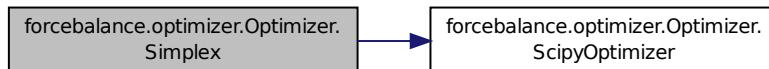
```
def forcebalance.optimizer.Optimizer.Simplex ( self ) Use SciPy's built-in simplex algorithm to optimize the parameters.
```

See Also

[Optimizer::ScipyOptimizer](#)

Definition at line 1088 of file optimizer.py.

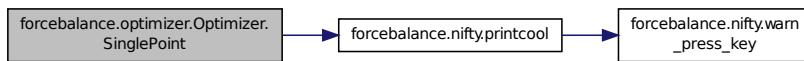
Here is the call graph for this function:



```
def forcebalance.optimizer.Optimizer.SinglePoint ( self ) A single-point objective function computation.
```

Definition at line 1191 of file optimizer.py.

Here is the call graph for this function:



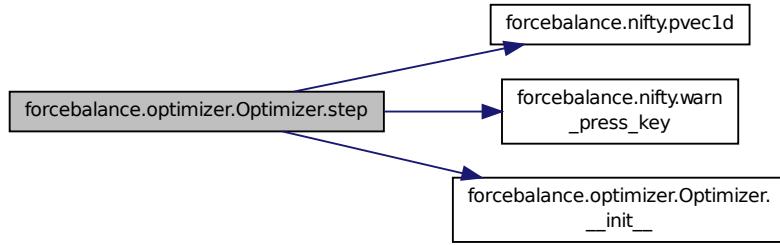
```
def forcebalance.optimizer.Optimizer.step ( self, xk, data, trust ) Computes the next step in the parameter space.
```

There are lots of tricks here that I will document later.

```
@param[in] G The gradient  
@param[in] H The Hessian  
@param[in] trust The trust radius
```

Definition at line 659 of file optimizer.py.

Here is the call graph for this function:



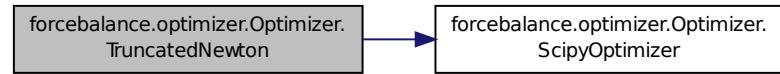
**def forcebalance.optimizer.Optimizer.TruncatedNewton ( *self* )** Use SciPy's built-in truncated Newton (`fmin_tnc`) algorithm to optimize the parameters.

See Also

[Optimizer::ScipyOptimizer](#)

Definition at line 1118 of file optimizer.py.

Here is the call graph for this function:



**def forcebalance.optimizer.Optimizer.writechk ( *self* )** Write the checkpoint file for the main optimizer.

Definition at line 1294 of file optimizer.py.

Here is the call graph for this function:



#### 8.55.4 Member Data Documentation

**forcebalance.optimizer.Optimizer.bakdir** Definition at line 178 of file optimizer.py.

**forcebalance.optimizer.Optimizer.bhyp** Whether the penalty function is hyperbolic.

Definition at line 172 of file optimizer.py.

**forcebalance.optimizer.Optimizer.chk** Definition at line 491 of file optimizer.py.

**forcebalance.optimizer.Optimizer.dx** Definition at line 688 of file optimizer.py.

**forcebalance.optimizer.Optimizer.excision** The indices to be excluded from the Hessian update.  
Definition at line 185 of file optimizer.py.

**forcebalance.optimizer.Optimizer.failmsg** Print a special message on failure.  
Definition at line 164 of file optimizer.py.

**forcebalance.optimizer.Optimizer.FF** The force field itself.  
Definition at line 174 of file optimizer.py.

**forcebalance.optimizer.Optimizer.Grad** Definition at line 690 of file optimizer.py.

**forcebalance.optimizer.Optimizer.h** Definition at line 339 of file optimizer.py.

**forcebalance.optimizer.Optimizer.H** Definition at line 687 of file optimizer.py.

**forcebalance.optimizer.Optimizer.Hess** Definition at line 691 of file optimizer.py.

**forcebalance.optimizer.Optimizer.mvals0** The original parameter values.  
Check derivatives by finite difference after the optimization is over (for good measure)  
Don't print a "result" force field if it's the same as the input.  
Determine the save file name.  
Parse the save file for mvals, if exist.  
Definition at line 192 of file optimizer.py.

**forcebalance.optimizer.Optimizer.mvals\_bak** The root directory.

Determine the output file name.  
The job type Initial step size trust radius Minimum trust radius (for noisy objective functions) Lower bound on Hessian eigenvalue (below this, we add in steepest descent) Lower bound on step size (will fail below this) Guess value for Brent Step size for numerical finite difference When the trust radius get smaller, the finite difference step might need to get smaller. Number of steps to average over Function value convergence threshold Step size convergence threshold Gradient convergence threshold Allow convergence on low quality steps Maximum number of optimization steps For scan[mp]vals: The parameter index to scan over For scan[mp]vals: The parameter name to scan over, it just looks up an index For scan[mp]vals: The values that are fed into the scanner Name of the checkpoint file that we're reading in Name of the checkpoint file that we're writing out Whether to write the checkpoint file at every step Adaptive trust radius adjustment factor Adaptive trust radius adjustment damping Whether to print gradient during each step of the optimization Whether to print Hessian during each step of the optimization Whether to print parameters during each step of the optimization Error tolerance (if objective function rises by less than this, then the optimizer will forge ahead!) Search tolerance (The Hessian diagonal search will stop if the change is below this threshold) Whether to make backup files Name of the original input file Number of convergence criteria that must be met Only backup the "mvals" input file once per calculation.

Clone the input file to the output,  
Definition at line 162 of file optimizer.py.

**forcebalance.optimizer.Optimizer.np** Number of parameters.  
Definition at line 188 of file optimizer.py.

**forcebalance.optimizer.Optimizer.Objective** The objective function (needs to pass in when I instantiate)  
Definition at line 170 of file optimizer.py.

**forcebalance.optimizer.Optimizer.OptTab** A list of all the things we can ask the optimizer to do.  
Definition at line 68 of file optimizer.py.

**forcebalance.optimizer.Optimizer.Penalty** Definition at line 692 of file optimizer.py.

**forcebalance.optimizer.Optimizer.prev\_bad** Definition at line 862 of file optimizer.py.

**forcebalance.BaseClass.PrintOptionDict** [**inherited**] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.optimizer.Optimizer.resdir** Definition at line 179 of file optimizer.py.

**forcebalance.optimizer.Optimizer.uncert** Target types which introduce uncertainty into the objective function.  
Will re-evaluate the objective function when an optimization step is rejected  
Definition at line 177 of file optimizer.py.

**forcebalance.optimizer.Optimizer.Val** Definition at line 689 of file optimizer.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.optimizer.Optimizer.x\_best** Definition at line 865 of file optimizer.py.

**forcebalance.optimizer.Optimizer.x\_prev** Definition at line 864 of file optimizer.py.

**forcebalance.optimizer.Optimizer.xk\_prev** Definition at line 863 of file optimizer.py.  
The documentation for this class was generated from the following file:

- [optimizer.py](#)

## 8.56 forcebalance.objective.Penalty Class Reference

[Penalty](#) functions for regularizing the force field optimizer.

### Public Member Functions

- def [\\_\\_init\\_\\_](#)
- def [compute](#)
- def [L2\\_norm](#)

*Harmonic L2-norm constraints.*
- def [HYP](#)

*Hyperbolic constraints.*
- def [FUSE](#)
- def [FUSE\\_BARRIER](#)
- def [FUSE\\_L0](#)

## Public Attributes

- [fadd](#)
- [fmul](#)
- [a](#)
- [b](#)
- [FF](#)
- [ptyp](#)
- [Pen\\_Tab](#)
- [spacings](#)

*Find exponential spacings.*

## Static Public Attributes

- [dictionary Pen\\_Names](#)

### 8.56.1 Detailed Description

[Penalty](#) functions for regularizing the force field optimizer.

The purpose for this module is to improve the behavior of our optimizer; essentially, our problem is fraught with 'linear dependencies', a.k.a. directions in the parameter space that the objective function does not respond to. This would happen if a parameter is just plain useless, or if there are two or more parameters that describe the same thing.

To accomplish these objectives, a penalty function is added to the objective function. Generally, the more the parameters change (i.e. the greater the norm of the parameter vector), the greater the penalty. Note that this is added on after all of the other contributions have been computed. This only matters if the penalty 'multiplies' the objective function:  $\text{Obj} + \text{Obj} * \text{Penalty}$ , but we also have the option of an additive penalty:  $\text{Obj} + \text{Penalty}$ .

Statistically, this is called regularization. If the penalty function is the norm squared of the parameter vector, it is called ridge regression. There is also the option of using simply the norm, and this is called lasso, but I think it presents problems for the optimizer that I need to work out.

Note that the penalty functions can be considered as part of a 'maximum likelihood' framework in which we assume a PRIOR PROBABILITY of the force field parameters around their initial values. The penalty function is related to the prior by an exponential. Ridge regression corresponds to a Gaussian prior and lasso corresponds to an exponential prior. There is also 'elastic net regression' which interpolates between Gaussian and exponential using a tuning parameter.

Our priors are adjustable too - there is one parameter, which is the width of the distribution. We can even use a noninformative prior for the distribution widths (hyperprior!). These are all important things to consider later.

Importantly, note that here there is no code that treats the distribution width. That is because the distribution width is wrapped up in the rescaling factors, which is essentially a coordinate transformation on the parameter space. More documentation on this will follow, perhaps in the 'rsmake' method.

Definition at line 318 of file `objective.py`.

### 8.56.2 Constructor & Destructor Documentation

```
def forcebalance.objective.Penalty.__init__( self, User_Option, ForceField, Factor_Add = 0.0, Factor_Mult = 0.0, Factor_B = 0.1, Alpha = 1.0 ) Definition at line 324 of file objective.py.
```

### 8.56.3 Member Function Documentation

```
def forcebalance.objective.Penalty.compute( self, mvals, Objective ) Definition at line 350 of file objective.py.
```

```
def forcebalance.objective.Penalty.FUSE ( self, mvals )
```

Definition at line 410 of file objective.py.  
Here is the call graph for this function:



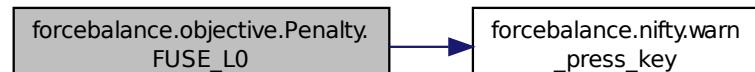
```
def forcebalance.objective.Penalty.FUSE_BARRIER ( self, mvals )
```

Definition at line 451 of file objective.py.  
Here is the call graph for this function:



```
def forcebalance.objective.Penalty.FUSE_L0 ( self, mvals )
```

Definition at line 493 of file objective.py.  
Here is the call graph for this function:



```
def forcebalance.objective.Penalty.HYP ( self, mvals )
```

Hyperbolic constraints.  
Depending on the 'b' parameter, the smaller it is, the closer we are to an L1-norm constraint. If we use these, we expect a properly-behaving optimizer to make several of the parameters very nearly zero (which would be cool).

#### Parameters

in	mvals	The parameter vector
----	-------	----------------------

#### Returns

DC0 The hyperbolic penalty

DC1 The gradient

DC2 The Hessian

Definition at line 402 of file objective.py.

```
def forcebalance.objective.Penalty.L2_norm ( self, mvals ) Harmonic L2-norm constraints.
```

These are the ones that I use the most often to regularize my optimization.

Parameters

in	mvals	The parameter vector
----	-------	----------------------

Returns

DC0 The norm squared of the vector

DC1 The gradient of DC0

DC2 The Hessian (just a constant)

Definition at line 382 of file objective.py.

#### 8.56.4 Member Data Documentation

**forcebalance.objective.Penalty.a** Definition at line 327 of file objective.py.

**forcebalance.objective.Penalty.b** Definition at line 328 of file objective.py.

**forcebalance.objective.Penalty.fadd** Definition at line 325 of file objective.py.

**forcebalance.objective.Penalty.FF** Definition at line 329 of file objective.py.

**forcebalance.objective.Penalty.fmul** Definition at line 326 of file objective.py.

**dictionary forcebalance.objective.Penalty.Pen\_Names [static] Initial value:**

```
1 = {'HYP' : 1, 'HYPER' : 1, 'HYPERBOLIC' : 1, 'L1' : 1, 'HYPERBOLA' : 1,
2           'PARA' : 2, 'PARABOLA' : 2, 'PARABOLIC' : 2, 'L2' : 2, 'QUADRATIC' : 2,
3           'FUSE' : 3, 'FUSION' : 3, 'FUSE_L0' : 4, 'FUSION_L0' : 4, 'FUSION-L0' : 4,
4           'FUSE-BARRIER' : 5, 'FUSE-BARRIER' : 5, 'FUSE_BARRIER' : 5, 'FUSION_BARRIER' : 5}
```

Definition at line 319 of file objective.py.

**forcebalance.objective.Penalty.Pen\_Tab** Definition at line 331 of file objective.py.

**forcebalance.objective.Penalty.ptyp** Definition at line 330 of file objective.py.

**forcebalance.objective.Penalty.spacings** Find exponential spacings.

Definition at line 347 of file objective.py.

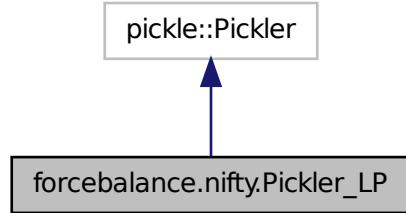
The documentation for this class was generated from the following file:

- [objective.py](#)

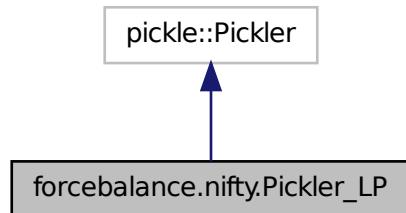
#### 8.57 forcebalance.nifty.Pickler\_LP Class Reference

A subclass of the python Pickler that implements pickling of \_ElementTree types.

Inheritance diagram for forcebalance.nifty.Pickler\_LP:



Collaboration diagram for forcebalance.nifty.Pickler\_LP:



## Public Member Functions

- def [\\_\\_init\\_\\_](#)

### 8.57.1 Detailed Description

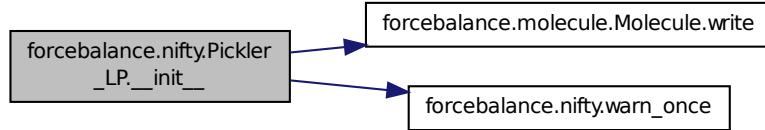
A subclass of the python Pickler that implements pickling of \_ElementTree types.

Definition at line 563 of file nifty.py.

### 8.57.2 Constructor & Destructor Documentation

**def forcebalance.nifty.Pickler\_LP.\_\_init\_\_( self, file, protocol = None )** Definition at line 564 of file nifty.py.

Here is the call graph for this function:

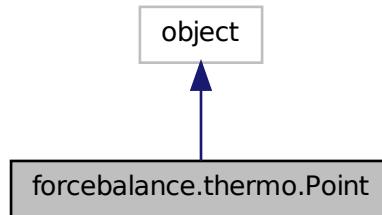


The documentation for this class was generated from the following file:

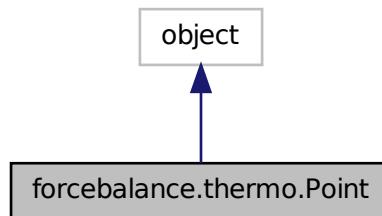
- [nifty.py](#)

## 8.58 forcebalance.thermo.Point Class Reference

Inheritance diagram for forcebalance.thermo.Point:



Collaboration diagram for forcebalance.thermo.Point:



## Public Member Functions

- def [\\_\\_init\\_\\_](#)
- def [\\_\\_str\\_\\_](#)

## Public Attributes

- [idnr](#)
- [ref](#)
- [temperature](#)
- [pressure](#)
- [data](#)

### 8.58.1 Detailed Description

Definition at line 426 of file thermo.py.

### 8.58.2 Constructor & Destructor Documentation

```
def forcebalance.thermo.Point.__init__ ( self, idnr, label = None, refs = None, weights = None, names = None, units = None, temperature = None, pressure = None, data = None )
```

Definition at line 428 of file thermo.py.

### 8.58.3 Member Function Documentation

```
def forcebalance.thermo.Point.__str__ ( self )
```

Definition at line 439 of file thermo.py.

### 8.58.4 Member Data Documentation

**forcebalance.thermo.Point.data** Definition at line 437 of file thermo.py.

**forcebalance.thermo.Point.idnr** Definition at line 429 of file thermo.py.

**forcebalance.thermo.Point.pressure** Definition at line 436 of file thermo.py.

**forcebalance.thermo.Point.ref** Definition at line 430 of file thermo.py.

**forcebalance.thermo.Point.temperature** Definition at line 435 of file thermo.py.

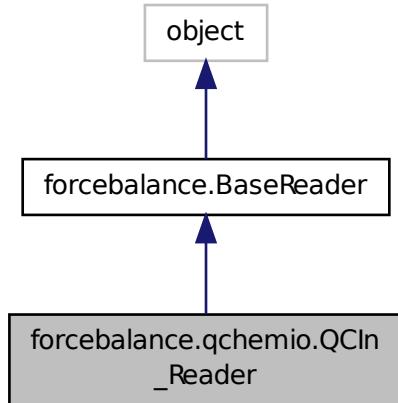
The documentation for this class was generated from the following file:

- [thermo.py](#)

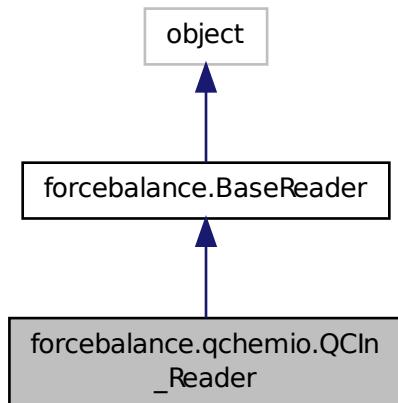
## 8.59 forcebalance.qchemio.QCIn\_ Reader Class Reference

Finite state machine for parsing Q-Chem input files.

Inheritance diagram for forcebalance.qchemio.QCIn\_Reader:



Collaboration diagram for forcebalance.qchemio.QCIn\_Reader:



#### Public Member Functions

- def `_init_`
- def `feed`  
*Feed in a line.*
- def `Split`
- def `Whites`

- def [build.pid](#)

*Returns the parameter type (e.g.*

## Public Attributes

- [atom](#)
- [snum](#)
- [cnum](#)
- [shell](#)
- [pdict](#)
- [sec](#)
- [itype](#)
- [suffix](#)
- [In](#)
- [adict](#)

*The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.*

- [molatom](#)

*The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.*

- [Molecules](#)
- [AtomTypes](#)

### 8.59.1 Detailed Description

Finite state machine for parsing Q-Chem input files.

Definition at line 31 of file qchemio.py.

### 8.59.2 Constructor & Destructor Documentation

**def forcebalance.qchemio.QCIn\_Reader.\_\_init\_\_ ( self, fnm )** Definition at line 33 of file qchemio.py.

### 8.59.3 Member Function Documentation

**def forcebalance.BaseReader.build\_pid ( self, pfd ) [inherited]** Returns the parameter type (e.g. K in BONDSK) based on the current interaction type.

Both the 'pdict' dictionary (see [gmxio.pdict](#)) and the interaction type 'state' (here, BONDS) are needed to get the parameter type.

If, however, 'pdict' does not contain the ptype value, a suitable substitute is simply the field number.

Note that if the interaction type state is not set, then it defaults to the file name, so a generic parameter ID is 'filename.line.num.field.num'

Definition at line 124 of file \_\_init\_\_.py.

**def forcebalance.qchemio.QCIn\_Reader.feed ( self, line )** Feed in a line.

Parameters

in	line	The line of data
----	------	------------------

Definition at line 48 of file qchemio.py.

**def forcebalance.BaseReader.Split ( self, line ) [inherited]** Definition at line 99 of file \_\_init\_\_.py.

**def forcebalance.BaseReader.Whites ( self, line ) [inherited]** Definition at line 102 of file \_\_init\_\_.py.

#### 8.59.4 Member Data Documentation

**forcebalance.BaseReader.adict [inherited]** The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.

Definition at line 89 of file `__init__.py`.

**forcebalance.qchemio.QCIn\_Reader.atom** Definition at line 36 of file `qchemio.py`.

**forcebalance.BaseReader.AtomTypes [inherited]** Definition at line 97 of file `__init__.py`.

**forcebalance.qchemio.QCIn\_Reader.cnum** Definition at line 38 of file `qchemio.py`.

**forcebalance.qchemio.QCIn\_Reader.itype** Definition at line 67 of file `qchemio.py`.

**forcebalance.BaseReader.In [inherited]** Definition at line 84 of file `__init__.py`.

**forcebalance.BaseReader.molatom [inherited]** The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.

`self.moleculedict = OrderedDict()` The listing of 'RES:ATOMNAMES' for atom names in the line This is obviously a placeholder.

Definition at line 94 of file `__init__.py`.

**forcebalance.BaseReader.Molecules [inherited]** Definition at line 96 of file `__init__.py`.

**forcebalance.qchemio.QCIn\_Reader.pdict** Definition at line 40 of file `qchemio.py`.

**forcebalance.qchemio.QCIn\_Reader.sec** Definition at line 58 of file `qchemio.py`.

**forcebalance.qchemio.QCIn\_Reader.shell** Definition at line 39 of file `qchemio.py`.

**forcebalance.qchemio.QCIn\_Reader.snum** Definition at line 37 of file `qchemio.py`.

**forcebalance.qchemio.QCIn\_Reader.suffix** Definition at line 72 of file `qchemio.py`.

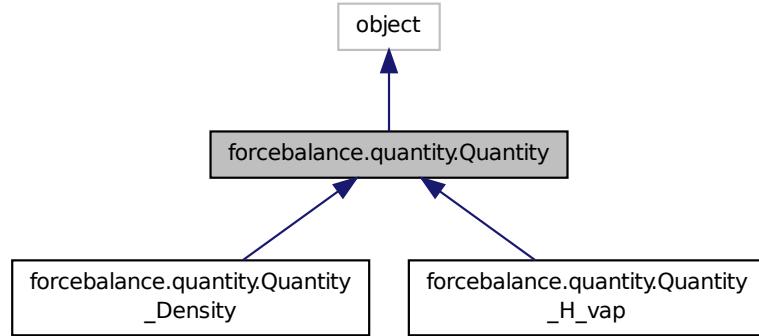
The documentation for this class was generated from the following file:

- [qchemio.py](#)

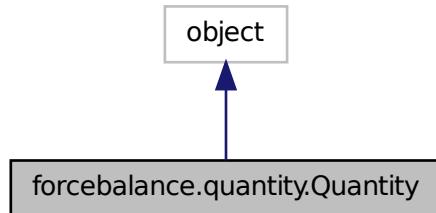
## 8.60 forcebalance.quantity.Quantity Class Reference

Base class for thermodynamical quantity used for fitting.

Inheritance diagram for forcebalance.quantity.Quantity:



Collaboration diagram for forcebalance.quantity.Quantity:



### Public Member Functions

- def `__init__`
- def `__str__`
- def `extract`

*Calculate and extract the quantity from MD results.*

### Public Attributes

- `name`
- `engname`
- `temperature`
- `pressure`

### 8.60.1 Detailed Description

Base class for thermodynamical quantity used for fitting.

This can be any experimental data that can be calculated as an ensemble average from a simulation.

**Data attributes** name : string Identifier for the quantity that is specified in quantities in Target options. engname : string Use this engine to extract the quantity from the simulation results. At present, only gromacs is supported. temperature : float Calculate the quantity at this temperature (in K). pressure : float Calculate the quantity at this pressure (in bar).

Definition at line 88 of file quantity.py.

### 8.60.2 Constructor & Destructor Documentation

```
def forcebalance.quantity.Quantity.__init__ ( self, engname, temperature, pressure, name = None )
```

Definition at line 89 of file quantity.py.

### 8.60.3 Member Function Documentation

```
def forcebalance.quantity.Quantity.__str__ ( self )
```

Definition at line 95 of file quantity.py.

**Returns** result : (float, float, np.array) The returned tuple is (Q, Qerr, Qgrad), where Q is the calculated quantity, Qerr is the calculated standard deviation of the quantity, and Qgrad is a M-array with the calculated gradients for the quantity, with M being the number of force field parameters that are being fitted.

How this is done depends on the quantity and the engine so this must be implemented in the subclass.

**Parameters** engines : list A list of Engine objects that are required to calculate the quantity. FF : FF Force field object. mvals : list Mathematical parameter values. h : float Finite difference step size. AGrad : Boolean Switch that turns derivatives on or off; if off, return all zeros.

**Returns** result : (float, float, np.array) The returned tuple is (Q, Qerr, Qgrad), where Q is the calculated quantity, Qerr is the calculated standard deviation of the quantity, and Qgrad is a M-array with the calculated gradients for the quantity, with M being the number of force field parameters that are being fitted.

Definition at line 126 of file quantity.py.

### 8.60.4 Member Data Documentation

**forcebalance.quantity.Quantity.engname** Definition at line 91 of file quantity.py.

**forcebalance.quantity.Quantity.name** Definition at line 90 of file quantity.py.

**forcebalance.quantity.Quantity.pressure** Definition at line 93 of file quantity.py.

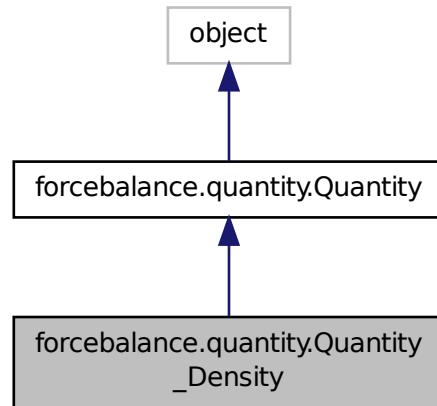
**forcebalance.quantity.Quantity.temperature** Definition at line 92 of file quantity.py.

The documentation for this class was generated from the following file:

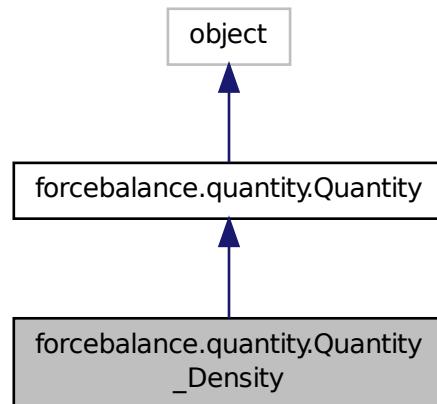
- [quantity.py](#)

## 8.61 forcebalance.quantity.Quantity\_Density Class Reference

Inheritance diagram for forcebalance.quantity.Quantity\_Density:



Collaboration diagram for forcebalance.quantity.Quantity\_Density:



### Public Member Functions

- def `__init__`  
*Density.*
- def `extract`
- def `__str__`

- def [extract](#)

*Calculate and extract the quantity from MD results.*

## Public Attributes

- [name](#)
- [engname](#)
- [temperature](#)
- [pressure](#)

### 8.61.1 Detailed Description

Definition at line 130 of file `quantity.py`.

### 8.61.2 Constructor & Destructor Documentation

```
def forcebalance.quantity.Quantity_Density.__init__( self, engname, temperature, pressure, name = None ) Density.
```

Definition at line 133 of file `quantity.py`.

### 8.61.3 Member Function Documentation

```
def forcebalance.quantity.Quantity.__str__( self ) [inherited] Definition at line 95 of file quantity.py.
```

```
def forcebalance.quantity.Quantity.extract( self, engines, FF, mvals, h, AGrad = True ) [inherited]
```

Calculate and extract the quantity from MD results.

How this is done depends on the quantity and the engine so this must be implemented in the subclass.

**Parameters** engines : list A list of Engine objects that are required to calculate the quantity. FF : FF Force field object. mvals : list Mathematical parameter values. h : float Finite difference step size. AGrad : Boolean Switch that turns derivatives on or off; if off, return all zeros.

**Returns** result : (float, float, np.array) The returned tuple is (Q, Qerr, Qgrad), where Q is the calculated quantity, Qerr is the calculated standard deviation of the quantity, and Qgrad is a M-array with the calculated gradients for the quantity, with M being the number of force field parameters that are being fitted.

Definition at line 126 of file `quantity.py`.

```
def forcebalance.quantity.Quantity_Density.extract( self, engines, FF, mvals, h, pgrad, AGrad = True )
```

Definition at line 138 of file `quantity.py`.

### 8.61.4 Member Data Documentation

**forcebalance.quantity.Quantity\_Density.engname** Definition at line 151 of file `quantity.py`.

**forcebalance.quantity.Quantity\_Density.name** Definition at line 136 of file `quantity.py`.

**forcebalance.quantity.Quantity.pressure** [inherited] Definition at line 93 of file `quantity.py`.

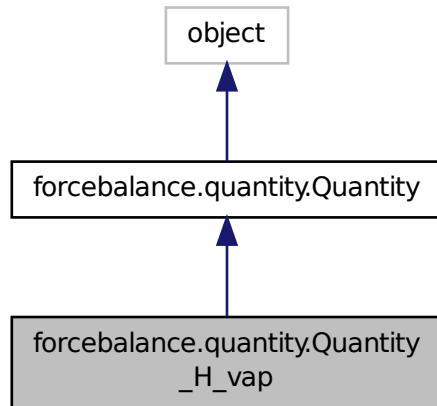
**forcebalance.quantity.Quantity.temperature** [inherited] Definition at line 92 of file `quantity.py`.

The documentation for this class was generated from the following file:

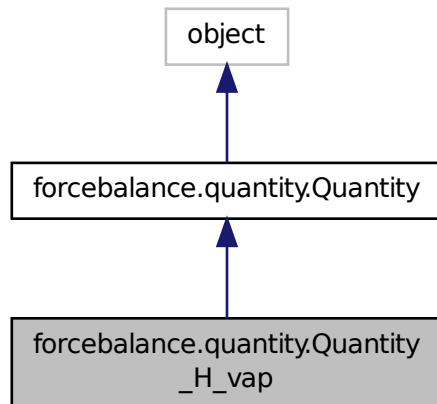
- [quantity.py](#)

## 8.62 forcebalance.quantity.Quantity\_H\_vap Class Reference

Inheritance diagram for forcebalance.quantity.Quantity\_H\_vap:



Collaboration diagram for forcebalance.quantity.Quantity\_H\_vap:



### Public Member Functions

- def `__init__`  
*Enthalpy of vaporization.*
- def `extract`
- def `__str__`

- def [extract](#)

*Calculate and extract the quantity from MD results.*

## Public Attributes

- [name](#)
- [engname](#)
- [temperature](#)
- [pressure](#)

### 8.62.1 Detailed Description

Definition at line 201 of file `quantity.py`.

### 8.62.2 Constructor & Destructor Documentation

**def forcebalance.quantity.Quantity.H\_vap.\_\_init\_\_ ( self, engname, temperature, pressure, name = None )**  
Enthalpy of vaporization.

Definition at line 204 of file `quantity.py`.

### 8.62.3 Member Function Documentation

**def forcebalance.quantity.Quantity.\_\_str\_\_ ( self ) [inherited]** Definition at line 95 of file `quantity.py`.

**def forcebalance.quantity.Quantity.extract ( self, engines, FF, mvals, h, AGrad = True ) [inherited]**  
Calculate and extract the quantity from MD results.

How this is done depends on the quantity and the engine so this must be implemented in the subclass.

**Parameters** engines : list A list of Engine objects that are required to calculate the quantity. FF : FF Force field object. mvals : list Mathematical parameter values. h : float Finite difference step size. AGrad : Boolean Switch that turns derivatives on or off; if off, return all zeros.

**Returns** result : (float, float, np.array) The returned tuple is (Q, Qerr, Qgrad), where Q is the calculated quantity, Qerr is the calculated standard deviation of the quantity, and Qgrad is a M-array with the calculated gradients for the quantity, with M being the number of force field parameters that are being fitted.

Definition at line 126 of file `quantity.py`.

**def forcebalance.quantity.Quantity.H\_vap.extract ( self, engines, FF, mvals, h, pgrad, AGrad = True )**  
Definition at line 209 of file `quantity.py`.

### 8.62.4 Member Data Documentation

**forcebalance.quantity.Quantity.H\_vap.engname** Definition at line 229 of file `quantity.py`.

**forcebalance.quantity.Quantity.H\_vap.name** Definition at line 207 of file `quantity.py`.

**forcebalance.quantity.Quantity.pressure [inherited]** Definition at line 93 of file `quantity.py`.

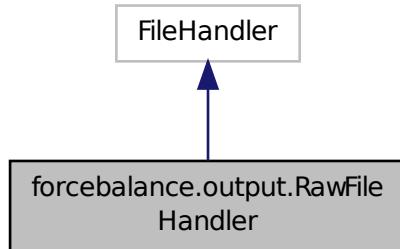
**forcebalance.quantity.Quantity.temperature [inherited]** Definition at line 92 of file `quantity.py`.

The documentation for this class was generated from the following file:

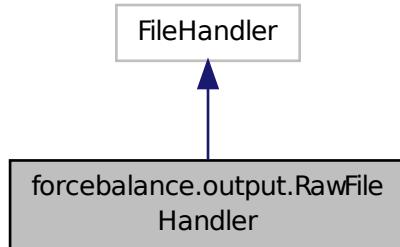
- [quantity.py](#)

## 8.63 forcebalance.output.RawFileHandler Class Reference

Exactly like output.FileHandler except it does no extra formatting before sending logging messages to the file.  
Inheritance diagram for forcebalance.output.RawFileHandler:



Collaboration diagram for forcebalance.output.RawFileHandler:



### Public Member Functions

- def [emit](#)

#### 8.63.1 Detailed Description

Exactly like output.FileHandler except it does no extra formatting before sending logging messages to the file.  
This is more compatible with how output has been displayed in ForceBalance.  
Definition at line 47 of file [output.py](#).

#### 8.63.2 Member Function Documentation

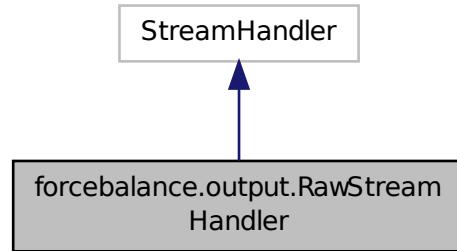
**def forcebalance.output.RawFileHandler.emit ( *self*, *record* )** Definition at line 48 of file [output.py](#).

The documentation for this class was generated from the following file:

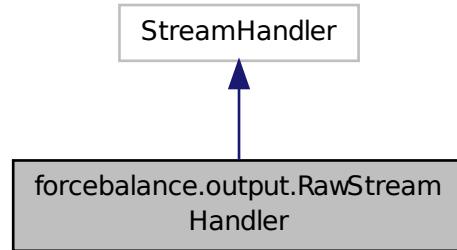
- [output.py](#)

## 8.64 forcebalance.output.RawStreamHandler Class Reference

Exactly like output.StreamHandler except it does no extra formatting before sending logging messages to the stream.  
Inheritance diagram for forcebalance.output.RawStreamHandler:



Collaboration diagram for forcebalance.output.RawStreamHandler:



### Public Member Functions

- def `__init__`
- def `emit`

#### 8.64.1 Detailed Description

Exactly like output.StreamHandler except it does no extra formatting before sending logging messages to the stream.  
This is more compatible with how output has been displayed in ForceBalance. Default stream has also been changed from stderr to stdout  
Definition at line 34 of file output.py.

#### 8.64.2 Constructor & Destructor Documentation

```
def forcebalance.output.RawStreamHandler.__init__ ( self, stream = sys.stdout ) Definition at line 35 of file output.py.
```

### 8.64.3 Member Function Documentation

```
def forcebalance.output.RawStreamHandler.emit ( self, record ) Definition at line 38 of file output.py.
```

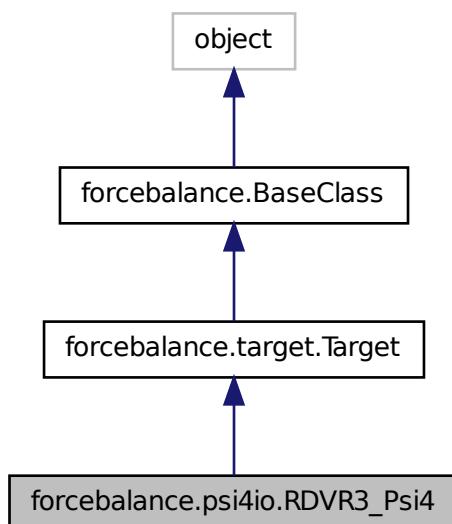
The documentation for this class was generated from the following file:

- [output.py](#)

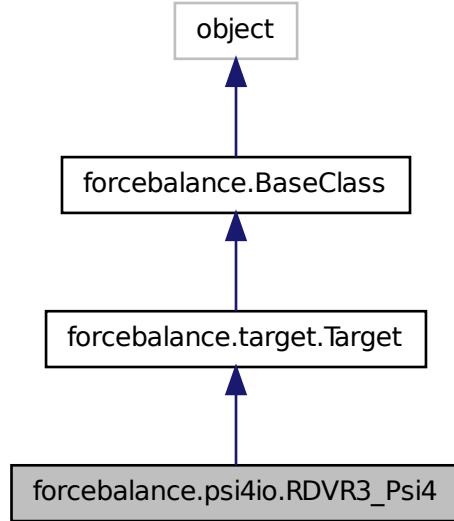
## 8.65 forcebalance.psi4io.RDVR3\_Psi4 Class Reference

Subclass of Target for R-DVR3 grid fitting.

Inheritance diagram for forcebalance.psi4io.RDVR3\_Psi4:



Collaboration diagram for forcebalance.psi4io.RDVR3\_Psi4:



## Public Member Functions

- def `_init_`
- def `indicate`
- def `submit_jobs`

*Create a grid file and a psi4 input file in the absolute path and submit it to the work queue.*

- def `driver`
- def `get`

*LPW 04-17-2013.*

- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*

- def `read_0grads`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*

- def `write_0grads`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*

- def `get_G`

*Computes the objective function contribution and its gradient.*

- def `get_H`

*Computes the objective function contribution and its gradient / Hessian.*

- def `link_from_tempdir`

- def `refresh_temp_directory`

*Back up the temporary directory if desired, delete it and then create a new one.*

- def `check_files`

*Check this directory for the presence of readable files when the 'read' option is set.*

- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `objfiles`  
*Which parameters are differentiated?*
- `objvals`
- `elements`
- `molecules`
- `callderivs`
- `factor`
- `bidirect`
- `tdir`
- `objd`
- `gradd`
- `hdiagd`
- `objective`
- `rd`  
*Root directory of the whole project.*
- `pgrad`  
*Iteration where we turn on zero-gradient skipping.*
- `tempbase`  
*Relative directory of target.*
- `tempdir`
- `rundir`  
`self.tempdir = os.path.join('temp',self.name)` *The directory in which the simulation is running - this can be updated.*
- `FF`  
*Need the forcefield (here for now)*
- `xct`  
*Counts how often the objective function was computed.*
- `gct`

- **hct**  
*Counts how often the gradient was computed.*
- **read\_indicate**  
*Whether to read indicate.log from file when restarting an aborted run.*
- **write\_indicate**  
*Whether to write indicate.log at every iteration (true for all but remote.)*
- **read\_objective**  
*Whether to read objective.p from file when restarting an aborted run.*
- **write\_objective**  
*Whether to write objective.p at every iteration (true for all but remote.)*
- **verbose\_options**
- **PrintOptionDict**

### 8.65.1 Detailed Description

Subclass of Target for R-DVR3 grid fitting.

Main features:

- Multiple molecules are treated as a single target.
- R-DVR3 can only print out the objective function, it cannot print out the residual vector.
- We should be smart enough to mask derivatives.

Definition at line 299 of file psi4io.py.

### 8.65.2 Constructor & Destructor Documentation

```
def forcebalance.psi4io.RDVR3_Psi4.__init__ ( self, options, tgt_opts, forcefield ) Definition at line 302 of file
psi4io.py.
```

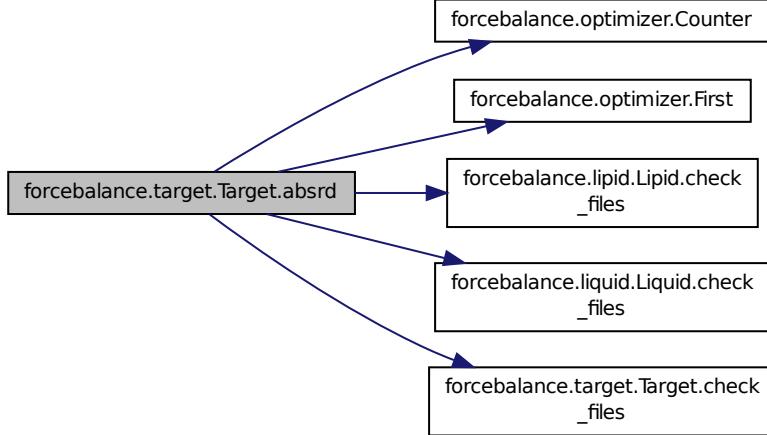
### 8.65.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory
specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:

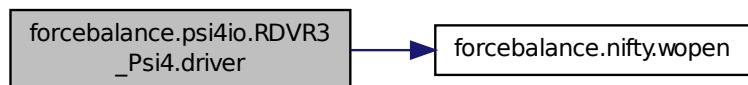


**def forcebalance.target.Target.check\_files( self, there ) [inherited]** Check this directory for the presence of readable files when the 'read' option is set.

Definition at line 364 of file target.py.

**def forcebalance.psi4io.RDVR3.Psi4.driver( self, mvals, d )** Definition at line 415 of file psi4io.py.

Here is the call graph for this function:



**def forcebalance.psi4io.RDVR3.Psi4.get( self, mvals, AGrad=False, AHess=False )** LPW 04-17-2013.

This subroutine builds the objective function from Psi4.

Parameters

in	<i>mvals</i>	Mathematical parameter values
in	<i>AGrad</i>	Switch to turn on analytic gradient

in	AHess	Switch to turn on analytic Hessian
----	-------	------------------------------------

Returns

Answer Contribution to the objective function

Definition at line 452 of file psi4io.py.

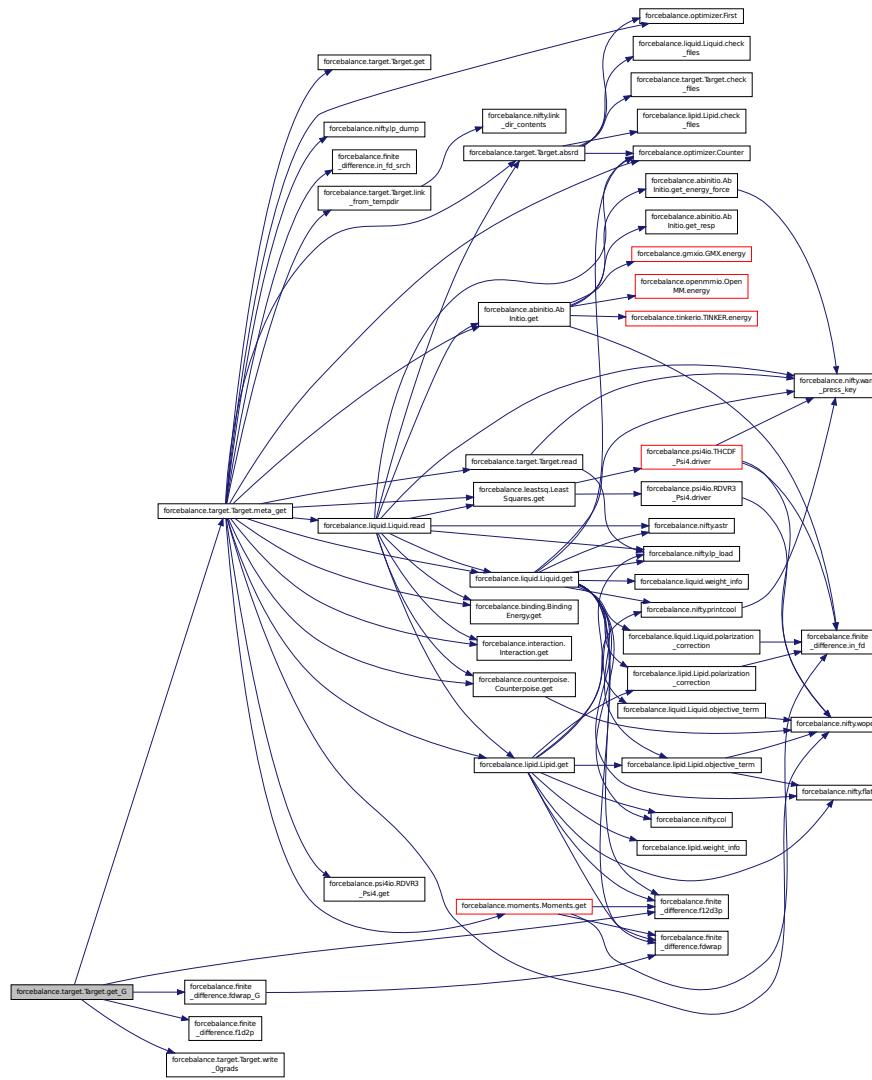
**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



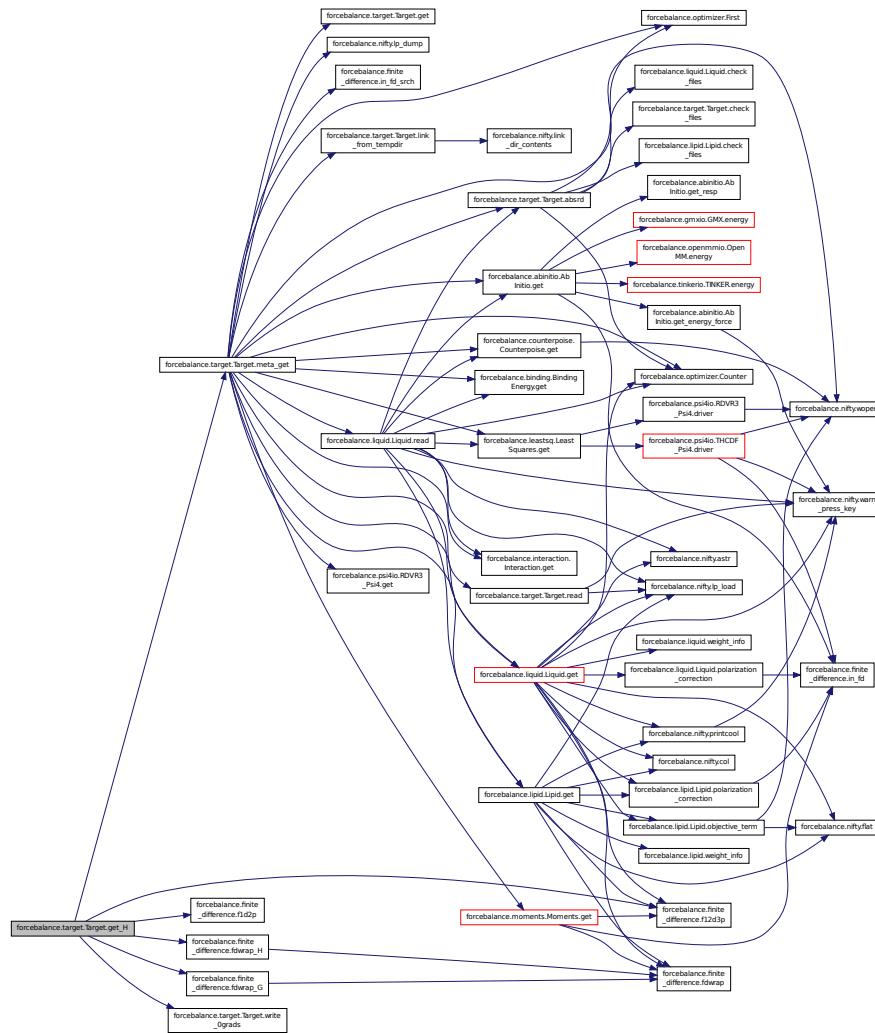
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

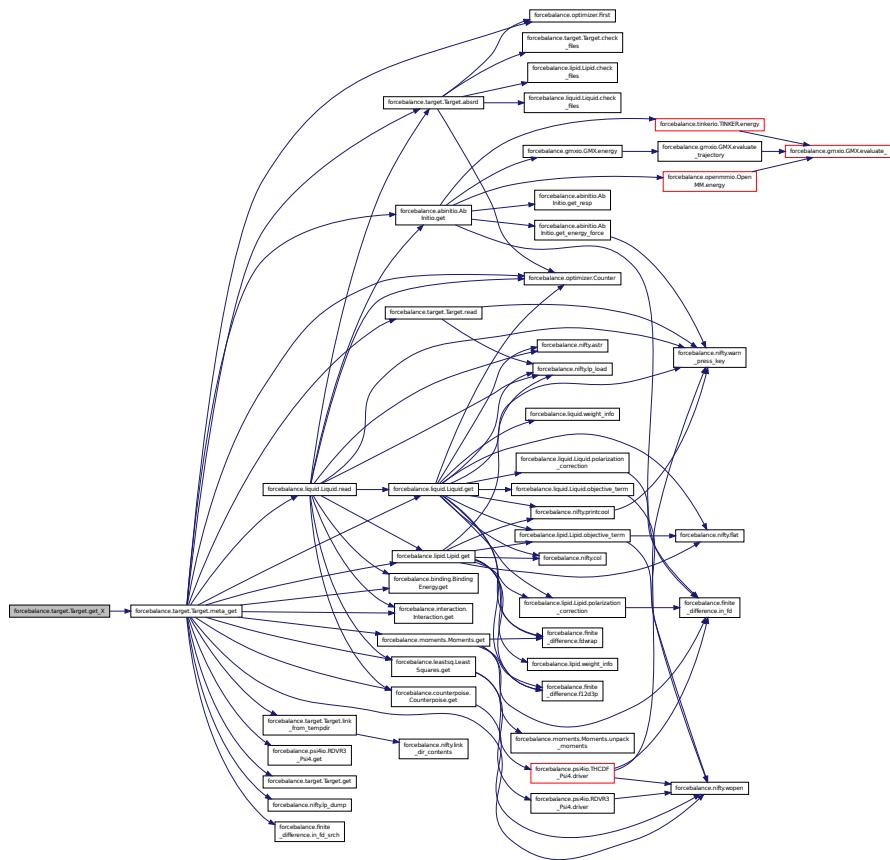
Here is the call graph for this function:



**def forcebalance.target.Target.get.X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

Here is the call graph for this function:



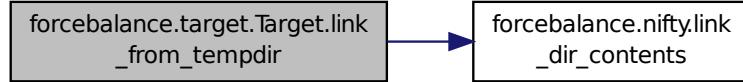
**def forcebalance.ps4io.RDVR3\_Psi4.indicate( self )** Definition at line 342 of file ps4io.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

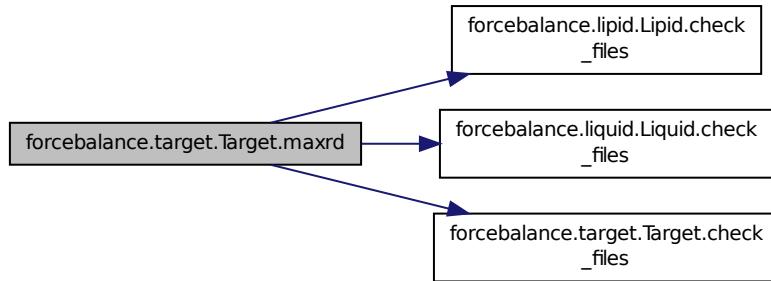
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

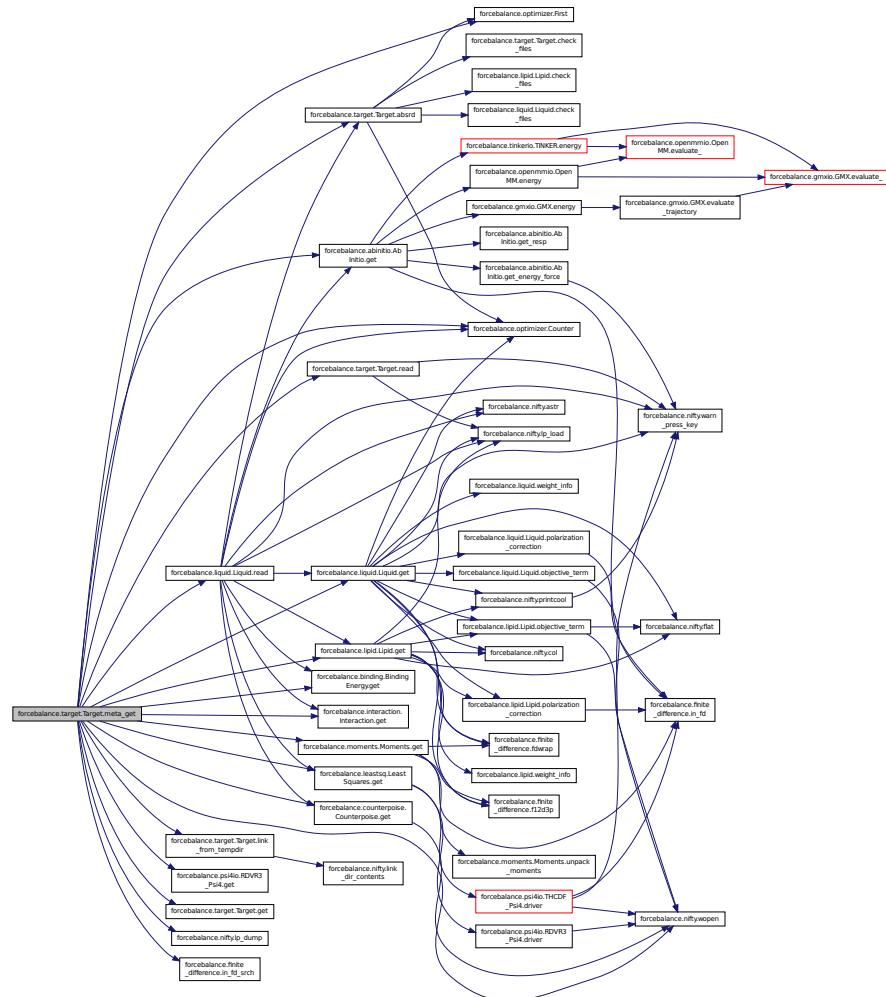


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

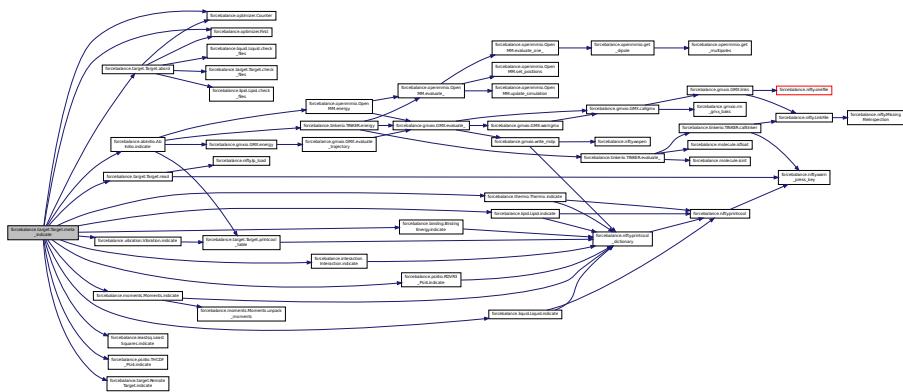
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

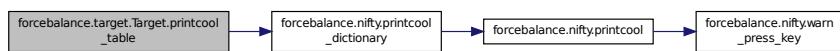
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

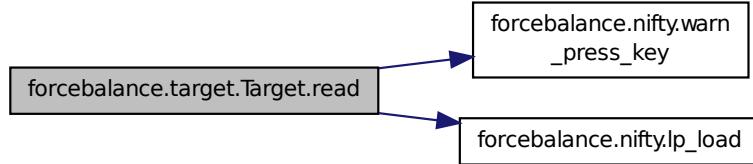
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

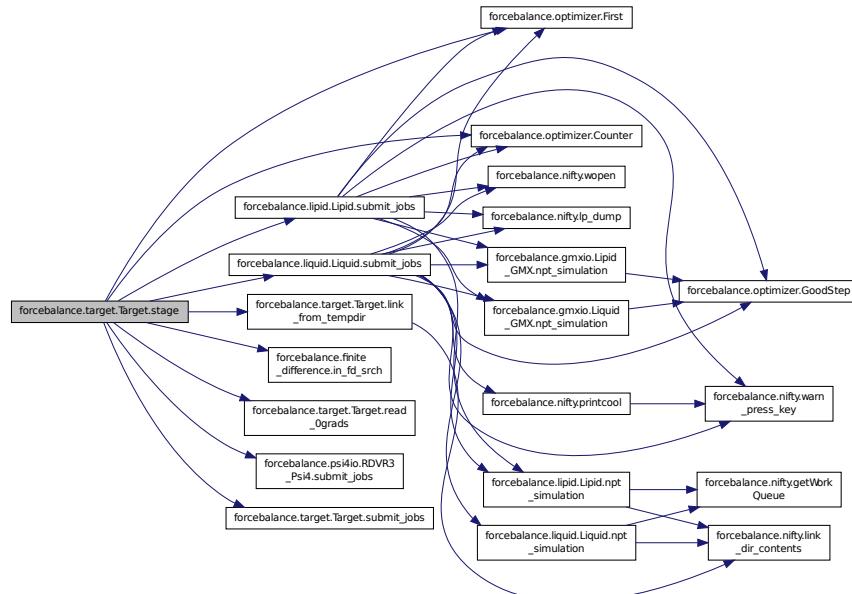
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



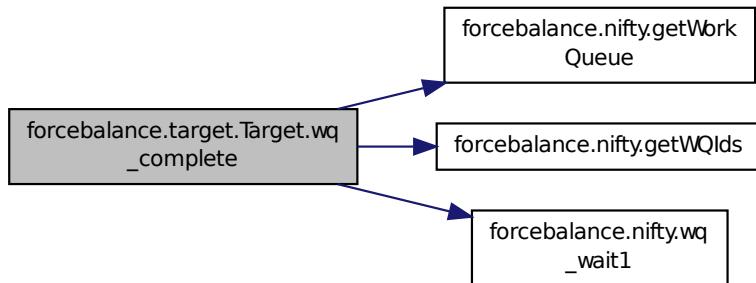
**def forcebalance.psi4io.RDVR3.Psi4.submit\_jobs ( self, mvals, AGrad = True, AHess = True )** Create a grid file and a psi4 input file in the absolute path and submit it to the work queue.

Definition at line 351 of file psi4io.py.

**def forcebalance.target.Target.wq\_complete ( self ) [inherited]** This method determines whether the Work Queue tasks for the current target have completed.

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.65.4 Member Data Documentation

**forcebalance.psi4io.RDVR3\_Psi4.bidirect** Definition at line 314 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.callderivs** Definition at line 312 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.elements** Definition at line 310 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.factor** Definition at line 313 of file psi4io.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.psi4io.RDVR3\_Psi4.gradd** Definition at line 462 of file psi4io.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.psi4io.RDVR3\_Psi4.hdiagd** Definition at line 463 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.molecules** Definition at line 311 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.objd** Definition at line 461 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.objective** Definition at line 538 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.objfiles** Which parameters are differentiated?

Definition at line 308 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.objvals** Definition at line 309 of file psi4io.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate** [**inherited**] Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [**inherited**] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.target.Target.rundir** [**inherited**] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number. The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.psi4io.RDVR3.Psi4.tdir** Definition at line 355 of file psi4io.py.

**forcebalance.target.Target.tempbase** [**inherited**] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [**inherited**] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate** [**inherited**] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

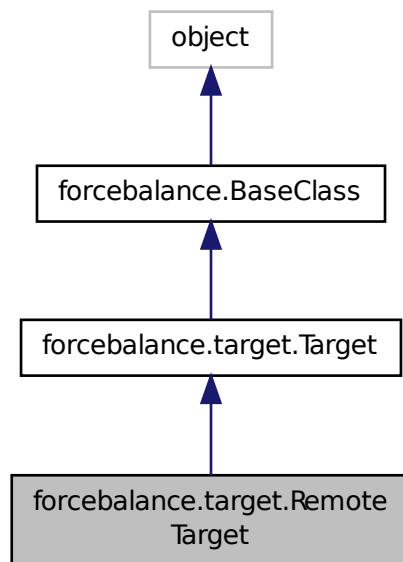
Definition at line 162 of file target.py.

The documentation for this class was generated from the following file:

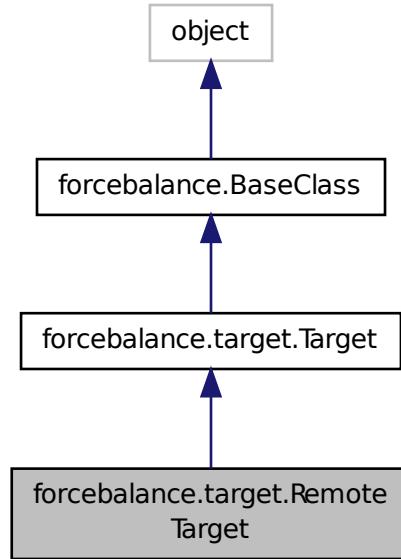
- [psi4io.py](#)

## 8.66 forcebalance.target.RemoteTarget Class Reference

Inheritance diagram for forcebalance.target.RemoteTarget:



Collaboration diagram for forcebalance.target.RemoteTarget:



## Public Member Functions

- def `__init__`
- def `submit_jobs`
- def `read`
- def `get`
- def `indicate`
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_0grads`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_0grads`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*
- def `get_H`

*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`

*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`

*Check this directory for the presence of readable files when the 'read' option is set.*
- def `absrd`

- def `maxrd`  
*Supply the correct directory specified by user's "read" option.*
- def `meta_indicate`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_get`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `stage`  
*Wrapper around the get function.*
- def `wq_complete`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `printcool_table`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `__setattr__`  
*Print target information in an organized table format.*
- def `set_option`

## Public Attributes

- `r_options`
- `r_tgt_opts`
- `remote_indicate`
- `read_indicate`
- `write_indicate`
- `write_objective`
- `rd`  
*Root directory of the whole project.*
- `pgrad`  
*Iteration where we turn on zero-gradient skipping.*
- `tempbase`  
*Relative directory of target.*
- `tempdir`
- `rundir`  
`self.tempdir = os.path.join('temp',self.name)` *The directory in which the simulation is running - this can be updated.*
- `FF`  
*Need the forcefield (here for now)*
- `xct`  
*Counts how often the objective function was computed.*
- `gct`  
*Counts how often the gradient was computed.*
- `hct`  
*Counts how often the Hessian was computed.*
- `read_objective`  
*Whether to read objective.p from file when restarting an aborted run.*
- `verbose_options`
- `PrintOptionDict`

### 8.66.1 Detailed Description

Definition at line 677 of file target.py.

## 8.66.2 Constructor & Destructor Documentation

```
def forcebalance.target.RemoteTarget.__init__ ( self, options, tgt_opts, forcefield ) Definition at line 678 of file target.py.
```

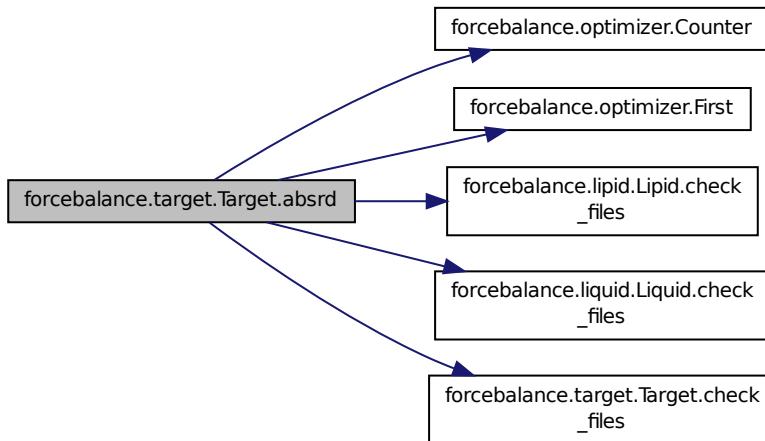
## 8.66.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:

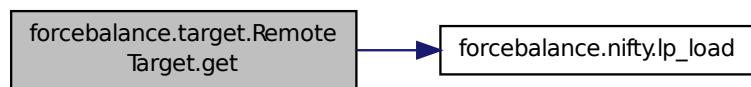


```
def forcebalance.target.Target.check_files ( self, there ) [inherited] Check this directory for the presence of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.target.RemoteTarget.get ( self, mvals, AGrad = False, AHess = False ) Definition at line 729 of file target.py.
```

Here is the call graph for this function:



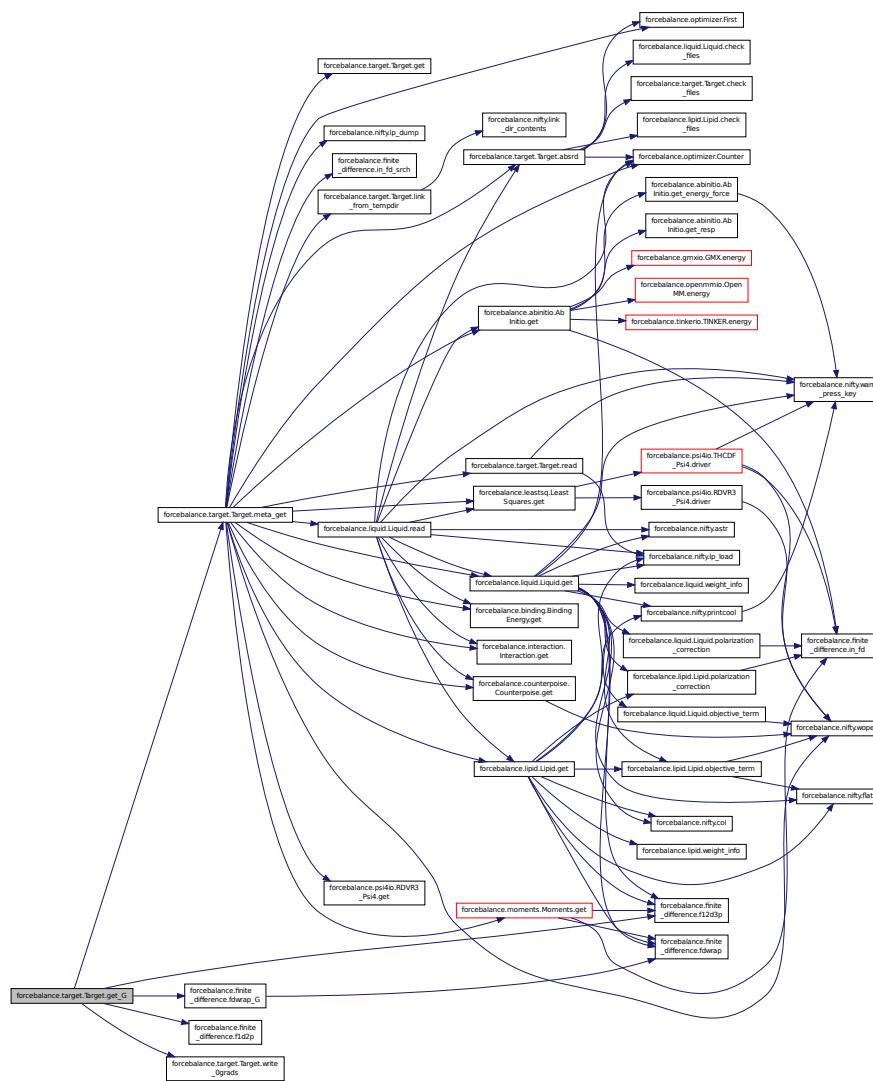
```
def forcebalance.target.Target.get_G ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.get_H ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient / Hessian.
```

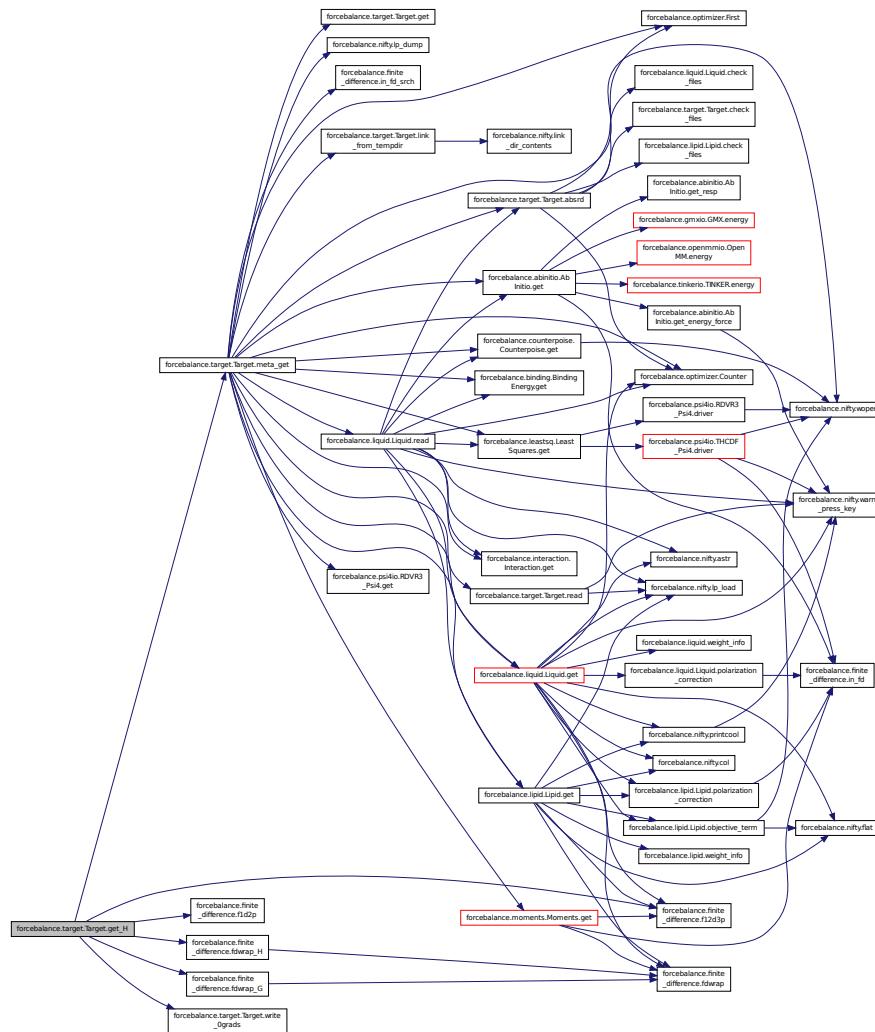
First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through

the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

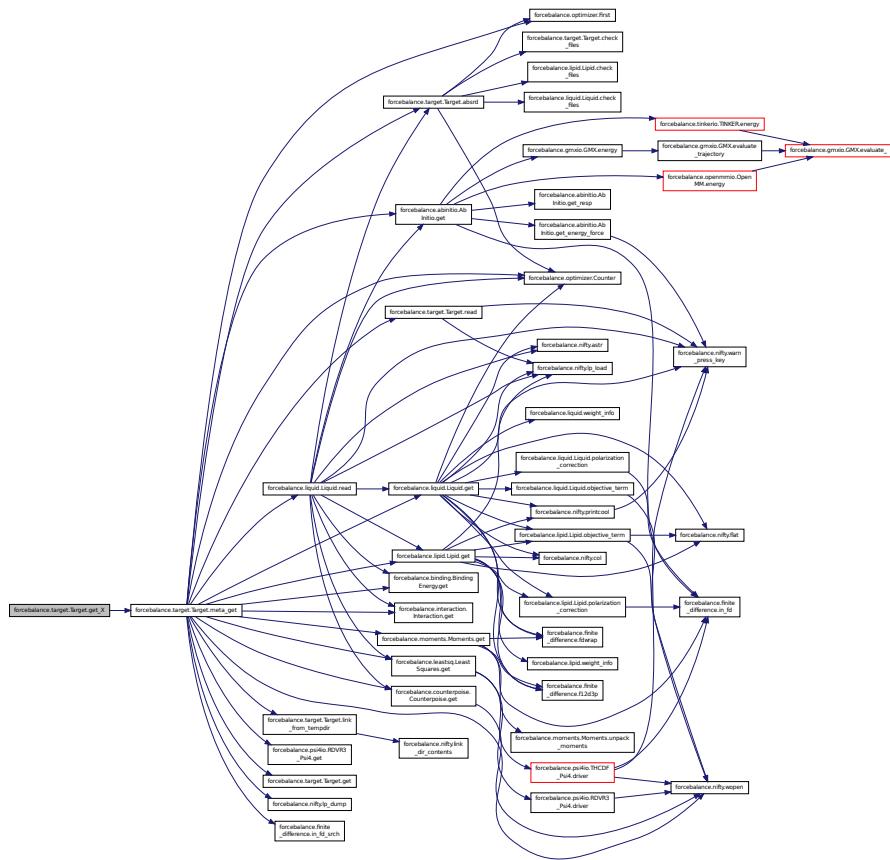
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

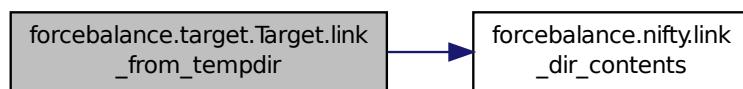
Here is the call graph for this function:



```
def forcebalance.target.RemoteTarget.indicate ( self ) Definition at line 735 of file target.py.
```

```
def forcebalance.target.Target.link_from_tempdir ( self, absdestdir ) [inherited] Definition at line 315 of file target.py.
```

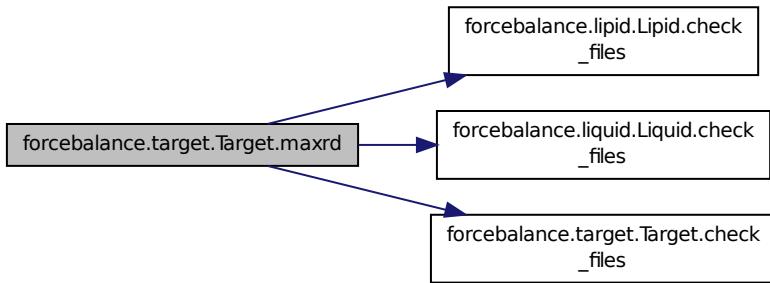
Here is the call graph for this function:



```
def forcebalance.target.Target.maxrd ( self ) [inherited] Supply the latest existing temp-directory containing valid data.
```

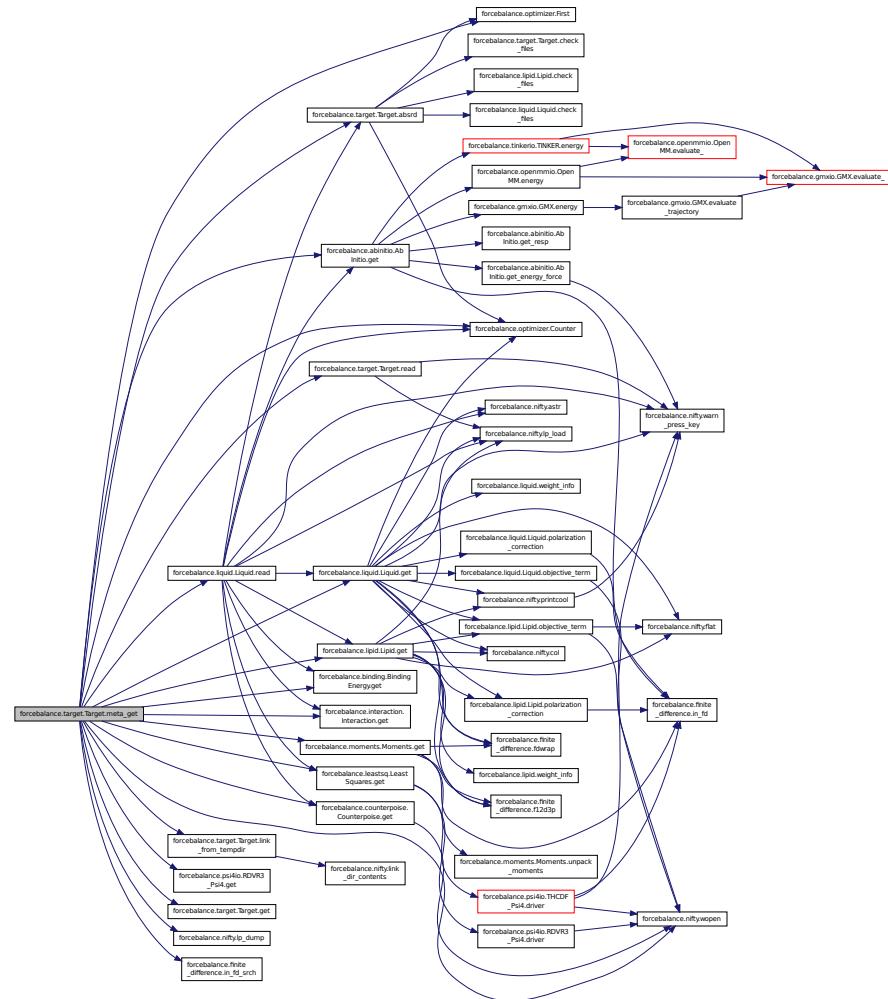
Definition at line 447 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.meta_get( self, mvals, AGrad=False, AHess=False, customdir=None ) [inherited] Wrapper around the get function.  
Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call read\(\) instead. The 'get' method should not worry about the directory that it's running in.  
Definition at line 511 of file target.py.
```

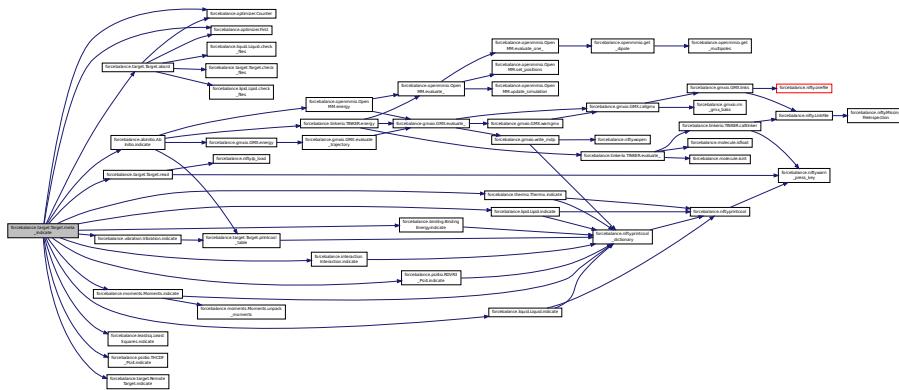
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

The input should be something like:

#### Parameters

<b>data</b>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<b>headings</b>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<b>banner</b>	Optional heading line, which will be printed at the top in the title.
<b>footnote</b>	Optional footnote line, which will be printed at the bottom.

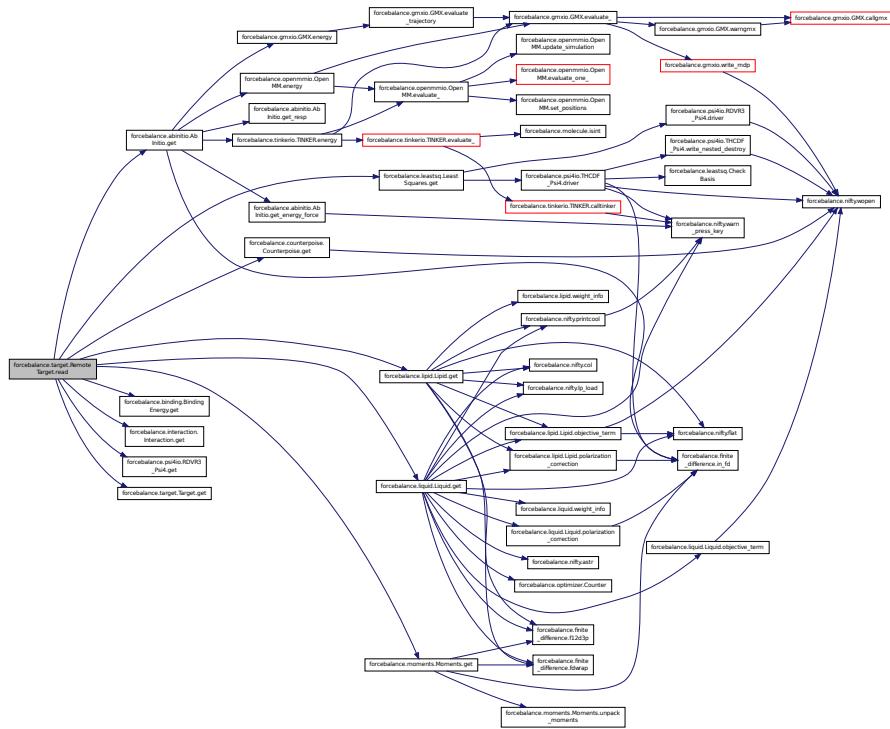
Definition at line 638 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.RemoteTarget.read( self, mvals, AGrad = False, AHess = False ) Definition at line 726 of file target.py.
```

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.target.Target.refresh\_temp\_directory ( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

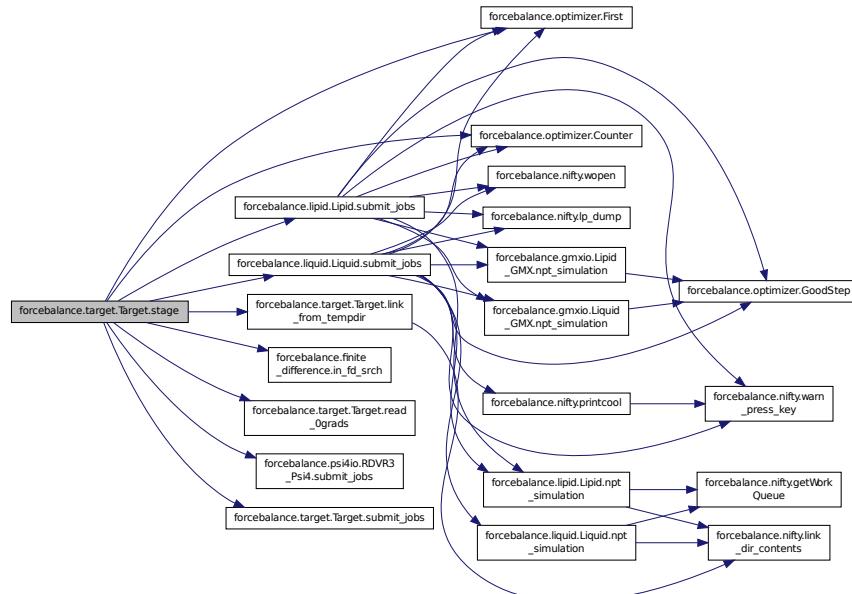
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

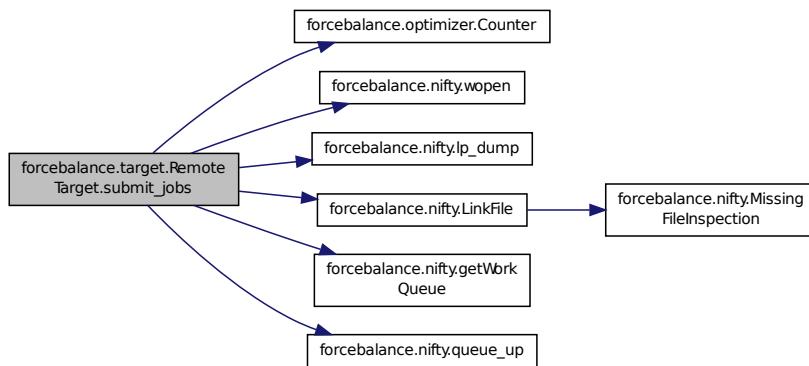
Definition at line 565 of file target.py.

Here is the call graph for this function:



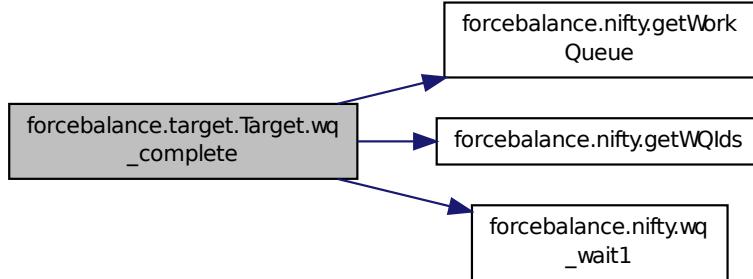
```
def forcebalance.target.RemoteTarget.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
Definition at line 702 of file target.py.
```

Here is the call graph for this function:



```
def forcebalance.target.Target.wq_complete ( self ) [inherited] This method determines whether the Work  
Queue tasks for the current target have completed.  
Definition at line 602 of file target.py.
```

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.66.4 Member Data Documentation

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.RemoteTarget.r\_options** Definition at line 681 of file target.py.

**forcebalance.target.RemoteTarget.r\_tgt\_opts** Definition at line 684 of file target.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.RemoteTarget.read\_indicate** Definition at line 698 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.target.RemoteTarget.remote\_indicate** Definition at line 691 of file target.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number. The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase [inherited]** Relative directory of target.

Temporary (working) directory; it is temp/(target\_name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir [inherited]** Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.RemoteTarget.write\_indicate** Definition at line 699 of file target.py.

**forcebalance.target.RemoteTarget.write\_objective** Definition at line 700 of file target.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 162 of file target.py.

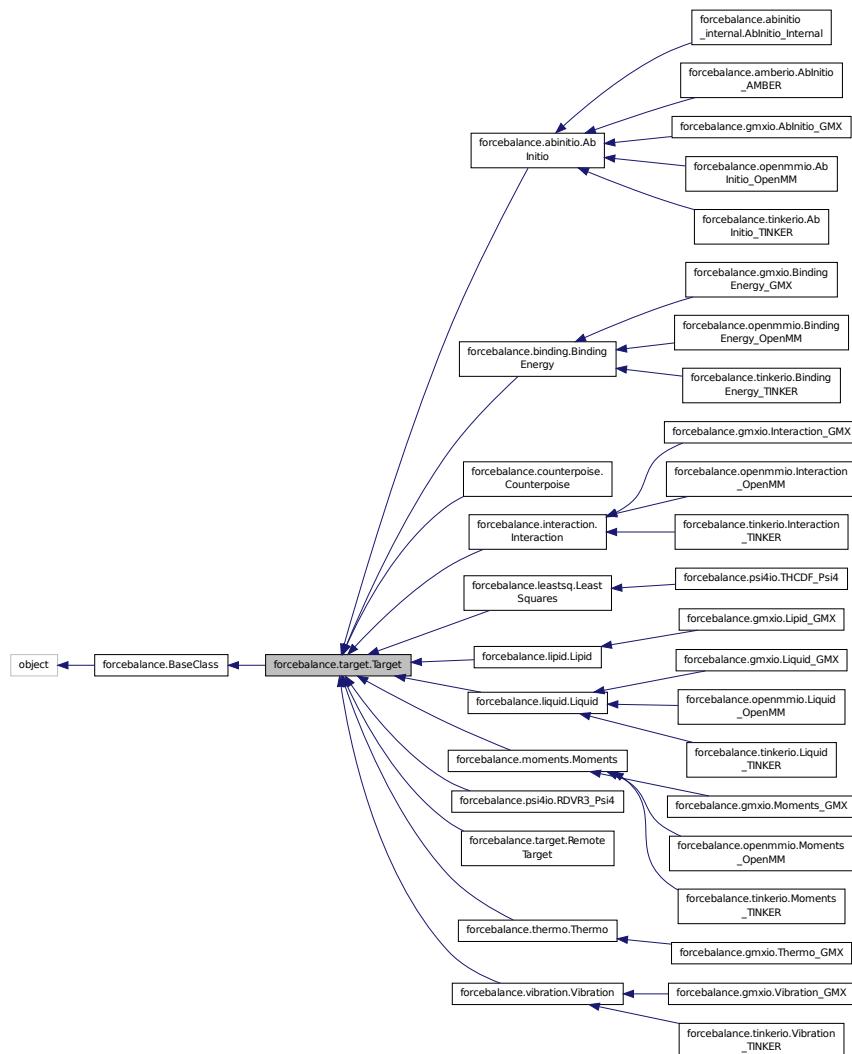
The documentation for this class was generated from the following file:

- [target.py](#)

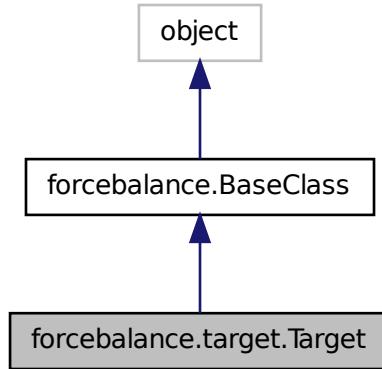
## 8.67 forcebalance.target.Target Class Reference

Base class for all fitting targets.

Inheritance diagram for forcebalance.target.Target:



Collaboration diagram for forcebalance.target.Target:



## Public Member Functions

- def `_init_`

All options here are intended to be usable by every conceivable type of target (in other words, only add content here if it's widely applicable.)
- def `get_X`

Computes the objective function contribution without any parametric derivatives.
- def `read_Ograds`

Read a file from the target directory containing names of parameters that don't contribute to the gradient.
- def `write_Ograds`

Write a file to the target directory containing names of parameters that don't contribute to the gradient.
- def `get_G`

Computes the objective function contribution and its gradient.
- def `get_H`

Computes the objective function contribution and its gradient / Hessian.
- def `link_from_tempdir`
- def `refresh_temp_directory`

Back up the temporary directory if desired, delete it and then create a new one.
- def `get`

Every target must be able to return a contribution to the objective function - however, this must be implemented in the specific subclass.
- def `check_files`

Check this directory for the presence of readable files when the 'read' option is set.
- def `read`

Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.
- def `absrd`

Supply the correct directory specified by user's "read" option.
- def `maxrd`

*Supply the latest existing temp-directory containing valid data.*

- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`

*Wrapper around the get function.*

- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*

- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*

- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `rd`  
*Root directory of the whole project.*
- `pgrad`  
*Iteration where we turn on zero-gradient skipping.*
- `tempbase`  
*Relative directory of target.*
- `tempdir`  
*`self.tempdir = os.path.join('temp', self.name)` The directory in which the simulation is running - this can be updated.*
- `FF`  
*Need the forcefield (here for now)*
- `xct`  
*Counts how often the objective function was computed.*
- `gct`  
*Counts how often the gradient was computed.*
- `hct`  
*Counts how often the Hessian was computed.*
- `read_indicate`  
*Whether to read indicate.log from file when restarting an aborted run.*
- `write_indicate`  
*Whether to write indicate.log at every iteration (true for all but remote.)*
- `read_objective`  
*Whether to read objective.p from file when restarting an aborted run.*
- `write_objective`  
*Whether to write objective.p at every iteration (true for all but remote.)*
- `verbose_options`
- `PrintOptionDict`

### 8.67.1 Detailed Description

Base class for all fitting targets.

In ForceBalance a [Target](#) is defined as a set of reference data plus a corresponding method to simulate that data using the force field.

The 'computable quantities' may include energies and forces where the reference values come from QM calculations (energy and force matching), energies from an EDA analysis (Maybe in the future, FDA?), molecular properties (like polarizability, refractive indices, multipole moments or vibrational frequencies), relative entropies, and bulk properties. Single-molecule or bulk properties can even come from the experiment!

The central idea in ForceBalance is that each quantity makes a contribution to the overall objective function. So we can build force fields that fit several quantities at once, rather than putting all of our chips behind energy and force matching. In the future ForceBalance may even include multiobjective optimization into the optimizer.

The optimization is done by way of minimizing an 'objective function', which is comprised of squared differences between the computed and reference values. These differences are not computed in this file, but rather in subclasses that use [Target](#) as a base class. Thus, the contents of [Target](#) itself are meant to be as general as possible, because the pertinent variables apply to all types of fitting targets.

An important note: [Target](#) requires that all subclasses have a method `get(self,mvals,AGrad=False,AHess=False)` that does the following:

Inputs: `mvals` = The parameter vector, which modifies the force field (Note to self: We include `mvals` with each [Target](#) because we can create copies of the force field and do finite difference derivatives) `AGrad`, `AHess` = Boolean switches for computing analytic gradients and Hessians

Outputs: `Answer = { 'X': Number, 'G': array(NP), 'H': array((NP,NP)) }` 'X' = The objective function itself 'G' = The gradient, elements not computed analytically are zero 'H' = The Hessian, elements not computed analytically are zero

This is the only global requirement of a [Target](#). Obviously 'get' itself is not defined here, because its calculation will depend entirely on specifically which target we wish to use. However, this should give us a unified framework which will facilitate rapid implementation of Targets.

Future work: Robert suggested that I could enable automatic detection of which parameters need to be computed by finite difference. Not a bad idea. :)

Definition at line 75 of file target.py.

### 8.67.2 Constructor & Destructor Documentation

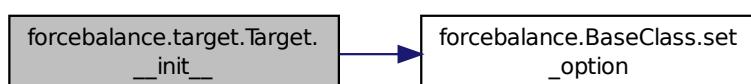
```
def forcebalance.target.Target.__init__ ( self, options, tgt_opts, forcefield )
```

All options here are intended to be usable by every conceivable type of target (in other words, only add content here if it's widely applicable.)

If we want to add attributes that are more specific (i.e. a set of reference forces for force matching), they are added in the subclass AblInitio that inherits from [Target](#).

Definition at line 92 of file target.py.

Here is the call graph for this function:



### 8.67.3 Member Function Documentation

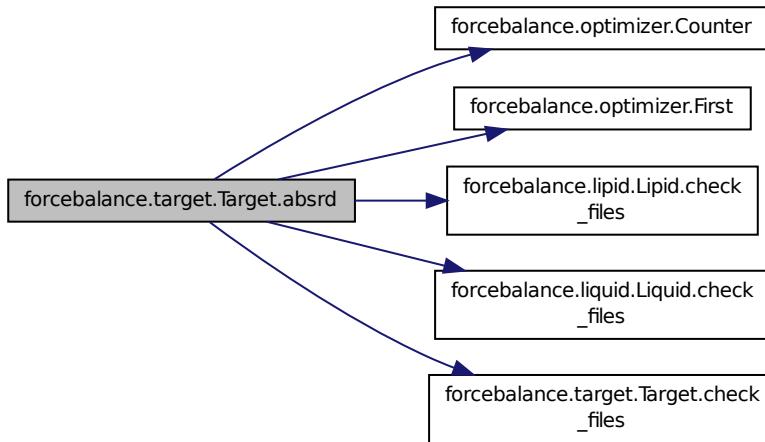
```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited]
```

Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.absrd ( self, inum = None ) Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.check_files ( self, there ) Check this directory for the presence of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.target.Target.get ( self, mvals, AGrad = False, AHess = False ) Every target must be able to return a contribution to the objective function - however, this must be implemented in the specific subclass.
```

See abinitio for an example.

Definition at line 357 of file target.py.

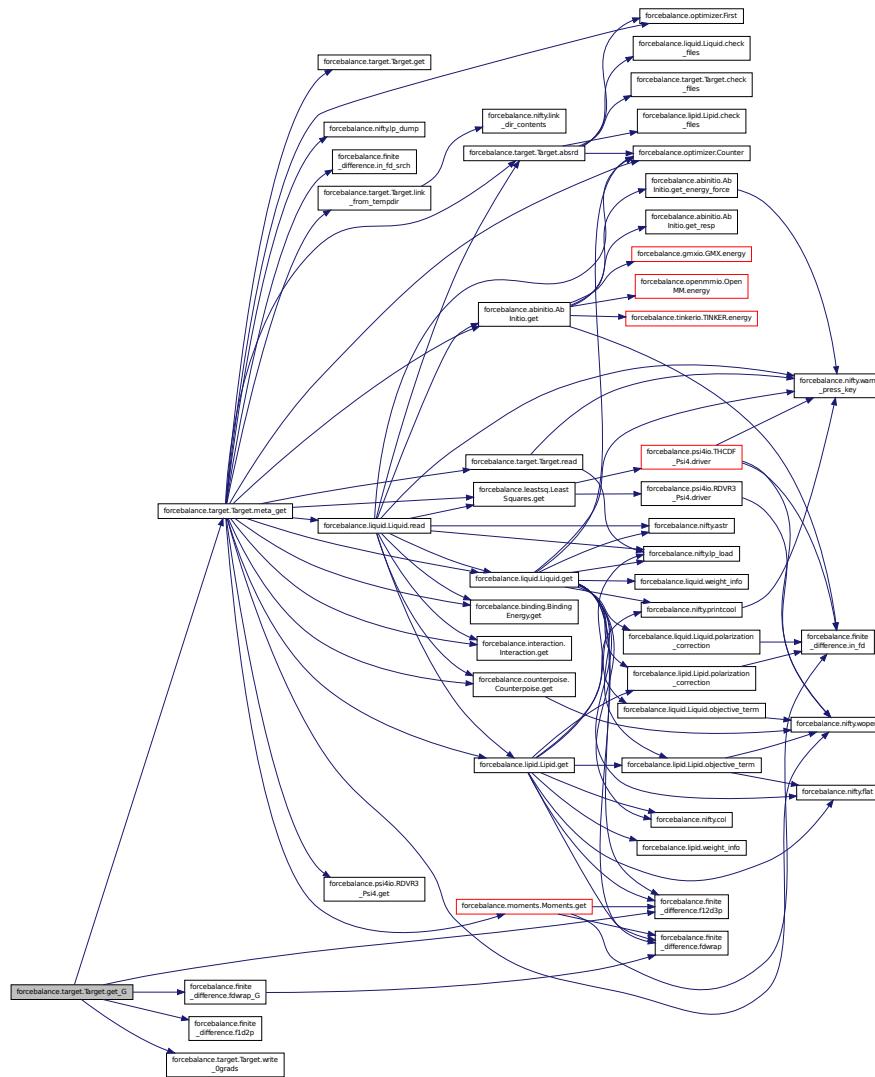
```
def forcebalance.target.Target.get_G ( self, mvals = None ) Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1.pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



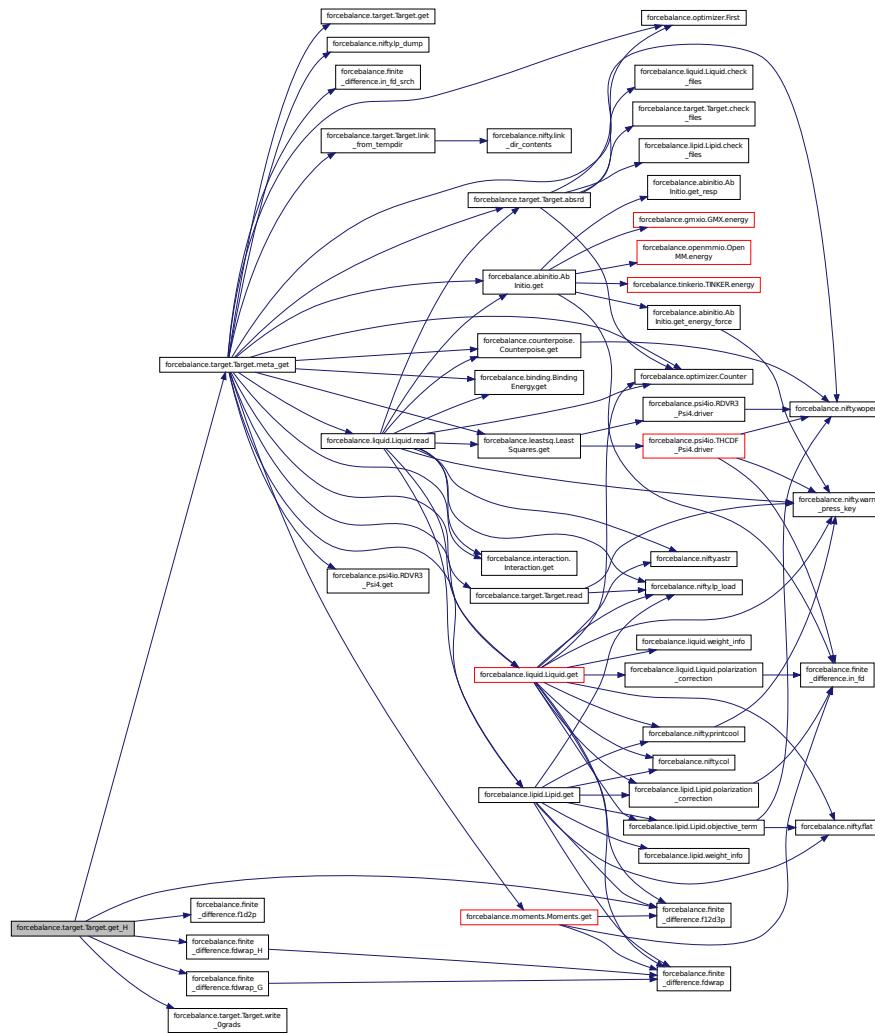
**def forcebalance.target.Target.get\_H ( self, mvals = None )** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

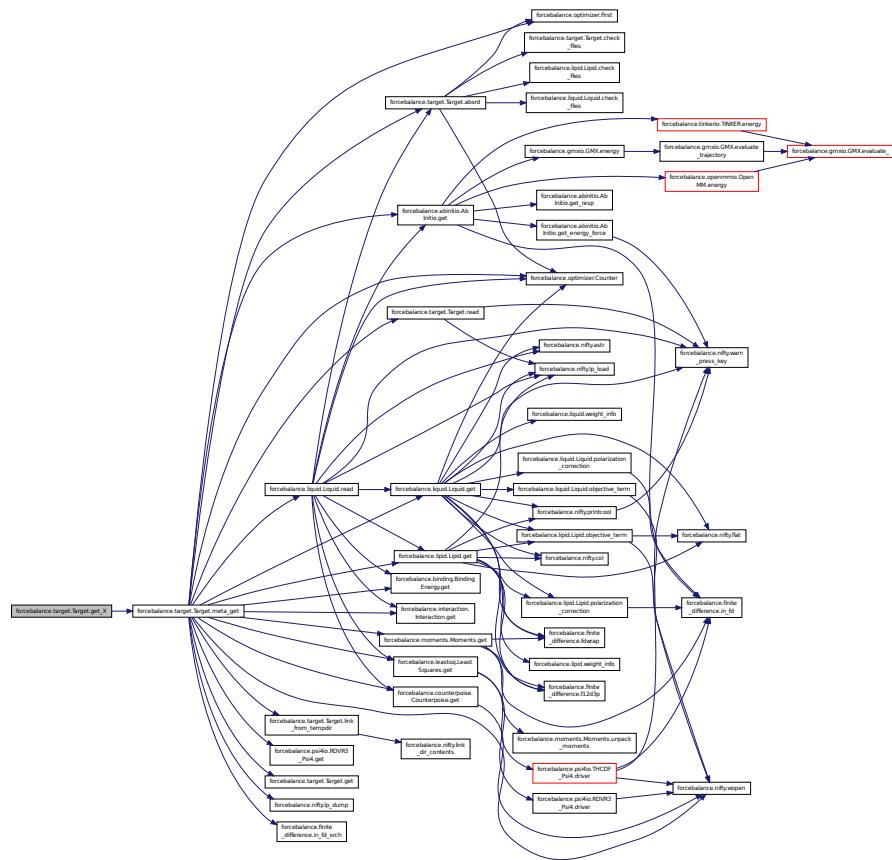
Here is the call graph for this function:



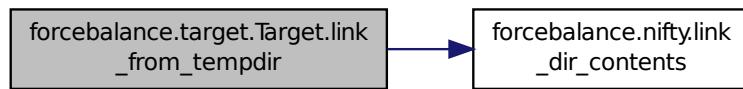
**def forcebalance.target.Target.get\_X ( self, mvals = None )** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

Here is the call graph for this function:

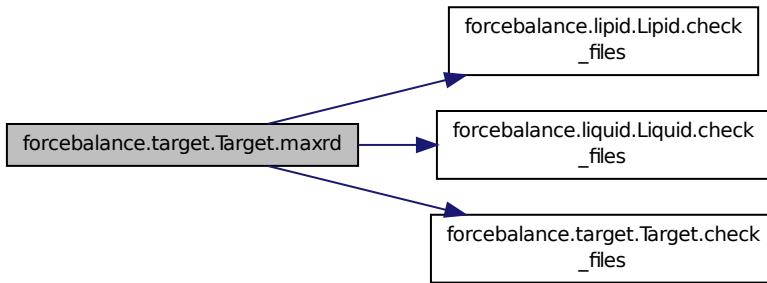


**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir )** Definition at line 315 of file target.py.  
Here is the call graph for this function:



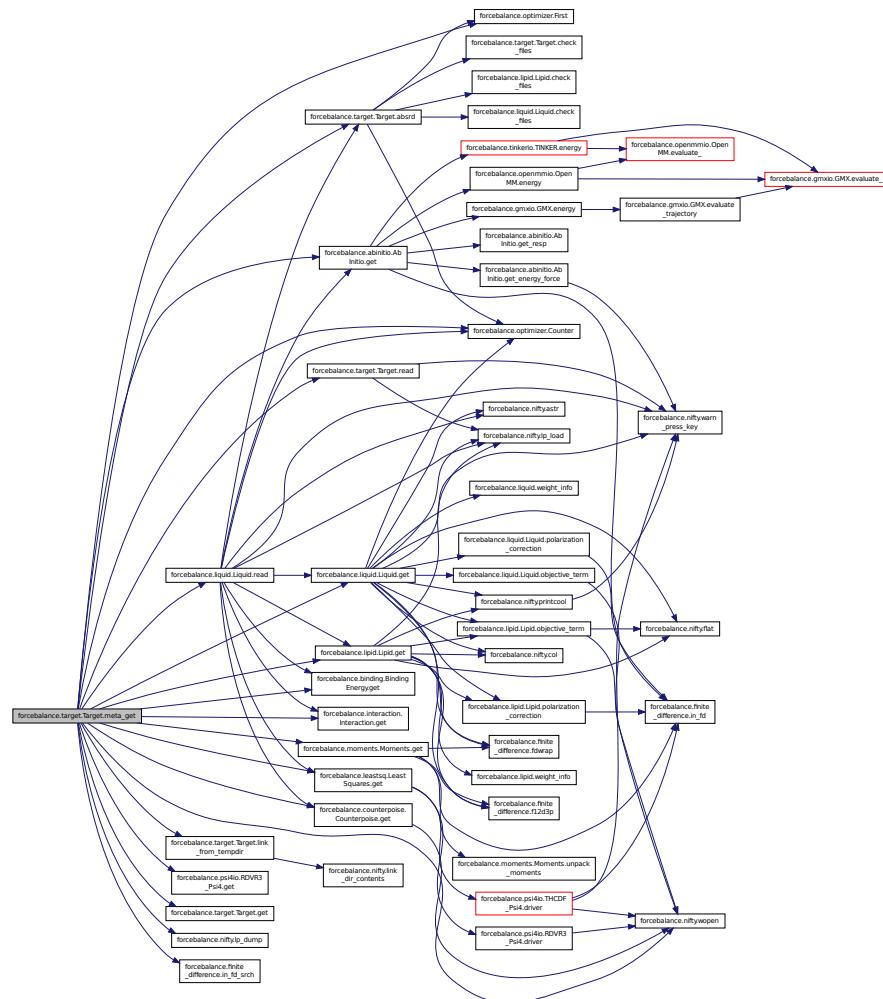
**def forcebalance.target.Target.maxrd ( self )** Supply the latest existing temp-directory containing valid data.  
Definition at line 447 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.meta_get( self, mvals, AGrad=False, AHess=False, customdir=None )
) Wrapper around the get function.
Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call read\(\) instead. The 'get' method should not worry about the directory that it's running in.
Definition at line 511 of file target.py.
```

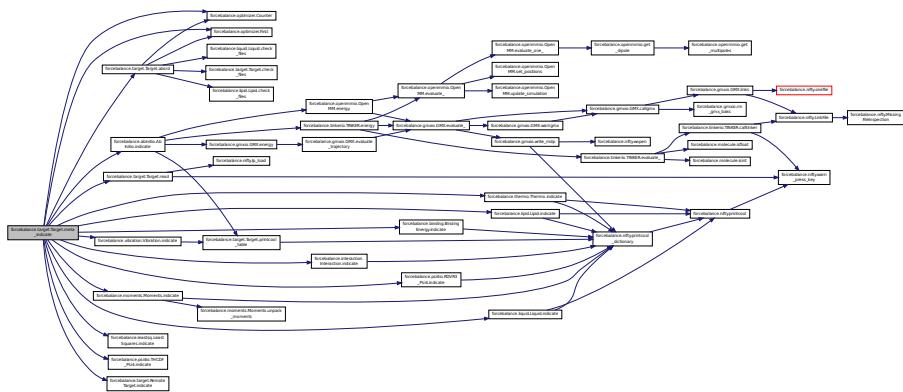
Here is the call graph for this function:



```
def forcebalance.target.Target.meta_indicate( self ) Wrap around the indicate function, so it can print to screen  
and also to a file.
```

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen. Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

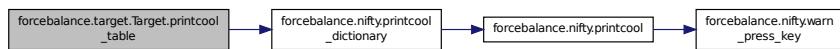
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

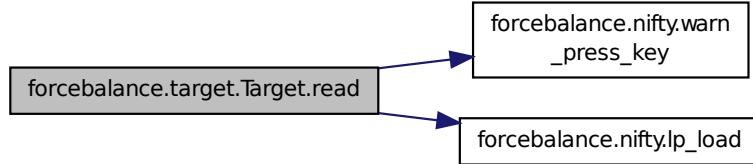
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False )** Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self )** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.target.Target.refresh\_temp\_directory ( self )** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

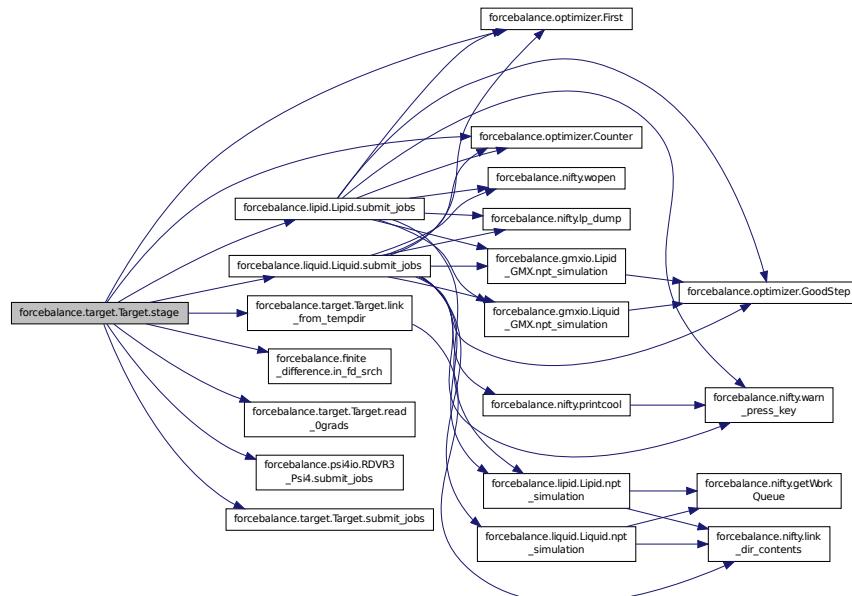
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )** Stages the directory for the target, and then launches Work Queue processes if any.

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

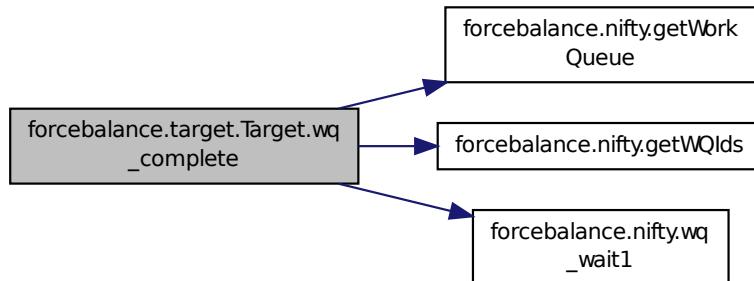


**def forcebalance.target.Target.submit\_jobs ( self, mvals, AGrad = False, AHess = False )** Definition at line 555 of file target.py.

**def forcebalance.target.Target.wq\_complete ( self )** This method determines whether the Work Queue tasks for the current target have completed.

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans )** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.67.4 Member Data Documentation

**forcebalance.target.Target.FF** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.target.Target.pgrad** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict** [**inherited**] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd** Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate** Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.target.Target.rundir** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate** Whether to write indicate.log at every iteration (true for all but remote.)  
Definition at line 170 of file target.py.

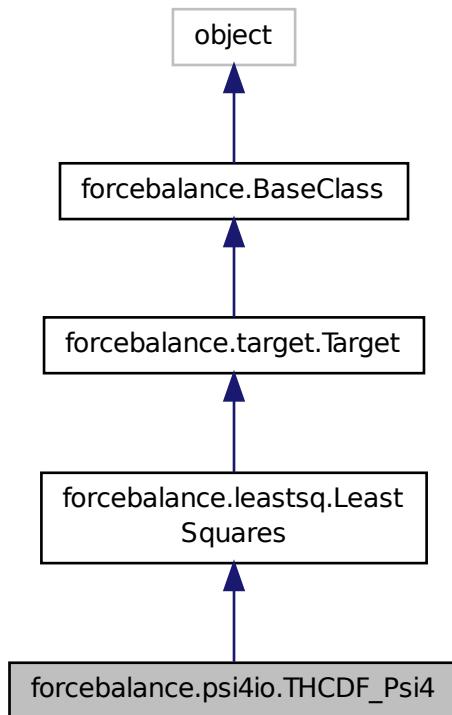
**forcebalance.target.Target.write\_objective** Whether to write objective.p at every iteration (true for all but remote.)  
Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** Counts how often the objective function was computed.  
Definition at line 162 of file target.py.  
The documentation for this class was generated from the following file:

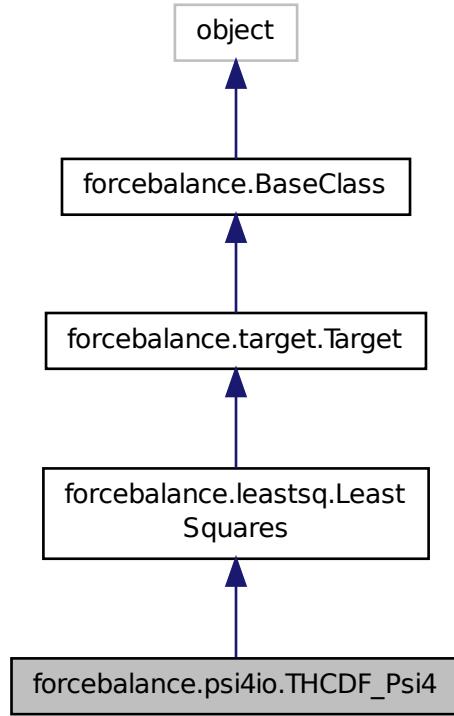
- [target.py](#)

## 8.68 forcebalance.psi4io.THCDF\_Psi4 Class Reference

Inheritance diagram for forcebalance.psi4io.THCDF\_Psi4:



Collaboration diagram for forcebalance.psi4io.THCDF\_Psi4:



### Public Member Functions

- def `_init_`
- def `prepare_temp_directory`
- def `indicate`
- def `write_nested_destroy`
- def `driver`
- def `get`  
    *LPW 05-30-2012.*
- def `get_X`  
    *Computes the objective function contribution without any parametric derivatives.*
- def `read_0grads`  
    *Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_0grads`  
    *Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`  
    *Computes the objective function contribution and its gradient.*
- def `get_H`  
    *Computes the objective function contribution and its gradient / Hessian.*

- def [link\\_from\\_tempdir](#)
- def [refresh\\_temp\\_directory](#)  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def [check\\_files](#)  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def [read](#)  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def [absrd](#)  
*Supply the correct directory specified by user's "read" option.*
- def [maxrd](#)  
*Supply the latest existing temp-directory containing valid data.*
- def [meta\\_indicate](#)  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def [meta\\_get](#)  
*Wrapper around the get function.*
- def [submit\\_jobs](#)
- def [stage](#)  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def [wq\\_complete](#)  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def [printcool\\_table](#)  
*Print target information in an organized table format.*
- def [\\_\\_setattr\\_\\_](#)
- def [set\\_option](#)

## Public Attributes

- [Molecules](#)
- [throw\\_outs](#)
- [Elements](#)
- [GBSfnm](#)  
*Psi4 basis set file.*
- [DATfnm](#)  
*Psi4 input file for calculation of linear dependencies This is actually a file in 'forcefield' until we can figure out a better system.*
- [MP2\\_Energy](#)  
*Actually run PSI4.*
- [DF\\_Energy](#)
- [MAQ](#)  
*Dictionary for derivative terms.*
- [D](#)
- [objective](#)
- [rd](#)  
*Root directory of the whole project.*
- [pgrad](#)  
*Iteration where we turn on zero-gradient skipping.*
- [tempbase](#)  
*Relative directory of target.*

- `tempdir`  
`self.tempdir = os.path.join('temp',self.name)` *The directory in which the simulation is running - this can be updated.*
- `FF`  
`Need the forcefield (here for now)`
- `xct`  
`Counts how often the objective function was computed.`
- `gct`  
`Counts how often the gradient was computed.`
- `hct`  
`Counts how often the Hessian was computed.`
- `read_indicate`  
`Whether to read indicate.log from file when restarting an aborted run.`
- `write_indicate`  
`Whether to write indicate.log at every iteration (true for all but remote.)`
- `read_objective`  
`Whether to read objective.p from file when restarting an aborted run.`
- `write_objective`  
`Whether to write objective.p at every iteration (true for all but remote.)`
- `verbose_options`
- `PrintOptionDict`

### 8.68.1 Detailed Description

Definition at line 97 of file psi4io.py.

### 8.68.2 Constructor & Destructor Documentation

`def forcebalance.psi4io.THCDF_Psi4.__init__ ( self, options, tgt_opts, forcefield )` Definition at line 99 of file psi4io.py.

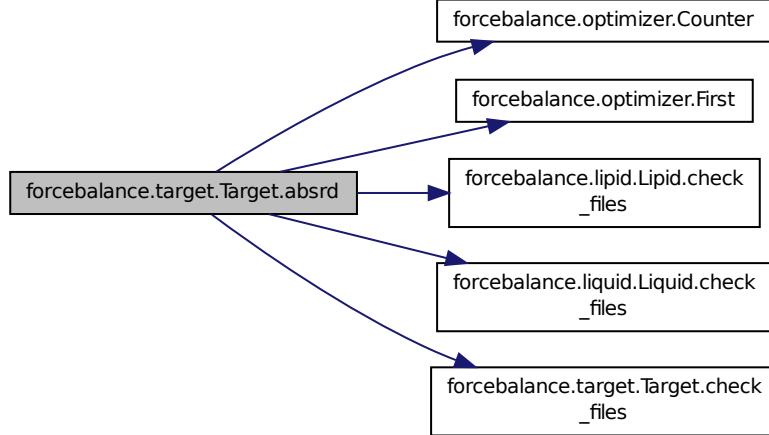
### 8.68.3 Member Function Documentation

`def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited]` Definition at line 28 of file \_\_init\_\_.py.

`def forcebalance.target.Target.absrd ( self, inum = None ) [inherited]` Supply the correct directory specified by user's "read" option.

Definition at line 393 of file target.py.

Here is the call graph for this function:

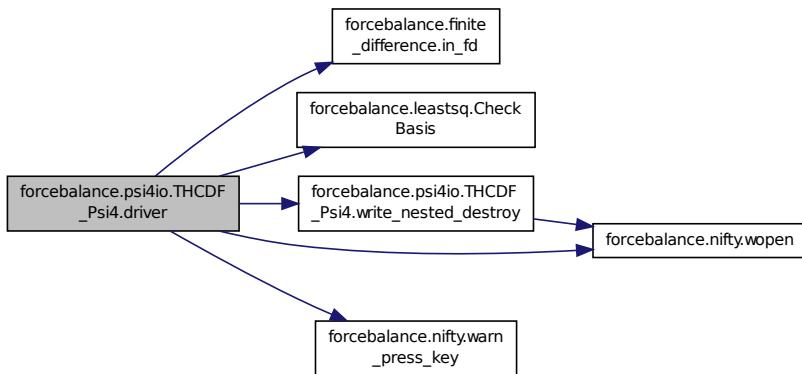


**def forcebalance.target.Target.check\_files( self, there ) [inherited]** Check this directory for the presence of readable files when the 'read' option is set.

Definition at line 364 of file target.py.

**def forcebalance.psi4io.THCDF\_Psi4.driver( self )** Definition at line 175 of file psi4io.py.

Here is the call graph for this function:



**def forcebalance.leastsq.LeastSquares.get( self, mvals, AGrad = False, AHess = False ) [inherited]** LPW 05-30-2012.

This subroutine builds the objective function (and optionally its derivatives) from a general software.

This subroutine interfaces with simulation software 'drivers'. The driver is expected to give exact values, fitting values, and weights.

## Parameters

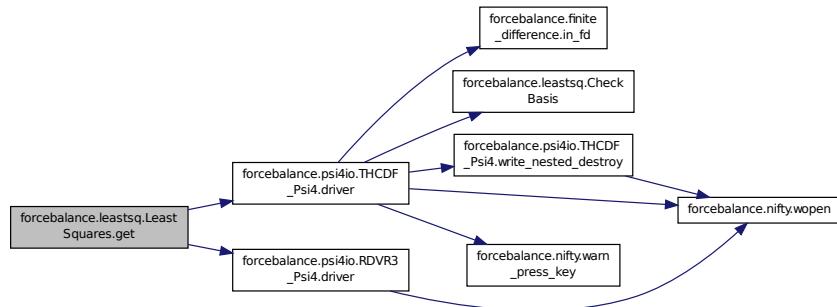
in	<i>mvals</i>	Mathematical parameter values
in	<i>AGrad</i>	Switch to turn on analytic gradient
in	<i>AHess</i>	Switch to turn on analytic Hessian

## Returns

Answer Contribution to the objective function

Definition at line 62 of file leastsq.py.

Here is the call graph for this function:



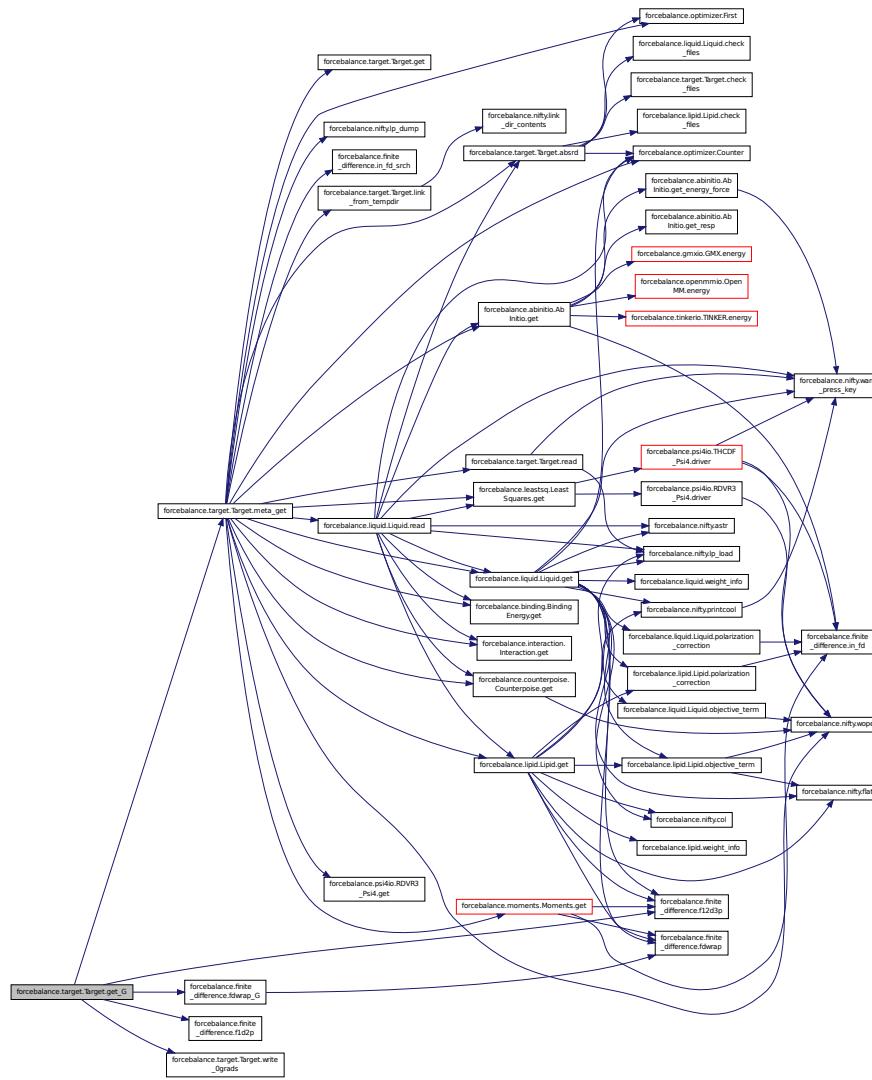
**def forcebalance.target.Target.get\_G ( *self*, *mvals = None* ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1.pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



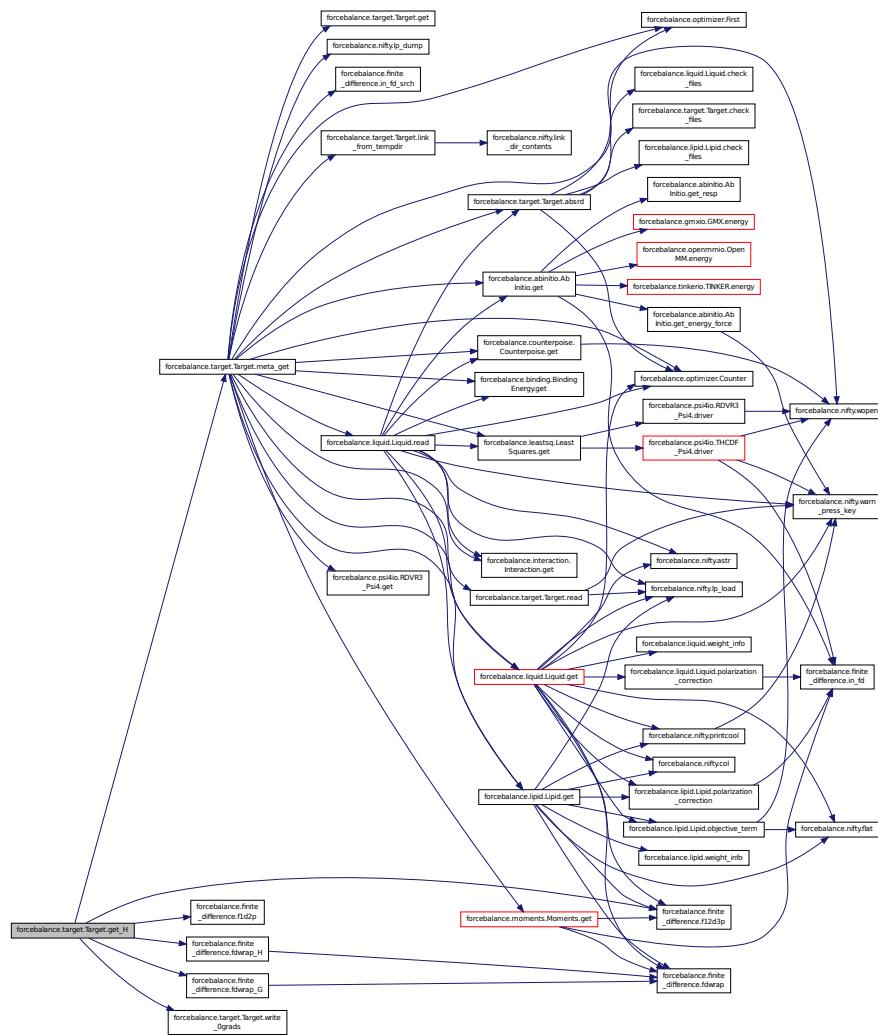
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

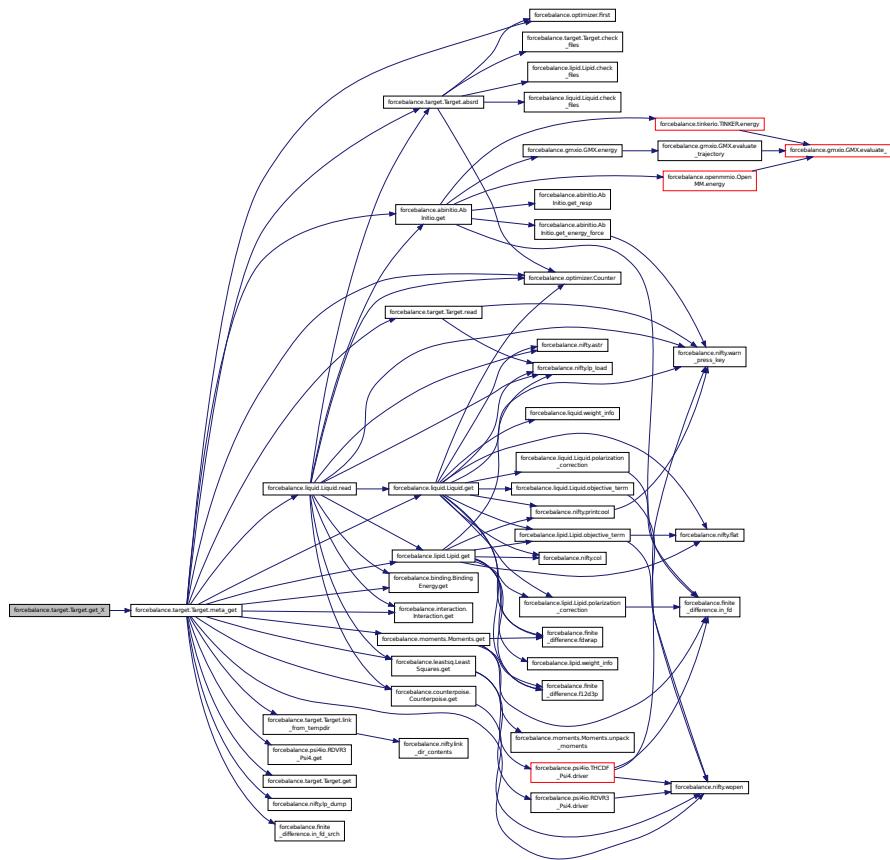
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

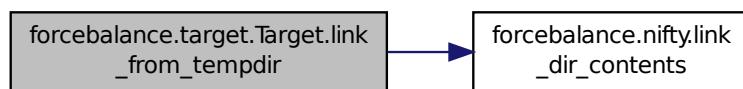
Here is the call graph for this function:



```
def forcebalance.psi4io.THCD4_Psi4.indicate ( self ) Definition at line 155 of file psi4io.py.
```

```
def forcebalance.target.Target.link_from_tempdir ( self, absdestdir ) [inherited] Definition at line 315 of file target.py.
```

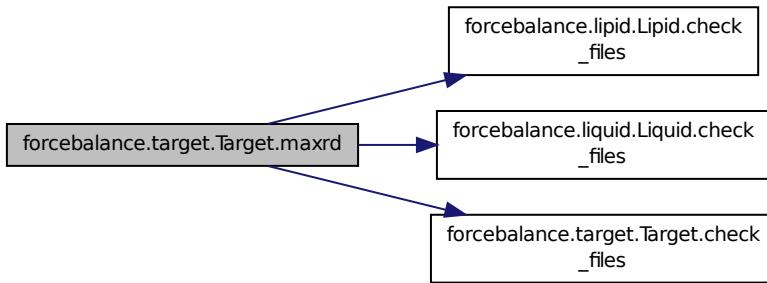
Here is the call graph for this function:



```
def forcebalance.target.Target.maxrd ( self ) [inherited] Supply the latest existing temp-directory containing valid data.
```

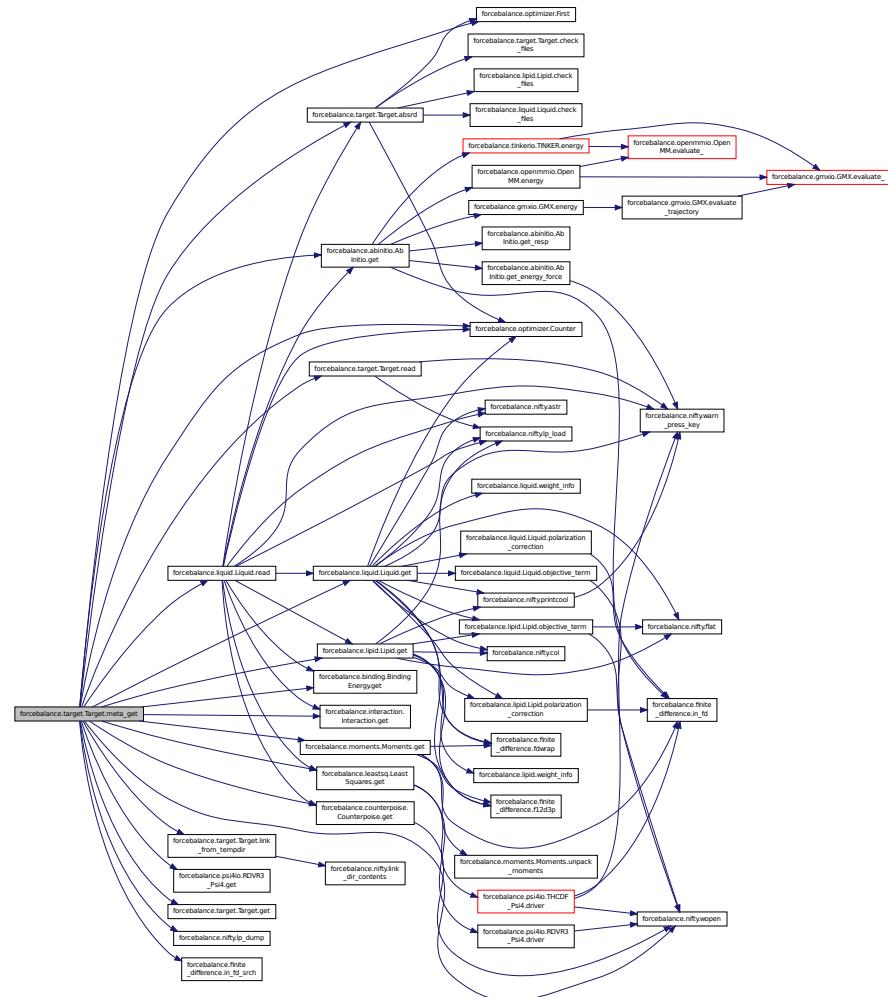
Definition at line 447 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.meta_get( self, mvals, AGrad=False, AHess=False, customdir=None ) [inherited] Wrapper around the get function.  
Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call read\(\) instead. The 'get' method should not worry about the directory that it's running in.  
Definition at line 511 of file target.py.
```

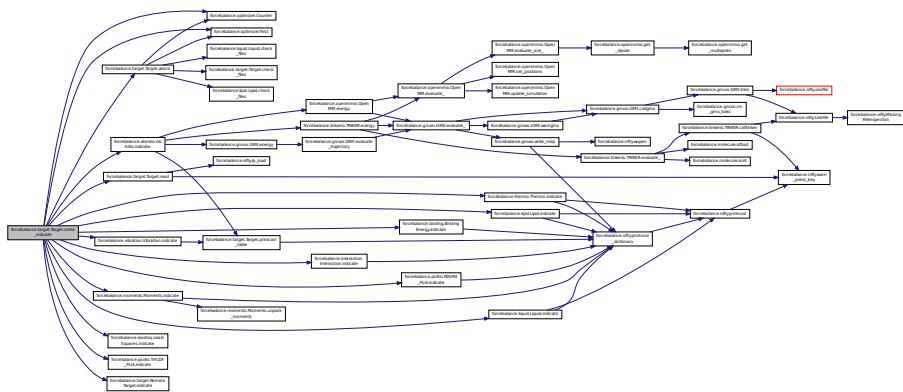
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.psi4io.THCDF_Psi4.prepare_temp_directory ( self, options, tgt_opts ) Definition at line 144  
of file psi4io.py.
```

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

Here is the call graph for this function:

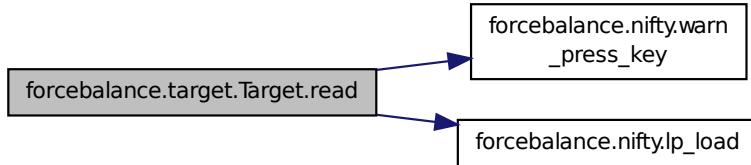


```
def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]
```

Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.read_0grads ( self ) [inherited] Read a file from the target directory containing names of parameters that don't contribute to the gradient.
```

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

```
def forcebalance.target.Target.refresh_temp_directory( self ) [inherited] Back up the temporary directory if desired, delete it and then create a new one.
```

Definition at line 321 of file target.py.

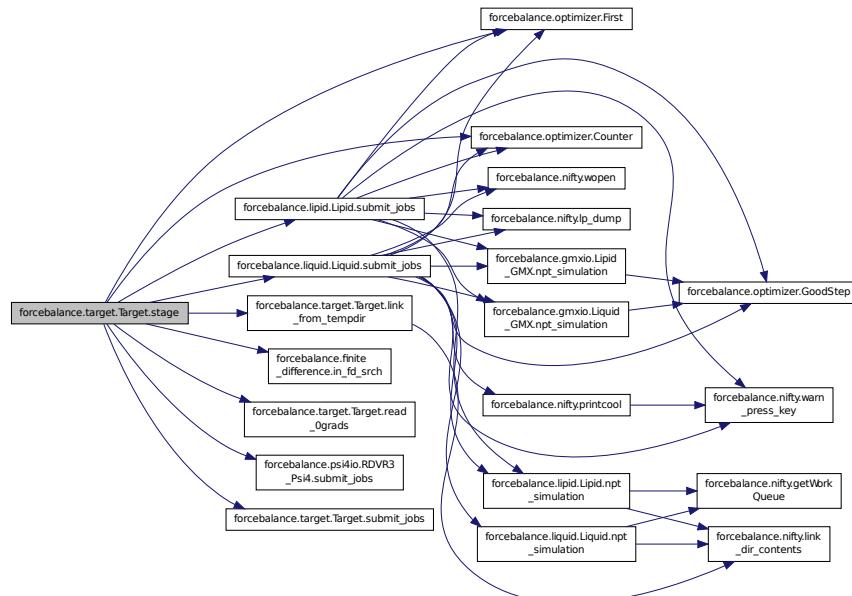
```
def forcebalance.BaseClass.set_option ( self, in_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:

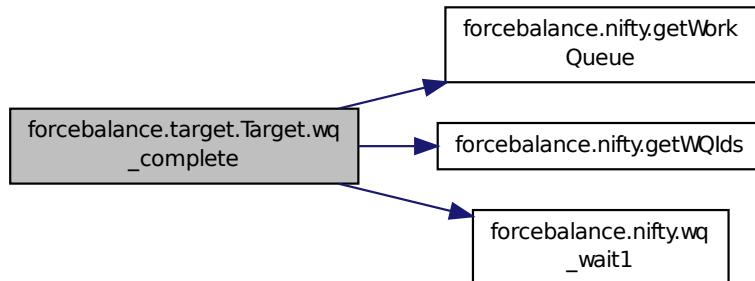


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:

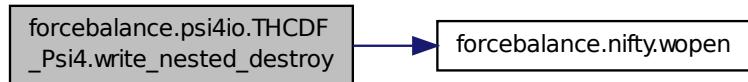


```
def forcebalance.target.Target.write_0grads( self, Ans ) [inherited] Write a file to the target directory containing names of parameters that don't contribute to the gradient.
```

Definition at line 225 of file target.py.

**def forcebalance.psi4io.THCDF\_Psi4.write\_nested\_destroy ( self, fnm, linedestroy )** Definition at line 160 of file psi4io.py.

Here is the call graph for this function:



#### 8.68.4 Member Data Documentation

**forcebalance.leastsq.LeastSquares.D [inherited]** Definition at line 126 of file leastsq.py.

**forcebalance.psi4io.THCDF\_Psi4.DATfnm** Psi4 input file for calculation of linear dependencies This is actually a file in 'forcefield' until we can figure out a better system.

Definition at line 140 of file psi4io.py.

**forcebalance.psi4io.THCDF\_Psi4.DF\_Energy** Definition at line 241 of file psi4io.py.

**forcebalance.psi4io.THCDF\_Psi4.Elements** Definition at line 117 of file psi4io.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.psi4io.THCDF\_Psi4.GBSfnm** Psi4 basis set file.

Definition at line 133 of file psi4io.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.leastsq.LeastSquares.MAQ [inherited]** Dictionary for derivative terms.

Definition at line 88 of file leastsq.py.

**forcebalance.psi4io.THCDF\_Psi4.Molecules** Definition at line 105 of file psi4io.py.

**forcebalance.psi4io.THCDF\_Psi4.MP2\_Energy** Actually run PSI4.

Read in the commented linindep.gbs file and ensure that these same lines are commented in the new .gbs file Now build a "Frankenstein" .gbs file composed of the original .gbs file but with data from the linindep.gbs file!

Definition at line 239 of file psi4io.py.

**forcebalance.leastsq.LeastSquares.objective [inherited]** Definition at line 127 of file leastsq.py.

**forcebalance.target.Target.pgrad** [**inherited**] Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict** [**inherited**] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd** [**inherited**] Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate** [**inherited**] Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [**inherited**] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.target.Target.rundir** [**inherited**] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number. The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [**inherited**] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [**inherited**] Definition at line 155 of file target.py.

**forcebalance.psi4io.THCDF\_Psi4.throw\_outs** Definition at line 106 of file psi4io.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.target.Target.write\_indicate** [**inherited**] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

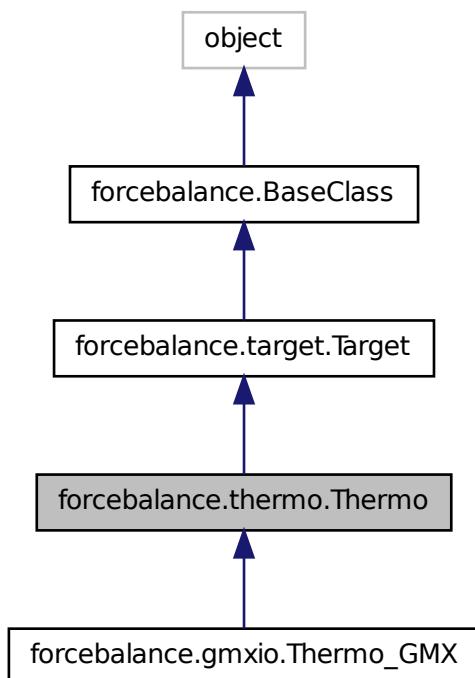
Definition at line 174 of file target.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.  
Definition at line 162 of file target.py.  
The documentation for this class was generated from the following file:  
• [psi4io.py](#)

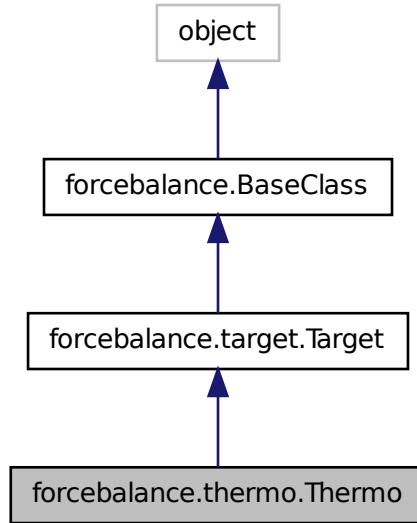
## 8.69 forcebalance.thermo.Thermo Class Reference

A target for fitting general experimental data sets.

Inheritance diagram for forcebalance.thermo.Thermo:



Collaboration diagram for forcebalance.thermo.Thermo:



## Public Member Functions

- def `_init_`
- def `retrieve`

*Retrieve the molecular dynamics (MD) results and store the calculated quantities in the `Point` object dp.*
- def `submit_jobs`

*This routine is called by `Objective.stage()` and will run before "get".*
- def `indicate`

*Shows optimization state.*
- def `objective_term`

*Calculates the contribution to the objective function (the term) for a given quantity.*
- def `get`

*Return the contribution to the total objective function.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_0grads`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_0grads`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*
- def `get_H`

*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`

- def [refresh\\_temp\\_directory](#)  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def [check\\_files](#)  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def [read](#)  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def [absrd](#)  
*Supply the correct directory specified by user's "read" option.*
- def [maxrd](#)  
*Supply the latest existing temp-directory containing valid data.*
- def [meta\\_indicate](#)  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def [meta\\_get](#)  
*Wrapper around the get function.*
- def [stage](#)  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def [wq\\_complete](#)  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def [printcool\\_table](#)  
*Print target information in an organized table format.*
- def [\\_\\_setattr\\_\\_](#)
- def [set\\_option](#)

## Public Attributes

- [simpfx](#)  
*Initialize base class.*
- [points](#)
- [denoms](#)
- [weights](#)
- [Xp](#)
- [Wp](#)
- [Pp](#)
- [Gp](#)
- [Objective](#)
- [rd](#)  
*Root directory of the whole project.*
- [pgrad](#)  
*Iteration where we turn on zero-gradient skipping.*
- [tempbase](#)  
*Relative directory of target.*
- [tempdir](#)
- [rundir](#)  
*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- [FF](#)  
*Need the forcefield (here for now)*
- [xct](#)  
*Counts how often the objective function was computed.*

- **gct**  
*Counts how often the gradient was computed.*
- **hct**  
*Counts how often the Hessian was computed.*
- **read\_indicate**  
*Whether to read indicate.log from file when restarting an aborted run.*
- **write\_indicate**  
*Whether to write indicate.log at every iteration (true for all but remote.)*
- **read\_objective**  
*Whether to read objective.p from file when restarting an aborted run.*
- **write\_objective**  
*Whether to write objective.p at every iteration (true for all but remote.)*
- **verbose\_options**
- **PrintOptionDict**

### 8.69.1 Detailed Description

A target for fitting general experimental data sets.

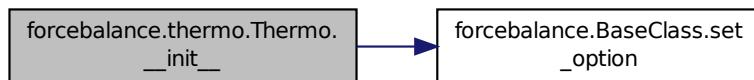
The experimental data is described in a .txt file and is handled with a `Quantity` subclass.

Definition at line 24 of file `thermo.py`.

### 8.69.2 Constructor & Destructor Documentation

**def forcebalance.thermo.Thermo.\_\_init\_\_ ( self, options, tgt\_opts, forcefield )** Definition at line 25 of file `thermo.py`.

Here is the call graph for this function:



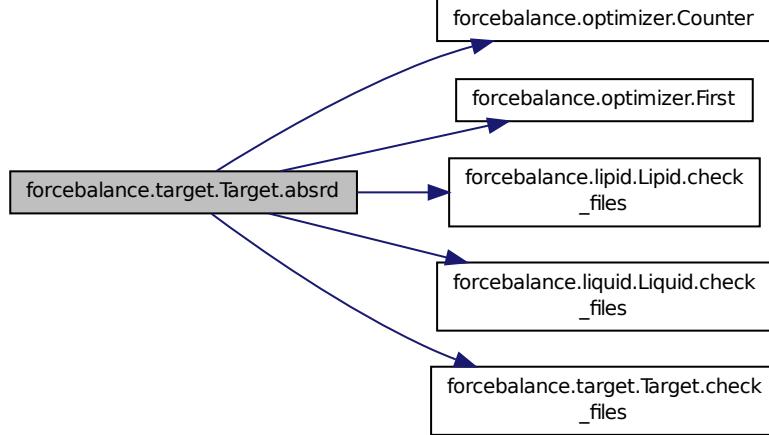
### 8.69.3 Member Function Documentation

**def forcebalance.BaseClass.\_\_setattr\_\_ ( self, key, value ) [inherited]** Definition at line 28 of file `__init__.py`.

**def forcebalance.target.Target.absrd ( self, inum = None ) [inherited]** Supply the correct directory specified by user's "read" option.

Definition at line 393 of file `target.py`.

Here is the call graph for this function:



**def forcebalance.target.Target.check\_files( self, there ) [inherited]** Check this directory for the presence of readable files when the 'read' option is set.

Definition at line 364 of file target.py.

**def forcebalance.thermo.Thermo.get( self, mvals, AGrad = True, AHess = True )** Return the contribution to the total objective function.

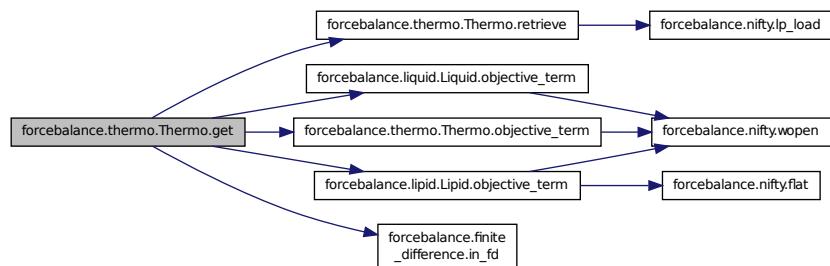
This is a weighted average of the calculated quantities.

**Parameters** mvals : list Mathematical parameter values. AGrad : Boolean Switch to turn on analytic gradient. AHess : Boolean Switch to turn on analytic Hessian.

**Returns** Answer : dict Contribution to the objective function. Answer is a dict with keys X for the objective function, G for its gradient and H for its Hessian.

Definition at line 361 of file thermo.py.

Here is the call graph for this function:



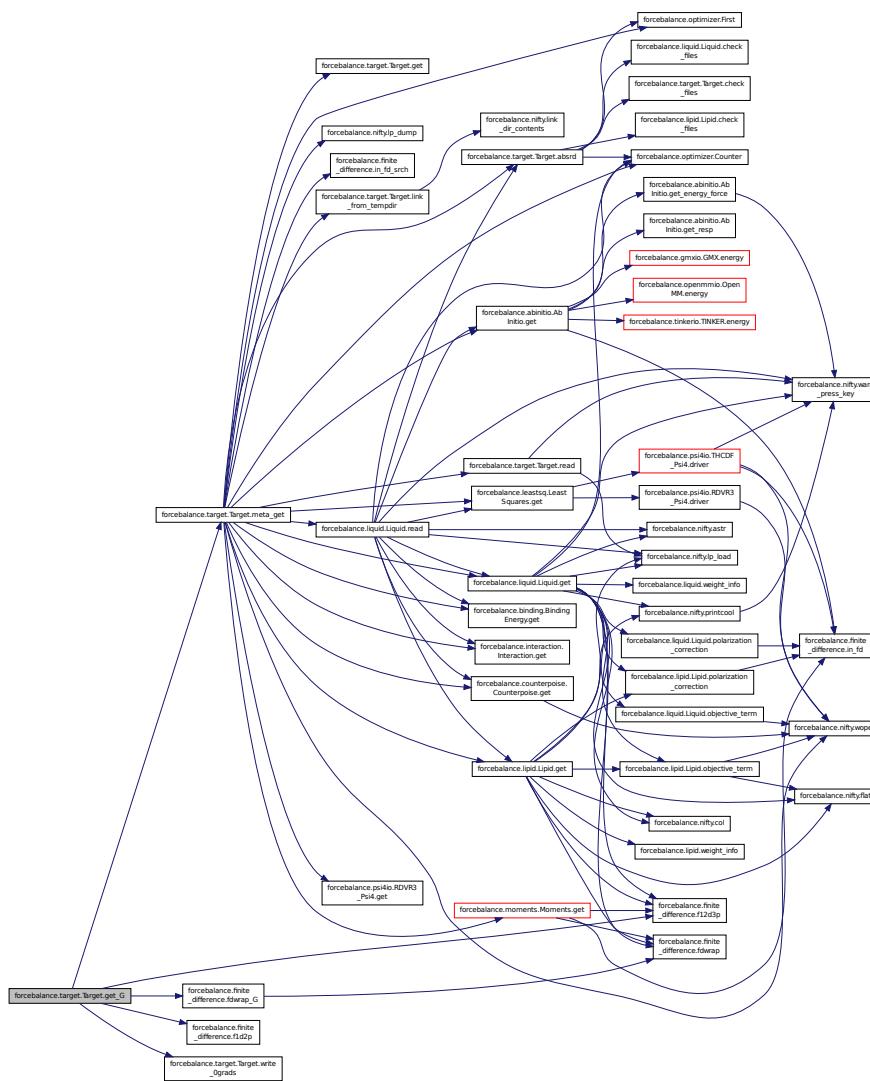
**def forcebalance.target.Target.get\_G( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the `fd1_pids` and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

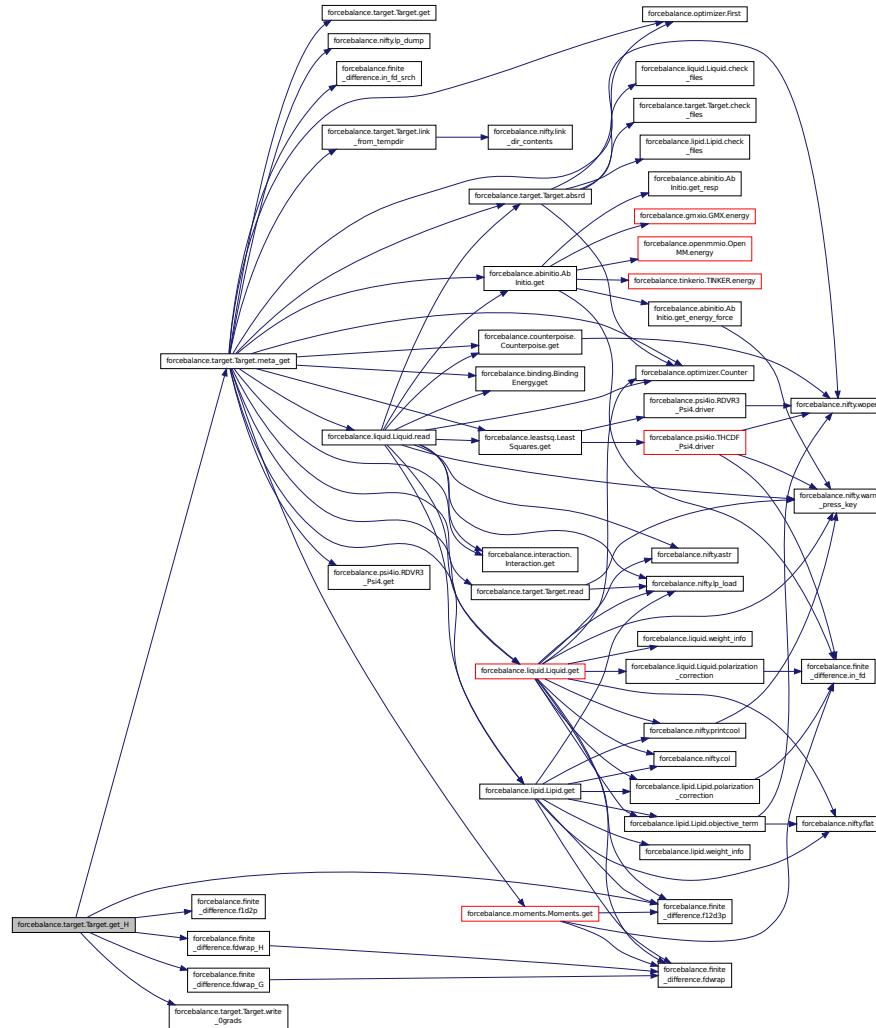
First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through

the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

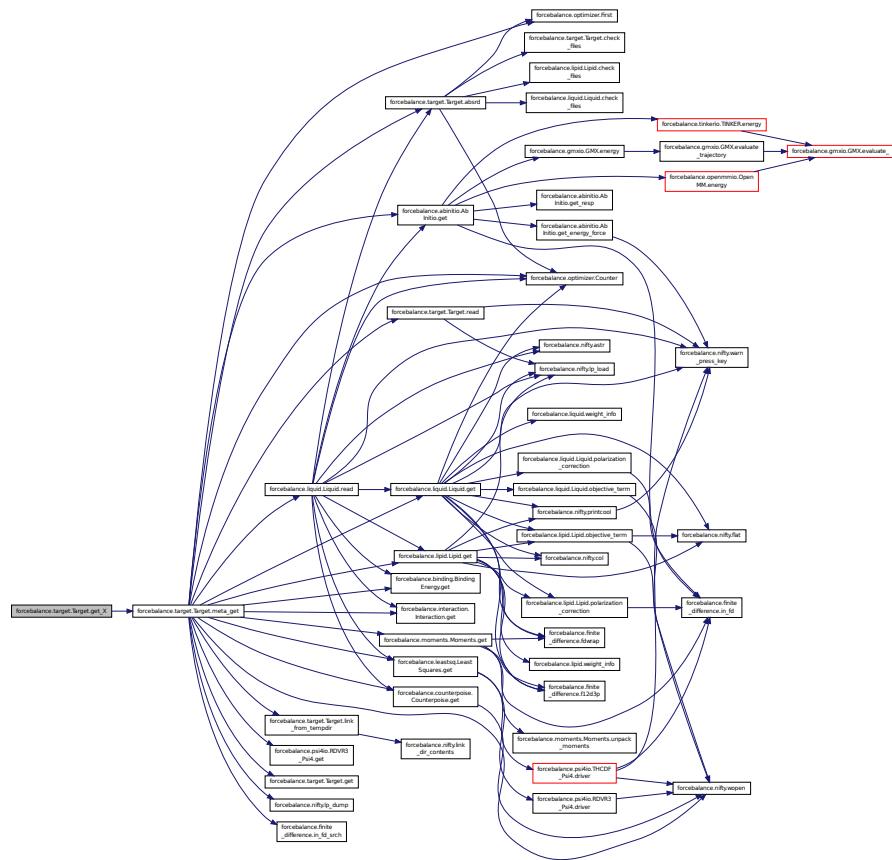
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

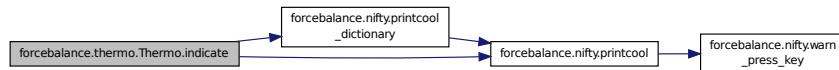
Here is the call graph for this function:



**def forcebalance.thermo.Thermo.indicate ( self )** Shows optimization state.

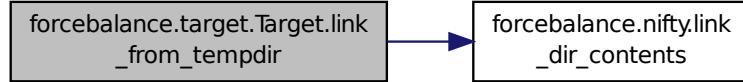
Definition at line 220 of file thermo.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

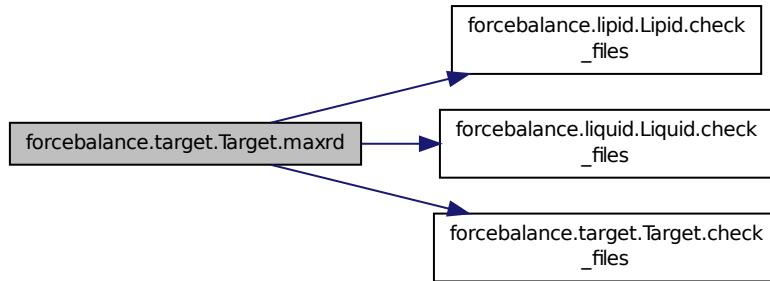
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

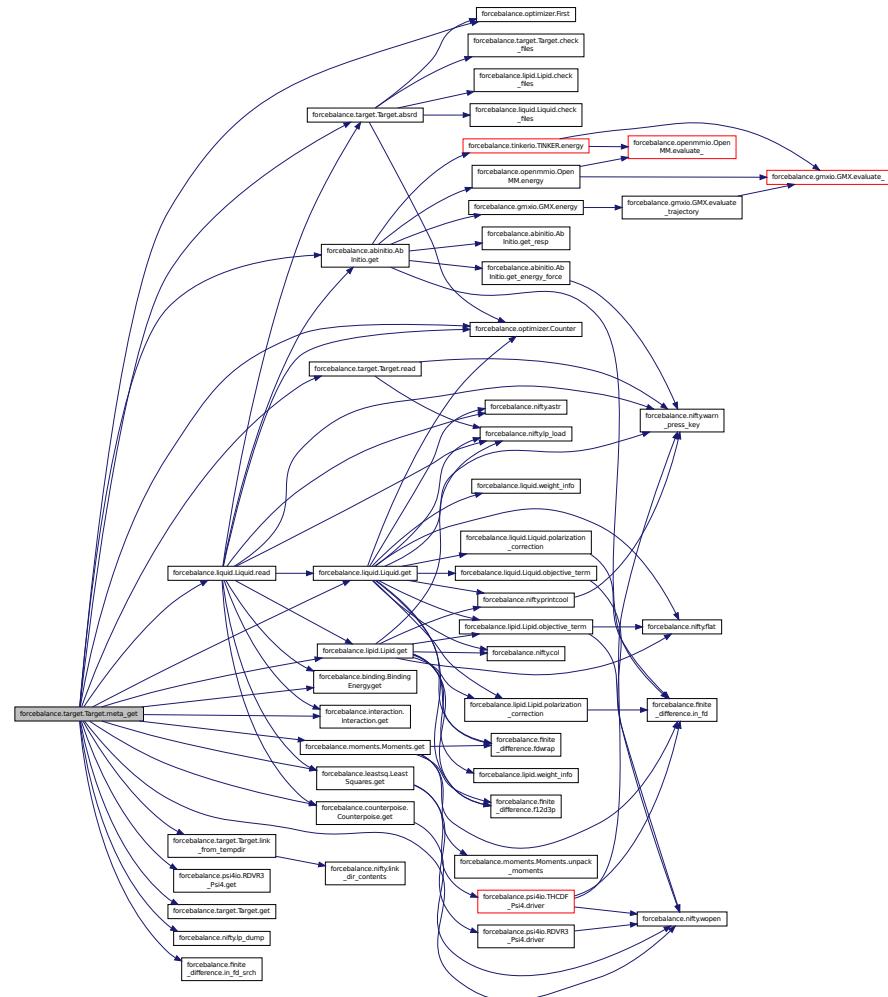


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

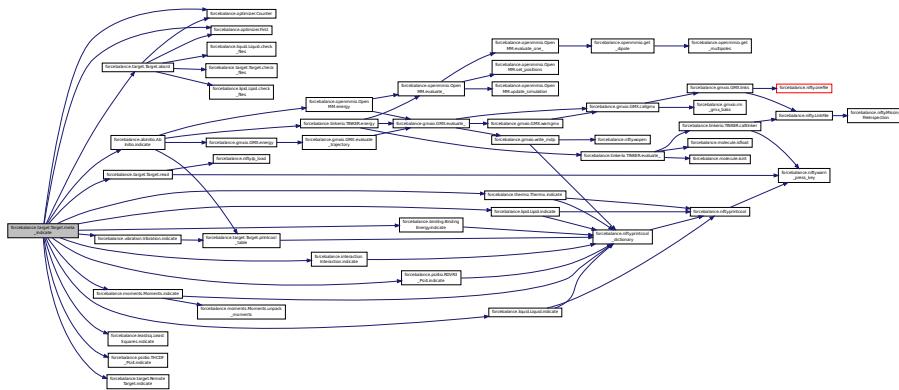
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



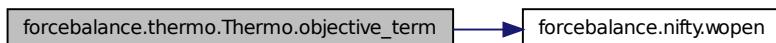
**def forcebalance.thermo.Thermo.objective\_term ( self, quantity )** Calculates the contribution to the objective function (the term) for a given quantity.

**Parameters** quantity : string Calculate the objective term for this quantity.

**Returns** term : dict term is a dict with keys X, G, H and info. The values of these keys are the objective term itself (X), its gradient (G), its Hessian (H), and an OrderedDict with print information on individual data points (info).

Definition at line 273 of file thermo.py.

Here is the call graph for this function:



**def forcebalance.target.Target.printcool\_table ( self, data = OrderedDict ([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited]** Print target information in an organized table format.

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool.dictionary.

The input should be something like:

**Parameters**

data	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
------	---

<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

Here is the call graph for this function:

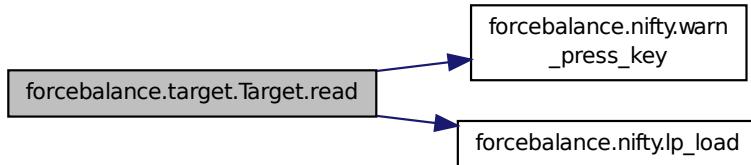


**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**

Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

**def forcebalance.thermo.Thermo.retrieve ( self, dp )** Retrieve the molecular dynamics (MD) results and store the calculated quantities in the [Point](#) object dp.

**Parameters** dp : [Point](#) Store the calculated quantities in this point.

**Returns** Nothing

Definition at line 140 of file thermo.py.

Here is the call graph for this function:



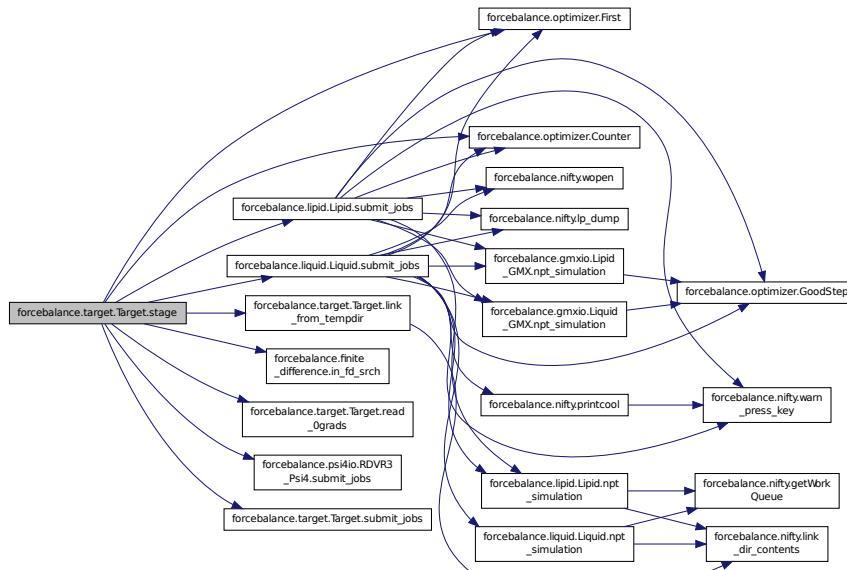
```
def forcebalance.BaseClass.set_option( self, in_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

```
def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



```
def forcebalance.thermo.Thermo.submit_jobs( self, mvals, AGrad = True, AHess = True ) This routine is called by Objective.stage() and will run before "get".
```

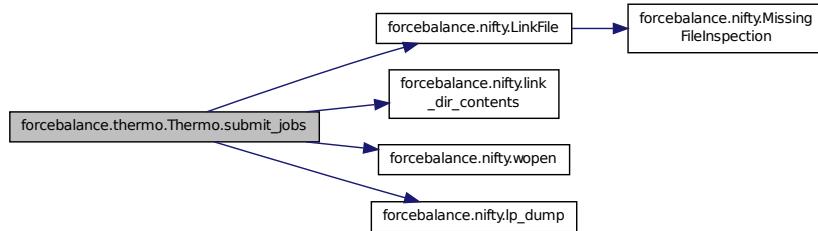
It submits the jobs and the [stage\(\)](#) function will wait for jobs to complete.

**Parameters** mvals : list Mathematical parameter values. AGrad : Boolean Switch to turn on analytic gradient. AHess : Boolean Switch to turn on analytic Hessian.

**Returns** Nothing.

Definition at line 179 of file thermo.py.

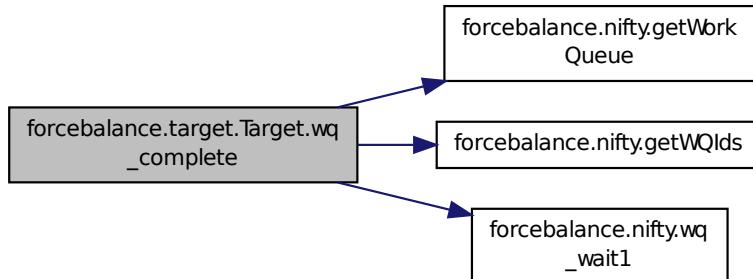
Here is the call graph for this function:



**def forcebalance.target.Target.wq\_complete( self ) [inherited]** This method determines whether the Work Queue tasks for the current target have completed.

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.69.4 Member Data Documentation

**forcebalance.thermo.Thermo.denoms** Definition at line 47 of file thermo.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.thermo.Thermo.Gp** Definition at line 418 of file thermo.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.thermo.Thermo.Objective** Definition at line 420 of file thermo.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.  
Dictionary of whether to call the derivatives.  
Definition at line 127 of file target.py.

**forcebalance.thermo.Thermo.points** Definition at line 45 of file thermo.py.

**forcebalance.thermo.Thermo.Pp** Definition at line 415 of file thermo.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd [inherited]** Root directory of the whole project.  
Submit jobs to the Work Queue.  
Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.  
Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate [inherited]** Whether to read indicate.log from file when restarting an aborted run.  
Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective [inherited]** Whether to read objective.p from file when restarting an aborted run.  
Definition at line 172 of file target.py.

**forcebalance.target.Target.rundir [inherited]** self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.  
Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number  
The 'customdir' is customizable and can go below anything.  
Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.  
Definition at line 158 of file target.py.

**forcebalance.thermo.Thermo.simpfx** Initialize base class.  
Parameters Reference experimental data Variables Prefix names for simulation data  
Definition at line 43 of file thermo.py.

**forcebalance.target.Target.tempbase [inherited]** Relative directory of target.  
Temporary (working) directory; it is temp/(target\_name) Used for storing temporary variables that don't change through the course of the optimization  
Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.thermo.Thermo.weights** Definition at line 49 of file thermo.py.

**forcebalance.thermo.Thermo.Wp** Definition at line 413 of file thermo.py.

**forcebalance.target.Target.write\_indicate** [inherited] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [inherited] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

**forcebalance.thermo.Thermo.Xp** Definition at line 412 of file thermo.py.

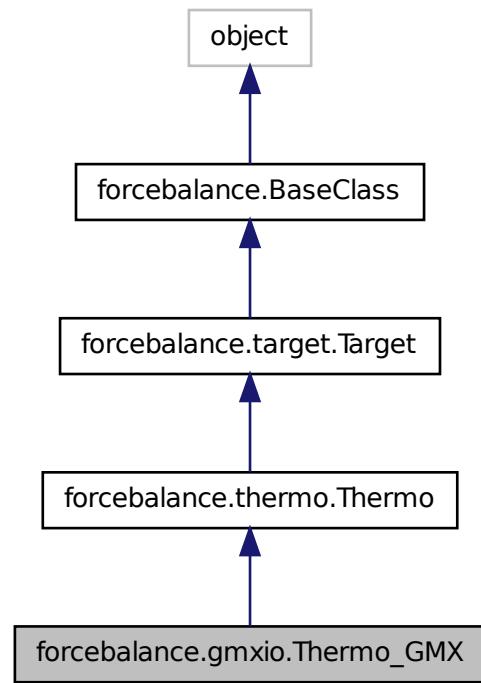
The documentation for this class was generated from the following file:

- [thermo.py](#)

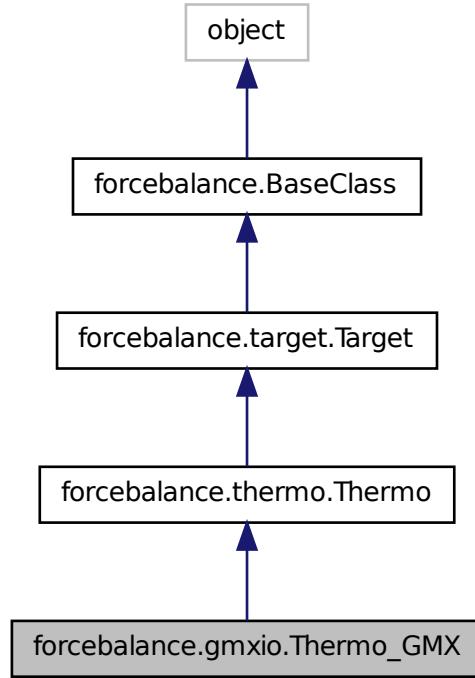
## 8.70 forcebalance.gmxio.Thermo\_GMX Class Reference

Thermodynamical property matching using GROMACS.

Inheritance diagram for forcebalance.gmxio.Thermo\_GMX:



Collaboration diagram for forcebalance.gmxio.Thermo\_GMX:



## Public Member Functions

- def `__init__`
- def `retrieve`

*Retrieve the molecular dynamics (MD) results and store the calculated quantities in the `Point` object dp.*
- def `submit_jobs`

*This routine is called by `Objective.stage()` and will run before "get".*
- def `indicate`

*Shows optimization state.*
- def `objective_term`

*Calculates the contribution to the objective function (the term) for a given quantity.*
- def `get`

*Return the contribution to the total objective function.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

- def `get_H`  
*Computes the objective function contribution and its gradient.*
- def `link_from_tempdir`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `engine_`
- `engname`
- `mdpx`
- `scripts`
- `simpfx`  
*Initialize base class.*
- `points`
- `denoms`
- `weights`
- `Xp`
- `Wp`
- `Pp`
- `Gp`
- `Objective`
- `rd`  
*Root directory of the whole project.*
- `pgrad`  
*Iteration where we turn on zero-gradient skipping.*
- `tempbase`

- Relative directory of target.*
- `tempdir`
  - `rundir`

*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
  - `FF`

*Need the forcefield (here for now)*
  - `xct`

*Counts how often the objective function was computed.*
  - `gct`

*Counts how often the gradient was computed.*
  - `hct`

*Counts how often the Hessian was computed.*
  - `read_indicate`

*Whether to read indicate.log from file when restarting an aborted run.*
  - `write_indicate`

*Whether to write indicate.log at every iteration (true for all but remote.)*
  - `read_objective`

*Whether to read objective.p from file when restarting an aborted run.*
  - `write_objective`

*Whether to write objective.p at every iteration (true for all but remote.)*
  - `verbose_options`
  - `PrintOptionDict`

### 8.70.1 Detailed Description

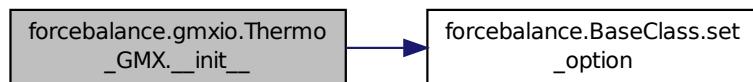
Thermodynamical property matching using GROMACS.

Definition at line 1503 of file gmxio.py.

### 8.70.2 Constructor & Destructor Documentation

**def forcebalance.gmxio.Thermo\_GMX.\_\_init\_\_( self, options, tgt\_opts, forcefield )** Definition at line 1504 of file gmxio.py.

Here is the call graph for this function:



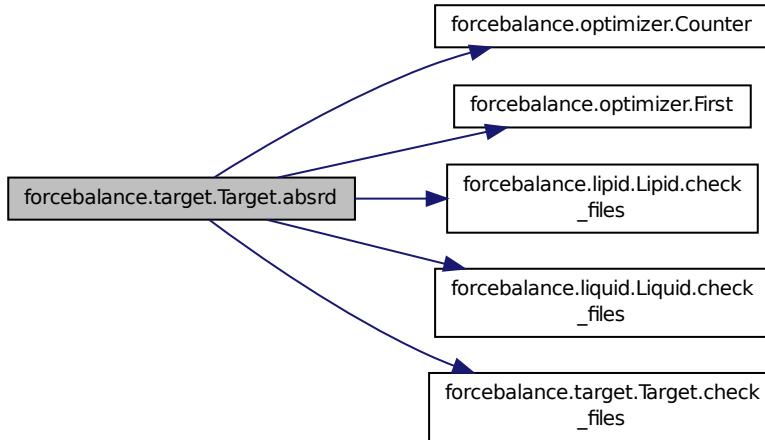
### 8.70.3 Member Function Documentation

**def forcebalance.BaseClass.\_\_setattr\_\_( self, key, value ) [inherited]** Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory  
specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.check_files ( self, there ) [inherited] Check this directory for the presence  
of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.thermo.Thermo.get ( self, mvals, AGrad = True, AHess = True ) [inherited]  
Return the contribution to the total objective function.
```

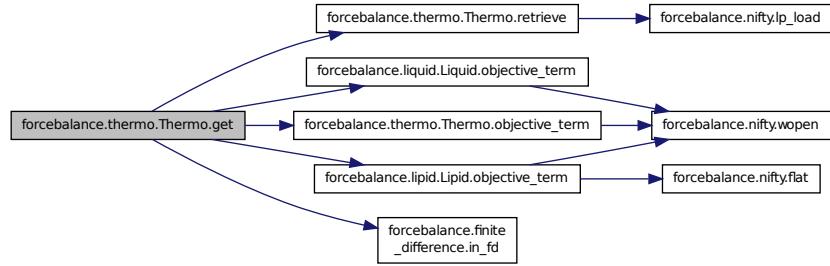
This is a weighted average of the calculated quantities.

**Parameters** mvals : list Mathematical parameter values. AGrad : Boolean Switch to turn on analytic gradient. AHess : Boolean Switch to turn on analytic Hessian.

**Returns** Answer : dict Contribution to the objective function. Answer is a dict with keys X for the objective function, G for its gradient and H for its Hessian.

Definition at line 361 of file thermo.py.

Here is the call graph for this function:



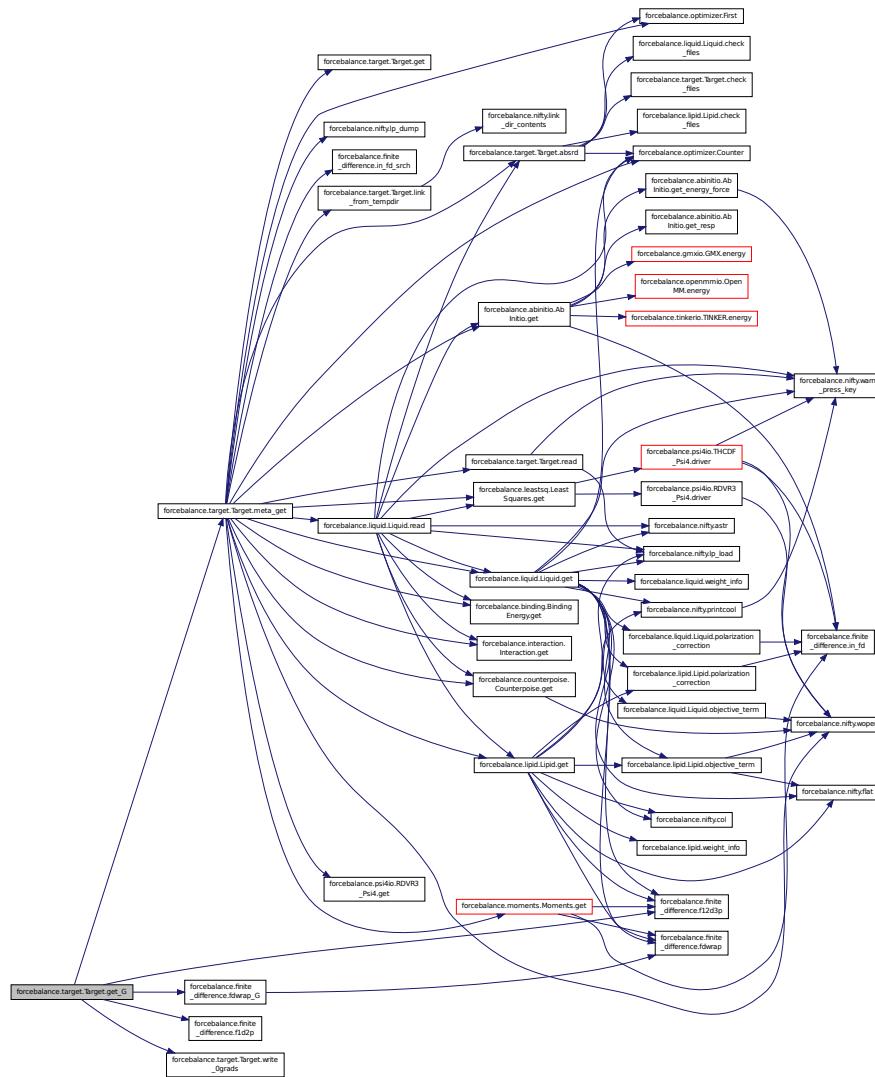
```
def forcebalance.target.Target.get_G( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the `fd1.pids` and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



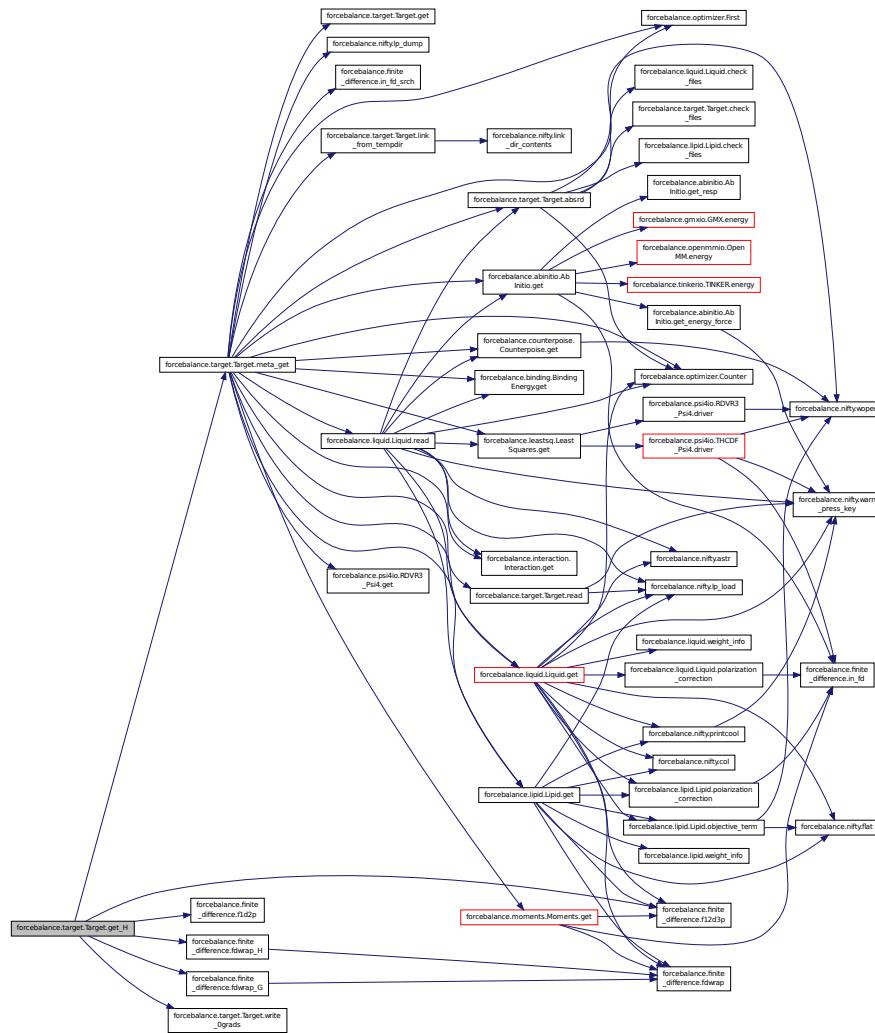
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

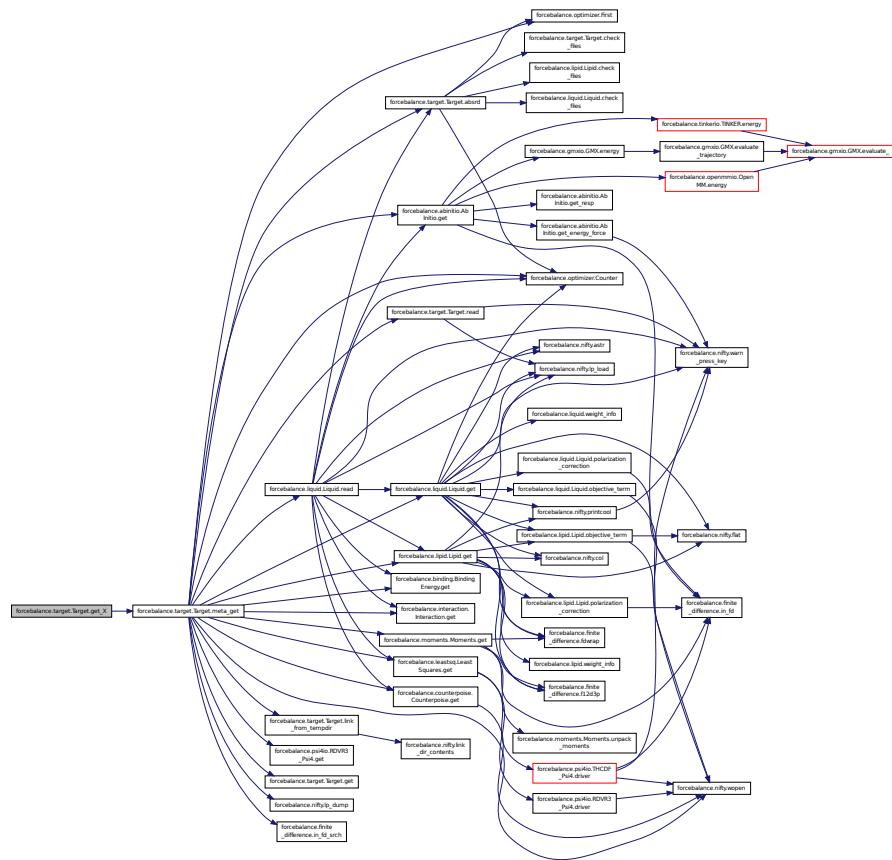
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

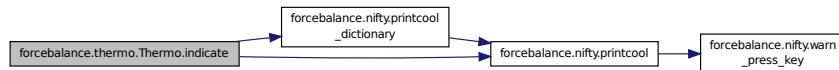
Here is the call graph for this function:



**def forcebalance.thermo.Thermo.indicate ( self ) [inherited]** Shows optimization state.

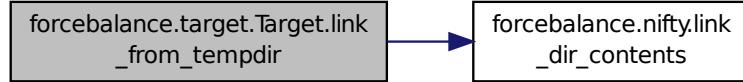
Definition at line 220 of file thermo.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

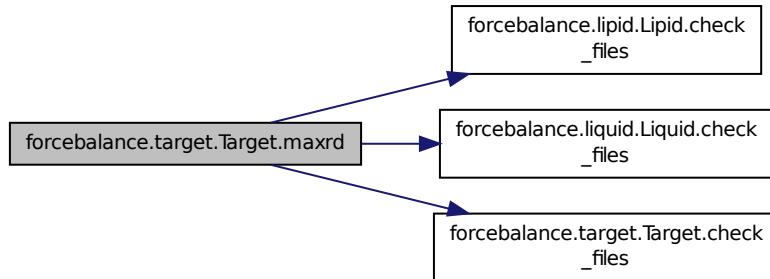
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

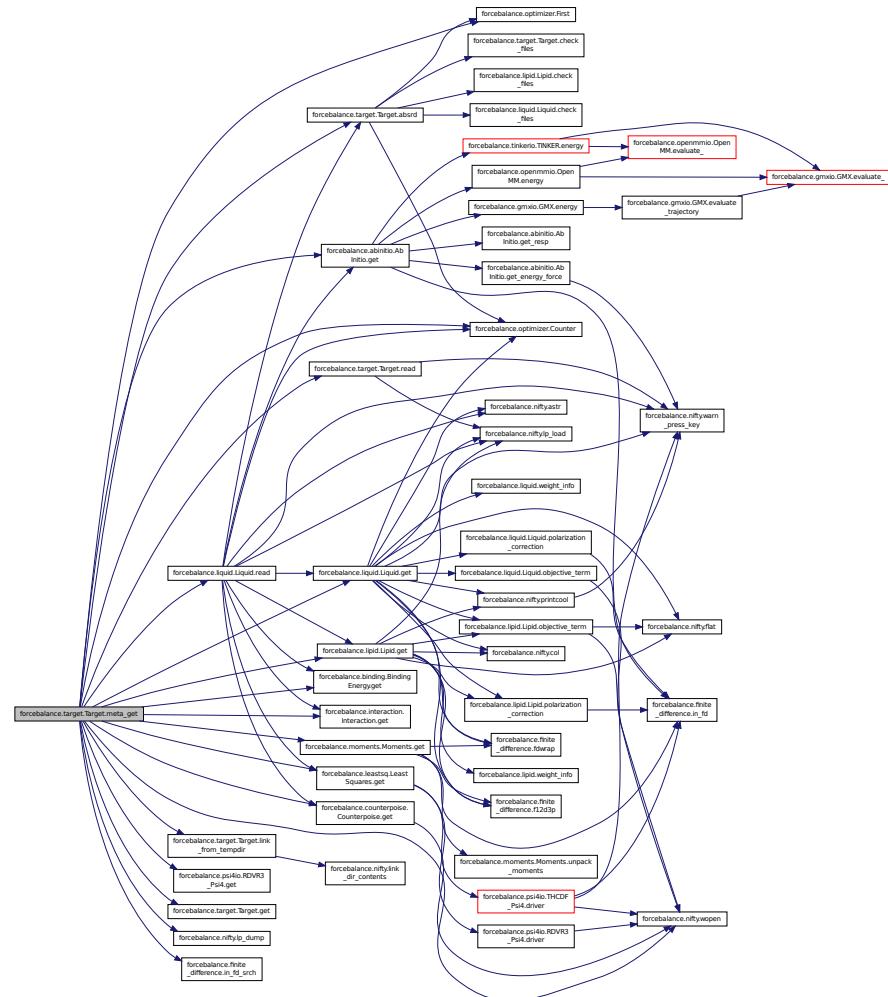


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

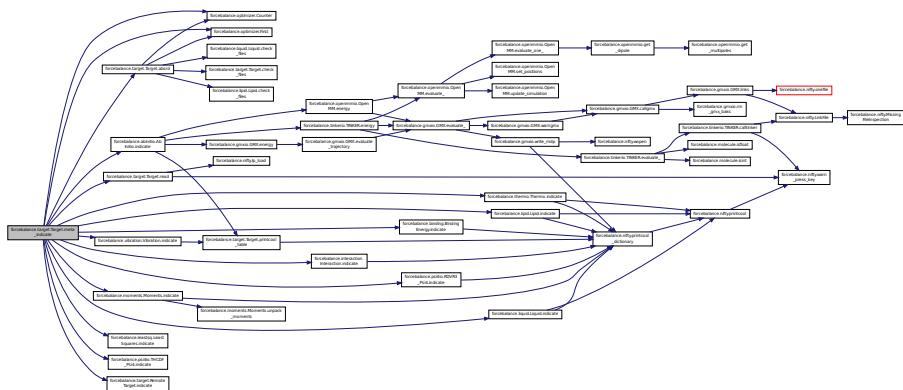
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



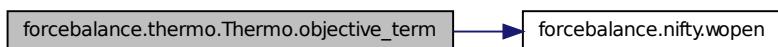
**def forcebalance.thermo.Thermo.objective\_term ( self, quantity ) [inherited]** Calculates the contribution to the objective function (the term) for a given quantity.

**Parameters** quantity : string Calculate the objective term for this quantity.

**Returns** term : dict term is a dict with keys X, G, H and info. The values of these keys are the objective term itself (X), its gradient (G), its Hessian (H), and an OrderedDict with print information on individual data points (info).

Definition at line 273 of file thermo.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

The input should be something like:

## Parameters

<b>data</b>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
-------------	---

<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

Here is the call graph for this function:

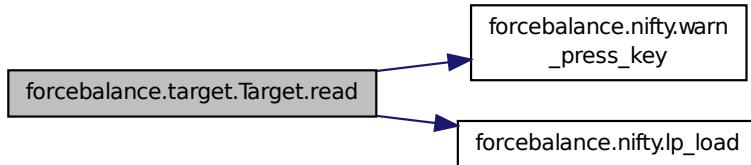


**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**

Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.target.Target.refresh\_temp\_directory( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

**def forcebalance.thermo.Thermo.retrieve ( self, dp ) [inherited]** Retrieve the molecular dynamics (MD) results and store the calculated quantities in the [Point](#) object dp.

**Parameters** dp : [Point](#) Store the calculated quantities in this point.

**Returns** Nothing

Definition at line 140 of file thermo.py.

Here is the call graph for this function:



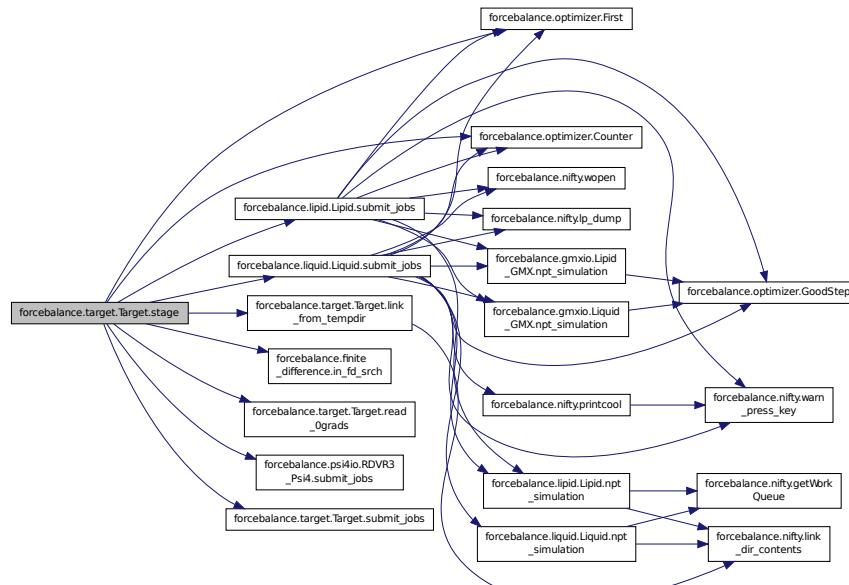
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 42 of file __init__.py.
```

```
def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



```
def forcebalance.thermo.Thermo.submit_jobs( self, mvals, AGrad = True, AHess = True ) [inherited] This routine is called by Objective.stage() and will run before "get".
```

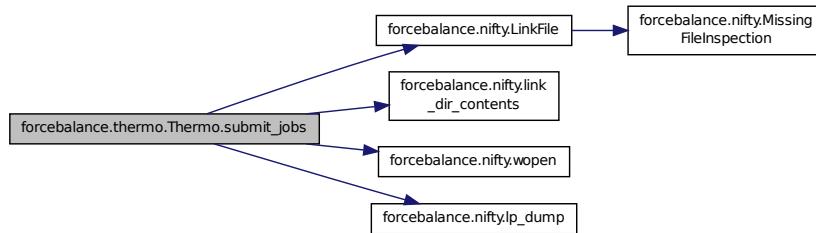
It submits the jobs and the [stage\(\)](#) function will wait for jobs to complete.

**Parameters** mvals : list Mathematical parameter values. AGrad : Boolean Switch to turn on analytic gradient. AHess : Boolean Switch to turn on analytic Hessian.

**Returns** Nothing.

Definition at line 179 of file thermo.py.

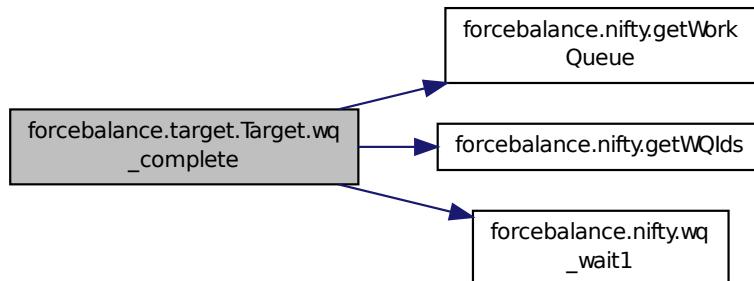
Here is the call graph for this function:



**def forcebalance.target.Target.wq\_complete( self ) [inherited]** This method determines whether the Work Queue tasks for the current target have completed.

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.70.4 Member Data Documentation

**forcebalance.thermo.Thermo.denoms [inherited]** Definition at line 47 of file thermo.py.

**forcebalance.gmxio.Thermo\_GMX.engine\_** Definition at line 1509 of file gmxio.py.

**forcebalance.gmxio.Thermo\_GMX.engname** Definition at line 1511 of file gmxio.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)  
Definition at line 160 of file target.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.  
Definition at line 164 of file target.py.

**forcebalance.thermo.Thermo.Gp** [inherited] Definition at line 418 of file thermo.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.  
Definition at line 166 of file target.py.

**forcebalance.gmxio.Thermo\_GMX.mdpfx** Definition at line 1513 of file gmxio.py.

**forcebalance.thermo.Thermo.Objective** [inherited] Definition at line 420 of file thermo.py.

**forcebalance.target.Target.pgrad** [inherited] Iteration where we turn on zero-gradient skipping.  
Dictionary of whether to call the derivatives.  
Definition at line 127 of file target.py.

**forcebalance.thermo.Thermo.points** [inherited] Definition at line 45 of file thermo.py.

**forcebalance.thermo.Thermo.Pp** [inherited] Definition at line 415 of file thermo.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd** [inherited] Root directory of the whole project.  
Submit jobs to the Work Queue.  
Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.  
Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate** [inherited] Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [inherited] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.  
Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number  
The 'customdir' is customizable and can go below anything.  
Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.  
Definition at line 158 of file target.py.

**forcebalance.gmxio.Thermo\_GMX.scripts** Definition at line 1515 of file gmxio.py.

**forcebalance.thermo.Thermo.simpfx** [**inherited**] Initialize base class.

Parameters Reference experimental data Variables Prefix names for simulation data  
Definition at line 43 of file thermo.py.

**forcebalance.target.Target.tempbase** [**inherited**] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [**inherited**] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.thermo.Thermo.weights** [**inherited**] Definition at line 49 of file thermo.py.

**forcebalance.thermo.Thermo.Wp** [**inherited**] Definition at line 413 of file thermo.py.

**forcebalance.target.Target.write\_indicate** [**inherited**] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

**forcebalance.thermo.Thermo.Xp** [**inherited**] Definition at line 412 of file thermo.py.

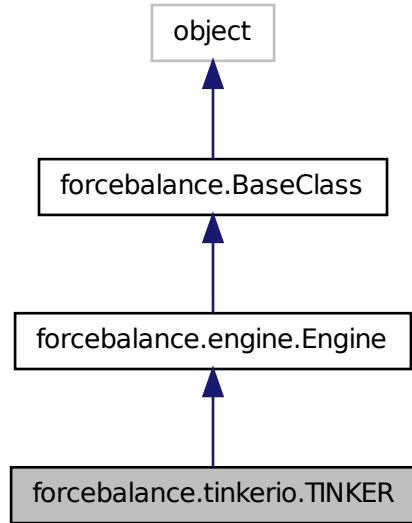
The documentation for this class was generated from the following file:

- [gmxio.py](#)

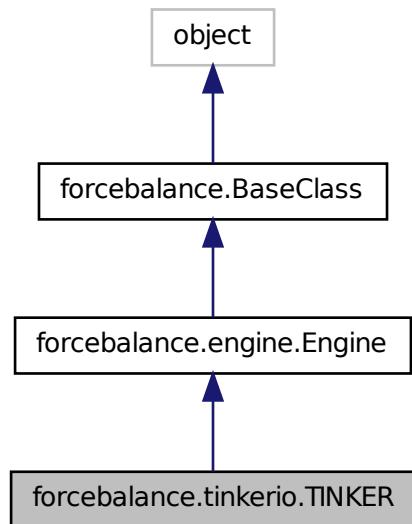
## 8.71 forcebalance.tinkerio.TINKER Class Reference

Engine for carrying out general purpose TINKER calculations.

Inheritance diagram for forcebalance.tinkerio.TINKER:



Collaboration diagram for forcebalance.tinkerio.TINKER:



## Public Member Functions

- def `_init_`  
*Called by `init` ; Set TINKER-specific options.*
- def `readsrc`  
*Called by `init` ; read files from the source directory.*
- def `calltinker`  
*Call TINKER; prepend the tinkerpath to calling the TINKER program.*
- def `prepare`  
*Called by `init` ; prepare the temp directory and figure out the topology.*
- def `optimize`  
*Optimize the geometry and align the optimized geometry to the starting geometry.*
- def `evaluate_`  
*Utility function for computing energy, and (optionally) forces and dipoles using TINKER.*
- def `energy_force_one`  
*Computes the energy and force using TINKER for one snapshot.*
- def `energy`  
*Computes the energy using TINKER over a trajectory.*
- def `energy_force`  
*Computes the energy and force using TINKER over a trajectory.*
- def `energy_dipole`  
*Computes the energy and dipole using TINKER over a trajectory.*
- def `normal_modes`
- def `multipole_moments`  
*Return the multipole moments of the 1st snapshot in Debye and Buckingham units.*
- def `energy_rmsd`  
*Calculate energy of the selected structure (optionally minimize and return the minimized energy and RMSD).*
- def `interaction_energy`  
*Calculate the interaction energy for two fragments.*
- def `molecular_dynamics`  
*Method for running a molecular dynamics simulation.*
- def `prepare`
- def `__setattr__`
- def `set_option`

## Public Attributes

- `valkwds`  
*Keyword args that aren't in this list are filtered out.*
- `warn_vn`
- `tinkerpath`  
*The directory containing TINKER executables (e.g.*
- `key`
- `prm`
- `mol`
- `rigid`  
*Attempt to set some TINKER options.*
- `pbc`

- `abskey`
- `AtomMask`  
*If the coordinates do not come with TINKER suffixes then throw an error.*
- `AtomLists`
- `A`
- `B`
- `mdtraj`
- `name`
- `verbose`
- `target`

*Engines can get properties from the Target that creates them.*

- `root`
- `srcdir`
- `tempdir`
- `FF`
- `verbose_options`
- `PrintOptionDict`

### 8.71.1 Detailed Description

Engine for carrying out general purpose TINKER calculations.

Definition at line 304 of file tinkerio.py.

### 8.71.2 Constructor & Destructor Documentation

```
def forcebalance.tinkerio.TINKER.__init__ ( self, name = "tinker", kwargs )
```

Definition at line 307 of file tinkerio.py.

### 8.71.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited]
```

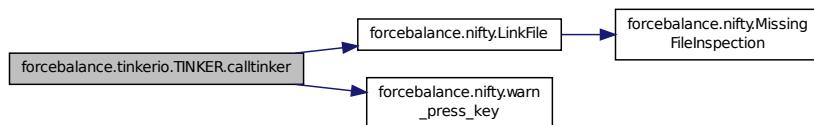
Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.tinkerio.TINKER.calltinker ( self, command, stdin = None, print_to_screen = False, print_command = False, kwargs )
```

Call TINKER; prepend the tinkerpath to calling the TINKER program.

Definition at line 351 of file tinkerio.py.

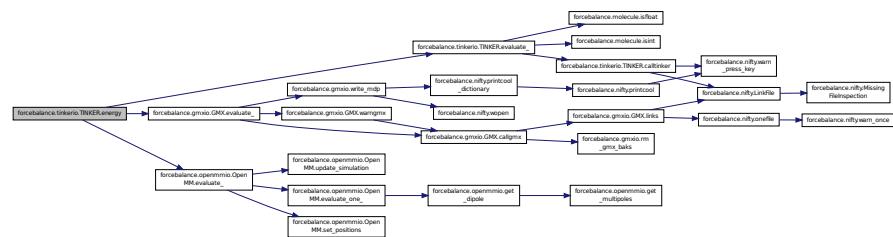
Here is the call graph for this function:



**def forcebalance.tinkerio.TINKER.energy ( self )** Computes the energy using [TINKER](#) over a trajectory.

Definition at line 654 of file tinkerio.py.

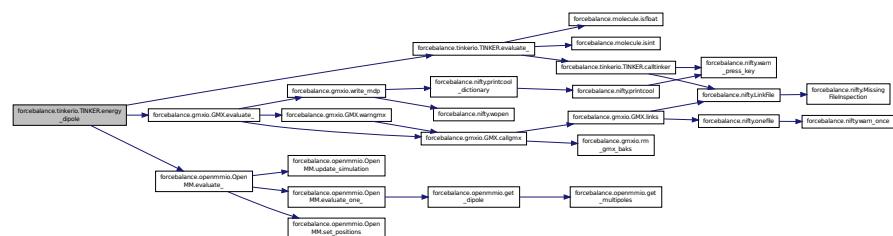
Here is the call graph for this function:



**def forcebalance.tinkerio.TINKER.energy\_dipole ( self )** Computes the energy and dipole using [TINKER](#) over a trajectory.

Definition at line 679 of file tinkerio.py.

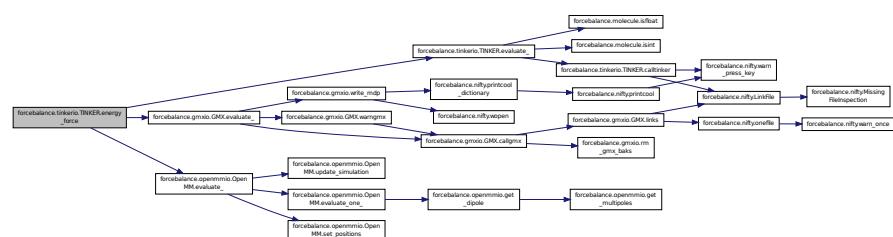
Here is the call graph for this function:



**def forcebalance.tinkerio.TINKER.energy\_force ( self )** Computes the energy and force using [TINKER](#) over a trajectory.

Definition at line 666 of file tinkerio.py.

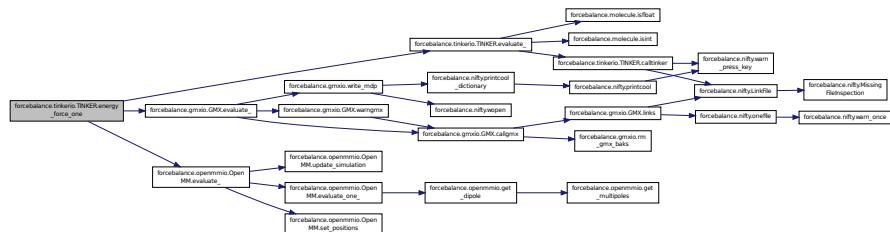
Here is the call graph for this function:



**def forcebalance.tinkerio.TINKER.energy\_force\_one ( self, shot )** Computes the energy and force using [TINKER](#) for one snapshot.

Definition at line 645 of file tinkerio.py.

Here is the call graph for this function:

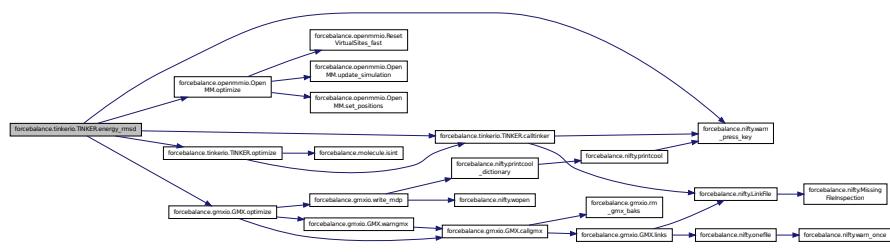


**def forcebalance.tinkerio.TINKER.energy\_rmsd ( self, shot = 0, optimize = True )** Calculate energy of the selected structure (optionally minimize and return the minimized energy and RMSD).

In kcal/mol.

Definition at line 791 of file tinkerio.py.

Here is the call graph for this function:



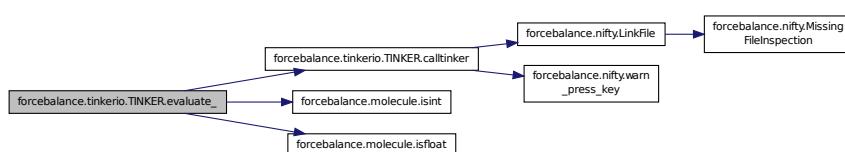
**def forcebalance.tinkerio.TINKER.evaluate\_( self, xyzin, force = False, dipole = False )** Utility function for computing energy, and (optionally) forces and dipoles using TINKER.

Inputs: xyzin: TINKER .xyz file name. force: Switch for calculating the force. dipole: Switch for calculating the dipole.

Outputs: Result: Dictionary containing energies, forces and/or dipoles.

Definition at line 596 of file tinkerio.py.

Here is the call graph for this function:



**def forcebalance.tinkerio.TINKER.interaction\_energy ( self, fraga, fragb )** Calculate the interaction energy for two fragments.

Definition at line 829 of file tinkerio.py

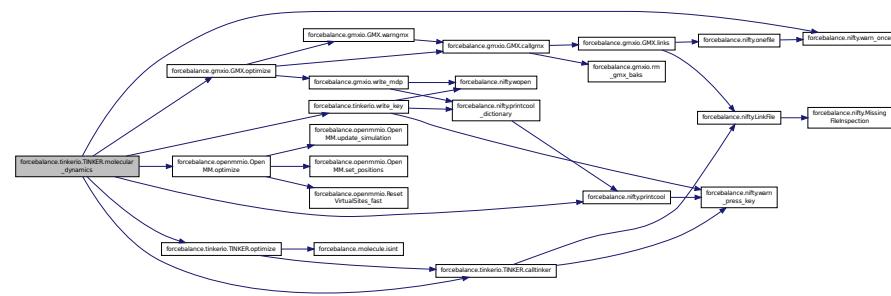
```
def forcebalance.tinkerio.TINKER.molecular_dynamics ( self, nsteps, timestep, temperature = None, pressure = None, nequil = 0, nsave = 1000, minimize = True, anisotropic = False, threads = 1, verbose = False, kwargs ) Method for running a molecular dynamics simulation.
```

Required arguments: nsteps = (int) Number of total time steps timestep = (float) Time step in FEMTOSECOND-S temperature = (float) Temperature control (Kelvin) pressure = (float) Pressure control (atmospheres) nequil = (int) Number of additional time steps at the beginning for equilibration nsave = (int) Step interval for saving and printing data minimize = (bool) Perform an energy minimization prior to dynamics threads = (int) Specify how many OpenMP threads to use

Returns simulation data: Rhos = (array) Density in kilogram m<sup>-3</sup> Potentials = (array) Potential energies Kinetics = (array) Kinetic energies Volumes = (array) Box volumes Dips = (3xN array) Dipole moments EComps = (dict) Energy components

Definition at line 860 of file tinkerio.py.

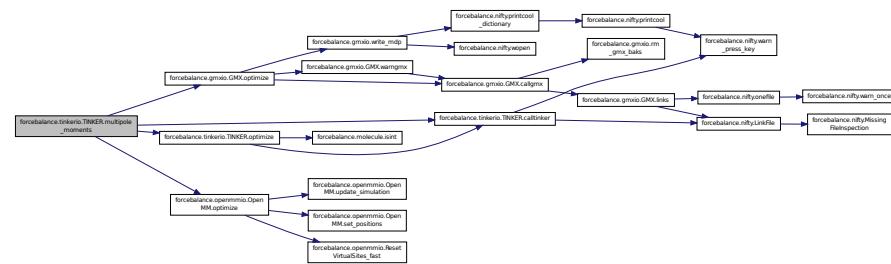
Here is the call graph for this function:



```
def forcebalance.tinkerio.TINKER.multipole_moments ( self, shot = 0, optimize = True, polarizability = False ) Return the multipole moments of the 1st snapshot in Debye and Buckingham units.
```

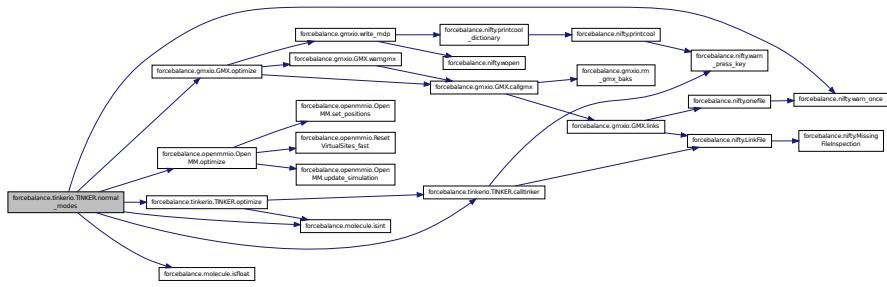
Definition at line 728 of file tinkerio.py.

Here is the call graph for this function:



```
def forcebalance.tinkerio.TINKER.normal_modes ( self, shot = 0, optimize = True ) Definition at line 690 of file tinkerio.py.
```

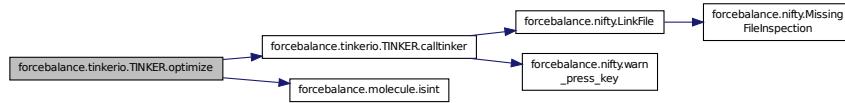
Here is the call graph for this function:



**def forcebalance.tinkerio.TINKER.optimize ( self, shot = 0, method = "newton", crit = 1e-4 )** Optimize the geometry and align the optimized geometry to the starting geometry.

Definition at line 543 of file tinkerio.py.

Here is the call graph for this function:



**def forcebalance.engine.Engine.prepare ( self, kwargs ) [inherited]** Definition at line 95 of file engine.py.

**def forcebalance.tinkerio.TINKER.prepare ( self, pbc = False, kwargs )** Called by **init** ; prepare the temp directory and figure out the topology.

Definition at line 393 of file tinkerio.py.

Here is the call graph for this function:



**def forcebalance.tinkerio.TINKER.readsrc ( self, kwargs )** Called by **init** ; read files from the source directory.

Definition at line 332 of file tinkerio.py.

**def forcebalance.BaseClass.set\_option ( self, in\_dict, src\_key, dest\_key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

**def forcebalance.tinkerio.TINKER.setopts ( self, kwargs )** Called by **init** ; Set TINKER-specific options.

Definition at line 315 of file tinkerio.py.

#### 8.71.4 Member Data Documentation

**forcebalance.tinkerio.TINKER.A** Definition at line 832 of file tinkerio.py.

**forcebalance.tinkerio.TINKER.abskey** Definition at line 481 of file tinkerio.py.

**forcebalance.tinkerio.TINKER.AtomLists** Definition at line 494 of file tinkerio.py.

**forcebalance.tinkerio.TINKER.AtomMask** If the coordinates do not come with TINKER suffixes then throw an error.

Call analyze to read information needed to build the atom lists. Parse the output of analyze.

Definition at line 493 of file tinkerio.py.

**forcebalance.tinkerio.TINKER.B** Definition at line 833 of file tinkerio.py.

**forcebalance.engine.Engine.FF** [inherited] Definition at line 65 of file engine.py.

**forcebalance.tinkerio.TINKER.key** Definition at line 335 of file tinkerio.py.

**forcebalance.tinkerio.TINKER.mdtraj** Definition at line 929 of file tinkerio.py.

**forcebalance.tinkerio.TINKER.mol** Definition at line 338 of file tinkerio.py.

**forcebalance.engine.Engine.name** [inherited] Definition at line 48 of file engine.py.

**forcebalance.tinkerio.TINKER.pbc** Definition at line 458 of file tinkerio.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.tinkerio.TINKER.prm** Definition at line 336 of file tinkerio.py.

**forcebalance.tinkerio.TINKER.rigid** Attempt to set some TINKER options.

Definition at line 399 of file tinkerio.py.

**forcebalance.engine.Engine.root** [inherited] Definition at line 56 of file engine.py.

**forcebalance.engine.Engine.srkdir** [inherited] Definition at line 57 of file engine.py.

**forcebalance.engine.Engine.target** [inherited] Engines can get properties from the Target that creates them.

Definition at line 55 of file engine.py.

**forcebalance.engine.Engine.tempdir** [inherited] Definition at line 58 of file engine.py.

**forcebalance.tinkerio.TINKER.tinkerpath** The directory containing TINKER executables (e.g. dynamic)

Definition at line 320 of file tinkerio.py.

**forcebalance.tinkerio.TINKER.valkwrd** Keyword args that aren't in this list are filtered out.

Definition at line 309 of file tinkerio.py.

`forcebalance.engine.Engine.verbose` [inherited] Definition at line 50 of file engine.py.

`forcebalance.BaseClass.verbose_options` [inherited] Definition at line 40 of file \_\_init\_\_.py.

`forcebalance.tinkerio.TINKER.warn_vn` Definition at line 310 of file tinkerio.py.

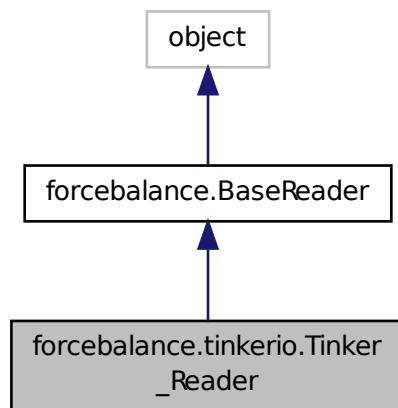
The documentation for this class was generated from the following file:

- `tinkerio.py`

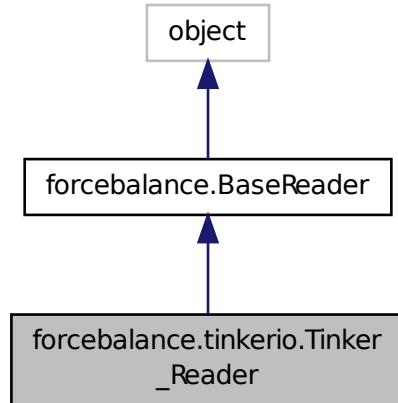
## 8.72 forcebalance.tinkerio.Tinker\_Reader Class Reference

Finite state machine for parsing TINKER force field files.

Inheritance diagram for forcebalance.tinkerio.Tinker\_Reader:



Collaboration diagram for forcebalance.tinkerio.Tinker\_Reader:



## Public Member Functions

- def `__init__`
- def `feed`

*Given a line, determine the interaction type and the atoms involved (the suffix).*
- def `Split`
- def `Whites`
- def `build.pid`

*Returns the parameter type (e.g.*

## Public Attributes

- `pdict`

*The parameter dictionary (defined in this file)*
- `atom`

*The atom numbers in the interaction (stored in the TINKER parser)*
- `itype`
- `suffix`
- `In`
- `adict`

*The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.*
- `molatom`

*The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.*
- `Molecules`
- `AtomTypes`

### 8.72.1 Detailed Description

Finite state machine for parsing [TINKER](#) force field files.

This class is instantiated when we begin to read in a file. The `feed(line)` method updates the state of the machine, informing it of the current interaction type. Using this information we can look up the interaction type and parameter type for building the parameter ID.

Definition at line 92 of file `tinkerio.py`.

### 8.72.2 Constructor & Destructor Documentation

`def forcebalance.tinkerio.Tinker_Reader.__init__ ( self, fnm )` Definition at line 94 of file `tinkerio.py`.

### 8.72.3 Member Function Documentation

`def forcebalance.BaseReader.build_pid ( self, pfd ) [inherited]` Returns the parameter type (e.g. K in BONDSK) based on the current interaction type.

Both the 'pdct' dictionary (see `gmxi0.pdict`) and the interaction type 'state' (here, BONDS) are needed to get the parameter type.

If, however, 'pdct' does not contain the ptype value, a suitable substitute is simply the field number.

Note that if the interaction type state is not set, then it defaults to the file name, so a generic parameter ID is 'filename.line\_num.field\_num'

Definition at line 124 of file `__init__.py`.

`def forcebalance.tinkerio.Tinker_Reader.feed ( self, line )` Given a line, determine the interaction type and the atoms involved (the suffix).

[TINKER](#) generally has stuff like this:

bond-cubic		-2.55			
bond-quartic		3.793125			
vdw	1		3.4050	0.1100	
vdw	2		2.6550	0.0135	0.910 # PRM 4
multipole	2	1	2	0.25983	
				-0.03859	0.00000 -0.05818
				-0.03673	
				0.00000	-0.10739
				-0.00203	0.00000 0.14412

The '#PRM 4' has no effect on [TINKER](#) but it indicates that we are tuning the fourth field on the line (the 0.910 value).

**Todo** Put the rescaling factors for [TINKER](#) parameters in here. Currently we're using the initial value to determine the rescaling factor which is not very good.

Every parameter line is prefaced by the interaction type except for 'multipole' which is on multiple lines. Because the lines that come after 'multipole' are predictable, we just determine the current line using the previous line.

Random note: Unit of force is kcal / mole / angstrom squared.

Definition at line 135 of file `tinkerio.py`.

`def forcebalance.BaseReader.Split ( self, line ) [inherited]` Definition at line 99 of file `__init__.py`.

`def forcebalance.BaseReader.Whites ( self, line ) [inherited]` Definition at line 102 of file `__init__.py`.

#### 8.72.4 Member Data Documentation

**forcebalance.BaseReader.adict [inherited]** The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.

Definition at line 89 of file `__init__.py`.

**forcebalance.tinkerio.Tinker\_Reader.atom** The atom numbers in the interaction (stored in the **TINKER** parser)

Definition at line 99 of file `tinkerio.py`.

**forcebalance.BaseReader.AtomTypes [inherited]** Definition at line 97 of file `__init__.py`.

**forcebalance.tinkerio.Tinker\_Reader.itype** Definition at line 143 of file `tinkerio.py`.

**forcebalance.BaseReader.In [inherited]** Definition at line 84 of file `__init__.py`.

**forcebalance.BaseReader.molatom [inherited]** The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.

`self.moleculerdict = OrderedDict()` The listing of 'RES:ATOMNAMES' for atom names in the line This is obviously a placeholder.

Definition at line 94 of file `__init__.py`.

**forcebalance.BaseReader.Molecules [inherited]** Definition at line 96 of file `__init__.py`.

**forcebalance.tinkerio.Tinker\_Reader.pdict** The parameter dictionary (defined in this file)

Definition at line 97 of file `tinkerio.py`.

**forcebalance.tinkerio.Tinker\_Reader.suffix** Definition at line 165 of file `tinkerio.py`.

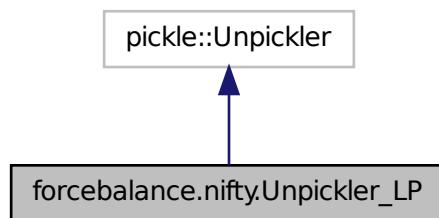
The documentation for this class was generated from the following file:

- [tinkerio.py](#)

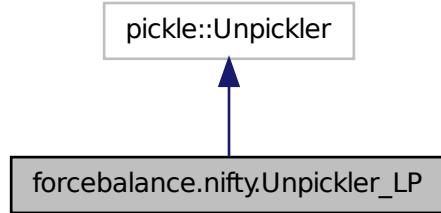
### 8.73 forcebalance.nifty.Unpickler\_LP Class Reference

A subclass of the python Unpickler that implements unpickling of `_ElementTree` types.

Inheritance diagram for `forcebalance.nifty.Unpickler_LP`:



Collaboration diagram for forcebalance.nifty.Unpickler\_LP:



### Public Member Functions

- def `__init__`

#### 8.73.1 Detailed Description

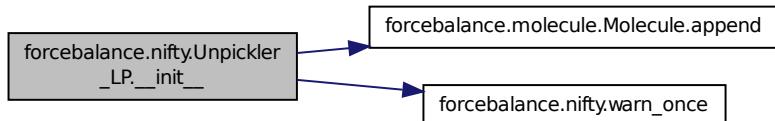
A subclass of the python Unpickler that implements unpickling of \_ElementTree types.

Definition at line 587 of file nifty.py.

#### 8.73.2 Constructor & Destructor Documentation

`def forcebalance.nifty.Unpickler_LP.__init__ ( self, file )` Definition at line 588 of file nifty.py.

Here is the call graph for this function:



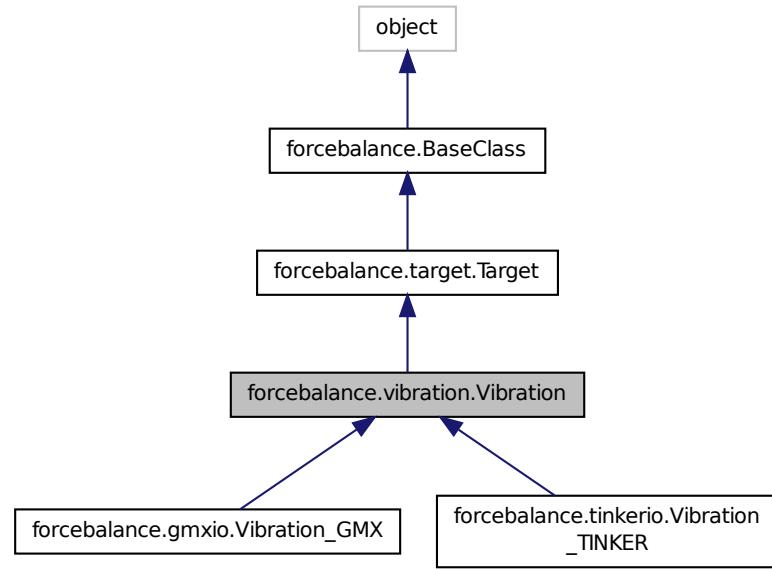
The documentation for this class was generated from the following file:

- [nifty.py](#)

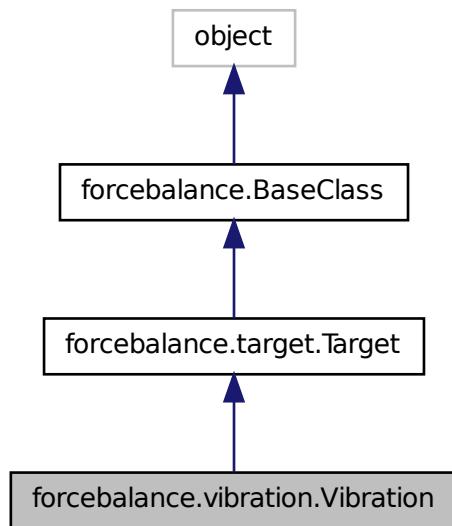
## 8.74 forcebalance.vibration.Vibration Class Reference

Subclass of Target for fitting force fields to vibrational spectra (from experiment or theory).

Inheritance diagram for forcebalance.vibration.Vibration:



Collaboration diagram for forcebalance.vibration.Vibration:



## Public Member Functions

- def `_init_`  
*Initialization.*
- def `read.reference_data`  
*Read the reference vibrational data from a file.*
- def `indicate`  
*Print qualitative indicator.*
- def `vibration_driver`
- def `process_vectors`  
*Return a set of normal and mass-weighted eigenvectors such that their outer product is the identity.*
- def `get`  
*Evaluate objective function.*
- def `get_X`  
*Computes the objective function contribution without any parametric derivatives.*
- def `read_0grads`  
*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_0grads`  
*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`  
*Computes the objective function contribution and its gradient.*
- def `get_H`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- **vfnm**  
*The vdata.txt file that contains the vibrations.*
- **engine**  
*Read in the reference data.*
- **na**  
*Number of atoms.*
- **ref\_eigvals**
- **ref\_eigvecs**
- **ref\_eigvecs\_nrm\_mw**
- **reassign**
- **c2r**
- **overlaps**
- **calc\_eigvals**
- **objective**
- **rd**  
*Root directory of the whole project.*
- **pgrad**  
*Iteration where we turn on zero-gradient skipping.*
- **tempbase**  
*Relative directory of target.*
- **tempdir**
- **rundir**  
*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- **FF**  
*Need the forcefield (here for now)*
- **xct**  
*Counts how often the objective function was computed.*
- **gct**  
*Counts how often the gradient was computed.*
- **hct**  
*Counts how often the Hessian was computed.*
- **read\_indicate**  
*Whether to read indicate.log from file when restarting an aborted run.*
- **write\_indicate**  
*Whether to write indicate.log at every iteration (true for all but remote.)*
- **read\_objective**  
*Whether to read objective.p from file when restarting an aborted run.*
- **write\_objective**  
*Whether to write objective.p at every iteration (true for all but remote.)*
- **verbose\_options**
- **PrintOptionDict**

### 8.74.1 Detailed Description

Subclass of Target for fitting force fields to vibrational spectra (from experiment or theory).

Currently Tinker is supported.

Definition at line 36 of file vibration.py.

## 8.74.2 Constructor & Destructor Documentation

```
def forcebalance.vibration.Vibration.__init__ ( self, options, tgt_opts, forcefield ) Initialization.
```

Definition at line 41 of file vibration.py.

Here is the call graph for this function:



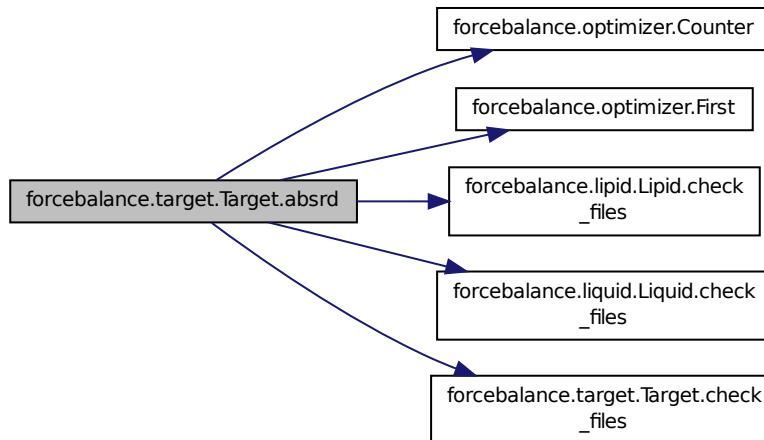
## 8.74.3 Member Function Documentation

```
def forcebalance.BaseClass.__setattr__ ( self, key, value ) [inherited] Definition at line 28 of file __init__.py.
```

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.check_files ( self, there ) [inherited] Check this directory for the presence of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.vibration.Vibration.get ( self, mvals, AGrad=False, AHess=False ) Evaluate objective function.
```

Definition at line 151 of file vibration.py.

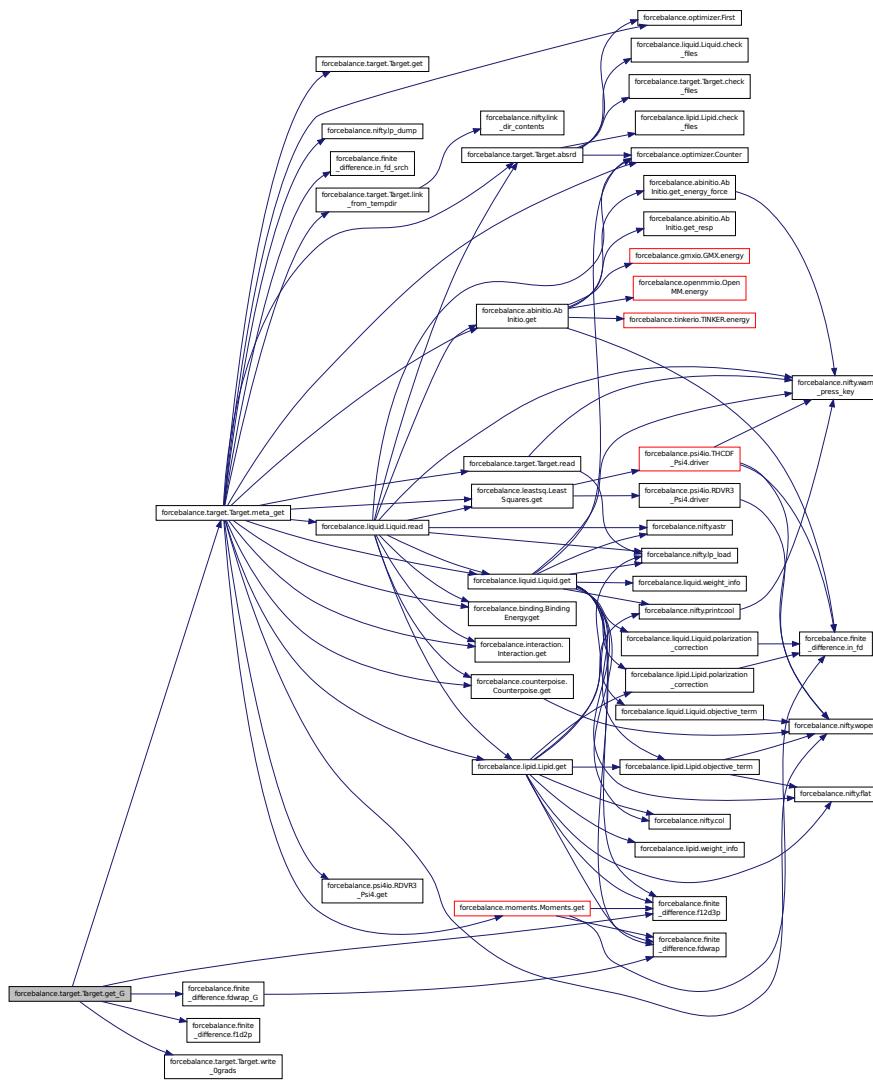
```
def forcebalance.target.Target.get_G ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.get_H ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient / Hessian.
```

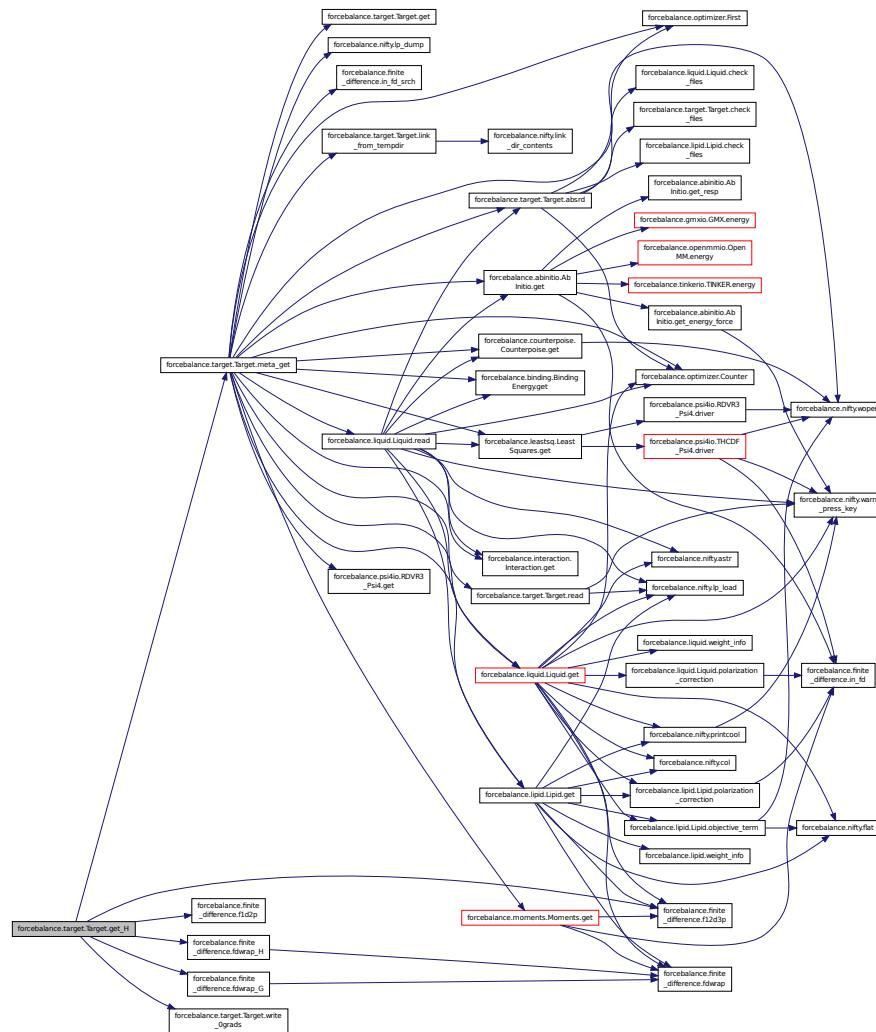
First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through

the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2\_pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

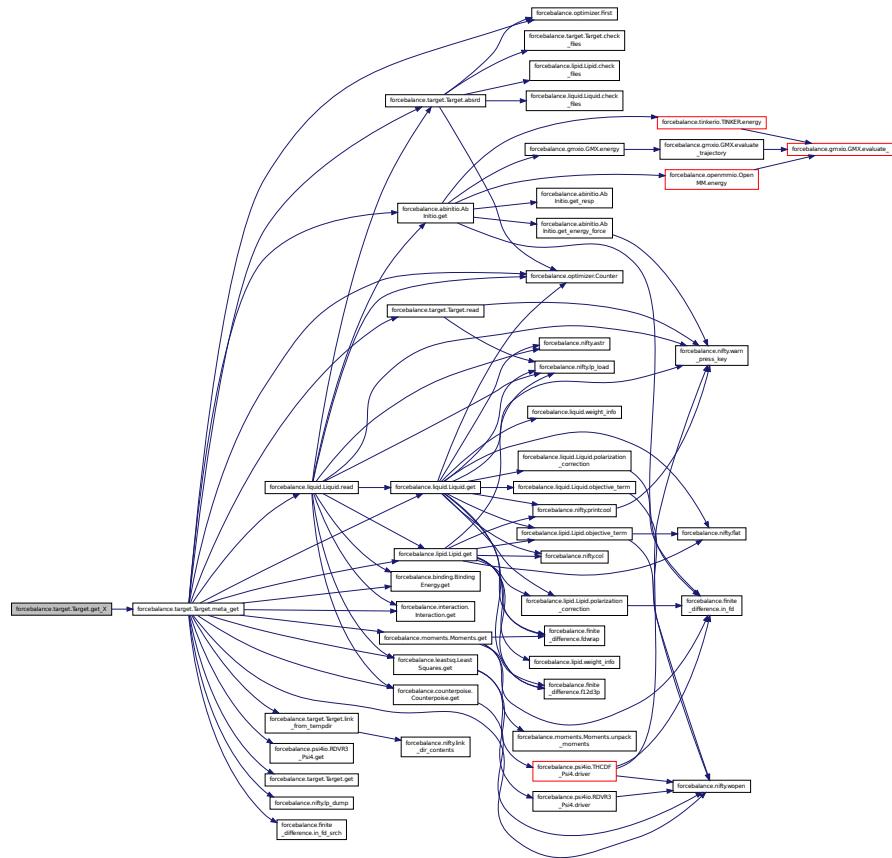
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

Here is the call graph for this function:



**def forcebalance.vibration.Vibration.indicate ( self )** Print qualitative indicator.

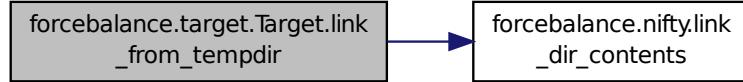
Definition at line 110 of file vibration.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

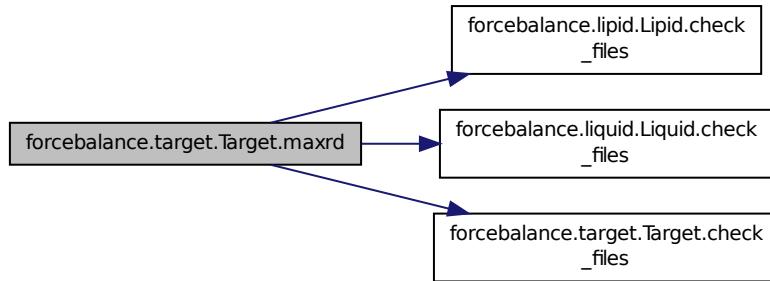
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

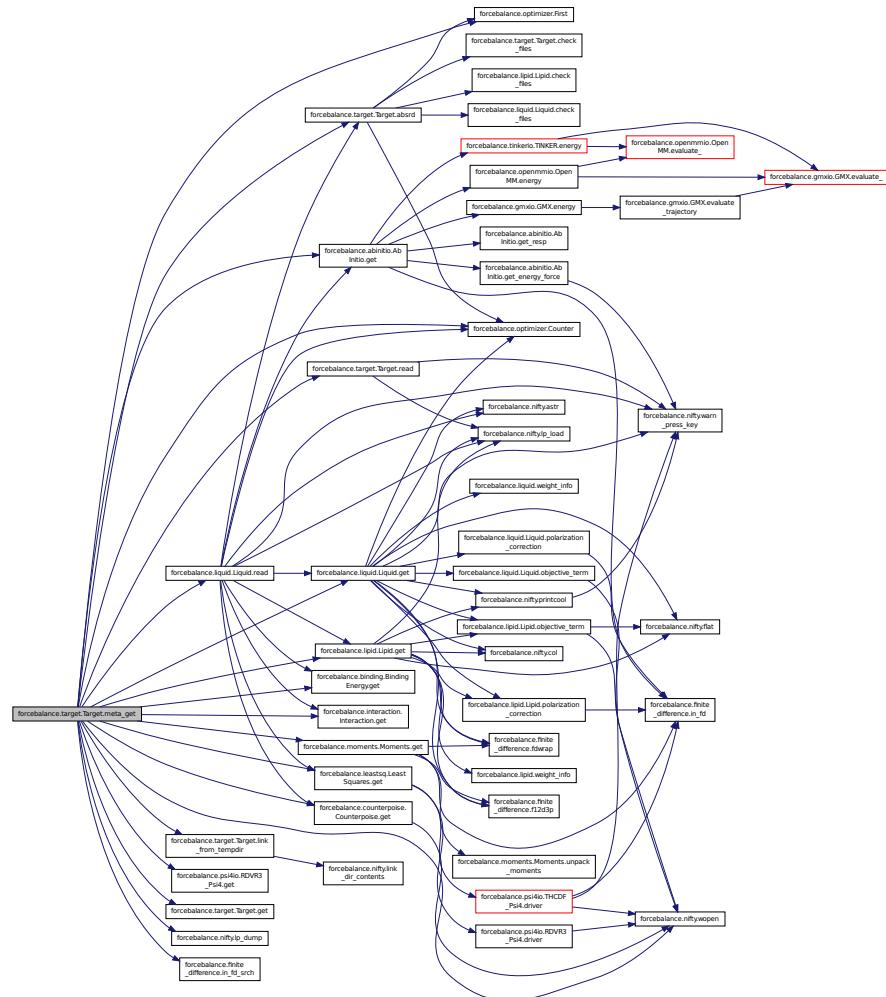


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

Here is the call graph for this function:

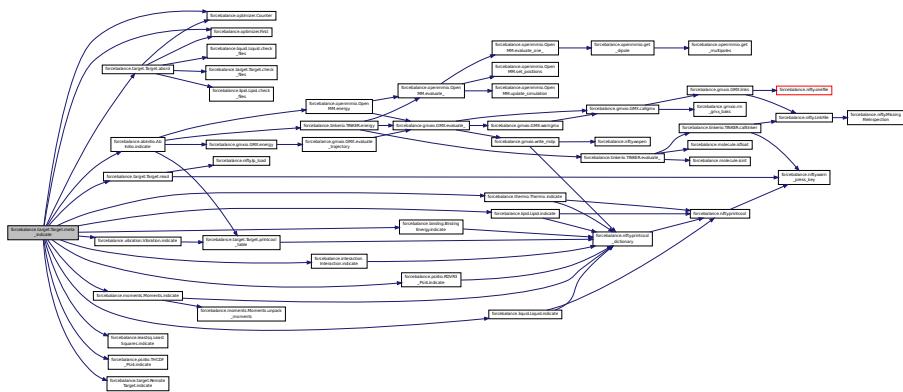


**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.

Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

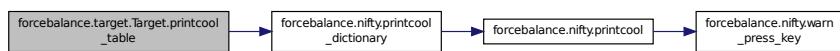
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

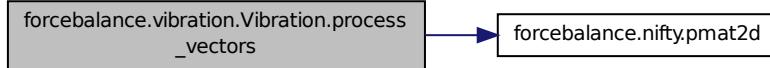
Here is the call graph for this function:



**def forcebalance.vibration.Vibration.process\_vectors ( self, vecs, verbose = False, check = False )**  
Return a set of normal and mass-weighted eigenvectors such that their outer product is the identity.

Definition at line 127 of file vibration.py.

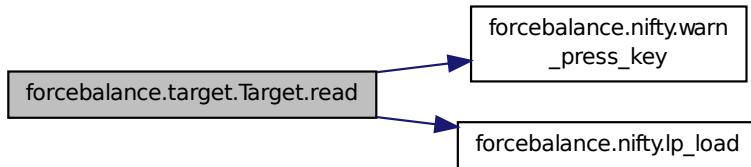
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.vibration.Vibration.read\_reference\_data ( self )** Read the reference vibrational data from a file.  
Definition at line 69 of file vibration.py.

**def forcebalance.target.Target.refresh\_temp\_directory ( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

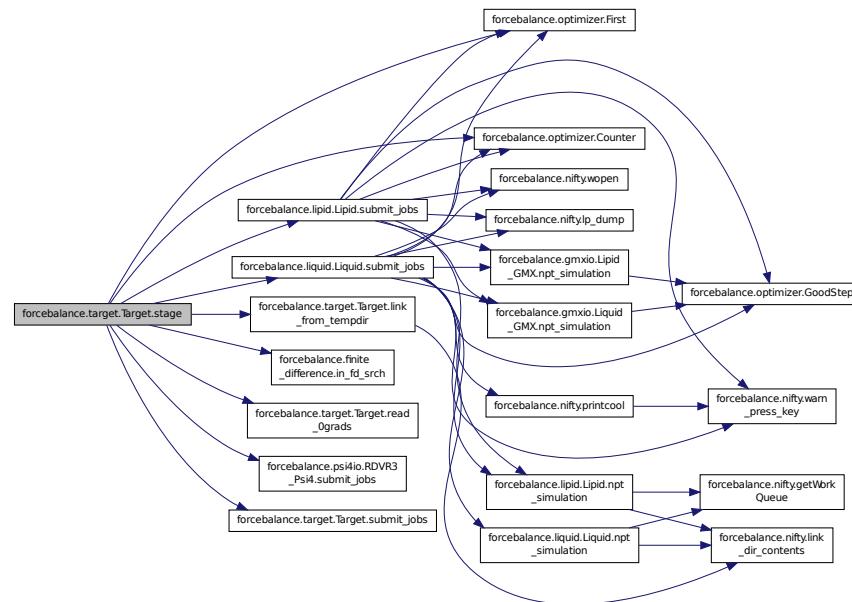
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None )  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



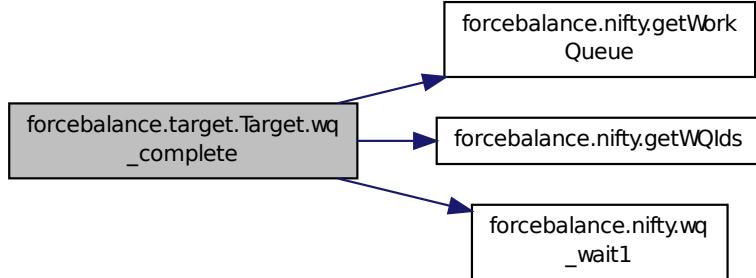
```
def forcebalance.target.Target.submit_jobs( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.vibration.Vibration.vibration_driver( self ) Definition at line 118 of file vibration.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.74.4 Member Data Documentation

**forcebalance.vibration.Vibration.c2r** Definition at line 178 of file vibration.py.

**forcebalance.vibration.Vibration.calc\_eigvals** Definition at line 198 of file vibration.py.

**forcebalance.vibration.Vibration.engine** Read in the reference data.

Build keyword dictionaries to pass to engine. Create engine object.

Definition at line 63 of file vibration.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.vibration.Vibration.na** Number of atoms.

Definition at line 71 of file vibration.py.

**forcebalance.vibration.Vibration.objective** Definition at line 199 of file vibration.py.

**forcebalance.vibration.Vibration.overlaps** Definition at line 183 of file vibration.py.

**forcebalance.target.Target.pgrad [inherited]** Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict** [**inherited**] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd** [**inherited**] Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate** [**inherited**] Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [**inherited**] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.vibration.Vibration.reassign** Definition at line 165 of file vibration.py.

**forcebalance.vibration.Vibration.ref\_eigvals** Definition at line 72 of file vibration.py.

**forcebalance.vibration.Vibration.ref\_eigvecs** Definition at line 73 of file vibration.py.

**forcebalance.vibration.Vibration.ref\_eigvecs\_nrm\_mw** Definition at line 155 of file vibration.py.

**forcebalance.target.Target.rundir** [**inherited**] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target\_name/iteration\_number The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [**inherited**] Relative directory of target.

Temporary (working) directory; it is temp/(target\_name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [**inherited**] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.vibration.Vibration.vfnm** The vdata.txt file that contains the vibrations.

Definition at line 56 of file vibration.py.

**forcebalance.target.Target.write\_indicate** [**inherited**] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

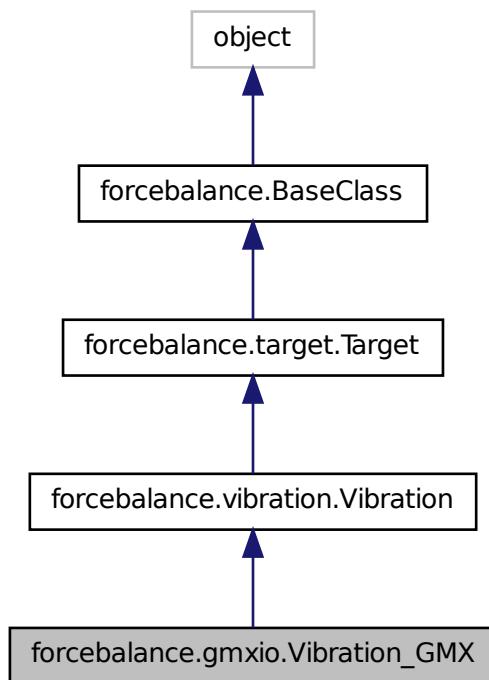
The documentation for this class was generated from the following file:

- [vibration.py](#)

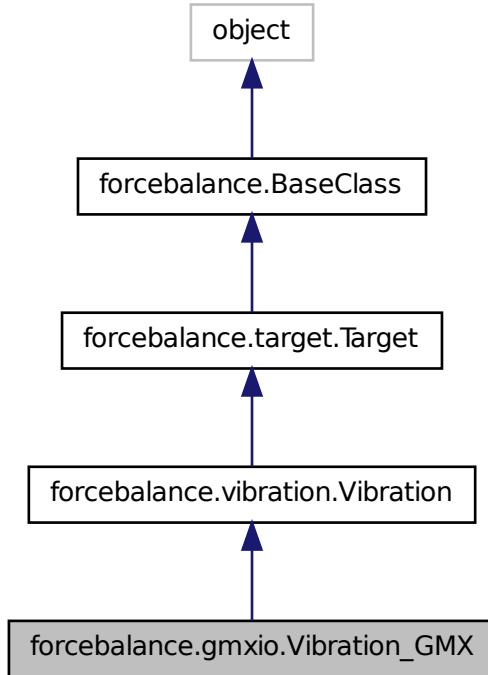
## 8.75 forcebalance.gmxio.Vibration\_GMX Class Reference

Vibrational frequency matching using GROMACS.

Inheritance diagram for forcebalance.gmxio.Vibration\_GMX:



Collaboration diagram for forcebalance.gmxio.Vibration\_GMX:



## Public Member Functions

- def `__init__`
- def `read_reference_data`

*Read the reference vibrational data from a file.*
- def `indicate`

*Print qualitative indicator.*
- def `vibration_driver`
- def `process_vectors`

*Return a set of normal and mass-weighted eigenvectors such that their outer product is the identity.*
- def `get`

*Evaluate objective function.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_0grads`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_0grads`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

*Computes the objective function contribution and its gradient.*

- def `get_H`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `link_from_tempdir`
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `engine_`  
*Default file names for coordinates and key file.*
- `vfnm`  
*The vdata.txt file that contains the vibrations.*
- `engine`  
*Read in the reference data.*
- `na`  
*Number of atoms.*
- `ref_eigvals`
- `ref_eigvecs`
- `ref_eigvecs_nrm_mw`
- `reassign`
- `c2r`
- `overlaps`
- `calc_eigvals`
- `objective`
- `rd`  
*Root directory of the whole project.*
- `pgrad`

*Iteration where we turn on zero-gradient skipping.*

- **tempbase**  
*Relative directory of target.*
- **tempdir**
- **rundir**  
*self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.*
- **FF**  
*Need the forcefield (here for now)*
- **xct**  
*Counts how often the objective function was computed.*
- **gct**  
*Counts how often the gradient was computed.*
- **hct**  
*Counts how often the Hessian was computed.*
- **read\_indicate**  
*Whether to read indicate.log from file when restarting an aborted run.*
- **write\_indicate**  
*Whether to write indicate.log at every iteration (true for all but remote.)*
- **read\_objective**  
*Whether to read objective.p from file when restarting an aborted run.*
- **write\_objective**  
*Whether to write objective.p at every iteration (true for all but remote.)*
- **verbose\_options**
- **PrintOptionDict**

### 8.75.1 Detailed Description

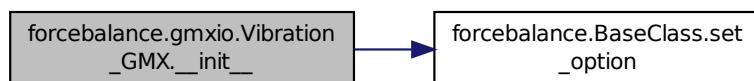
Vibrational frequency matching using GROMACS.

Definition at line 1493 of file gmxio.py.

### 8.75.2 Constructor & Destructor Documentation

**def forcebalance.gmxio.Vibration\_GMX.\_\_init\_\_ ( self, options, tgt\_opts, forcefield )** Definition at line 1494 of file gmxio.py.

Here is the call graph for this function:



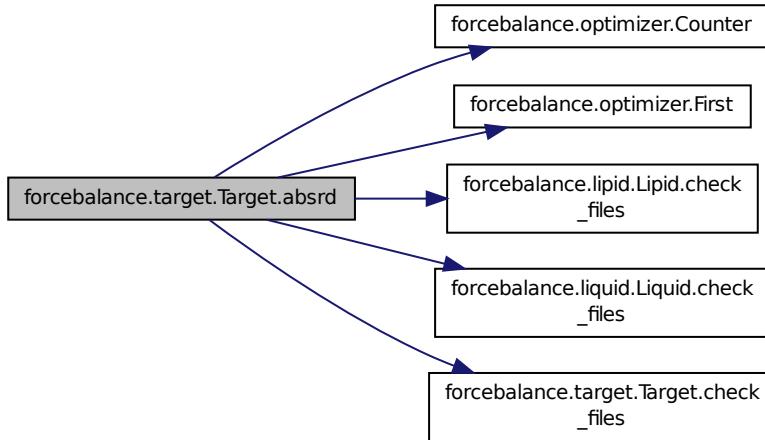
### 8.75.3 Member Function Documentation

**def forcebalance.BaseClass.\_\_setattr\_\_ ( self, key, value ) [inherited]** Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.check_files ( self, there ) [inherited] Check this directory for the presence of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.vibration.Vibration.get ( self, mvals, AGrad=False, AHess=False ) [inherited]  
Evaluate objective function.
```

Definition at line 151 of file vibration.py.

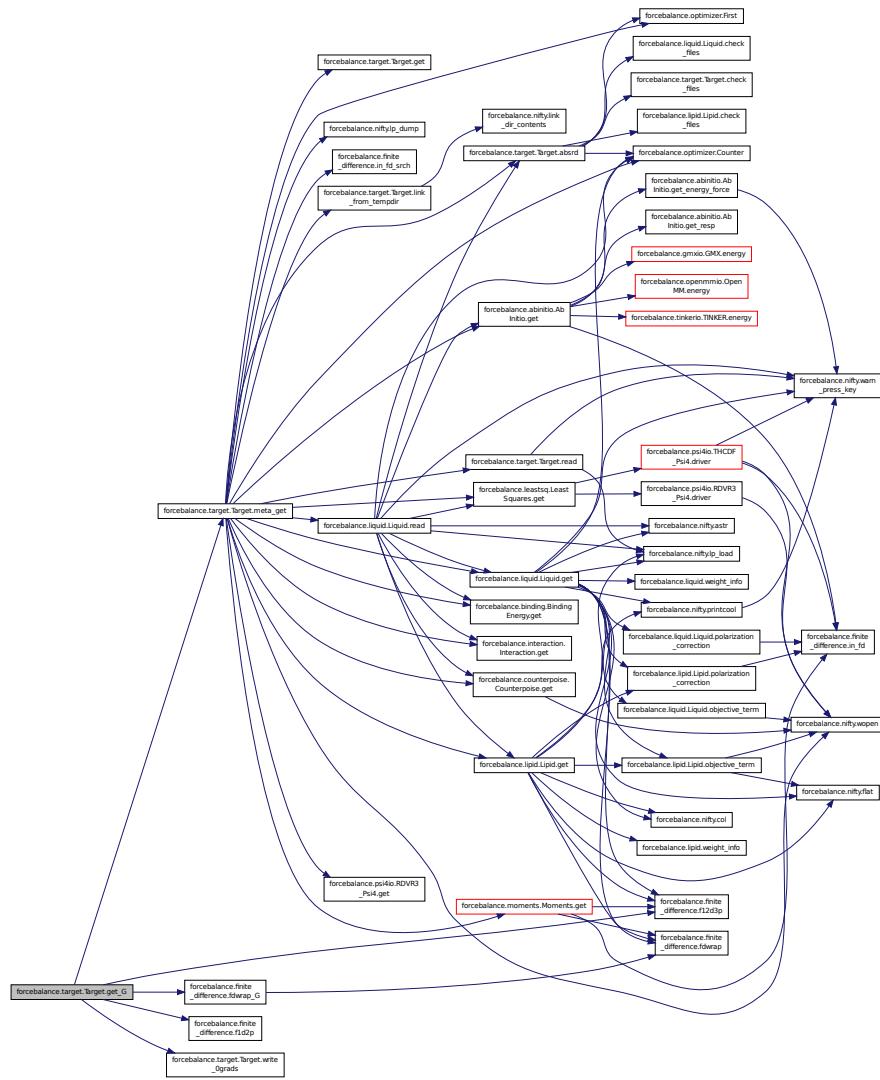
```
def forcebalance.target.Target.get_G ( self, mvals=None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



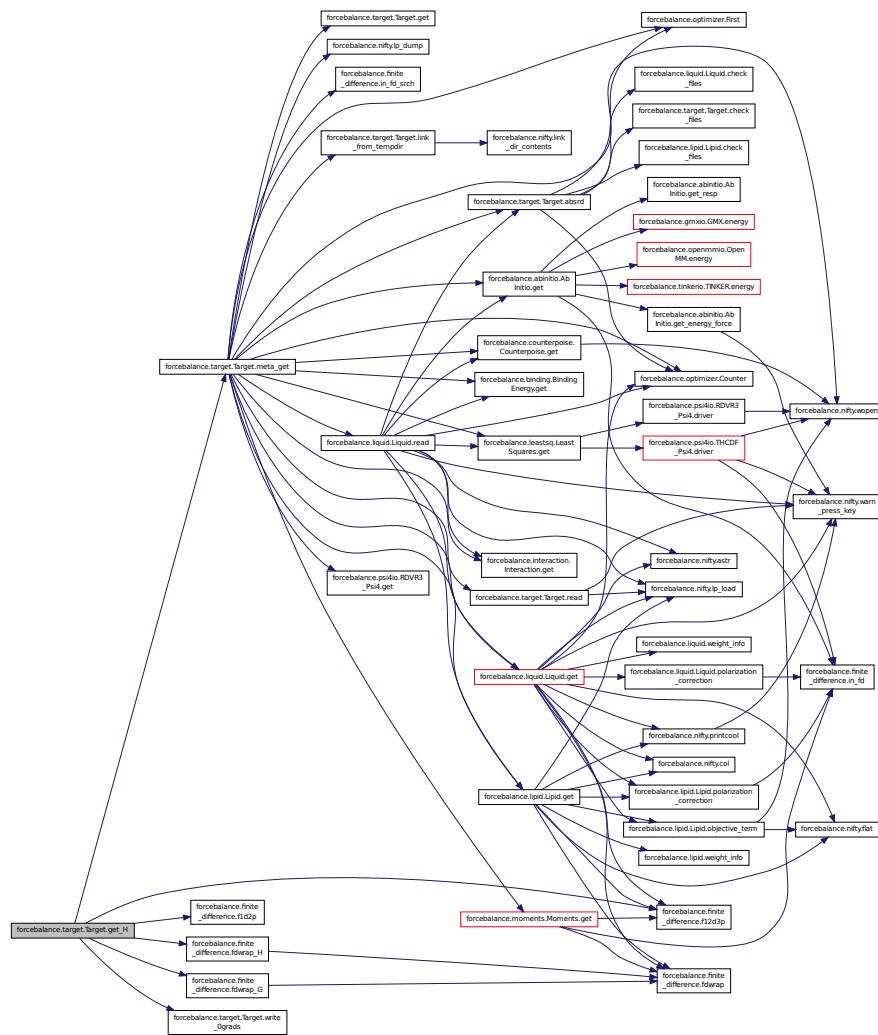
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the `fd2.pids` and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

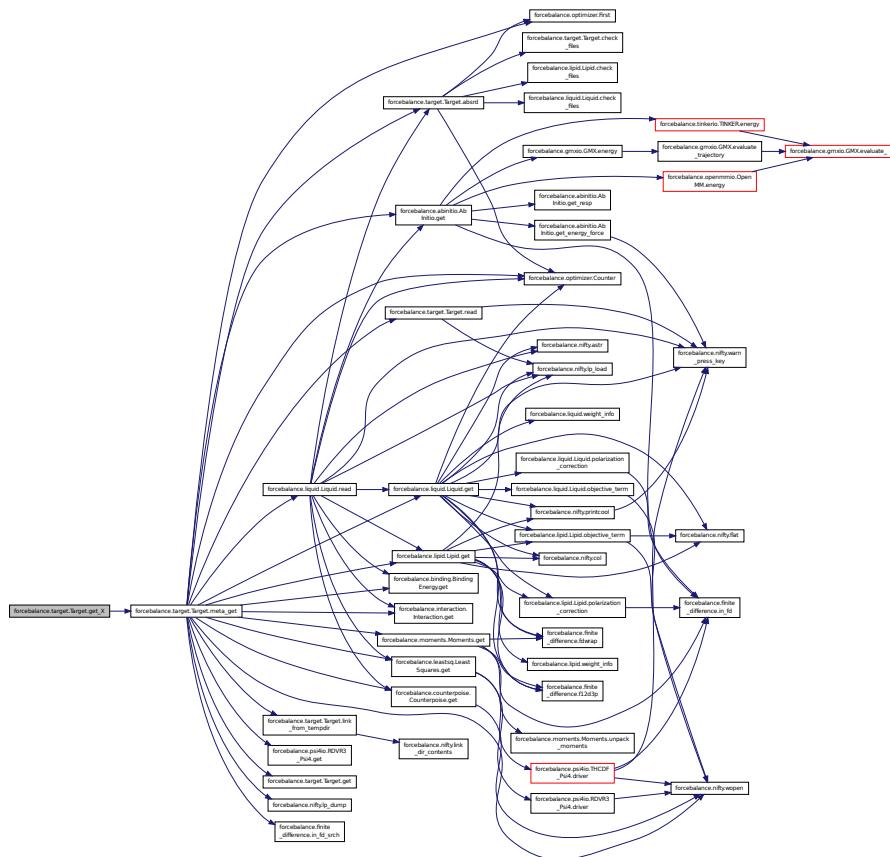
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

Here is the call graph for this function:



**def** forcebalance.vibration.Vibration.indicate (*self*) [**inherited**] Print qualitative indicator.

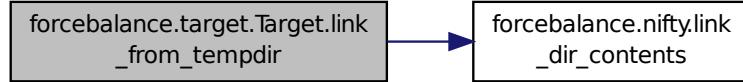
Definition at line 110 of file vibration.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

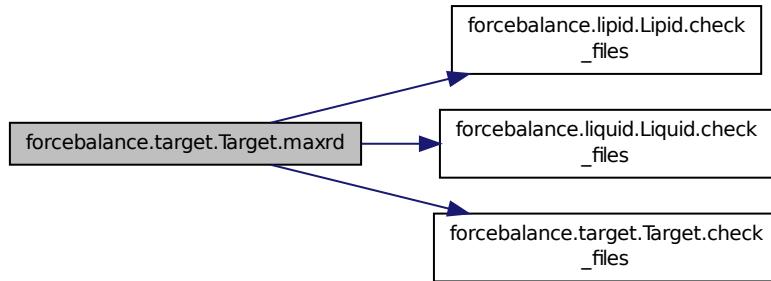
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

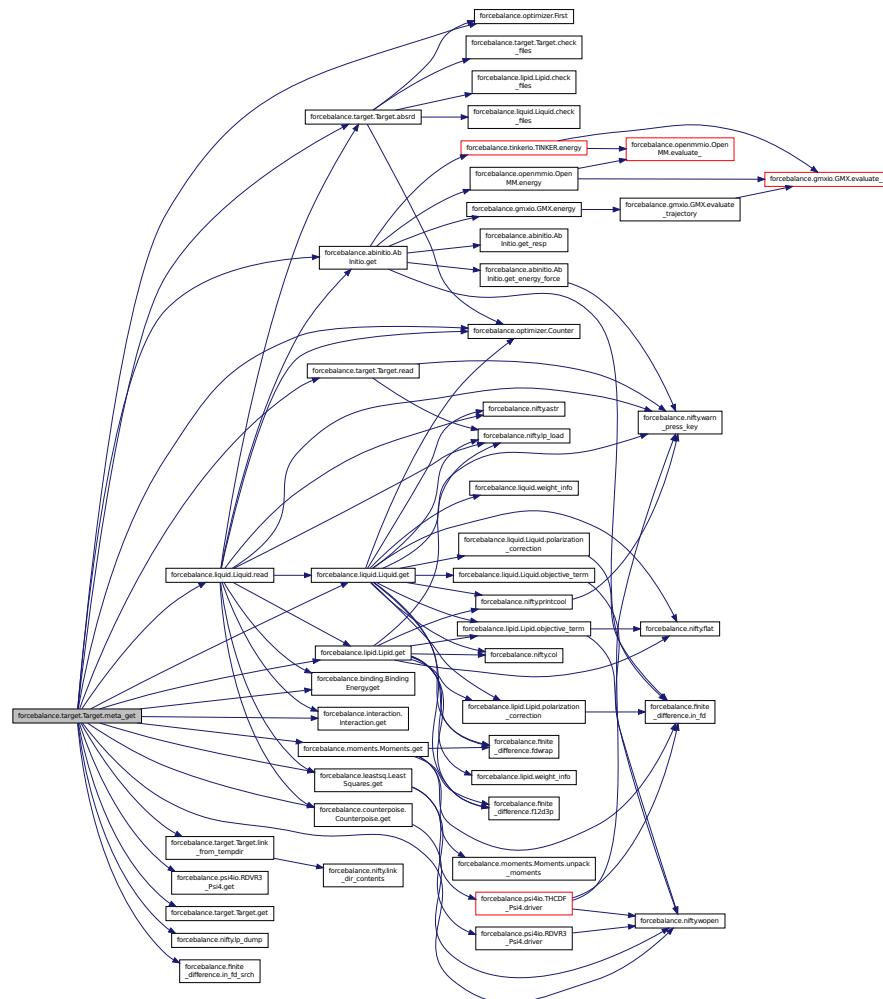


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

Here is the call graph for this function:

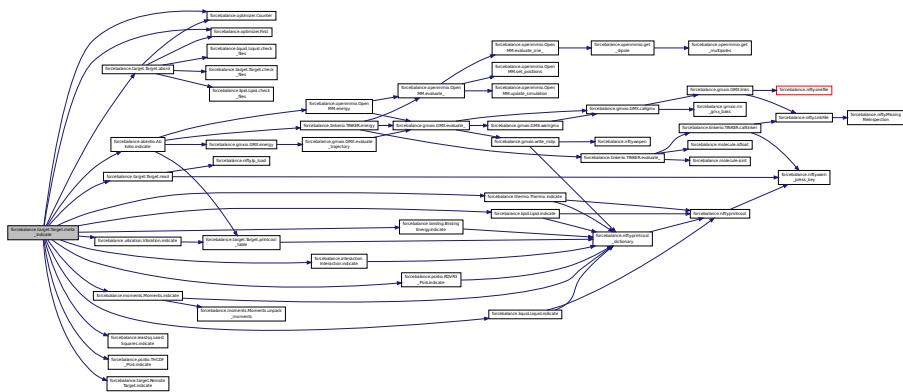


**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.

Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

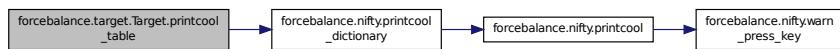
The input should be something like:

## Parameters

<i>data</i>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<i>headings</i>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<i>banner</i>	Optional heading line, which will be printed at the top in the title.
<i>footnote</i>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

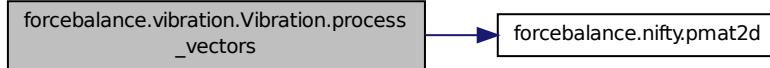
Here is the call graph for this function:



**def forcebalance.vibration.Vibration.process\_vectors ( self, vecs, verbose = False, check = False )**  
[inherited] Return a set of normal and mass-weighted eigenvectors such that their outer product is the identity.

Definition at line 127 of file vibration.py.

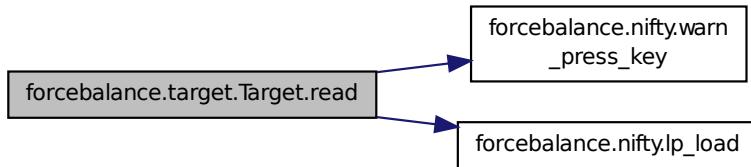
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.vibration.Vibration.read\_reference\_data ( self ) [inherited]** Read the reference vibrational data from a file.

Definition at line 69 of file vibration.py.

**def forcebalance.target.Target.refresh\_temp\_directory ( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

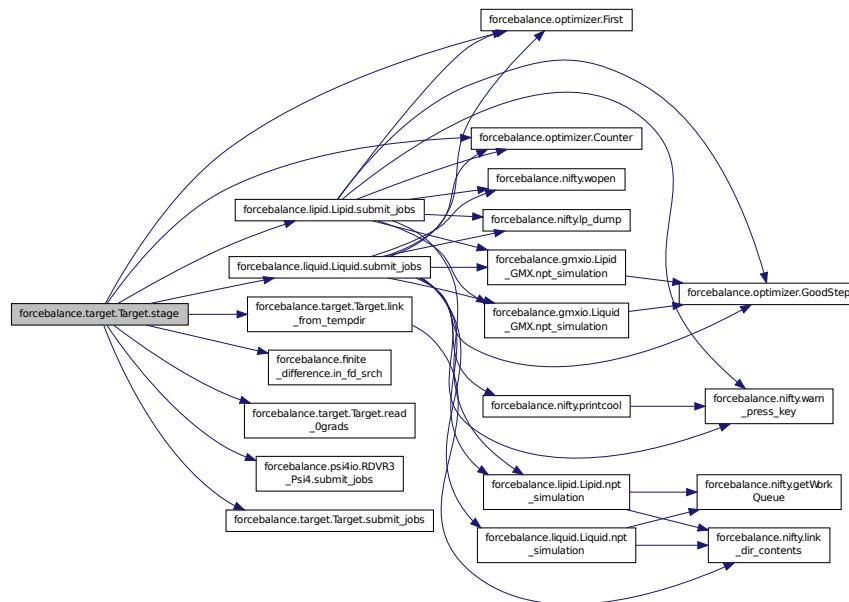
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src\_key, dest\_key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None )  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



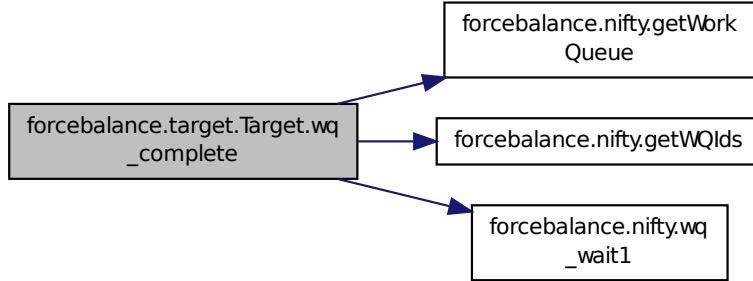
```
def forcebalance.target.Target.submit_jobs( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.vibration.Vibration.vibration_driver( self ) [inherited] Definition at line 118 of file vibration.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.75.4 Member Data Documentation

**forcebalance.vibration.Vibration.c2r [inherited]** Definition at line 178 of file vibration.py.

**forcebalance.vibration.Vibration.calc\_eigvals [inherited]** Definition at line 198 of file vibration.py.

**forcebalance.vibration.Vibration.engine [inherited]** Read in the reference data.

Build keyword dictionaries to pass to engine. Create engine object.

Definition at line 63 of file vibration.py.

**forcebalance.gmxio.Vibration\_GMX.engine** Default file names for coordinates and key file.

Definition at line 1497 of file gmxio.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.vibration.Vibration.na [inherited]** Number of atoms.

Definition at line 71 of file vibration.py.

**forcebalance.vibration.Vibration.objective [inherited]** Definition at line 199 of file vibration.py.

**forcebalance.vibration.Vibration.overlaps [inherited]** Definition at line 183 of file vibration.py.

**forcebalance.target.Target.pgrad** [inherited] Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd** [inherited] Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate** [inherited] Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [inherited] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.vibration.Vibration.reassign** [inherited] Definition at line 165 of file vibration.py.

**forcebalance.vibration.Vibration.ref\_eigvals** [inherited] Definition at line 72 of file vibration.py.

**forcebalance.vibration.Vibration.ref\_eigvecs** [inherited] Definition at line 73 of file vibration.py.

**forcebalance.vibration.Vibration.ref\_eigvecs\_nrm\_mw** [inherited] Definition at line 155 of file vibration.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number. The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.vibration.Vibration.vfnm** [inherited] The vdata.txt file that contains the vibrations.

Definition at line 56 of file vibration.py.

**forcebalance.target.Target.write\_indicate** [**inherited**] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

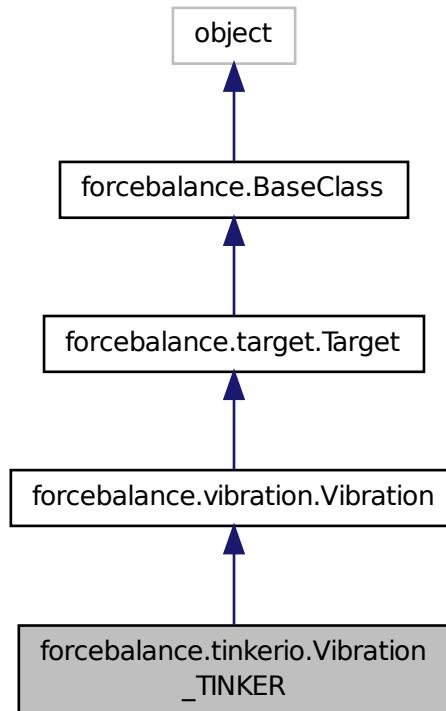
The documentation for this class was generated from the following file:

- [gmxio.py](#)

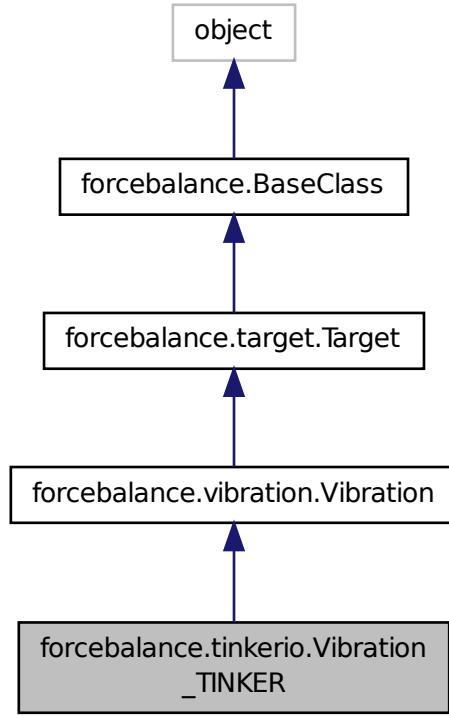
## 8.76 forcebalance.tinkerio.Vibration\_TINKER Class Reference

Vibrational frequency matching using [TINKER](#).

Inheritance diagram for forcebalance.tinkerio.Vibration\_TINKER:



Collaboration diagram for forcebalance.tinkerio.Vibration\_TINKER:



## Public Member Functions

- def `__init__`
- def `read_reference_data`

*Read the reference vibrational data from a file.*
- def `indicate`

*Print qualitative indicator.*
- def `vibration_driver`
- def `process_vectors`

*Return a set of normal and mass-weighted eigenvectors such that their outer product is the identity.*
- def `get`

*Evaluate objective function.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- def `read_Ograds`

*Read a file from the target directory containing names of parameters that don't contribute to the gradient.*
- def `write_Ograds`

*Write a file to the target directory containing names of parameters that don't contribute to the gradient.*
- def `get_G`

- def `get_H`  
*Computes the objective function contribution and its gradient.*
- def `link_from_tempdir`  
*Computes the objective function contribution and its gradient / Hessian.*
- def `refresh_temp_directory`  
*Back up the temporary directory if desired, delete it and then create a new one.*
- def `check_files`  
*Check this directory for the presence of readable files when the 'read' option is set.*
- def `read`  
*Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.*
- def `absrd`  
*Supply the correct directory specified by user's "read" option.*
- def `maxrd`  
*Supply the latest existing temp-directory containing valid data.*
- def `meta_indicate`  
*Wrap around the indicate function, so it can print to screen and also to a file.*
- def `meta_get`  
*Wrapper around the get function.*
- def `submit_jobs`
- def `stage`  
*Stages the directory for the target, and then launches Work Queue processes if any.*
- def `wq_complete`  
*This method determines whether the Work Queue tasks for the current target have completed.*
- def `printcool_table`  
*Print target information in an organized table format.*
- def `__setattr__`
- def `set_option`

## Public Attributes

- `engine_`  
*Default file names for coordinates and key file.*
- `vfnm`  
*The vdata.txt file that contains the vibrations.*
- `engine`  
*Read in the reference data.*
- `na`  
*Number of atoms.*
- `ref_eigvals`
- `ref_eigvecs`
- `ref_eigvecs_nrm_mw`
- `reassign`
- `c2r`
- `overlaps`
- `calc_eigvals`
- `objective`
- `rd`  
*Root directory of the whole project.*

- **pgrad**  
*Iteration where we turn on zero-gradient skipping.*
- **tempbase**  
*Relative directory of target.*
- **tempdir**  
 $\text{self.tempdir} = \text{os.path.join('temp', self.name)}$  *The directory in which the simulation is running - this can be updated.*
- **FF**  
*Need the forcefield (here for now)*
- **xct**  
*Counts how often the objective function was computed.*
- **gct**  
*Counts how often the gradient was computed.*
- **hct**  
*Counts how often the Hessian was computed.*
- **read\_indicate**  
*Whether to read indicate.log from file when restarting an aborted run.*
- **write\_indicate**  
*Whether to write indicate.log at every iteration (true for all but remote.)*
- **read\_objective**  
*Whether to read objective.p from file when restarting an aborted run.*
- **write\_objective**  
*Whether to write objective.p at every iteration (true for all but remote.)*
- **verbose\_options**
- **PrintOptionDict**

### 8.76.1 Detailed Description

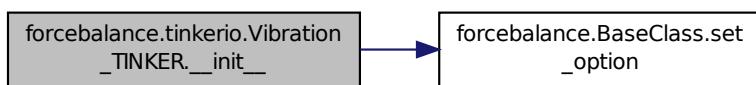
Vibrational frequency matching using [TINKER](#).

Definition at line 1108 of file tinkerio.py.

### 8.76.2 Constructor & Destructor Documentation

**def forcebalance.tinkerio.Vibration\_TINKER.\_\_init\_\_ ( *self*, *options*, *tgt\_opts*, *forcefield* )** Definition at line 1109 of file tinkerio.py.

Here is the call graph for this function:



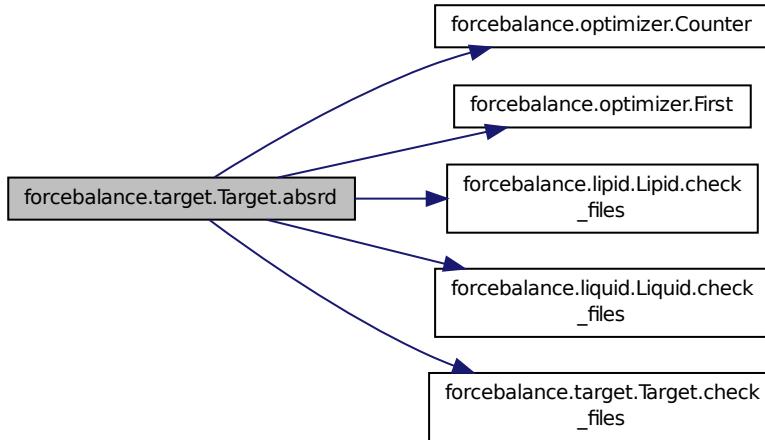
### 8.76.3 Member Function Documentation

**def forcebalance.BaseClass.\_\_setattr\_\_ ( *self*, *key*, *value* ) [inherited]** Definition at line 28 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.absrd ( self, inum = None ) [inherited] Supply the correct directory specified by user's "read" option.
```

Definition at line 393 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.check_files ( self, there ) [inherited] Check this directory for the presence of readable files when the 'read' option is set.
```

Definition at line 364 of file target.py.

```
def forcebalance.vibration.Vibration.get ( self, mvals, AGrad=False, AHess=False ) [inherited]  
Evaluate objective function.
```

Definition at line 151 of file vibration.py.

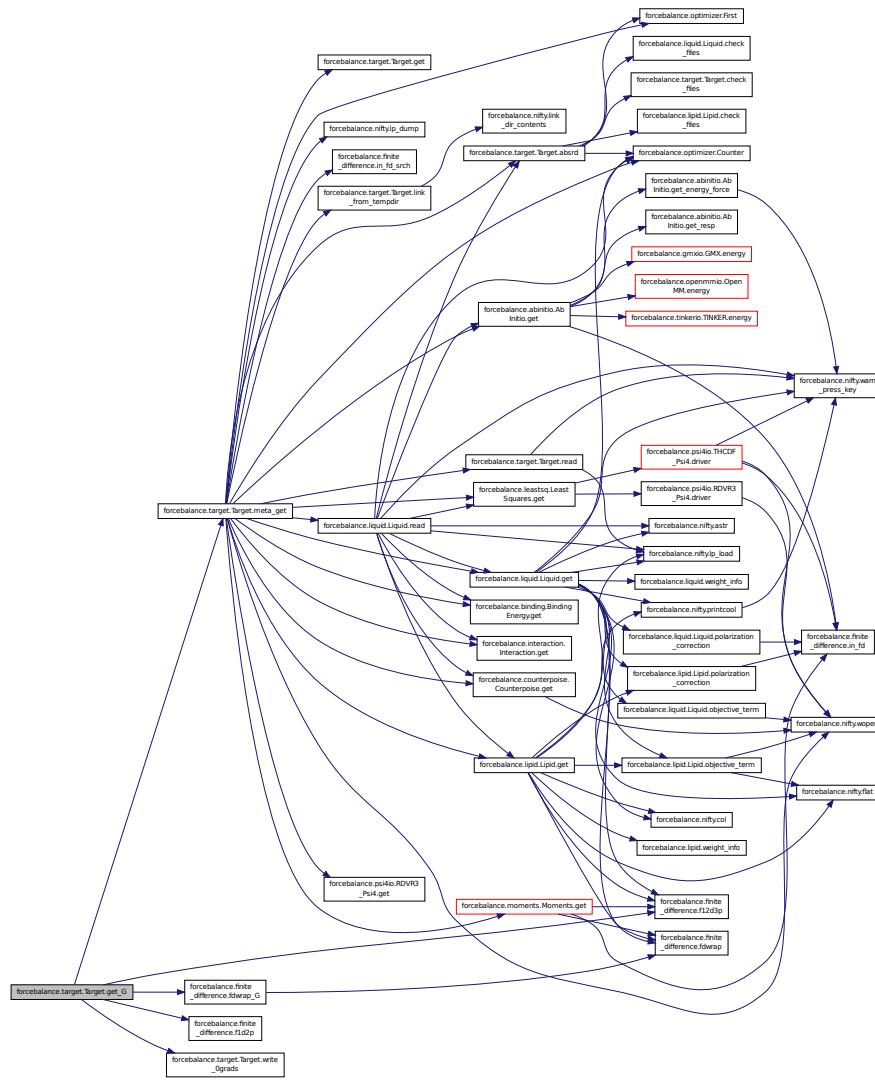
```
def forcebalance.target.Target.get_G ( self, mvals = None ) [inherited] Computes the objective function contribution and its gradient.
```

First the low-level 'get' method is called with the analytic gradient switch turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on. Alternately we can compute the gradient elements and diagonal Hessian elements at the same time using central difference if 'fdhessdiag' is turned on.

In this function we also record which parameters cause a nonzero change in the objective function contribution. Parameters which do not change the objective function will not be differentiated in subsequent calculations. This is recorded in a text file in the targets directory.

Definition at line 272 of file target.py.

Here is the call graph for this function:



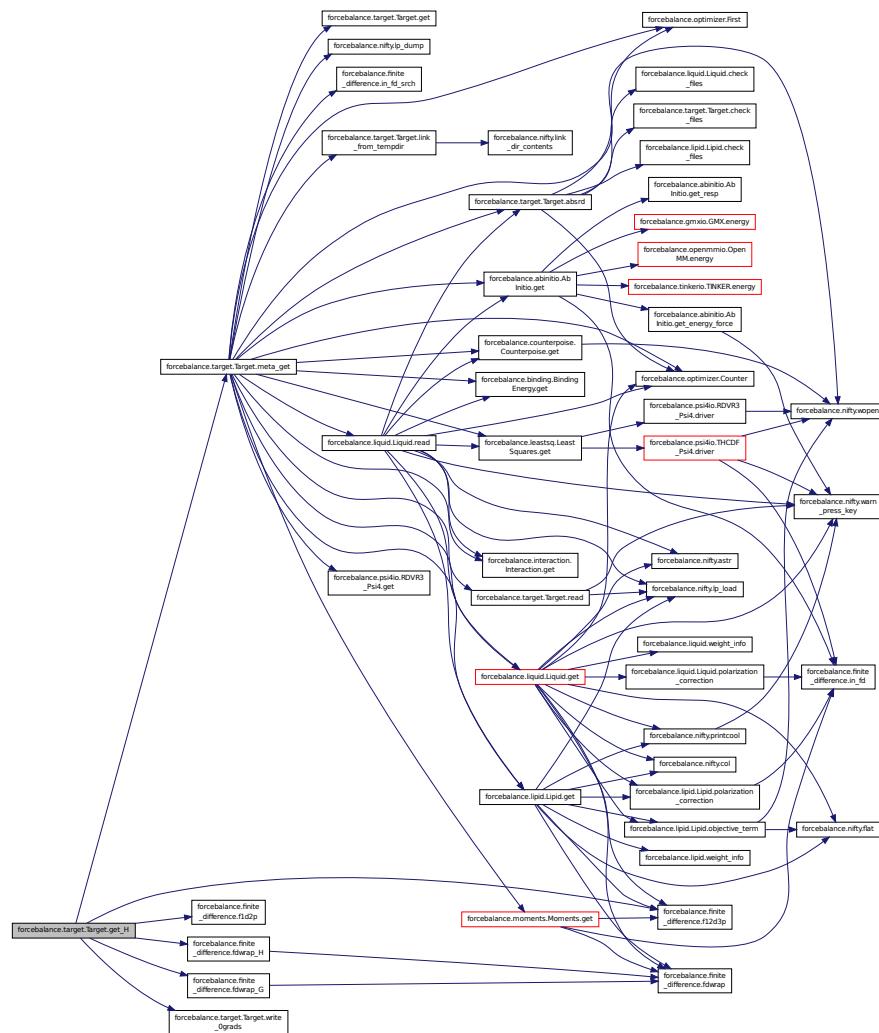
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

First the low-level 'get' method is called with the analytic gradient and Hessian both turned on. Then we loop through the fd1\_pids and compute the corresponding elements of the gradient by finite difference, if the 'fdgrad' switch is turned on.

This is followed by looping through the fd2.pids and computing the corresponding Hessian elements by finite difference. Forward finite difference is used throughout for the sake of speed.

Definition at line 296 of file target.py.

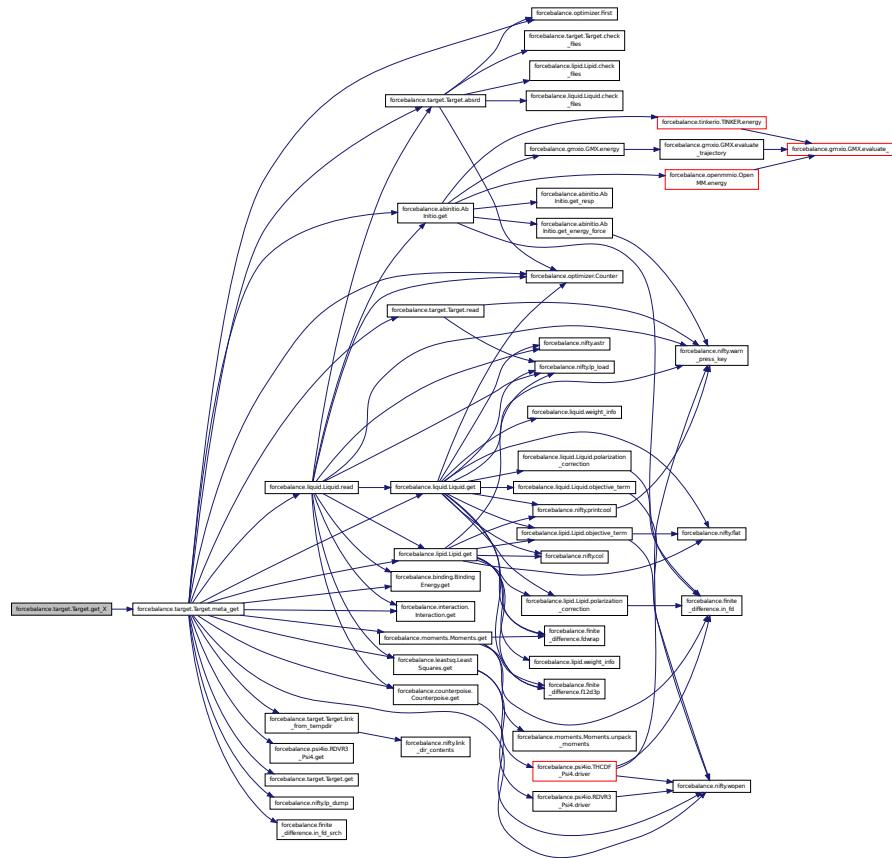
Here is the call graph for this function:



**def forcebalance.target.Target.get\_X ( self, mvals = None ) [inherited]** Computes the objective function contribution without any parametric derivatives.

Definition at line 184 of file target.py.

Here is the call graph for this function:



**def forcebalance.vibration.Vibration.indicate ( self ) [inherited]** Print qualitative indicator.

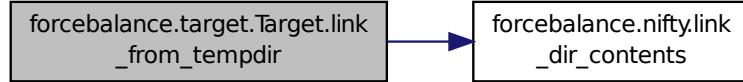
Definition at line 110 of file vibration.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 315 of file target.py.

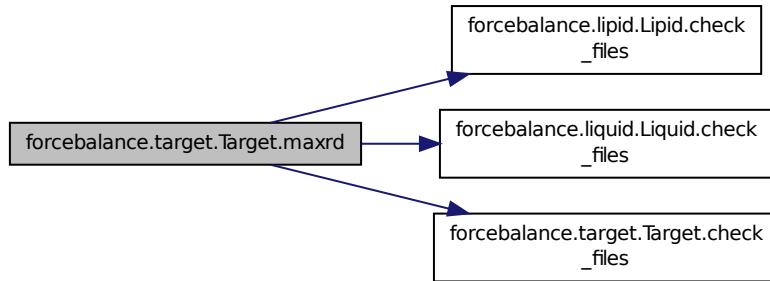
Here is the call graph for this function:



**def forcebalance.target.Target.maxrd ( self ) [inherited]** Supply the latest existing temp-directory containing valid data.

Definition at line 447 of file target.py.

Here is the call graph for this function:

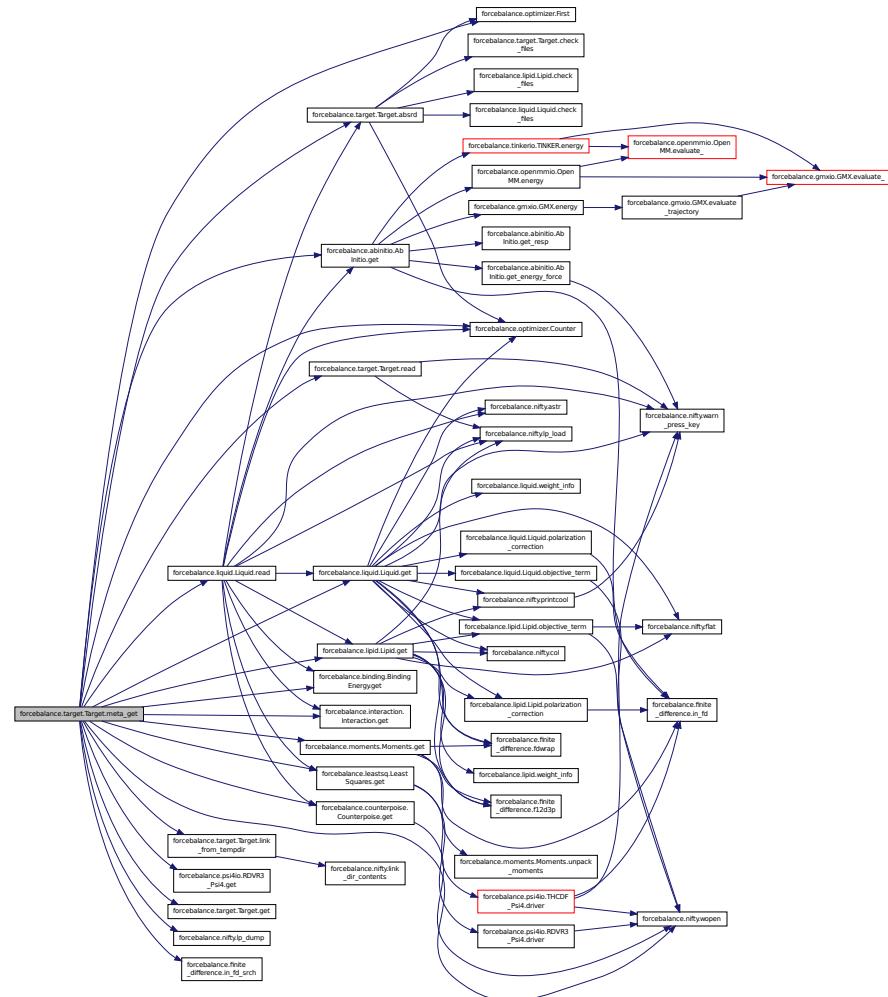


**def forcebalance.target.Target.meta\_get ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Wrapper around the get function.

Create the directory for the target, and then calls 'get'. If we are reading existing data, go into the appropriate read directory and call [read\(\)](#) instead. The 'get' method should not worry about the directory that it's running in.

Definition at line 511 of file target.py.

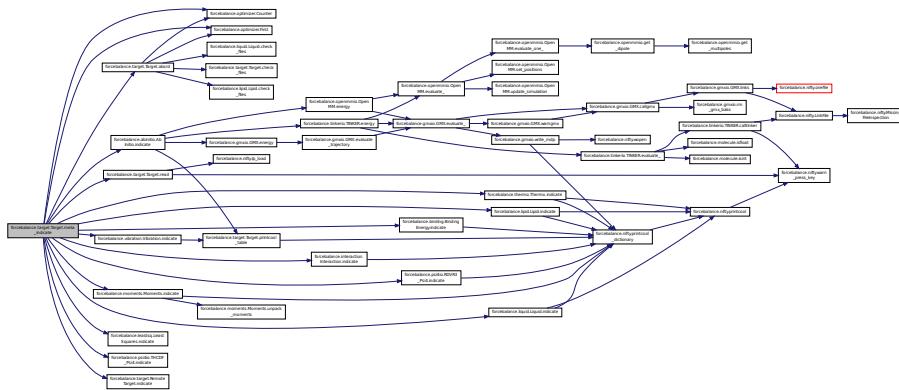
Here is the call graph for this function:



**def forcebalance.target.Target.meta\_indicate ( self ) [inherited]** Wrap around the indicate function, so it can print to screen and also to a file.

If reading from checkpoint file, don't call the indicate() function, instead just print the file contents to the screen.  
Definition at line 469 of file target.py.

Here is the call graph for this function:



```
def forcebalance.target.Target.printcool_table( self, data = OrderedDict([]), headings = [], banner = None, footnote = None, color = 0 ) [inherited] Print target information in an organized table format.
```

Implemented 6/30 because multiple targets are already printing out tabulated information in very similar ways. This method is a simple wrapper around printcool\_dictionary.

The input should be something like:

#### Parameters

<b>data</b>	Column contents in the form of an OrderedDict, with string keys and list vals. The key is printed in the leftmost column and the vals are printed in the other columns. If non-strings are passed, they will be converted to strings (not recommended).
<b>headings</b>	Column headings in the form of a list. It must be equal to the number to the list length for each of the "vals" in OrderedDict, plus one. Use "\n" characters to specify long column names that may take up more than one line.
<b>banner</b>	Optional heading line, which will be printed at the top in the title.
<b>footnote</b>	Optional footnote line, which will be printed at the bottom.

Definition at line 638 of file target.py.

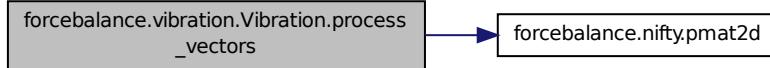
Here is the call graph for this function:



```
def forcebalance.vibration.Vibration.process_vectors( self, vecs, verbose = False, check = False ) [inherited] Return a set of normal and mass-weighted eigenvectors such that their outer product is the identity.
```

Definition at line 127 of file vibration.py.

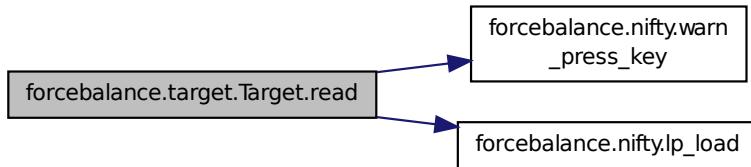
Here is the call graph for this function:



**def forcebalance.target.Target.read ( self, mvals, AGrad = False, AHess = False ) [inherited]**  
Read data from disk for the initial optimization step if the user has provided the directory to the "read" option.

Definition at line 379 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.read\_0grads ( self ) [inherited]** Read a file from the target directory containing names of parameters that don't contribute to the gradient.

Note that we are checking the derivatives of the objective function, and not the derivatives of the quantities that go into building the objective function. However, it is the quantities that we actually differentiate. Since there is a simple chain rule relationship, the parameters that do/don't contribute to the objective function/quantities are the same.

However, property gradients do contribute to objective function Hessian elements, so we cannot use the same mechanism for excluding the calculation of property Hessians. This is mostly fine since we rarely if ever calculate an explicit property Hessian.

Definition at line 207 of file target.py.

**def forcebalance.vibration.Vibration.read\_reference\_data ( self ) [inherited]** Read the reference vibrational data from a file.

Definition at line 69 of file vibration.py.

**def forcebalance.target.Target.refresh\_temp\_directory ( self ) [inherited]** Back up the temporary directory if desired, delete it and then create a new one.

Definition at line 321 of file target.py.

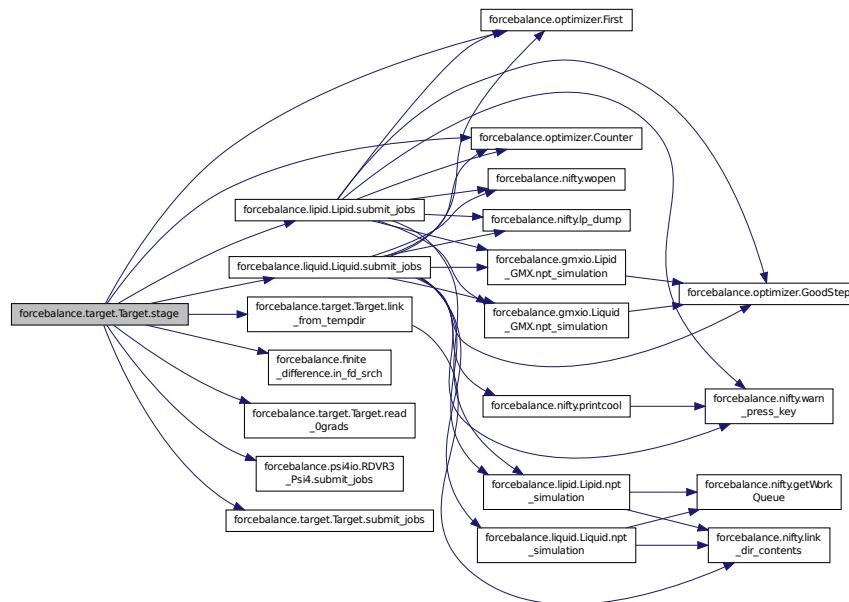
**def forcebalance.BaseClass.set\_option ( self, in\_dict, src\_key, dest\_key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 42 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None )  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

The 'get' method should not worry about the directory that it's running in.

Definition at line 565 of file target.py.

Here is the call graph for this function:



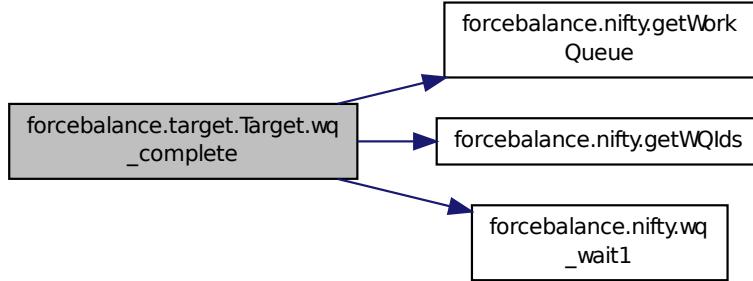
```
def forcebalance.target.Target.submit_jobs( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 555 of file target.py.
```

```
def forcebalance.vibration.Vibration.vibration_driver( self ) [inherited] Definition at line 118 of file vibration.py.
```

```
def forcebalance.target.Target.wq_complete( self ) [inherited] This method determines whether the Work Queue tasks for the current target have completed.
```

Definition at line 602 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.write\_0grads ( self, Ans ) [inherited]** Write a file to the target directory containing names of parameters that don't contribute to the gradient.

Definition at line 225 of file target.py.

#### 8.76.4 Member Data Documentation

**forcebalance.vibration.Vibration.c2r [inherited]** Definition at line 178 of file vibration.py.

**forcebalance.vibration.Vibration.calc\_eigvals [inherited]** Definition at line 198 of file vibration.py.

**forcebalance.vibration.Vibration.engine [inherited]** Read in the reference data.

Build keyword dictionaries to pass to engine. Create engine object.

Definition at line 63 of file vibration.py.

**forcebalance.tinkerio.Vibration.TINKER.engine** Default file names for coordinates and key file.

Definition at line 1113 of file tinkerio.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 160 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 164 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 166 of file target.py.

**forcebalance.vibration.Vibration.na [inherited]** Number of atoms.

Definition at line 71 of file vibration.py.

**forcebalance.vibration.Vibration.objective [inherited]** Definition at line 199 of file vibration.py.

**forcebalance.vibration.Vibration.overlaps [inherited]** Definition at line 183 of file vibration.py.

**forcebalance.target.Target.pgrad** [inherited] Iteration where we turn on zero-gradient skipping.

Dictionary of whether to call the derivatives.

Definition at line 127 of file target.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 44 of file \_\_init\_\_.py.

**forcebalance.target.Target.rd** [inherited] Root directory of the whole project.

Submit jobs to the Work Queue.

Name of the target Type of target Relative weight of the target Switch for finite difference gradients Switch for finite difference Hessians Switch for FD gradients + Hessian diagonals How many seconds to sleep (if any) Parameter types that trigger FD gradient elements Parameter types that trigger FD Hessian elements Finite difference step size Whether to make backup files Directory to read data from.

Definition at line 123 of file target.py.

**forcebalance.target.Target.read\_indicate** [inherited] Whether to read indicate.log from file when restarting an aborted run.

Definition at line 168 of file target.py.

**forcebalance.target.Target.read\_objective** [inherited] Whether to read objective.p from file when restarting an aborted run.

Definition at line 172 of file target.py.

**forcebalance.vibration.Vibration.reassign** [inherited] Definition at line 165 of file vibration.py.

**forcebalance.vibration.Vibration.ref\_eigvals** [inherited] Definition at line 72 of file vibration.py.

**forcebalance.vibration.Vibration.ref\_eigvecs** [inherited] Definition at line 73 of file vibration.py.

**forcebalance.vibration.Vibration.ref\_eigvecs\_nrm\_mw** [inherited] Definition at line 155 of file vibration.py.

**forcebalance.target.Target.rundir** [inherited] self.tempdir = os.path.join('temp',self.name) The directory in which the simulation is running - this can be updated.

Directory of the current iteration; if not None, then the simulation runs under temp/target.name/iteration\_number. The 'customdir' is customizable and can go below anything.

Not expecting more than ten thousand iterations Go into the directory where [get\(\)](#) will be executed. Write mathematical parameters to file; will be used to checkpoint calculation. Read in file that specifies which derivatives may be skipped.

Definition at line 158 of file target.py.

**forcebalance.target.Target.tempbase** [inherited] Relative directory of target.

Temporary (working) directory; it is temp/(target.name) Used for storing temporary variables that don't change through the course of the optimization

Definition at line 152 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Definition at line 155 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 40 of file \_\_init\_\_.py.

**forcebalance.vibration.Vibration.vfnm** [inherited] The vdata.txt file that contains the vibrations.

Definition at line 56 of file vibration.py.

**forcebalance.target.Target.write\_indicate** [**inherited**] Whether to write indicate.log at every iteration (true for all but remote.)

Definition at line 170 of file target.py.

**forcebalance.target.Target.write\_objective** [**inherited**] Whether to write objective.p at every iteration (true for all but remote.)

Definition at line 174 of file target.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 162 of file target.py.

The documentation for this class was generated from the following file:

- [tinkerio.py](#)

## 9 File Documentation

### 9.1 \_\_init\_\_.py File Reference

#### Classes

- class [forcebalance.BaseClass](#)

*Provides some nifty functions that are common to all ForceBalance classes.*

- class [forcebalance.BaseReader](#)

*The 'reader' class.*

#### Namespaces

- [forcebalance](#)

#### Variables

- tuple [forcebalance.\\_\\_version\\_\\_](#) = pkg\_resources.get\_distribution("forcebalance")

### 9.2 abinitio.py File Reference

#### Classes

- class [forcebalance.abinitio.AblInitio](#)

*Subclass of Target for fitting force fields to ab initio data.*

#### Namespaces

- [forcebalance.abinitio](#)

*Ab-initio fitting module (energies, forces, resp).*

#### Functions

- def [forcebalance.abinitio.weighted\\_variance](#)

*A more generalized version of build\_objective which is callable for derivatives, but the covariance is not there anymore.*

- def [forcebalance.abinitio.weighted\\_variance2](#)

*A bit of a hack, since we have to subtract out two mean quantities to get Hessian elements.*

- def [forcebalance.abinitio.build\\_objective](#)

*This function builds an objective function (number) from the complicated polytensor and covariance matrices.*

## Variables

- tuple `forcebalance.abinitio.logger` = getLogger(\_\_name\_\_)

## 9.3 abinitio\_internal.py File Reference

### Classes

- class `forcebalance.abinitio_internal.AblInitio_Internal`

*Subclass of Target for force and energy matching using an internal implementation.*

### Namespaces

- `forcebalance.abinitio_internal`  
*Internal implementation of energy matching (for TIP3P water only)*

## 9.4 amberio.py File Reference

### Classes

- class `forcebalance.amberio.Mol2_Reader`  
*Finite state machine for parsing Mol2 force field file.*
- class `forcebalance.amberio.FrcMod_Reader`  
*Finite state machine for parsing FrcMod force field file.*
- class `forcebalance.amberio.AblInitio_AMBER`  
*Subclass of Target for force and energy matching using AMBER.*

### Namespaces

- `forcebalance.amberio`  
*AMBER force field input/output.*

### Functions

- def `forcebalance.amberio.is_mol2_atom`

## Variables

- tuple `forcebalance.amberio.logger` = getLogger(\_\_name\_\_)
- dictionary `forcebalance.amberio.mol2_pdct` = {'COUL': {'Atom': [1], 8: ''}}
- dictionary `forcebalance.amberio.frcmod_pdct`

## 9.5 api.dox File Reference

## 9.6 binding.py File Reference

### Classes

- class `forcebalance.binding.BindingEnergy`  
*Improved subclass of Target for fitting force fields to binding energies.*

### Namespaces

- `forcebalance.binding`  
*Binding energy fitting module.*

## Functions

- def `forcebalance.binding.parse_interactions`

*Parse through the interactions input file.*

## Variables

- tuple `forcebalance.binding.logger` = getLogger(\_\_name\_\_)

## 9.7 chemistry.py File Reference

### Namespaces

- `forcebalance.chemistry`

### Functions

- def `forcebalance.chemistry.LookupByMass`
- def `forcebalance.chemistry.BondStrengthByLength`

### Variables

- tuple `forcebalance.chemistry.BondEnergies` = defaultdict(lambda:defaultdict(dict))
- list `forcebalance.chemistry.Radii`

*Covalent radii from Cordero et al.*
- tuple `forcebalance.chemistry.PeriodicTable`
- list `forcebalance.chemistry.Elements`
- list `forcebalance.chemistry.BondChars` = [ '-', '=', '3' ]
- string `forcebalance.chemistry.data_from_web`
- tuple `forcebalance.chemistry.line` = line.expandtabs()
- tuple `forcebalance.chemistry.BE` = float(line.split()[1])
- tuple `forcebalance.chemistry.L` = float(line.split()[2])
- tuple `forcebalance.chemistry.atoms` = re.split('[-=3]', line.split()[0])
- list `forcebalance.chemistry.A` = atoms[0]
- list `forcebalance.chemistry.B` = atoms[1]
- tuple `forcebalance.chemistry.bo` = BondChars.index(re.findall('[-=3]', line.split()[0])[0])

## 9.8 contact.py File Reference

### Namespaces

- `forcebalance.contact`

### Functions

- def `forcebalance.contact.atom_distances`

*For each frame in xyzlist, compute the (euclidean) distance between pairs of atoms whos indices are given in contacts.*

- def `forcebalance.contact.residue_distances`

*For each frame in xyzlist, and for each pair of residues in the array contact, compute the distance between the closest pair of atoms such that one of them belongs to each residue.*

## 9.9 counterpoise.py File Reference

### Classes

- class `forcebalance.counterpoise.Counterpoise`

*Target subclass for matching the counterpoise correction.*

### Namespaces

- `forcebalance.counterpoise`

*Match an empirical potential to the counterpoise correction for basis set superposition error (BSSE).*

### Variables

- tuple `forcebalance.counterpoise.logger = getLogger(__name__)`

## 9.10 custom.io.py File Reference

### Classes

- class `forcebalance.custom_io.Gen_Reader`

*Finite state machine for parsing custom GROMACS force field files.*

### Namespaces

- `forcebalance.custom_io`

*Custom force field parser.*

### Variables

- list `forcebalance.custom_io.cptypes = [None, 'CPGAUSS', 'CPEXPG', 'CPGEXP']`

*Types of counterpoise correction.*

- list `forcebalance.custom_io.ndtypes = [None]`

*Types of NDDO correction.*

- dictionary `forcebalance.custom.io.fdict`

*Section -> Interaction type dictionary.*

- dictionary `forcebalance.custom.io.pdict`

*Interaction type -> Parameter Dictionary.*

## 9.11 engine.py File Reference

### Classes

- class `forcebalance.engine.Engine`

*Base class for all engines.*

### Namespaces

- `forcebalance.engine`

### Variables

- tuple `forcebalance.engine.logger = getLogger(__name__)`

## 9.12 finite\_difference.py File Reference

### Namespaces

- `forcebalance.finite_difference`

### Functions

- def `forcebalance.finite_difference.f1d2p`  
*A two-point finite difference stencil.*
- def `forcebalance.finite_difference.f1d5p`  
*A highly accurate five-point finite difference stencil for computing derivatives of a function.*
- def `forcebalance.finite_difference.f1d7p`  
*A highly accurate seven-point finite difference stencil for computing derivatives of a function.*
- def `forcebalance.finite_difference.f12d7p`
- def `forcebalance.finite_difference.f12d3p`  
*A three-point finite difference stencil.*
- def `forcebalance.finite_difference.f2var`  
*A finite difference stencil for a function of two variables.*
- def `forcebalance.finite_difference.in_fd`  
*Invoking this function from anywhere will tell us whether we're being called by a finite-difference function.*
- def `forcebalance.finite_difference.in_fd_srcb`  
*Invoking this function from anywhere will tell us whether we're being called by a finite-difference function.*
- def `forcebalance.finite_difference.fdwrap`  
*A function wrapper for finite difference designed for differentiating 'get'-type functions.*
- def `forcebalance.finite_difference.fdwrap_G`  
*A driver to fdwrap for gradients (see documentation for fdwrap) Inputs: tgt = The Target containing the objective function that we want to differentiate mvals0 = The 'central' values of the mathematical parameters - i.e.*
- def `forcebalance.finite_difference.fdwrap_H`  
*A driver to fdwrap for Hessians (see documentation for fdwrap) Inputs: tgt = The Target containing the objective function that we want to differentiate mvals0 = The 'central' values of the mathematical parameters - i.e.*

### Variables

- tuple `forcebalance.finite_difference.logger` = getLogger(\_\_name\_\_)

## 9.13 forcefield.py File Reference

### Classes

- class `forcebalance.forcefield.BackedUpDict`
- class `forcebalance.forcefield.FF`

*Force field class.*

### Namespaces

- `forcebalance.forcefield`

*Force field module.*

## Functions

- def `forcebalance.forcefield.determine_fftype`  
*Determine the type of a force field file.*
- def `forcebalance.forcefield.rs_override`  
*This function takes in a dictionary (`rsfactors`) and a string (`termtype`).*

## Variables

- tuple `forcebalance.forcefield.logger` = getLogger(`_name_`)
- dictionary `forcebalance.forcefield.FF_Extensions`
- dictionary `forcebalance.forcefield.FF_IOModules`

## 9.14 gmxio.py File Reference

### Classes

- class `forcebalance.gmxio.ITP_Reader`  
*Finite state machine for parsing GROMACS force field files.*
- class `forcebalance.gmxio.GMX`  
*Derived from Engine object for carrying out general purpose GROMACS calculations.*
- class `forcebalance.gmxio.Liquid_GMX`
- class `forcebalance.gmxio.Lipid_GMX`
- class `forcebalance.gmxio.AbInitio_GMX`  
*Subclass of AbInitio for force and energy matching using GROMACS.*
- class `forcebalance.gmxio.BindingEnergy_GMX`  
*Binding energy matching using Gromacs.*
- class `forcebalance.gmxio.Interaction_GMX`  
*Interaction energy matching using GROMACS.*
- class `forcebalance.gmxio.Moments_GMX`  
*Multipole moment matching using GROMACS.*
- class `forcebalance.gmxio.Vibration_GMX`  
*Vibrational frequency matching using GROMACS.*
- class `forcebalance.gmxio.Thermo_GMX`  
*Thermodynamical property matching using GROMACS.*

### Namespaces

- `forcebalance.gmxio`  
*GROMACS input/output.*

## Functions

- def `forcebalance.gmxio.write_mdp`  
*Create or edit a Gromacs MDP file.*
- def `forcebalance.gmxio.write_ndx`  
*Create or edit a Gromacs ndx file.*
- def `forcebalance.gmxio.parse_atomtype_line`  
*Parses the 'atomtype' line.*
- def `forcebalance.gmxio.rm_gmx_baks`

## Variables

- tuple `forcebalance.gmxio.logger` = getLogger(`_name_`)
- list `forcebalance.gmxio.nftypes` = [None, 'VDW', 'VDW\_BHAM']  
*VdW interaction function types.*
- list `forcebalance.gmxio.pftypes` = [None, 'VPAIR', 'VPAIR\_BHAM']  
*Pairwise interaction function types.*
- list `forcebalance.gmxio.bftypes` = [None, 'BONDS', 'G96BONDS', 'MORSE']  
*Bonded interaction function types.*
- list `forcebalance.gmxio.aftypes`  
*Angle interaction function types.*
- list `forcebalance.gmxio.dftypes` = [None, 'PDIHS', 'IDIHS', 'RBDIHS', 'PIMPDIHS', 'FOURDIHS', None, None, 'TABDIHS', 'PDIHMULS']  
*Dihedral interaction function types.*
- dictionary `forcebalance.gmxio.fdict`  
*Section -> Interaction type dictionary.*
- dictionary `forcebalance.gmxio.pdict`  
*Interaction type -> Parameter Dictionary.*

## 9.15 interaction.py File Reference

### Classes

- class `forcebalance.interaction.Interaction`  
*Subclass of Target for fitting force fields to interaction energies.*

### Namespaces

- `forcebalance.interaction`  
*Interaction energy fitting module.*

### Variables

- tuple `forcebalance.interaction.logger` = getLogger(`_name_`)

## 9.16 leastsq.py File Reference

### Classes

- class `forcebalance.leastsq.LeastSquares`  
*Subclass of Target for general least squares fitting.*

### Namespaces

- `forcebalance.leastsq`
- `forcebalance.abinitio`  
*Ab-initio fitting module (energies, forces, resp).*

### Functions

- def `forcebalance.leastsq.CheckBasis`
- def `forcebalance.leastsq.LastMvals`

## Variables

- tuple `forcebalance.leastsq.logger` = getLogger(`_name_`)
- `forcebalance.leastsq.CHECK_BASIS` = False
- `forcebalance.leastsq.LAST_MVALS` = None

## 9.17 lipid.py File Reference

### Classes

- class `forcebalance.lipid.Lipid`

*Subclass of Target for lipid property matching.*

### Namespaces

- `forcebalance.lipid`

*Matching of lipid bulk properties.*

### Functions

- def `forcebalance.lipid.weight_info`

## Variables

- tuple `forcebalance.lipid.logger` = getLogger(`_name_`)

## 9.18 liquid.py File Reference

### Classes

- class `forcebalance.liquid.Liquid`

*Subclass of Target for liquid property matching.*

### Namespaces

- `forcebalance.liquid`

*Matching of liquid bulk properties.*

### Functions

- def `forcebalance.liquid.weight_info`

## Variables

- tuple `forcebalance.liquid.logger` = getLogger(`_name_`)

## 9.19 Mol2.py File Reference

### Classes

- class `forcebalance.Mol2.mol2_atom`

*This is to manage mol2 atomic lines on the form: 1 C1 5.4790 42.2880 49.5910 C.ar 1 <1> 0.0424.*

- class `forcebalance.Mol2.mol2_bond`

*This is to manage mol2 bond lines on the form: 1 1 2 ar.*

- class `forcebalance.Mol2.mol2`

*This is to manage one mol2 series of lines on the form:*

- class `forcebalance.Mol2.mol2_set`

## Namespaces

- `forcebalance.Mol2`

## Variables

- tuple `forcebalance.Mol2.data = mol2_set(sys.argv[1], subset=["RNase.xray.inh8.1QHC"])`

## 9.20 mol2io.py File Reference

### Classes

- class `forcebalance.mol2io.Mol2_Reader`

*Finite state machine for parsing Mol2 force field file.*

## Namespaces

- `forcebalance.mol2io`  
*Mol2 I/O.*

## Variables

- dictionary `forcebalance.mol2io.mol2_pdct = {'COUL': {'Atom': [1], 6: ''}}`

## 9.21 molecule.py File Reference

### Classes

- class `forcebalance.molecule.MyG`
- class `forcebalance.molecule.MolfileTimestep`  
*Wrapper for the timestep C structure used in molfile plugins.*
- class `forcebalance.molecule.Molecule`  
*Lee-Ping's general file format conversion class.*

## Namespaces

- `forcebalance.molecule`

### Functions

- def `forcebalance.molecule.getElement`
- def `forcebalance.molecule.elem_from_atomname`  
*Given an atom name, attempt to get the element in most cases.*
- def `forcebalance.molecule.nodematch`
- def `forcebalance.molecule.isint`  
*ONLY matches integers! If you have a decimal point? None shall pass!*
- def `forcebalance.molecule.isfloat`  
*Matches ANY number; it can be a decimal, scientific notation, integer, or what have you.*
- def `forcebalance.molecule.CubicLattice`  
*This function takes in three lattice lengths and three lattice angles, and tries to return a complete box specification.*

- def `forcebalance.molecule.BuildLatticeFromLengthsAngles`  
*This function takes in three lattice lengths and three lattice angles, and tries to return a complete box specification.*
- def `forcebalance.molecule.BuildLatticeFromVectors`  
*This function takes in three lattice vectors and tries to return a complete box specification.*
- def `forcebalance.molecule.format_xyz_coord`  
*Print a line consisting of (element, x, y, z) in accordance with .xyz file format.*
- def `forcebalance.molecule.format_gro_coord`  
*Print a line in accordance with .gro file format, with six decimal points of precision.*
- def `forcebalance.molecule.format_xyzgen_coord`  
*Print a line consisting of (element, p, q, r, s, t, ...) where (p, q, r) are arbitrary atom-wise data (this might happen, for instance, with atomic charges)*
- def `forcebalance.molecule.format_gro_box`  
*Print a line corresponding to the box vector in accordance with .gro file format.*
- def `forcebalance.molecule.is_gro_coord`  
*Determines whether a line contains GROMACS data or not.*
- def `forcebalance.molecule.is_charmm_coord`  
*Determines whether a line contains CHARMM data or not.*
- def `forcebalance.molecule.is_gro_box`  
*Determines whether a line contains a GROMACS box vector or not.*
- def `forcebalance.molecule.add_strip_to_mat`
- def `forcebalance.molecule.pvec`
- def `forcebalance.molecule.grouper`  
*Groups a big long iterable into groups of ten or what have you.*
- def `forcebalance.molecule.even_list`  
*Creates a list of number sequences divided as evenly as possible.*
- def `forcebalance.molecule.both`
- def `forcebalance.molecule.diff`
- def `forcebalance.molecule.either`
- def `forcebalance.molecule.EulerMatrix`  
*Constructs an Euler matrix from three Euler angles.*
- def `forcebalance.molecule.ComputeOverlap`  
*Computes an 'overlap' between two molecules based on some fictitious density.*
- def `forcebalance.molecule.AlignToDensity`  
*Computes a "overlap density" from two frames.*
- def `forcebalance.molecule.AlignToMoments`  
*Pre-aligns molecules to 'moment of inertia'.*
- def `forcebalance.molecule.get_rotate_translate`
- def `forcebalance.molecule.cartesian_product2`  
*Form a Cartesian product of two NumPy arrays.*
- def `forcebalance.molecule.main`

## Variables

- tuple `forcebalance.molecule.FrameVariableNames`
- tuple `forcebalance.molecule.AtomVariableNames` = set(['elem', 'partial\_charge', 'atomname', 'atomtype', 'tinker-suf', 'resid', 'resname', 'qcsuf', 'qm\_ghost', 'chain', 'altloc', 'icode'])
- tuple `forcebalance.molecule.MetaVariableNames` = set(['fnm', 'ftype', 'qcrems', 'qctemplate', 'qcerr', 'charge', 'mult', 'bonds'])

- tuple `forcebalance.molecule.QuantumVariableNames` = set(['qcrems', 'qctemplate', 'charge', 'mult', 'qcsuf', 'qm-ghost'])
- `forcebalance.molecule.AllVariableNames` = QuantumVariableNames|AtomVariableNames|MetaVariableNames|FrameVariableNames
- list `forcebalance.molecule.Radii`
- list `forcebalance.molecule.Elements`
- tuple `forcebalance.molecule.PeriodicTable`
- float `forcebalance.molecule.bohrang` = 0.529177249  
*One bohr equals this many angstroms.*
- tuple `forcebalance.molecule.splitter` = re.compile(r'(\s+|\S+)')
- tuple `forcebalance.molecule.Box` = namedtuple('Box',[‘a’;‘b’;‘c’;‘alpha’;‘beta’;‘gamma’;‘A’;‘B’;‘C’;‘V’])
- int `forcebalance.molecule.radian` = 180
- int `forcebalance.molecule.have_contact` = 0

## 9.22 moments.py File Reference

### Classes

- class `forcebalance.moments.Moments`

*Subclass of Target for fitting force fields to multipole moments (from experiment or theory).*

### Namespaces

- `forcebalance.moments`

*Multipole moment fitting module.*

### Variables

- tuple `forcebalance.moments.logger` = getLogger(\_\_name\_\_)

## 9.23 nifty.py File Reference

### Classes

- class `forcebalance.nifty.Pickler_LP`

*A subclass of the python Pickler that implements pickling of \_ElementTree types.*

- class `forcebalance.nifty.Unpickler_LP`

*A subclass of the python Unpickler that implements unpickling of \_ElementTree types.*

- class `forcebalance.nifty.LineChunker`

### Namespaces

- `forcebalance.nifty`

*Nifty functions, intended to be imported by any module within ForceBalance.*

### Functions

- def `forcebalance.nifty.pvec1d`

*Printout of a 1-D vector.*

- def `forcebalance.nifty.astr`

*Write an array to a string so we can use it to key a dictionary.*

- def `forcebalance.nifty.pmat2d`

- def `forcebalance.nifty.grouper`  
*Printout of a 2-D matrix.*
- def `forcebalance.nifty.encode`  
*Get the representative integer value from an array.*
- def `forcebalance.nifty.segments`  
*Cool-looking printout for slick formatting of output.*
- def `forcebalance.nifty.commadash`  
*See documentation for printcool; this is a nice way to print out keys/values in a dictionary.*
- def `forcebalance.nifty.uncommadash`  
*ONLY matches integers! If you have a decimal point? None shall pass!*
- def `forcebalance.nifty.isfloat`  
*Matches ANY number; it can be a decimal, scientific notation, what have you CAUTION - this will also match an integer.*
- def `forcebalance.nifty.isdecimal`  
*Matches things with a decimal only; see isint and isfloat.*
- def `forcebalance.nifty.floatoran`  
*Returns a big number if we encounter NaN.*
- def `forcebalance.nifty.col`  
*Given any list, array, or matrix, return a 1-column matrix.*
- def `forcebalance.nifty.row`  
*Given any list, array, or matrix, return a 1-row matrix.*
- def `forcebalance.nifty.flat`  
*Given any list, array, or matrix, return a single-index array.*
- def `forcebalance.nifty.monotonic`  
*Invert a matrix using singular value decomposition.*
- def `forcebalance.nifty.get_least_squares`  
*Compute the (cross) statistical inefficiency of (two) timeseries.*
- def `forcebalance.nifty.statisticalInefficiency`  
*Use this instead of pickle.dump for pickling anything that contains \_ElementTree types.*
- def `forcebalance.nifty.lp_dump`  
*Use this instead of pickle.load for unpickling anything that contains \_ElementTree types.*
- def `forcebalance.nifty.lp_load`  
*Submit a job to the Work Queue.*
- def `forcebalance.nifty.queue_up_src_dest`  
*Submit a job to the Work Queue.*
- def `forcebalance.nifty.wq_wait1`

*This function waits ten seconds to see if a task in the Work Queue has finished.*

- def `forcebalance.nifty.wq_wait`

*This function waits until the work queue is completely empty.*
- def `forcebalance.nifty.click`

*Stopwatch function for timing.*
- def `forcebalance.nifty.bak`
- def `forcebalance.nifty.onefile`
- def `forcebalance.nifty.GoInto`
- def `forcebalance.nifty.allsplit`
- def `forcebalance.nifty.Leave`
- def `forcebalance.nifty.MissingFileInspection`
- def `forcebalance.nifty.wopen`

*If trying to write to a symbolic link, remove it first.*
- def `forcebalance.nifty.LinkFile`
- def `forcebalance.nifty.CopyFile`
- def `forcebalance.nifty.link_dir_contents`
- def `forcebalance.nifty.remove_if_exists`

*Remove the file if it exists (doesn't return an error).*
- def `forcebalance.nifty.which`
- def `forcebalance.nifty.warn_press_key`
- def `forcebalance.nifty.warn_once`

*Prints a warning but will only do so once in a given run.*
- def `forcebalance.nifty.concurrent_map`

*Similar to the builtin function map().*

## Variables

- tuple `forcebalance.nifty.logger` = getLogger(\_\_name\_\_)
- float `forcebalance.nifty.kb` = 0.0083144100163

*Boltzmann constant.*
- float `forcebalance.nifty.eqcgmx` = 2625.5002

*Q-Chem to GMX unit conversion for energy.*
- float `forcebalance.nifty.fqcgmx` = -49621.9

*Q-Chem to GMX unit conversion for force.*
- float `forcebalance.nifty.bohrang` = 0.529177249

*One bohr equals this many angstroms.*
- string `forcebalance.nifty.XMLFILE` = 'x'

*Pickle uses 'flags' to pickle and unpickle different variable types.*
- `forcebalance.nifty.WORK_QUEUE` = None
- tuple `forcebalance.nifty.WQIDS` = defaultdict(list)
- list `forcebalance.nifty.specific_lst`
- tuple `forcebalance.nifty.specific_dct` = dict(list(itertools.chain(\*[[j,i[1]] for j in i[0]] for i in specific\_lst)))

## 9.24 objective.py File Reference

### Classes

- class `forcebalance.objective.Objective`

*Objective function.*
- class `forcebalance.objective.Penalty`

*Penalty functions for regularizing the force field optimizer.*

## Namespaces

- `forcebalance.objective`

*ForceBalance objective function.*

## Variables

- tuple `forcebalance.objective.logger` = getLogger(`__name__`)
- dictionary `forcebalance.objective.Implemented_Targets`

*The table of implemented Targets.*

- list `forcebalance.objective.Letters` = ['X','G','H']

*This is the canonical lettering that corresponds to : objective function, gradient, Hessian.*

## 9.25 openmmio.py File Reference

### Classes

- class `forcebalance.openmmio.OpenMM_Reader`  
*Class for parsing OpenMM force field files.*
- class `forcebalance.openmmio.OpenMM`  
*Derived from Engine object for carrying out general purpose OpenMM calculations.*
- class `forcebalance.openmmio.Liquid_OpenMM`  
*Condensed phase property matching using OpenMM.*
- class `forcebalance.openmmio.AbInitio_OpenMM`  
*Force and energy matching using OpenMM.*
- class `forcebalance.openmmio.BindingEnergy_OpenMM`  
*Binding energy matching using OpenMM.*
- class `forcebalance.openmmio.Interaction_OpenMM`  
*Interaction matching using OpenMM.*
- class `forcebalance.openmmio.Moments_OpenMM`  
*Multipole moment matching using OpenMM.*

### Namespaces

- `forcebalance.openmmio`  
*OpenMM input/output.*

### Functions

- def `forcebalance.openmmio.energy_components`
- def `forcebalance.openmmio.get_multipoles`  
*Return the current multipole moments in Debye and Buckingham units.*
- def `forcebalance.openmmio.get_dipole`  
*Return the current dipole moment in Debye.*
- def `forcebalance.openmmio.PrepareVirtualSites`  
*Prepare a list of function wrappers and vsite parameters from the system.*
- def `forcebalance.openmmio.ResetVirtualSites.fast`  
*Given a set of OpenMM-compatible positions and a System object, compute the correct virtual site positions according to the System.*
- def `forcebalance.openmmio.ResetVirtualSites`

*Given a set of OpenMM-compatible positions and a System object, compute the correct virtual site positions according to the System.*

- def `forcebalance.openmmio.GetVirtualSiteParameters`  
*Return an array of all virtual site parameters in the system.*
- def `forcebalance.openmmio.CopyAmoebaBondParameters`
- def `forcebalance.openmmio.CopyAmoebaOutOfPlaneBendParameters`
- def `forcebalance.openmmio.CopyAmoebaAngleParameters`
- def `forcebalance.openmmio.CopyAmoebaInPlaneAngleParameters`
- def `forcebalance.openmmio.CopyAmoebaVdwParameters`
- def `forcebalance.openmmio.CopyAmoebaMultipoleParameters`
- def `forcebalance.openmmio.CopyHarmonicBondParameters`
- def `forcebalance.openmmio.CopyHarmonicAngleParameters`
- def `forcebalance.openmmio.CopyPeriodicTorsionParameters`
- def `forcebalance.openmmio.CopyNonbondedParameters`
- def `forcebalance.openmmio.do_nothing`
- def `forcebalance.openmmio.CopySystemParameters`  
*Copy parameters from one system (i.e.*
- def `forcebalance.openmmio.UpdateSimulationParameters`
- def `forcebalance.openmmio.SetAmoebaVirtualExclusions`
- def `forcebalance.openmmio.MTSVVVRIntegrator`  
*Create a multiple timestep velocity verlet with velocity randomization (VVVR) integrator.*

## Variables

- tuple `forcebalance.openmmio.logger` = getLogger(\_\_name\_\_)
- dictionary `forcebalance.openmmio.suffix_dict`
- string `forcebalance.openmmio.pdict` = "XML\_Override"  
*pdict is a useless variable if the force field is XML.*

## 9.26 optimizer.py File Reference

### Classes

- class `forcebalance.optimizer.Optimizer`  
*Optimizer class.*

### Namespaces

- `forcebalance.optimizer`  
*Optimization algorithms.*

### Functions

- def `forcebalance.optimizer.Counter`
- def `forcebalance.optimizer.First`
- def `forcebalance.optimizer.GoodStep`

### Variables

- tuple `forcebalance.optimizer.logger` = getLogger(\_\_name\_\_)
- int `forcebalance.optimizer.ITERATION` = 0
- int `forcebalance.optimizer.GOODSTEP` = 0
- int `forcebalance.optimizer.ITERINIT` = 0

## 9.27 output.py File Reference

### Classes

- class `forcebalance.output.ForceBalanceLogger`

*This logger starts out with a default handler that writes to stdout addHandler removes this default the first time another handler is added.*
- class `forcebalance.output.RawStreamHandler`

*Exactly like output.StreamHandler except it does no extra formatting before sending logging messages to the stream.*
- class `forcebalance.output.RawFileHandler`

*Exactly like output.FileHandler except it does no extra formatting before sending logging messages to the file.*
- class `forcebalance.output.CleanStreamHandler`

*Similar to RawStreamHandler except it does not write terminal escape codes.*
- class `forcebalance.output.CleanFileHandler`

*File handler that does not write terminal escape codes and carriage returns to files.*
- class `forcebalance.output.ModLogger`

### Namespaces

- `forcebalance.output`

## 9.28 parser.py File Reference

### Namespaces

- `forcebalance.parser`

*Input file parser for ForceBalance jobs.*

### Functions

- def `forcebalance.parser.read_mvals`
- def `forcebalance.parser.read_pvals`
- def `forcebalance.parser.read_priors`
- def `forcebalance.parser.read_internals`
- def `forcebalance.parser.printsection`

*Print out a section of the input file in a parser-compliant and readable format.*
- def `forcebalance.parser.parse_inputs`

*Parse through the input file and read all user-supplied options.*

### Variables

- tuple `forcebalance.parser.logger` = getLogger(\_\_name\_\_)
- dictionary `forcebalance.parser.gen_opts_types`

*Default general options.*
- dictionary `forcebalance.parser.tgt_opts_types`

*Default fitting target options.*
- tuple `forcebalance.parser.all_opts_names` = list(itertools.chain(\*[i.keys() for i in gen\_opts\_types.values()]))
- list `forcebalance.parser.iocc` = []

*Check for uniqueness of option names.*
- dictionary `forcebalance.parser.gen_opts_defaults` = {}

*Default general options - basically a collapsed version of gen\_opts\_types.*
- dictionary `forcebalance.parser.subdict` = {}

- dictionary `forcebalance.parser.tgt_opts_defaults = {}`  
*Default target options - basically a collapsed version of tgt\_opts\_types.*
- dictionary `forcebalance.parser.bkwd`  
*Option maps for maintaining backward compatibility.*
- list `forcebalance.parser.mainsections = ["SIMULATION","TARGET","OPTIONS","END","NONE"]`  
*Listing of sections in the input file.*
- dictionary `forcebalance.parser.ParsTab`  
*ParsTab that refers to subsection parsers.*

## 9.29 psi4io.py File Reference

### Classes

- class `forcebalance.psi4io.GBS_Reader`  
*Interaction type -> Parameter Dictionary.*
- class `forcebalance.psi4io.THCDF_Psi4`
- class `forcebalance.psi4io.Grid_Reader`  
*Finite state machine for parsing DVR grid files.*
- class `forcebalance.psi4io.RDVR3_Psi4`  
*Subclass of Target for R-DVR3 grid fitting.*

### Namespaces

- `forcebalance.psi4io`  
*PSI4 force field input/output.*

### Variables

- tuple `forcebalance.psi4io.logger = getLogger(__name__)`

## 9.30 PT.py File Reference

### Namespaces

- `forcebalance.PT`

### Variables

- dictionary `forcebalance.PT.PeriodicTable`
- list `forcebalance.PT.Elements`

## 9.31 qchemio.py File Reference

### Classes

- class `forcebalance.qchemio.QCIn_Reader`  
*Finite state machine for parsing Q-Chem input files.*

### Namespaces

- `forcebalance.qchemio`  
*Q-Chem input file parser.*

## Functions

- def `forcebalance.qchemio.QChem_Dielectric_Energy`

## Variables

- tuple `forcebalance.qchemio.logger` = getLogger(\_\_name\_\_)
- list `forcebalance.qchemio.ndtypes` = [None]  
*Types of counterpoise correction cptypes = [None, 'BASS', 'BASSP'] Types of NDDO correction.*
- dictionary `forcebalance.qchemio.pdict`  
*Section -> Interaction type dictionary.*

## 9.32 quantity.py File Reference

### Classes

- class `forcebalance.quantity.Quantity`  
*Base class for thermodynamical quantity used for fitting.*
- class `forcebalance.quantity.Quantity_Density`
- class `forcebalance.quantity.Quantity_H_vap`

### Namespaces

- `forcebalance.quantity`

### Functions

- def `forcebalance.quantity.mean_stderr`  
*Return mean and standard deviation of a time series ts.*
- def `forcebalance.quantity.energy_derivatives`  
*Compute the first derivatives of a set of snapshot energies with respect to the force field parameters.*

### Variables

- tuple `forcebalance.quantity.logger` = getLogger(\_\_name\_\_)

## 9.33 target.py File Reference

### Classes

- class `forcebalance.target.Target`  
*Base class for all fitting targets.*
- class `forcebalance.target.RemoteTarget`

### Namespaces

- `forcebalance.target`

### Variables

- tuple `forcebalance.target.logger` = getLogger(\_\_name\_\_)

## 9.34 thermo.py File Reference

### Classes

- class `forcebalance.thermo.Thermo`  
*A target for fitting general experimental data sets.*
- class `forcebalance.thermo.Point`

### Namespaces

- `forcebalance.thermo`

### Variables

- tuple `forcebalance.thermo.logger` = getLogger(`_name_`)

## 9.35 tinkerio.py File Reference

### Classes

- class `forcebalance.tinkerio.Tinker_Reader`  
*Finite state machine for parsing `TINKER` force field files.*
- class `forcebalance.tinkerio.TINKER`  
*Engine for carrying out general purpose `TINKER` calculations.*
- class `forcebalance.tinkerio.Liquid_TINKER`  
*Condensed phase property matching using `TINKER`.*
- class `forcebalance.tinkerio.AbInitio_TINKER`  
*Subclass of Target for force and energy matching using `TINKER`.*
- class `forcebalance.tinkerio.BindingEnergy_TINKER`  
*Binding energy matching using `TINKER`.*
- class `forcebalance.tinkerio.Interaction_TINKER`  
*Subclass of Target for interaction matching using `TINKER`.*
- class `forcebalance.tinkerio.Moments_TINKER`  
*Subclass of Target for multipole moment matching using `TINKER`.*
- class `forcebalance.tinkerio.Vibration_TINKER`  
*Vibrational frequency matching using `TINKER`.*

### Namespaces

- `forcebalance.tinkerio`  
*`TINKER` input/output.*

### Functions

- def `forcebalance.tinkerio.write_key`  
*Create or edit a `TINKER` .key file.*

### Variables

- list `forcebalance.tinkerio.allp`
- list `forcebalance.tinkerio.eckeys`
- tuple `forcebalance.tinkerio.logger` = getLogger(`_name_`)
- dictionary `forcebalance.tinkerio.pdict`

## 9.36 vibration.py File Reference

### Classes

- class `forcebalance.vibration.Vibration`

*Subclass of Target for fitting force fields to vibrational spectra (from experiment or theory).*

### Namespaces

- `forcebalance.vibration`

*Vibrational mode fitting module.*

### Functions

- def `forcebalance.vibration.count_assignment`

### Variables

- tuple `forcebalance.vibration.logger` = getLogger(`_name_`)

## Index

`_add_`  
    `forcebalance::molecule::Molecule`, 543  
`_delitem_`  
    `forcebalance::molecule::Molecule`, 543  
`_enter_`  
    `forcebalance::nifty::LineChunker`, 409  
`_eq_`  
    `forcebalance::forcefield::FF`, 296  
    `forcebalance::molecule::MyG`, 621  
`_exit_`  
    `forcebalance::nifty::LineChunker`, 409  
`_getattr_`  
    `forcebalance::molecule::Molecule`, 543  
`_getitem_`  
    `forcebalance::molecule::Molecule`, 543  
`_hash_`  
    `forcebalance::molecule::MyG`, 621  
`_iadd_`  
    `forcebalance::molecule::Molecule`, 544  
`_init_`  
    `forcebalance::abinitio::AbInitio`, 79  
    `forcebalance::abinitio_internal::AbInitio_Internal`, 146  
    `forcebalance::amberio::AbInitio_AMBER`, 102  
    `forcebalance::amberio::FrcMod_Reader`, 309  
    `forcebalance::amberio::Mol2_Reader`, 537  
    `forcebalance::BaseClass`, 209  
    `forcebalance::BaseReader`, 211  
    `forcebalance::binding::BindingEnergy`, 215  
    `forcebalance::counterpoise::Counterpoise`, 277  
    `forcebalance::custom_io::Gen_Reader`, 315  
    `forcebalance::engine::Engine`, 292  
    `forcebalance::forcefield::BackedUpDict`, 207  
    `forcebalance::forcefield::FF`, 295  
    `forcebalance::gmxio::AbInitio_GMX`, 124  
    `forcebalance::gmxio::BindingEnergy_GMX`, 230  
    `forcebalance::gmxio::GMX`, 319  
    `forcebalance::gmxio::Interaction_GMX`, 348  
    `forcebalance::gmxio::ITP_Reader`, 392  
    `forcebalance::gmxio::Lipid_GMX`, 433  
    `forcebalance::gmxio::Liquid_GMX`, 468  
    `forcebalance::gmxio::Moments_GMX`, 578  
    `forcebalance::gmxio::Thermo_GMX`, 751  
    `forcebalance::gmxio::Vibration_GMX`, 795  
    `forcebalance::interaction::Interaction`, 333  
    `forcebalance::leastsq::LeastSquares`, 397  
    `forcebalance::lipid::Lipid`, 414  
    `forcebalance::liquid::Liquid`, 450  
    `forcebalance::Mol2::mol2`, 524  
    `forcebalance::Mol2::mol2_atom`, 528  
    `forcebalance::Mol2::mol2_bond`, 531  
    `forcebalance::Mol2::mol2_set`, 538

`forcebalance::mol2io::Mol2_Reader`, 534  
`forcebalance::molecule::Molecule`, 543  
`forcebalance::molecule::MyG`, 621  
`forcebalance::moments::Moments`, 563  
`forcebalance::nifty::LineChunker`, 409  
`forcebalance::nifty::Pickler_LP`, 657  
`forcebalance::nifty::Unpickler_LP`, 777  
`forcebalance::objective::Objective`, 624  
`forcebalance::objective::Penalty`, 654  
`forcebalance::openmmio::AbInitio_OpenMM`, 168  
`forcebalance::openmmio::BindingEnergy_OpenMM`, 245  
`forcebalance::openmmio::Interaction_OpenMM`, 363  
`forcebalance::openmmio::Liquid_OpenMM`, 488  
`forcebalance::openmmio::Moments_OpenMM`, 593  
`forcebalance::openmmio::OpenMM`, 629  
`forcebalance::openmmio::OpenMM_Reader`, 638  
`forcebalance::optimizer::Optimizer`, 642  
`forcebalance::output::CleanStreamHandler`, 273  
`forcebalance::output::ForceBalanceLogger`, 307  
`forcebalance::output::RawStreamHandler`, 670  
`forcebalance::psi4io::GBS_Reader`, 312  
`forcebalance::psi4io::Grid_Reader`, 329  
`forcebalance::psi4io::RDVR3_Psi4`, 674  
`forcebalance::psi4io::THCDF_Psi4`, 718  
`forcebalance::qcchemio::QCIn_Reader`, 661  
`forcebalance::quantity::Quantity`, 664  
`forcebalance::quantity::Quantity_Density`, 666  
`forcebalance::quantity::Quantity_H_vap`, 668  
`forcebalance::target::RemoteTarget`, 690  
`forcebalance::target::Target`, 704  
`forcebalance::thermo::Point`, 659  
`forcebalance::thermo::Thermo`, 735  
`forcebalance::tinkerio::AbInitio_TINKER`, 190  
`forcebalance::tinkerio::BindingEnergy_TINKER`, 260  
`forcebalance::tinkerio::Interaction_TINKER`, 378  
`forcebalance::tinkerio::Liquid_TINKER`, 507  
`forcebalance::tinkerio::Moments_TINKER`, 608  
`forcebalance::tinkerio::TINKER`, 767  
`forcebalance::tinkerio::Tinker_Reader`, 775  
`forcebalance::tinkerio::Vibration_TINKER`, 810  
`forcebalance::vibration::Vibration`, 781

`_init_.py`, 822  
`_iter_`  
    `forcebalance::molecule::Molecule`, 544  
`_len_`  
    `forcebalance::molecule::Molecule`, 544  
`_missing_`  
    `forcebalance::forcefield::BackedUpDict`, 207  
`_repr_`  
    `forcebalance::Mol2::mol2`, 524

forcebalance::Mol2::mol2\_atom, 528  
 forcebalance::Mol2::mol2\_bond, 531  
 \_\_setattr\_\_  
     forcebalance::abinitio::AbInitio, 79  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 102  
     forcebalance::BaseClass, 209  
     forcebalance::binding::BindingEnergy, 215  
     forcebalance::counterpoise::Counterpoise, 277  
     forcebalance::engine::Engine, 292  
     forcebalance::forcefield::FF, 296  
     forcebalance::gmxio::AbInitio\_GMX, 124  
     forcebalance::gmxio::BindingEnergy\_GMX, 230  
     forcebalance::gmxio::GMX, 319  
     forcebalance::gmxio::Interaction\_GMX, 349  
     forcebalance::gmxio::Lipid\_GMX, 433  
     forcebalance::gmxio::Liquid\_GMX, 468  
     forcebalance::gmxio::Moments\_GMX, 578  
     forcebalance::gmxio::Thermo\_GMX, 751  
     forcebalance::gmxio::Vibration\_GMX, 795  
     forcebalance::interaction::Interaction, 333  
     forcebalance::leastsq::LeastSquares, 397  
     forcebalance::lipid::Lipid, 415  
     forcebalance::liquid::Liquid, 450  
     forcebalance::molecule::Molecule, 544  
     forcebalance::moments::Moments, 563  
     forcebalance::objective::Objective, 624  
     forcebalance::openmmio::AbInitio\_OpenMM, 168  
     forcebalance::openmmio::BindingEnergy\_OpenMM,  
         245  
     forcebalance::openmmio::Interaction\_OpenMM, 364  
     forcebalance::openmmio::Liquid\_OpenMM, 488  
     forcebalance::openmmio::Moments\_OpenMM, 593  
     forcebalance::openmmio::OpenMM, 629  
     forcebalance::optimizer::Optimizer, 642  
     forcebalance::psi4io::RDVR3\_Psi4, 674  
     forcebalance::psi4io::THCDF\_Psi4, 718  
     forcebalance::target::RemoteTarget, 690  
     forcebalance::target::Target, 704  
     forcebalance::thermo::Thermo, 735  
     forcebalance::tinkerio::AbInitio\_TINKER, 190  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 260  
     forcebalance::tinkerio::Interaction\_TINKER, 379  
     forcebalance::tinkerio::Liquid\_TINKER, 507  
     forcebalance::tinkerio::Moments\_TINKER, 608  
     forcebalance::tinkerio::TINKER, 767  
     forcebalance::tinkerio::Vibration\_TINKER, 810  
     forcebalance::vibration::Vibration, 781  
 \_\_str\_\_  
     forcebalance::quantity::Quantity, 664  
     forcebalance::quantity::Quantity\_Density, 666  
     forcebalance::quantity::Quantity\_H\_vap, 668  
     forcebalance::thermo::Point, 659  
 \_\_version\_\_  
     forcebalance, 15

A

forcebalance::chemistry, 20  
 forcebalance::openmmio::OpenMM, 635  
 forcebalance::tinkerio::TINKER, 772

a

forcebalance::objective::Penalty, 656

AMOEBA

forcebalance::openmmio::OpenMM, 635

AStr

forcebalance::molecule::MyG, 621

abinitio.py, 822

abinitio\_internal.py, 823

abskey

forcebalance::tinkerio::TINKER, 772

absrd

forcebalance::abinitio::AbInitio, 79  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
 forcebalance::amberio::AbInitio\_AMBER, 102  
 forcebalance::binding::BindingEnergy, 215  
 forcebalance::counterpoise::Counterpoise, 277  
 forcebalance::gmxio::AbInitio\_GMX, 124  
 forcebalance::gmxio::BindingEnergy\_GMX, 230  
 forcebalance::gmxio::Interaction\_GMX, 349  
 forcebalance::gmxio::Lipid\_GMX, 433  
 forcebalance::gmxio::Liquid\_GMX, 468  
 forcebalance::gmxio::Moments\_GMX, 578  
 forcebalance::gmxio::Thermo\_GMX, 751  
 forcebalance::gmxio::Vibration\_GMX, 795  
 forcebalance::interaction::Interaction, 333  
 forcebalance::leastsq::LeastSquares, 397  
 forcebalance::lipid::Lipid, 415  
 forcebalance::liquid::Liquid, 450  
 forcebalance::moments::Moments, 563  
 forcebalance::openmmio::AbInitio\_OpenMM, 168  
 forcebalance::openmmio::BindingEnergy\_OpenMM,  
 245  
 forcebalance::openmmio::Interaction\_OpenMM, 364  
 forcebalance::openmmio::Liquid\_OpenMM, 488  
 forcebalance::openmmio::Moments\_OpenMM, 593  
 forcebalance::psi4io::RDVR3\_Psi4, 674  
 forcebalance::psi4io::THCDF\_Psi4, 718  
 forcebalance::target::RemoteTarget, 690  
 forcebalance::target::Target, 704  
 forcebalance::thermo::Thermo, 735  
 forcebalance::tinkerio::AbInitio\_TINKER, 190  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 260  
 forcebalance::tinkerio::Interaction\_TINKER, 379  
 forcebalance::tinkerio::Liquid\_TINKER, 507  
 forcebalance::tinkerio::Moments\_TINKER, 608  
 forcebalance::tinkerio::Vibration\_TINKER, 810  
 forcebalance::vibration::Vibration, 781

add\_quantum

```

    forcebalance::molecule::Molecule, 544
add_strip_to_mat
    forcebalance::molecule, 38
add_virtual_site
    forcebalance::molecule::Molecule, 544
addHandler
    forcebalance::output::ForceBalanceLogger, 307
addff
    forcebalance::forcefield::FF, 296
addff_txt
    forcebalance::forcefield::FF, 297
addff_xml
    forcebalance::forcefield::FF, 298
adict
    forcebalance::amberio::FrcMod_Reader, 310
    forcebalance::amberio::Mol2_Reader, 537
    forcebalance::BaseReader, 212
    forcebalance::custom_io::Gen_Reader, 316
    forcebalance::gmxio::ITP_Reader, 393
    forcebalance::mol2io::Mol2_Reader, 534
    forcebalance::openmmio::OpenMM_Reader, 639
    forcebalance::psi4io::GBS_Reader, 313
    forcebalance::psi4io::Grid_Reader, 329
    forcebalance::qchemio::QCIn_Reader, 662
    forcebalance::tinkerio::Tinker_Reader, 776
adjh
    forcebalance::optimizer::Optimizer, 642
aftypes
    forcebalance::gmxio, 32
align
    forcebalance::molecule::Molecule, 544
align_by_moments
    forcebalance::molecule::Molecule, 545
align_center
    forcebalance::molecule::Molecule, 545
AlignToDensity
    forcebalance::molecule, 38
AlignToMoments
    forcebalance::molecule, 38
Alive
    forcebalance::molecule::MyG, 622
all_at_once
    forcebalance::amberio::AbInitio_AMBER, 114
all_opts_names
    forcebalance::parser, 66
all_pairwise_rmsd
    forcebalance::molecule::Molecule, 545
AllResults
    forcebalance::gmxio::Liquid_GMX, 481
    forcebalance::liquid::Liquid, 462
    forcebalance::openmmio::Liquid_OpenMM, 500
    forcebalance::tinkerio::Liquid_TINKER, 520
AllVariableNames
    forcebalance::molecule, 42
allp
    forcebalance::tinkerio, 73
allsplit
    forcebalance::nifty, 47
amberio.py, 823
amom
    forcebalance::psi4io::GBS_Reader, 313
Anneal
    forcebalance::optimizer::Optimizer, 642
api.dox, 823
append
    forcebalance::molecule::Molecule, 546
assign_field
    forcebalance::forcefield::FF, 299
assign_p0
    forcebalance::forcefield::FF, 299
astr
    forcebalance::nifty, 47
atom
    forcebalance::amberio::FrcMod_Reader, 310
    forcebalance::amberio::Mol2_Reader, 537
    forcebalance::mol2io::Mol2_Reader, 534
    forcebalance::qchemio::QCIn_Reader, 662
    forcebalance::tinkerio::Tinker_Reader, 776
atom_distances
    forcebalance::contact, 21
atom_id
    forcebalance::Mol2::mol2_atom, 530
atom_name
    forcebalance::Mol2::mol2_atom, 530
atom_select
    forcebalance::molecule::Molecule, 546
atom_stack
    forcebalance::molecule::Molecule, 546
atom_type
    forcebalance::Mol2::mol2_atom, 530
AtomLists
    forcebalance::abinitio::AbInitio, 92
    forcebalance::abinitio_internal::AbInitio_Internal, 158
    forcebalance::amberio::AbInitio_AMBER, 114
    forcebalance::gmxio::AbInitio_GMX, 136
    forcebalance::gmxio::GMX, 325
    forcebalance::openmmio::AbInitio_OpenMM, 180
    forcebalance::openmmio::OpenMM, 635
    forcebalance::tinkerio::AbInitio_TINKER, 202
    forcebalance::tinkerio::TINKER, 772
AtomMask
    forcebalance::abinitio::AbInitio, 92
    forcebalance::abinitio_internal::AbInitio_Internal, 158
    forcebalance::amberio::AbInitio_AMBER, 114
    forcebalance::gmxio::AbInitio_GMX, 136
    forcebalance::gmxio::GMX, 325
    forcebalance::openmmio::AbInitio_OpenMM, 180
    forcebalance::openmmio::OpenMM, 635

```

forcebalance::tinkerio::AbInitio\_TINKER, 202  
 forcebalance::tinkerio::TINKER, 772  
**AtomTypes**  
 forcebalance::amberio::FrcMod\_Reader, 310  
 forcebalance::amberio::Mol2\_Reader, 537  
 forcebalance::BaseReader, 212  
 forcebalance::custom.io::Gen\_Reader, 316  
 forcebalance::gmxio::ITP\_Reader, 393  
 forcebalance::mol2io::Mol2\_Reader, 535  
 forcebalance::openmmio::OpenMM\_Reader, 639  
 forcebalance::psi4io::GBS\_Reader, 313  
 forcebalance::psi4io::Grid\_Reader, 329  
 forcebalance::qchemio::QCIn\_Reader, 662  
 forcebalance::tinkerio::Tinker\_Reader, 776  
**AtomVariableNames**  
 forcebalance::molecule, 42  
**atomnames**  
 forcebalance::amberio::Mol2\_Reader, 537  
 forcebalance::forcefield::FF, 304  
 forcebalance::gmxio::ITP\_Reader, 393  
**atoms**  
 forcebalance::chemistry, 20  
 forcebalance::Mol2::mol2, 527  
**atomtype\_to\_mass**  
 forcebalance::gmxio::ITP\_Reader, 393  
**atomtypes**  
 forcebalance::gmxio::ITP\_Reader, 393  
**B**  
 forcebalance::chemistry, 20  
 forcebalance::openmmio::OpenMM, 635  
 forcebalance::tinkerio::TINKER, 772  
**b**  
 forcebalance::objective::Penalty, 656  
**BE**  
 forcebalance::chemistry, 20  
**BFGS**  
 forcebalance::optimizer::Optimizer, 643  
**backup\_dict**  
 forcebalance::forcefield::BackedUpDict, 207  
**bak**  
 forcebalance::nifty, 47  
**bakdir**  
 forcebalance::optimizer::Optimizer, 651  
**BasinHopping**  
 forcebalance::optimizer::Optimizer, 643  
**basis\_number**  
 forcebalance::psi4io::GBS\_Reader, 313  
**bftypes**  
 forcebalance::gmxio, 32  
**bhyp**  
 forcebalance::optimizer::Optimizer, 651  
**bidirect**  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
**binding.py**, 823  
**bkwd**  
 forcebalance::parser, 66  
**bo**  
 forcebalance::chemistry, 20  
**bohrang**  
 forcebalance::molecule, 42  
 forcebalance::nifty, 56  
**boltz\_wts**  
 forcebalance::abinitio::AbInitio, 92  
 forcebalance::abinitio.internal::AbInitio\_Internal, 158  
 forcebalance::amberio::AbInitio\_AMBER, 114  
 forcebalance::gmxio::AbInitio\_GMX, 136  
 forcebalance::openmmio::AbInitio\_OpenMM, 180  
 forcebalance::tinkerio::AbInitio\_TINKER, 202  
**bond\_id**  
 forcebalance::Mol2::mol2\_bond, 532  
**bond\_type**  
 forcebalance::Mol2::mol2\_bond, 532  
**BondChars**  
 forcebalance::chemistry, 20  
**BondEnergies**  
 forcebalance::chemistry, 20  
**BondStrengthByLength**  
 forcebalance::chemistry, 20  
**bonds**  
 forcebalance::Mol2::mol2, 527  
**both**  
 forcebalance::molecule, 39  
**Box**  
 forcebalance::molecule, 42  
**boxes**  
 forcebalance::molecule::Molecule, 558  
**buf**  
 forcebalance::nifty::LineChunker, 410  
**build\_invdist**  
 forcebalance::abinitio::AbInitio, 80  
 forcebalance::abinitio.internal::AbInitio\_Internal, 146  
 forcebalance::amberio::AbInitio\_AMBER, 102  
 forcebalance::gmxio::AbInitio\_GMX, 124  
 forcebalance::openmmio::AbInitio\_OpenMM, 168  
 forcebalance::tinkerio::AbInitio\_TINKER, 190  
**build\_objective**  
 forcebalance::abinitio, 16  
**build\_pid**  
 forcebalance::amberio::FrcMod\_Reader, 309  
 forcebalance::amberio::Mol2\_Reader, 537  
 forcebalance::BaseReader, 211  
 forcebalance::custom.io::Gen\_Reader, 315  
 forcebalance::gmxio::ITP\_Reader, 392  
 forcebalance::mol2io::Mol2\_Reader, 534  
 forcebalance::openmmio::OpenMM\_Reader, 638  
 forcebalance::psi4io::GBS\_Reader, 312  
 forcebalance::psi4io::Grid\_Reader, 329

forcebalance::qchemio::QCIn\_Reader, 661  
 forcebalance::tinkerio::Tinker\_Reader, 775  
**build\_topology**  
 forcebalance::molecule::Molecule, 546  
**BuildLatticeFromLengthsAngles**  
 forcebalance::molecule, 39  
**BuildLatticeFromVectors**  
 forcebalance::molecule, 39  
**built\_bonds**  
 forcebalance::molecule::Molecule, 558  
  
**c2r**  
 forcebalance::gmxio::Vibration\_GMX, 805  
 forcebalance::tinkerio::Vibration\_TINKER, 820  
 forcebalance::vibration::Vibration, 790  
**CHECK\_BASIS**  
 forcebalance::leastsq, 34  
**calc\_eigvals**  
 forcebalance::gmxio::Vibration\_GMX, 805  
 forcebalance::tinkerio::Vibration\_TINKER, 820  
 forcebalance::vibration::Vibration, 790  
**calc\_moments**  
 forcebalance::gmxio::Moments\_GMX, 588  
 forcebalance::moments::Moments, 573  
 forcebalance::openmmio::Moments\_OpenMM, 603  
 forcebalance::tinkerio::Moments\_TINKER, 618  
**calc\_scd**  
 forcebalance::gmxio::GMX, 319  
**callback**  
 forcebalance::nifty::LineChunker, 410  
**callderivs**  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
**callgmx**  
 forcebalance::gmxio::GMX, 319  
**calltinker**  
 forcebalance::tinkerio::TINKER, 767  
**cartesian\_product2**  
 forcebalance::molecule, 39  
**center\_of\_mass**  
 forcebalance::molecule::Molecule, 546  
**charge**  
 forcebalance::Mol2::mol2\_atom, 530  
**charge\_type**  
 forcebalance::Mol2::mol2, 527  
**check\_files**  
 forcebalance::abinitio::AbInitio, 80  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
 forcebalance::amberio::AbInitio\_AMBER, 102  
 forcebalance::binding::BindingEnergy, 216  
 forcebalance::counterpoise::Counterpoise, 278  
 forcebalance::gmxio::AbInitio\_GMX, 124  
 forcebalance::gmxio::BindingEnergy\_GMX, 231  
 forcebalance::gmxio::Interaction\_GMX, 349  
 forcebalance::gmxio::Lipid\_GMX, 433  
  
 forcebalance::gmxio::Liquid\_GMX, 468  
 forcebalance::gmxio::Moments\_GMX, 579  
 forcebalance::gmxio::Thermo\_GMX, 752  
 forcebalance::gmxio::Vibration\_GMX, 796  
 forcebalance::interaction::Interaction, 334  
 forcebalance::leastsq::LeastSquares, 397  
 forcebalance::lipid::Lipid, 415  
 forcebalance::liquid::Liquid, 450  
 forcebalance::moments::Moments, 564  
 forcebalance::openmmio::AbInitio\_OpenMM, 168  
 forcebalance::openmmio::BindingEnergy\_OpenMM,  
     246  
 forcebalance::openmmio::Interaction\_OpenMM, 364  
 forcebalance::openmmio::Liquid\_OpenMM, 488  
 forcebalance::openmmio::Moments\_OpenMM, 594  
 forcebalance::psi4io::RDVR3\_Psi4, 675  
 forcebalance::psi4io::THCDF\_Psi4, 719  
 forcebalance::target::RemoteTarget, 690  
 forcebalance::target::Target, 705  
 forcebalance::thermo::Thermo, 736  
 forcebalance::tinkerio::AbInitio\_TINKER, 190  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 261  
 forcebalance::tinkerio::Interaction\_TINKER, 379  
 forcebalance::tinkerio::Liquid\_TINKER, 507  
 forcebalance::tinkerio::Moments\_TINKER, 609  
 forcebalance::tinkerio::Vibration\_TINKER, 811  
 forcebalance::vibration::Vibration, 781  
**CheckBasis**  
 forcebalance::leastsq, 34  
**chemistry.py**, 824  
**chk**  
 forcebalance::optimizer::Optimizer, 651  
**click**  
 forcebalance::nifty, 47  
**close**  
 forcebalance::nifty::LineChunker, 409  
**cnum**  
 forcebalance::qchemio::QCIn\_Reader, 662  
**col**  
 forcebalance::nifty, 47  
**commadash**  
 forcebalance::nifty, 47  
**comments**  
 forcebalance::Mol2::mol2, 527  
 forcebalance::Mol2::mol2\_set, 538  
**comms**  
 forcebalance::molecule::Molecule, 558  
**compounds**  
 forcebalance::Mol2::mol2\_set, 538  
**compute**  
 forcebalance::objective::Penalty, 654  
**compute\_mass**  
 forcebalance::openmmio::OpenMM, 629  
**compute\_netforce\_torque**

forcebalance::abinitio::AbInitio, 80  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
 forcebalance::amberio::AbInitio\_AMBER, 102  
 forcebalance::gmxio::AbInitio\_GMX, 124  
 forcebalance::openmmio::AbInitio\_OpenMM, 168  
 forcebalance::tinkerio::AbInitio\_TINKER, 190  
 compute\_volume  
     forcebalance::openmmio::OpenMM, 629  
 ComputeOverlap  
     forcebalance::molecule, 39  
 concurrent\_map  
     forcebalance::nifty, 47  
 ConjugateGradient  
     forcebalance::optimizer::Optimizer, 643  
 contact.py, 824  
 contraction\_number  
     forcebalance::psi4io::GBS\_Reader, 313  
 coords  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 158  
     forcebalance::amberio::AbInitio\_AMBER, 114  
 CopyAmoebaAngleParameters  
     forcebalance::openmmio, 59  
 CopyAmoebaBondParameters  
     forcebalance::openmmio, 59  
 CopyAmoebaInPlaneAngleParameters  
     forcebalance::openmmio, 59  
 CopyAmoebaMultipoleParameters  
     forcebalance::openmmio, 59  
 CopyAmoebaOutOfPlaneBendParameters  
     forcebalance::openmmio, 59  
 CopyAmoebaVdwParameters  
     forcebalance::openmmio, 59  
 CopyFile  
     forcebalance::nifty, 47  
 CopyHarmonicAngleParameters  
     forcebalance::openmmio, 59  
 CopyHarmonicBondParameters  
     forcebalance::openmmio, 60  
 CopyNonbondedParameters  
     forcebalance::openmmio, 60  
 CopyPeriodicTorsionParameters  
     forcebalance::openmmio, 60  
 CopySystemParameters  
     forcebalance::openmmio, 60  
 count\_assignment  
     forcebalance::vibration, 74  
 Counter  
     forcebalance::optimizer, 62  
 counterpoise.py, 825  
 cpqm  
     forcebalance::counterpoise::Counterpoise, 289  
 cotypes  
     forcebalance::custom\_io, 23  
 create\_mvals

forcebalance::forcefield::FF, 299  
 create\_pvals  
     forcebalance::forcefield::FF, 300  
 create\_simulation  
     forcebalance::openmmio::OpenMM, 629  
 createWorkQueue  
     forcebalance::nifty, 48  
 CubicLattice  
     forcebalance::molecule, 39  
 custom\_io.py, 825

D

forcebalance::leastsq::LeastSquares, 407  
 forcebalance::psi4io::THCDF\_Psi4, 730

DATfnm

forcebalance::psi4io::THCDF\_Psi4, 730

DF\_Energy

forcebalance::psi4io::THCDF\_Psi4, 730

Data

forcebalance::molecule::Molecule, 558

data

forcebalance::Mol2, 36  
 forcebalance::thermo::Point, 659

data\_from\_web

forcebalance::chemistry, 20

defaultHandler

forcebalance::output::ForceBalanceLogger, 307

denoms

forcebalance::gmxio::Moments\_GMX, 588  
 forcebalance::gmxio::Thermo\_GMX, 762  
 forcebalance::moments::Moments, 573  
 forcebalance::openmmio::Moments\_OpenMM, 603  
 forcebalance::thermo::Thermo, 745  
 forcebalance::tinkerio::Moments\_TINKER, 618

destroy

forcebalance::psi4io::GBS\_Reader, 313

destroyWorkQueue

forcebalance::nifty, 48

determine\_fftype

forcebalance::forcefield, 28

dftypes

forcebalance::gmxio, 32

diff

forcebalance::molecule, 39

dihe

forcebalance::amberio::FrcMod\_Reader, 310

divisor

forcebalance::gmxio::Interaction\_GMX, 358  
 forcebalance::interaction::Interaction, 343  
 forcebalance::openmmio::Interaction\_OpenMM, 373  
 forcebalance::tinkerio::Interaction\_TINKER, 388

do\_nothing

forcebalance::openmmio, 60

do\_self\_pol

forcebalance::gmxio::Lipid\_GMX, 444  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::lipid::Lipid, 426  
 forcebalance::liquid::Liquid, 462  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**double**  
 forcebalance::gmxio::GMX, 326  
**driver**  
 forcebalance::psi4io::RDVR3\_Psi4, 675  
 forcebalance::psi4io::THCDF\_Psi4, 719  
**dx**  
 forcebalance::optimizer::Optimizer, 652  
**DynDict**  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**DynDict\_New**  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**e**  
 forcebalance::molecule::MyG, 621  
**e\_err**  
 forcebalance::abinitio::AbInitio, 92  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 158  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 136  
 forcebalance::gmxio::Interaction\_GMX, 358  
 forcebalance::interaction::Interaction, 343  
 forcebalance::openmmio::AbInitio\_OpenMM, 180  
 forcebalance::openmmio::Interaction\_OpenMM, 373  
 forcebalance::tinkerio::AbInitio\_TINKER, 202  
 forcebalance::tinkerio::Interaction\_TINKER, 388  
**e\_err\_pct**  
 forcebalance::abinitio::AbInitio, 92  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 136  
 forcebalance::gmxio::Interaction\_GMX, 358  
 forcebalance::interaction::Interaction, 343  
 forcebalance::openmmio::AbInitio\_OpenMM, 180  
 forcebalance::openmmio::Interaction\_OpenMM, 373  
 forcebalance::tinkerio::AbInitio\_TINKER, 202  
 forcebalance::tinkerio::Interaction\_TINKER, 388  
**e\_ref**  
 forcebalance::abinitio::AbInitio, 92  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 136  
 forcebalance::openmmio::AbInitio\_OpenMM, 180  
 forcebalance::tinkerio::AbInitio\_TINKER, 202  
**eckeys**  
 forcebalance::tinkerio, 73  
**edit\_qcrems**  
 forcebalance::molecule::Molecule, 546  
**ef**

forcebalance::molecule::MyG, 621  
**either**  
 forcebalance::molecule, 39  
**elem\_from\_atomname**  
 forcebalance::molecule, 39  
**element**  
 forcebalance::psi4io::GBS\_Reader, 313  
 forcebalance::psi4io::Grid\_Reader, 329  
**Elements**  
 forcebalance::chemistry, 20  
 forcebalance::molecule, 42  
 forcebalance::psi4io::THCDF\_Psi4, 730  
 forcebalance::PT, 68  
**elements**  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
**emd0**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::openmmio::AbInitio\_OpenMM, 181  
 forcebalance::tinkerio::AbInitio\_TINKER, 203  
**emit**  
 forcebalance::output::CleanFileHandler, 272  
 forcebalance::output::CleanStreamHandler, 274  
 forcebalance::output::RawFileHandler, 669  
 forcebalance::output::RawStreamHandler, 671  
**emm**  
 forcebalance::gmxio::Interaction\_GMX, 358  
 forcebalance::interaction::Interaction, 343  
 forcebalance::openmmio::Interaction\_OpenMM, 373  
 forcebalance::tinkerio::Interaction\_TINKER, 388  
**encode**  
 forcebalance::nifty, 48  
**energy**  
 forcebalance::gmxio::GMX, 320  
 forcebalance::openmmio::OpenMM, 629  
 forcebalance::tinkerio::TINKER, 767  
**energy\_all**  
 forcebalance::abinitio::AbInitio, 80  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
 forcebalance::amberio::AbInitio\_AMBER, 102  
 forcebalance::gmxio::AbInitio\_GMX, 125  
 forcebalance::openmmio::AbInitio\_OpenMM, 169  
 forcebalance::tinkerio::AbInitio\_TINKER, 191  
**energy\_components**  
 forcebalance::openmmio, 60  
**energy\_derivatives**  
 forcebalance::quantity, 70  
**energy\_dipole**  
 forcebalance::gmxio::GMX, 320  
 forcebalance::openmmio::OpenMM, 630  
 forcebalance::tinkerio::TINKER, 768  
**energy\_force**

forcebalance::gmxio::GMX, 320  
 forcebalance::openmmio::OpenMM, 630  
 forcebalance::tinkerio::TINKER, 768  
**energy\_force\_all**  
     forcebalance::abinitio::AbInitio, 80  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
     forcebalance::amberio::AbInitio\_AMBER, 102  
     forcebalance::gmxio::AbInitio\_GMX, 125  
     forcebalance::openmmio::AbInitio\_OpenMM, 169  
     forcebalance::tinkerio::AbInitio\_TINKER, 191  
**energy\_force\_driver\_all**  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
     forcebalance::amberio::AbInitio\_AMBER, 102  
**energy\_force\_driver\_all\_external**  
     forcebalance::amberio::AbInitio\_AMBER, 103  
**energy\_force\_one**  
     forcebalance::abinitio::AbInitio, 80  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
     forcebalance::amberio::AbInitio\_AMBER, 103  
     forcebalance::gmxio::AbInitio\_GMX, 125  
     forcebalance::gmxio::GMX, 320  
     forcebalance::openmmio::AbInitio\_OpenMM, 169  
     forcebalance::openmmio::OpenMM, 630  
     forcebalance::tinkerio::AbInitio\_TINKER, 191  
     forcebalance::tinkerio::TINKER, 768  
**energy\_force\_transform**  
     forcebalance::abinitio::AbInitio, 80  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
     forcebalance::amberio::AbInitio\_AMBER, 103  
     forcebalance::gmxio::AbInitio\_GMX, 125  
     forcebalance::openmmio::AbInitio\_OpenMM, 169  
     forcebalance::tinkerio::AbInitio\_TINKER, 191  
**energy\_force\_transform\_one**  
     forcebalance::abinitio::AbInitio, 81  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
     forcebalance::amberio::AbInitio\_AMBER, 103  
     forcebalance::gmxio::AbInitio\_GMX, 125  
     forcebalance::openmmio::AbInitio\_OpenMM, 169  
     forcebalance::tinkerio::AbInitio\_TINKER, 191  
**energy\_one**  
     forcebalance::abinitio::AbInitio, 81  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 148  
     forcebalance::amberio::AbInitio\_AMBER, 104  
     forcebalance::gmxio::AbInitio\_GMX, 126  
     forcebalance::gmxio::GMX, 321  
     forcebalance::openmmio::AbInitio\_OpenMM, 170  
     forcebalance::openmmio::OpenMM, 631  
     forcebalance::tinkerio::AbInitio\_TINKER, 192  
**energy\_part**  
     forcebalance::binding::BindingEnergy, 225  
     forcebalance::gmxio::BindingEnergy\_GMX, 240  
     forcebalance::openmmio::BindingEnergy\_OpenMM,  
         255  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 270  
**energy\_rmsd**  
     forcebalance::gmxio::GMX, 321  
     forcebalance::openmmio::OpenMM, 631  
     forcebalance::tinkerio::TINKER, 769  
**energy\_termnames**  
     forcebalance::gmxio::GMX, 321  
**engine**  
     forcebalance::abinitio::AbInitio, 93  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
     forcebalance::amberio::AbInitio\_AMBER, 115  
     forcebalance::gmxio::AbInitio\_GMX, 137  
     forcebalance::gmxio::Interaction\_GMX, 358  
     forcebalance::gmxio::Moments\_GMX, 588  
     forcebalance::gmxio::Vibration\_GMX, 805  
     forcebalance::interaction::Interaction, 343  
     forcebalance::moments::Moments, 573  
     forcebalance::openmmio::AbInitio\_OpenMM, 181  
     forcebalance::openmmio::Interaction\_OpenMM, 373  
     forcebalance::openmmio::Moments\_OpenMM, 603  
     forcebalance::tinkerio::AbInitio\_TINKER, 203  
     forcebalance::tinkerio::Interaction\_TINKER, 388  
     forcebalance::tinkerio::Moments\_TINKER, 618  
     forcebalance::tinkerio::Vibration\_TINKER, 820  
     forcebalance::vibration::Vibration, 790  
**engine.py**, 825  
**engine**  
     forcebalance::gmxio::AbInitio\_GMX, 137  
     forcebalance::gmxio::BindingEnergy\_GMX, 240  
     forcebalance::gmxio::Interaction\_GMX, 358  
     forcebalance::gmxio::Lipid\_GMX, 444  
     forcebalance::gmxio::Liquid\_GMX, 481  
     forcebalance::gmxio::Moments\_GMX, 588  
     forcebalance::gmxio::Thermo\_GMX, 762  
     forcebalance::gmxio::Vibration\_GMX, 805  
     forcebalance::openmmio::AbInitio\_OpenMM, 181  
     forcebalance::openmmio::BindingEnergy\_OpenMM,  
         255  
     forcebalance::openmmio::Interaction\_OpenMM, 373  
     forcebalance::openmmio::Liquid\_OpenMM, 500  
     forcebalance::openmmio::Moments\_OpenMM, 603  
     forcebalance::tinkerio::AbInitio\_TINKER, 203  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 270  
     forcebalance::tinkerio::Interaction\_TINKER, 388  
     forcebalance::tinkerio::Liquid\_TINKER, 520  
     forcebalance::tinkerio::Moments\_TINKER, 618  
     forcebalance::tinkerio::Vibration\_TINKER, 820  
**engines**  
     forcebalance::binding::BindingEnergy, 225  
     forcebalance::gmxio::BindingEnergy\_GMX, 240  
     forcebalance::openmmio::BindingEnergy\_OpenMM,  
         255  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 270  
**engname**  
     forcebalance::gmxio::Lipid\_GMX, 444

forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::gmxio::Thermo\_GMX, 762  
 forcebalance::lipid::Lipid, 426  
 forcebalance::liquid::Liquid, 462  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::quantity::Quantity, 664  
 forcebalance::quantity::Quantity\_Density, 666  
 forcebalance::quantity::Quantity\_H\_vap, 668  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**eqcgmx**  
 forcebalance::nifty, 56  
**eqm**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::gmxio::Interaction\_GMX, 358  
 forcebalance::interaction::Interaction, 343  
 forcebalance::openmmio::AbInitio\_OpenMM, 181  
 forcebalance::openmmio::Interaction\_OpenMM, 373  
 forcebalance::tinkerio::AbInitio\_TINKER, 203  
 forcebalance::tinkerio::Interaction\_TINKER, 388  
**error**  
 forcebalance::output::ModLogger, 523  
**esp\_err**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::openmmio::AbInitio\_OpenMM, 181  
 forcebalance::tinkerio::AbInitio\_TINKER, 203  
**espval**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::openmmio::AbInitio\_OpenMM, 181  
 forcebalance::tinkerio::AbInitio\_TINKER, 203  
**espxyz**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::openmmio::AbInitio\_OpenMM, 181  
 forcebalance::tinkerio::AbInitio\_TINKER, 203  
**EulerMatrix**  
 forcebalance::molecule, 40  
**evaluate\_**  
 forcebalance::gmxio::GMX, 321  
 forcebalance::openmmio::OpenMM, 631  
 forcebalance::tinkerio::TINKER, 769  
**evaluate\_one\_**  
 forcebalance::openmmio::OpenMM, 632  
**evaluate\_snapshot**

forcebalance::gmxio::GMX, 322  
**evaluate\_trajectory**  
 forcebalance::gmxio::GMX, 322  
**even\_list**  
 forcebalance::molecule, 40  
**excision**  
 forcebalance::forcefield::FF, 304  
 forcebalance::optimizer::Optimizer, 652  
**extra\_output**  
 forcebalance::gmxio::Lipid\_GMX, 444  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::lipid::Lipid, 426  
 forcebalance::liquid::Liquid, 462  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**extract**  
 forcebalance::quantity::Quantity, 664  
 forcebalance::quantity::Quantity\_Density, 666  
 forcebalance::quantity::Quantity\_H\_vap, 668  
**extract\_int**  
 forcebalance::nifty, 48  
**f12d3p**  
 forcebalance::finite\_difference, 25  
**f12d7p**  
 forcebalance::finite\_difference, 25  
**f1d2p**  
 forcebalance::finite\_difference, 25  
**f1d5p**  
 forcebalance::finite\_difference, 25  
**f1d7p**  
 forcebalance::finite\_difference, 25  
**f2var**  
 forcebalance::finite\_difference, 25  
**f\_err**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::openmmio::AbInitio\_OpenMM, 181  
 forcebalance::tinkerio::AbInitio\_TINKER, 203  
**f\_err\_pct**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::openmmio::AbInitio\_OpenMM, 181  
 forcebalance::tinkerio::AbInitio\_TINKER, 203  
**f\_ref**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::openmmio::AbInitio\_OpenMM, 181

forcebalance::tinkerio::AbInitio\_TINKER, 203  
**FDCheckG**  
 forcebalance::optimizer::Optimizer, 644  
**FDCheckH**  
 forcebalance::optimizer::Optimizer, 644  
**FF**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::binding::BindingEnergy, 225  
 forcebalance::counterpoise::Counterpoise, 289  
 forcebalance::engine::Engine, 292  
 forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::gmxio::BindingEnergy\_GMX, 240  
 forcebalance::gmxio::GMX, 326  
 forcebalance::gmxio::Interaction\_GMX, 358  
 forcebalance::gmxio::Lipid\_GMX, 444  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::gmxio::Moments\_GMX, 588  
 forcebalance::gmxio::Thermo\_GMX, 762  
 forcebalance::gmxio::Vibration\_GMX, 805  
 forcebalance::interaction::Interaction, 343  
 forcebalance::leastsq::LeastSquares, 407  
 forcebalance::lipid::Lipid, 426  
 forcebalance::liquid::Liquid, 462  
 forcebalance::moments::Moments, 573  
 forcebalance::objective::Objective, 625  
 forcebalance::objective::Penalty, 656  
 forcebalance::openmmio::AbInitio\_OpenMM, 181  
 forcebalance::openmmio::BindingEnergy\_OpenMM, 255  
 forcebalance::openmmio::Interaction\_OpenMM, 373  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::openmmio::Moments\_OpenMM, 603  
 forcebalance::openmmio::OpenMM, 635  
 forcebalance::optimizer::Optimizer, 652  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
 forcebalance::psi4io::THCDF\_Psi4, 730  
 forcebalance::target::RemoteTarget, 699  
 forcebalance::target::Target, 714  
 forcebalance::thermo::Thermo, 745  
 forcebalance::tinkerio::AbInitio\_TINKER, 203  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 270  
 forcebalance::tinkerio::Interaction\_TINKER, 388  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
 forcebalance::tinkerio::Moments\_TINKER, 618  
 forcebalance::tinkerio::TINKER, 772  
 forcebalance::tinkerio::Vibration\_TINKER, 820  
 forcebalance::vibration::Vibration, 790  
**FF\_Extensions**  
 forcebalance::forcefield, 29  
**FF\_IOModules**  
 forcebalance::forcefield, 29  
**FFAtomTypes**

forcebalance::forcefield::FF, 304  
**FFMolecules**  
 forcebalance::forcefield::FF, 304  
**FUSE**  
 forcebalance::objective::Penalty, 654  
**FUSE\_BARRIER**  
 forcebalance::objective::Penalty, 655  
**FUSE\_L0**  
 forcebalance::objective::Penalty, 655  
**factor**  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
**fadd**  
 forcebalance::objective::Penalty, 656  
**failmsg**  
 forcebalance::optimizer::Optimizer, 652  
**fdict**  
 forcebalance::custom\_io, 23  
 forcebalance::gmxio, 32  
**fdwrap**  
 forcebalance::finite\_difference, 25  
**fdwrap\_G**  
 forcebalance::finite\_difference, 26  
**fdwrap\_H**  
 forcebalance::finite\_difference, 26  
**feed**  
 forcebalance::amberio::FrcMod\_Reader, 309  
 forcebalance::amberio::Mol2\_Reader, 537  
 forcebalance::BaseReader, 211  
 forcebalance::custom\_io::Gen\_Reader, 315  
 forcebalance::gmxio::ITP\_Reader, 393  
 forcebalance::mol2io::Mol2\_Reader, 534  
 forcebalance::openmmio::OpenMM\_Reader, 638  
 forcebalance::psi4io::GBS\_Reader, 312  
 forcebalance::psi4io::Grid\_Reader, 329  
 forcebalance::qchemio::QCIn\_Reader, 661  
 forcebalance::tinkerio::Tinker\_Reader, 775  
**ffdata**  
 forcebalance::forcefield::FF, 304  
**ffdata\_isxml**  
 forcebalance::forcefield::FF, 304  
**ffxml**  
 forcebalance::openmmio::OpenMM, 635  
**find\_angles**  
 forcebalance::molecule::Molecule, 546  
**find\_dihedrals**  
 forcebalance::molecule::Molecule, 547  
**find\_spacings**  
 forcebalance::forcefield::FF, 300  
**finite\_difference.py**, 826  
**First**  
 forcebalance::optimizer, 62  
**fitatoms**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159

forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::openmmio::AbInitio\_OpenMM, 181  
 forcebalance::tinkerio::AbInitio\_TINKER, 203  
 flat  
     forcebalance::nifty, 48  
 floatornan  
     forcebalance::nifty, 48  
 fmul  
     forcebalance::objective::Penalty, 656  
 force  
     forcebalance::abinitio::AbInitio, 93  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
     forcebalance::amberio::AbInitio\_AMBER, 115  
     forcebalance::gmxio::AbInitio\_GMX, 137  
     forcebalance::openmmio::AbInitio\_OpenMM, 181  
     forcebalance::tinkerio::AbInitio\_TINKER, 203  
 force\_map  
     forcebalance::abinitio::AbInitio, 93  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
     forcebalance::amberio::AbInitio\_AMBER, 115  
     forcebalance::gmxio::AbInitio\_GMX, 137  
     forcebalance::openmmio::AbInitio\_OpenMM, 181  
     forcebalance::tinkerio::AbInitio\_TINKER, 203  
 forcebalance, 14  
     .\_version..., 15  
 forcebalance.abinitio, 16  
 forcebalance.abinitio.AbInitio, 75  
 forcebalance.abinitio\_internal, 16  
 forcebalance.abinitio\_internal.AbInitio\_Internal, 140  
 forcebalance.amberio, 17  
 forcebalance.amberio.AbInitio\_AMBER, 96  
 forcebalance.amberio.FrcMod\_Reader, 307  
 forcebalance.amberio.Mol2\_Reader, 535  
 forcebalance.BaseClass, 207  
 forcebalance.BaseReader, 209  
 forcebalance.binding, 18  
 forcebalance.binding.BindingEnergy, 212  
 forcebalance.chemistry, 19  
 forcebalance.contact, 21  
 forcebalance.counterpoise, 22  
 forcebalance.counterpoise.Counterpoise, 274  
 forcebalance.custom\_io, 23  
 forcebalance.custom\_io.Gen\_Reader, 314  
 forcebalance.engine, 24  
 forcebalance.engine.Engine, 290  
 forcebalance.finite\_difference, 24  
 forcebalance.forcefield, 27  
 forcebalance.forcefield.BackedUpDict, 206  
 forcebalance.forcefield.FF, 293  
 forcebalance.gmxio, 29  
 forcebalance.gmxio.AbInitio\_GMX, 118  
 forcebalance.gmxio.BindingEnergy\_GMX, 226  
 forcebalance.gmxio.GMX, 316  
 forcebalance.gmxio.ITP\_Reader, 390  
 forcebalance.gmxio.Interaction\_GMX, 345  
 forcebalance.gmxio.Lipid\_GMX, 429  
 forcebalance.gmxio.Liquid\_GMX, 464  
 forcebalance.gmxio.Moments\_GMX, 574  
 forcebalance.gmxio.Thermo\_GMX, 747  
 forcebalance.gmxio.Vibration\_GMX, 792  
 forcebalance.interaction, 33  
 forcebalance.interaction.Interaction, 330  
 forcebalance.leastsq, 34  
 forcebalance.leastsq.LeastSquares, 394  
 forcebalance.lipid, 34  
 forcebalance.lipid.Lipid, 410  
 forcebalance.liquid, 35  
 forcebalance.liquid.Liquid, 447  
 forcebalance.Mol2, 36  
 forcebalance.Mol2.mol2, 523  
 forcebalance.Mol2.mol2\_atom, 527  
 forcebalance.Mol2.mol2\_bond, 530  
 forcebalance.Mol2.mol2\_set, 538  
 forcebalance.mol2io, 36  
 forcebalance.mol2io.Mol2\_Reader, 533  
 forcebalance.molecule, 37  
 forcebalance.molecule.Molecule, 539  
 forcebalance.molecule.MolfileTimestep, 559  
 forcebalance.molecule.MyG, 620  
 forcebalance.moments, 44  
 forcebalance.moments.Moments, 560  
 forcebalance.nifty, 44  
 forcebalance.nifty.LineChunker, 408  
 forcebalance.nifty.Pickler\_LP, 656  
 forcebalance.nifty.Unpickler\_LP, 776  
 forcebalance.objective, 57  
 forcebalance.objective.Objective, 622  
 forcebalance.objective.Penalty, 653  
 forcebalance.openmmio, 58  
 forcebalance.openmmio.AbInitio\_OpenMM, 162  
 forcebalance.openmmio.BindingEnergy\_OpenMM, 241  
 forcebalance.openmmio.Interaction\_OpenMM, 360  
 forcebalance.openmmio.Liquid\_OpenMM, 483  
 forcebalance.openmmio.Moments\_OpenMM, 589  
 forcebalance.openmmio.OpenMM, 626  
 forcebalance.openmmio.OpenMM\_Reader, 637  
 forcebalance.optimizer, 62  
 forcebalance.optimizer.Optimizer, 639  
 forcebalance.output, 63  
 forcebalance.output.CleanFileHandler, 271  
 forcebalance.output.CleanStreamHandler, 273  
 forcebalance.output.ForceBalanceLogger, 306  
 forcebalance.output.ModLogger, 522  
 forcebalance.output.RawFileHandler, 669  
 forcebalance.output.RawStreamHandler, 670  
 forcebalance.PT, 68  
 forcebalance.parser, 63

forcebalance.psi4io, 67  
 forcebalance.psi4io.GBS\_Reader, 310  
 forcebalance.psi4io.Grid\_Reader, 327  
 forcebalance.psi4io.RDVR3\_Psi4, 671  
 forcebalance.psi4io.THCDF\_Psi4, 715  
 forcebalance.qchemio, 68  
 forcebalance.qchemio.QCIn\_Reader, 659  
 forcebalance.quantity, 70  
 forcebalance.quantity.Quantity, 662  
 forcebalance.quantity.Quantity\_Density, 665  
 forcebalance.quantity.Quantity\_H\_vap, 667  
 forcebalance.target, 71  
 forcebalance.target.RemoteTarget, 687  
 forcebalance.target.Target, 700  
 forcebalance.thermo, 71  
 forcebalance.thermo.Point, 658  
 forcebalance.thermo.Thermo, 732  
 forcebalance.tinkerio, 71  
 forcebalance.tinkerio.ABInitio\_TINKER, 184  
 forcebalance.tinkerio.BindingEnergy\_TINKER, 256  
 forcebalance.tinkerio.Interaction\_TINKER, 375  
 forcebalance.tinkerio.Liquid\_TINKER, 502  
 forcebalance.tinkerio.Moments\_TINKER, 604  
 forcebalance.tinkerio.TINKER, 764  
 forcebalance.tinkerio.Tinker\_Reader, 773  
 forcebalance.tinkerio.Vibration\_TINKER, 807  
 forcebalance.vibration, 74  
 forcebalance.vibration.Vibration, 777  
 forcebalance::BaseClass  
     \_\_init\_\_, 209  
     \_\_setattr\_\_, 209  
     PrintOptionDict, 209  
     set\_option, 209  
     verbose\_options, 209  
 forcebalance::BaseReader  
     \_\_init\_\_, 211  
     adict, 212  
     AtomTypes, 212  
     build\_pid, 211  
     feed, 211  
     itype, 212  
     In, 212  
     molatom, 212  
     Molecules, 212  
     pdict, 212  
     Split, 211  
     suffix, 212  
     Whites, 211  
 forcebalance::Mol2  
     data, 36  
 forcebalance::Mol2::mol2  
     \_\_init\_\_, 524  
     \_\_repr\_\_, 524  
     atoms, 527  
     bonds, 527  
     charge\_type, 527  
     comments, 527  
     get\_atom, 524  
     get\_bonded\_atoms, 525  
     mol\_name, 527  
     mol\_type, 527  
     num\_atoms, 527  
     num\_bonds, 527  
     num\_feat, 527  
     num\_sets, 527  
     num\_subst, 527  
     out, 525  
     parse, 525  
     set\_charge\_type, 526  
     set\_donor\_acceptor\_atoms, 526  
     set\_mol\_name, 526  
     set\_mol\_type, 527  
     set\_num\_atoms, 527  
     set\_num\_bonds, 527  
     set\_num\_feat, 527  
     set\_num\_sets, 527  
     set\_num\_subst, 527  
 forcebalance::Mol2::mol2\_atom  
     \_\_init\_\_, 528  
     \_\_repr\_\_, 528  
     atom\_id, 530  
     atom\_name, 530  
     atom\_type, 530  
     charge, 530  
     parse, 528  
     set\_atom\_id, 529  
     set\_atom\_name, 529  
     set\_atom\_type, 529  
     set\_charge, 529  
     set\_crds, 530  
     set\_status\_bit, 530  
     set\_subst\_id, 530  
     set\_subst\_name, 530  
     status\_bit, 530  
     subst\_id, 530  
     subst\_name, 530  
     x, 530  
     y, 530  
     z, 530  
 forcebalance::Mol2::mol2\_bond  
     \_\_init\_\_, 531  
     \_\_repr\_\_, 531  
     bond\_id, 532  
     bond\_type, 532  
     origin\_atom\_id, 532  
     parse, 531  
     set\_bond\_id, 532  
     set\_bond\_type, 532

```

set_origin_atom_id, 532
set_status_bit, 532
set_target_atom_id, 532
status_bit, 532
target_atom_id, 532
forcebalance::Mol2::mol2_set
    __init__, 538
    comments, 538
    compounds, 538
    num_compounds, 538
    parse, 538
forcebalance::PT
    Elements, 68
    PeriodicTable, 68
forcebalance::abinitio
    build_objective, 16
    logger, 16
    weighted_variance, 16
    weighted_variance2, 16
forcebalance::abinitio::AbInitio
    __init__, 79
    __setattr__, 79
    absrd, 79
    AtomLists, 92
    AtomMask, 92
    boltz_wts, 92
    build_invdist, 80
    check_files, 80
    compute_netforce_torque, 80
    e_err, 92
    e_err_pct, 92
    e_ref, 92
    emd0, 93
    energy_all, 80
    energy_force_all, 80
    energy_force_one, 80
    energy_force_transform, 80
    energy_force_transform_one, 81
    energy_one, 81
    engine, 93
    eqm, 93
    esp_err, 93
    espval, 93
    espxyz, 93
    f_err, 93
    f_err_pct, 93
    f_ref, 93
    FF, 93
    fitatoms, 93
    force, 93
    force_map, 93
    fqm, 93
    fref, 93
    gct, 93
    get, 81
    get_G, 83
    get_H, 84
    get_X, 85
    get_energy_force, 82
    get_resp, 85
    hct, 93
    indicate, 86
    invdists, 93
    link_from_tempdir, 86
    maxrd, 87
    meta_get, 87
    meta_indicate, 88
    mol, 93
    nesp, 94
    new_vsites, 94
    nf_err, 94
    nf_err_pct, 94
    nf_ref, 94
    nftqm, 94
    nnf, 94
    nparticles, 94
    ns, 94
    ntq, 94
    objective, 94
    pgrad, 94
    PrintOptionDict, 94
    printcool_table, 89
    qfnm, 94
    qmatoms, 94
    qmboltz_wts, 94
    rd, 94
    read, 89
    read_0grads, 90
    read_indicate, 95
    read_objective, 95
    read_reference_data, 90
    refresh_temp_directory, 91
    respterm, 95
    rundir, 95
    save_vmvalls, 95
    set_option, 91
    stage, 91
    submit_jobs, 91
    tempbase, 95
    tempdir, 95
    tq_err, 95
    tq_err_pct, 95
    tq_ref, 95
    use_nft, 95
    verbose_options, 95
    w_energy, 95
    w_force, 95
    w_netforce, 96

```

w\_resp, 96  
 w\_torque, 96  
 whamboltz, 96  
 wq\_complete, 92  
 write\_0grads, 92  
 write\_indicate, 96  
 write\_objective, 96  
 xct, 96  
**forcebalance::abinitio\_internal::AbInitio\_Internal**  
 \_\_init\_\_, 146  
 \_\_setattr\_\_, 146  
 absrd, 146  
 AtomLists, 158  
 AtomMask, 158  
 boltz\_wts, 158  
 build\_invdist, 146  
 check\_files, 146  
 compute\_netforce\_torque, 146  
 coords, 158  
 e\_err, 158  
 e\_err\_pct, 159  
 e\_ref, 159  
 emd0, 159  
 energy\_all, 146  
 energy\_force\_all, 147  
 energy\_force\_driver\_all, 147  
 energy\_force\_one, 147  
 energy\_force\_transform, 147  
 energy\_force\_transform\_one, 147  
 energy\_one, 148  
 engine, 159  
 eqm, 159  
 esp\_err, 159  
 espval, 159  
 espxyz, 159  
 f\_err, 159  
 f\_err\_pct, 159  
 f\_ref, 159  
 FF, 159  
 fitatoms, 159  
 force, 159  
 force\_map, 159  
 fqm, 159  
 fref, 159  
 gct, 159  
 get, 148  
 get\_G, 149  
 get\_H, 150  
 get\_X, 151  
 get\_energy\_force, 148  
 get\_resp, 151  
 hct, 160  
 indicate, 152  
 invdists, 160  
 link\_from\_tempdir, 152  
 maxrd, 153  
 meta\_get, 153  
 meta\_indicate, 154  
 mol, 160  
 nesp, 160  
 new\_vsites, 160  
 nf\_err, 160  
 nf\_err\_pct, 160  
 nf\_ref, 160  
 nftqm, 160  
 nmf, 160  
 nparticles, 160  
 ns, 160  
 ntq, 160  
 objective, 160  
 pgrad, 160  
 PrintOptionDict, 160  
 printcool\_table, 155  
 qfnm, 160  
 qmatoms, 160  
 qmboltz\_wts, 160  
 rd, 161  
 read, 155  
 read\_0grads, 156  
 read\_indicate, 161  
 read\_objective, 161  
 read\_reference\_data, 156  
 refresh\_temp\_directory, 157  
 respterm, 161  
 rundir, 161  
 save\_vmvalls, 161  
 set\_option, 157  
 stage, 157  
 submit\_jobs, 157  
 tempbase, 161  
 tempdir, 161  
 tq\_err, 161  
 tq\_err\_pct, 161  
 tq\_ref, 161  
 use\_nft, 161  
 verbose\_options, 161  
 w\_energy, 161  
 w\_force, 162  
 w\_netforce, 162  
 w\_resp, 162  
 w\_torque, 162  
 whamboltz, 162  
 wq\_complete, 158  
 write\_0grads, 158  
 write\_indicate, 162  
 write\_objective, 162  
 xct, 162  
**forcebalance::amberio**

```

frcmod_pdct, 18
is_mol2_atom, 18
logger, 18
mol2_pdct, 18
forcebalance::amberio::AbInitio_AMBER
    __init__, 102
    __setattr__, 102
    absrd, 102
    all_at_once, 114
    AtomLists, 114
    AtomMask, 114
    boltz_wts, 114
    build_invdist, 102
    check_files, 102
    compute_netforce_torque, 102
    coords, 114
    e_err, 115
    e_err_pct, 115
    e_ref, 115
    emd0, 115
    energy_all, 102
    energy_force_all, 102
    energy_force_driver_all, 102
    energy_force_one, 103
    energy_force_transform, 103
    energy_force_transform_one, 103
    energy_one, 104
    engine, 115
    eqm, 115
    esp_err, 115
    espval, 115
    espxyz, 115
    f_err, 115
    f_err_pct, 115
    f_ref, 115
    FF, 115
    fitatoms, 115
    force, 115
    force_map, 115
    fqm, 115
    fref, 115
    gct, 116
    get, 104
    get_G, 105
    get_H, 106
    get_X, 107
    get_energy_force, 104
    get_resp, 107
    hct, 116
    indicate, 108
    invdists, 116
    link_from_tempdir, 108
    maxrd, 109
    meta_get, 109
    meta_indicate, 110
    mol, 116
    nesp, 116
    new_vsites, 116
    nf_err, 116
    nf_err_pct, 116
    nf_ref, 116
    nftqm, 116
    nnf, 116
    nparticles, 116
    ns, 116
    ntq, 116
    objective, 116
    pgrad, 116
    prepare_temp_directory, 111
    PrintOptionDict, 116
    printcool_table, 111
    qfnm, 116
    qmatoms, 116
    qmboltz_wts, 117
    rd, 117
    read, 111
    read_0grads, 112
    read_indicate, 117
    read_objective, 117
    read_reference_data, 112
    refresh_temp_directory, 113
    respterm, 117
    rundir, 117
    save_vmvalls, 117
    set_option, 113
    stage, 113
    submit_jobs, 114
    tempbase, 117
    tempdir, 117
    tq_err, 117
    tq_err_pct, 117
    tq_ref, 117
    use_nft, 117
    verbose_options, 118
    w_energy, 118
    w_force, 118
    w_netforce, 118
    w_resp, 118
    w_torque, 118
    whamboltz, 118
    wq_complete, 114
    write_0grads, 114
    write_indicate, 118
    write_objective, 118
    xct, 118
forcebalance::amberio::FrcMod_Reader
    __init__, 309
    adict, 310

```

atom, 310  
 AtomTypes, 310  
 build\_pid, 309  
 dihe, 310  
 feed, 309  
 itype, 310  
 In, 310  
 molatom, 310  
 Molecules, 310  
 pdict, 310  
 Split, 309  
 suffix, 310  
 Whites, 310  
**forcebalance::amberio::Mol2\_Reader**  
 \_\_init\_\_, 537  
 adict, 537  
 atom, 537  
 AtomTypes, 537  
 atomnames, 537  
 build\_pid, 537  
 feed, 537  
 itype, 537  
 In, 537  
 mol, 537  
 molatom, 537  
 Molecules, 537  
 pdict, 537  
 section, 538  
 Split, 537  
 suffix, 538  
 Whites, 537  
**forcebalance::binding**  
 logger, 19  
 parse\_interactions, 19  
**forcebalance::binding::BindingEnergy**  
 \_\_init\_\_, 215  
 \_\_setattr\_\_, 215  
 absrd, 215  
 check\_files, 216  
 energy\_part, 225  
 engines, 225  
 FF, 225  
 gct, 225  
 get, 216  
 get\_G, 216  
 get\_H, 217  
 get\_X, 218  
 hct, 225  
 indicate, 219  
 inter\_opts, 225  
 link\_from\_tempdir, 219  
 maxrd, 220  
 meta\_get, 220  
 meta\_indicate, 221  
 objective, 225  
 pgrad, 225  
 PrintDict, 225  
 PrintOptionDict, 225  
 printcool\_table, 222  
 RMSDDict, 225  
 rd, 225  
 read, 222  
 read\_0grads, 223  
 read\_indicate, 225  
 read\_objective, 225  
 refresh\_temp\_directory, 223  
 rmsd\_part, 225  
 rundir, 226  
 set\_option, 223  
 stage, 223  
 submit\_jobs, 224  
 system\_driver, 224  
 tempbase, 226  
 tempdir, 226  
 verbose\_options, 226  
 wq\_complete, 224  
 write\_0grads, 224  
 write\_indicate, 226  
 write\_objective, 226  
 xct, 226  
**forcebalance::chemistry**  
 A, 20  
 atoms, 20  
 B, 20  
 BE, 20  
 bo, 20  
 BondChars, 20  
 BondEnergies, 20  
 BondStrengthByLength, 20  
 data\_from\_web, 20  
 Elements, 20  
 L, 20  
 line, 20  
 LookupByMass, 20  
 PeriodicTable, 20  
 Radii, 21  
**forcebalance::contact**  
 atom\_distances, 21  
 residue\_distances, 22  
**forcebalance::counterpoise**  
 logger, 22  
**forcebalance::counterpoise::Counterpoise**  
 \_\_init\_\_, 277  
 \_\_setattr\_\_, 277  
 absrd, 277  
 check\_files, 278  
 cpqm, 289  
 FF, 289

```

gct, 289
get, 278
get_G, 279
get_H, 280
get_X, 281
hct, 289
link_from_tempdir, 282
load_cp, 282
loadxyz, 282
maxrd, 284
meta_get, 284
meta_indicate, 285
na, 289
ns, 289
pgrad, 289
PrintOptionDict, 289
printcool_table, 286
rd, 289
read, 286
read_0grads, 287
read_indicate, 289
read_objective, 289
refresh_temp_directory, 287
rundir, 289
set_option, 287
stage, 287
submit_jobs, 288
tempbase, 289
tempdir, 290
verbose_options, 290
wq_complete, 288
write_0grads, 288
write_indicate, 290
write_objective, 290
xct, 290
xyzs, 290
forcebalance::custom_io
    cotypes, 23
    fdict, 23
    ndtypes, 23
    pdict, 23
forcebalance::custom_io::Gen_Reader
    __init__, 315
    adict, 316
    AtomTypes, 316
    build_pid, 315
    feed, 315
    itype, 316
    In, 316
    molatom, 316
    Molecules, 316
    pdict, 316
    sec, 316
    Split, 316
suffix, 316
Whites, 316
forcebalance::engine
    logger, 24
forcebalance::engine::Engine
    __init__, 292
    __setattr__, 292
    FF, 292
    name, 292
    prepare, 292
    PrintOptionDict, 292
    readsrc, 292
    root, 292
    set_option, 292
    setopts, 292
    srkdir, 292
    target, 292
    tempdir, 292
    verbose, 292
    verbose_options, 292
forcebalance::finite_difference
    f12d3p, 25
    f12d7p, 25
    f1d2p, 25
    f1d5p, 25
    f1d7p, 25
    f2var, 25
    fdwrap, 25
    fdwrap_G, 26
    fdwrap_H, 26
    in_fd, 26
    in_fd_srch, 27
    logger, 27
forcebalance::forcefield
    determine_fftype, 28
    FF_Extensions, 29
    FF_IOModules, 29
    logger, 29
    rs_override, 28
forcebalance::forcefield::BackedUpDict
    __init__, 207
    __missing__, 207
    backup_dict, 207
forcebalance::forcefield::FF
    __eq__, 296
    __init__, 295
    __setattr__, 296
    addff, 296
    addff_txt, 297
    addff_xml, 298
    assign_field, 299
    assign_p0, 299
    atomnames, 304
    create_mvals, 299

```

```

create_pvls, 300
excision, 304
FFAtomTypes, 304
FFMolecules, 304
ffdata, 304
ffdata_isxml, 304
find_spacings, 300
linedestroy_save, 304
linedestroy_this, 304
list_map, 301
make, 301
make_redirect, 302
map, 304
mktransmat, 302
np, 304
openmmxml, 304
patoms, 304
pfields, 305
plist, 305
print_map, 302
PrintOptionDict, 305
prmdestroy_save, 305
prmdestroy_this, 305
pvls0, 305
qid, 305
qid2, 305
qmap, 305
Readers, 305
redirect, 305
rs, 305
rsmake, 303
set_option, 303
sprint_map, 303
tinkerprm, 305
tm, 305
tml, 305
verbose_options, 305
forcebalance::gmxio
    aftytypes, 32
    bftytypes, 32
    dftytypes, 32
    fdict, 32
    logger, 33
    nftytypes, 33
    parse_atomtype_line, 31
    pdict, 33
    pftypes, 33
    rm_gmx_baks, 31
    write_mdp, 31
    write_ndx, 32
forcebalance::gmxio::AbInitio_GMX
    __init__, 124
    __setattr__, 124
    absrd, 124
    AtomLists, 136
    AtomMask, 136
    boltz_wts, 136
    build_invdist, 124
    check_files, 124
    compute_netforce_torque, 124
    e_err, 136
    e_err_pct, 136
    e_ref, 136
    emd0, 137
    energy_all, 125
    energy_force_all, 125
    energy_force_one, 125
    energy_force_transform, 125
    energy_force_transform_one, 125
    energy_one, 126
    engine, 137
    engine_, 137
    eqm, 137
    esp_err, 137
    espval, 137
    espxyz, 137
    f_err, 137
    f_err_pct, 137
    f_ref, 137
    FF, 137
    fitatoms, 137
    force, 137
    force_map, 137
    fqm, 137
    fref, 137
    gct, 137
    get, 126
    get_G, 127
    get_H, 128
    get_X, 129
    get_energy_force, 126
    get_resp, 129
    hct, 137
    indicate, 130
    invdists, 138
    link_from_tempdir, 130
    maxrd, 131
    meta_get, 131
    meta_indicate, 132
    mol, 138
    nesp, 138
    new_vsites, 138
    nf_err, 138
    nf_err_pct, 138
    nf_ref, 138
    nftqm, 138
    nnf, 138
    nparticles, 138

```

ns, 138  
 ntq, 138  
 objective, 138  
 pgrad, 138  
 PrintOptionDict, 138  
 printcool\_table, 133  
 qfnm, 138  
 qatoms, 138  
 qmboltz\_wts, 138  
 rd, 138  
 read, 133  
 read\_0grads, 134  
 read\_indicate, 139  
 read\_objective, 139  
 read\_reference\_data, 134  
 refresh\_temp\_directory, 135  
 respterm, 139  
 rundir, 139  
 save\_vmvalls, 139  
 set\_option, 135  
 stage, 135  
 submit\_jobs, 135  
 tempbase, 139  
 tempdir, 139  
 tq\_err, 139  
 tq\_err\_pct, 139  
 tq\_ref, 139  
 use\_nft, 139  
 verbose\_options, 139  
 w\_energy, 139  
 w\_force, 139  
 w\_netforce, 140  
 w\_resp, 140  
 w\_torque, 140  
 whamboltz, 140  
 wq\_complete, 136  
 write\_0grads, 136  
 write\_indicate, 140  
 write\_objective, 140  
 xct, 140  
 forcebalance::gmxio::BindingEnergy\_GMX  
     \_\_init\_\_, 230  
     \_\_setattr\_\_, 230  
     absrd, 230  
     check\_files, 231  
     energy\_part, 240  
     engine\_, 240  
     engines, 240  
     FF, 240  
     gct, 240  
     get, 231  
     get\_G, 231  
     get\_H, 232  
     get\_X, 233  
     hct, 240  
     indicate, 234  
     inter\_opts, 240  
     link\_from\_tempdir, 234  
     maxrd, 235  
     meta\_get, 235  
     meta\_indicate, 236  
     objective, 240  
     pgrad, 240  
     PrintDict, 240  
     PrintOptionDict, 240  
     printcool\_table, 237  
     RMSDDict, 241  
     rd, 240  
     read, 237  
     read\_0grads, 238  
     read\_indicate, 240  
     read\_objective, 240  
     refresh\_temp\_directory, 238  
     rmsd\_part, 240  
     rundir, 241  
     set\_option, 238  
     stage, 238  
     submit\_jobs, 239  
     system\_driver, 239  
     tempbase, 241  
     tempdir, 241  
     verbose\_options, 241  
     wq\_complete, 239  
     write\_0grads, 239  
     write\_indicate, 241  
     write\_objective, 241  
     xct, 241  
 forcebalance::gmxio::GMX  
     \_\_init\_\_, 319  
     \_\_setattr\_\_, 319  
     AtomLists, 325  
     AtomMask, 325  
     calc\_scd, 319  
     callgmx, 319  
     double, 326  
     energy, 320  
     energy\_dipole, 320  
     energy\_force, 320  
     energy\_force\_one, 320  
     energy\_one, 321  
     energy\_rmsd, 321  
     energy\_ternnames, 321  
     evaluate\_, 321  
     evaluate\_snapshot, 322  
     evaluate\_trajectory, 322  
     FF, 326  
     generate\_vsites\_positions, 322  
     gmx\_defs, 326

gmxpath, 326  
 gmxsuffix, 326  
 interaction\_energy, 322  
 links, 323  
 md, 323  
 mdene, 326  
 mdp, 326  
 mdtraj, 326  
 mol, 326  
 molecular\_dynamics, 323  
 multipole\_moments, 324  
 n\_snaps, 324  
 name, 326  
 normal\_modes, 324  
 optimize, 324  
 pbc, 326  
 prepare, 325  
 PrintOptionDict, 326  
 readsrc, 325  
 root, 326  
 scd\_persnap, 325  
 set\_option, 325  
 setopts, 325  
 srkdir, 326  
 target, 326  
 tempdir, 326  
 top, 326  
 valkwrd, 326  
 verbose, 327  
 verbose\_options, 327  
 warnmgmx, 325  
 forcebalance::gmxio::ITP\_Reader  
   \_\_init\_\_, 392  
   adict, 393  
   AtomTypes, 393  
   atomnames, 393  
   atomtype\_to\_mass, 393  
   atomtypes, 393  
   build\_pid, 392  
   feed, 393  
   itype, 393  
   In, 393  
   mol, 393  
   molatom, 393  
   Molecules, 393  
   nbtype, 393  
   pdict, 394  
   sec, 394  
   Split, 393  
   suffix, 394  
   Whites, 393  
 forcebalance::gmxio::Interaction\_GMX  
   \_\_init\_\_, 348  
   \_\_setattr\_\_, 349  
                     absrd, 349  
                     check\_files, 349  
                     divisor, 358  
                     e\_err, 358  
                     e\_err\_pct, 358  
                     emm, 358  
                     engine, 358  
                     engine\_, 358  
                     eqm, 358  
                     FF, 358  
                     gct, 358  
                     get, 349  
                     get\_G, 349  
                     get\_H, 350  
                     get\_X, 351  
                     hct, 358  
                     indicate, 352  
                     label, 358  
                     link\_from\_tempdir, 352  
                     maxrd, 353  
                     meta\_get, 353  
                     meta\_indicate, 354  
                     mol, 358  
                     ns, 358  
                     objective, 358  
                     pgrad, 358  
                     prefactor, 358  
                     PrintOptionDict, 358  
                     printcool\_table, 355  
                     qfnm, 358  
                     rd, 359  
                     read, 355  
                     read\_0grads, 356  
                     read\_indicate, 359  
                     read\_objective, 359  
                     read\_reference\_data, 356  
                     refresh\_temp\_directory, 356  
                     rundir, 359  
                     select1, 359  
                     select2, 359  
                     set\_option, 356  
                     stage, 356  
                     submit\_jobs, 357  
                     tempbase, 359  
                     tempdir, 359  
                     verbose\_options, 359  
                     weight, 359  
                     wq\_complete, 357  
                     write\_0grads, 357  
                     write\_indicate, 359  
                     write\_objective, 359  
                     xct, 360  
 forcebalance::gmxio::Lipid\_GMX  
   \_\_init\_\_, 433

\_\_setattr\_\_, 433  
 absrd, 433  
 check\_files, 433  
 do\_self\_pol, 444  
 engine\_, 444  
 engname, 444  
 extra\_output, 444  
 FF, 444  
 gas\_engine, 444  
 gct, 444  
 get, 433  
 get\_G, 434  
 get\_H, 435  
 get\_X, 436  
 Gp, 444  
 hct, 444  
 indicate, 437  
 Labels, 444  
 last\_traj, 444  
 LfDict, 445  
 LfDict\_New, 445  
 link\_from\_tempdir, 437  
 lipid\_mol, 445  
 MBarEnergy, 445  
 maxrd, 438  
 meta\_get, 438  
 meta\_indicate, 439  
 npt\_simulation, 440  
 nptfiles, 445  
 nptpx, 445  
 Objective, 445  
 objective\_term, 440  
 pgrad, 445  
 PhasePoints, 445  
 polarization\_correction, 440  
 Pp, 445  
 prepare\_temp\_directory, 441  
 PrintOptionDict, 445  
 printcool\_table, 441  
 rd, 445  
 read, 441  
 read\_0grads, 442  
 read\_data, 442  
 read\_indicate, 445  
 read\_objective, 445  
 RefData, 445  
 refresh\_temp\_directory, 442  
 rundir, 445  
 SavedMVal, 445  
 SavedTraj, 446  
 scripts, 446  
 set\_option, 442  
 stage, 442  
 submit\_jobs, 443  
 tempbase, 446  
 tempdir, 446  
 verbose\_options, 446  
 w\_al, 446  
 w\_alpha, 446  
 w\_cp, 446  
 w\_eps0, 446  
 w\_kappa, 446  
 w\_rho, 446  
 w\_scd, 446  
 Wp, 446  
 wq\_complete, 443  
 write\_0grads, 444  
 write\_indicate, 446  
 write\_objective, 446  
 xct, 446  
 Xp, 446  
 forcebalance::gmxi::Liquid\_GMX  
 \_\_init\_\_, 468  
 \_\_setattr\_\_, 468  
 absrd, 468  
 AllResults, 481  
 check\_files, 468  
 do\_self\_pol, 481  
 engine\_, 481  
 engname, 481  
 extra\_output, 481  
 FF, 481  
 gas\_engine, 481  
 gas\_engine\_args, 481  
 gas\_mol, 481  
 gct, 481  
 get, 468  
 get\_G, 470  
 get\_H, 471  
 get\_X, 472  
 Gp, 481  
 hct, 481  
 indicate, 473  
 Labels, 481  
 last\_traj, 481  
 LfDict, 481  
 LfDict\_New, 481  
 link\_from\_tempdir, 473  
 liquid\_mol, 481  
 MBarEnergy, 481  
 maxrd, 474  
 meta\_get, 474  
 meta\_indicate, 475  
 npt\_simulation, 476  
 nptfiles, 481  
 nptpx, 481  
 Objective, 481  
 objective\_term, 476

pgrad, 482  
 PhasePoints, 482  
 polarization\_correction, 476  
 Pp, 482  
 prepare\_temp\_directory, 477  
 PrintOptionDict, 482  
 printcool\_table, 477  
 rd, 482  
 read, 477  
 read\_0grads, 478  
 read\_data, 478  
 read\_indicate, 482  
 read\_objective, 482  
 RefData, 482  
 refresh\_temp\_directory, 478  
 rundir, 482  
 SavedTraj, 482  
 scripts, 482  
 set\_option, 478  
 stage, 478  
 submit\_jobs, 479  
 tempbase, 482  
 tempdir, 482  
 verbose\_options, 482  
 w\_alpha, 482  
 w\_cp, 483  
 w\_eps0, 483  
 w\_hvap, 483  
 w\_kappa, 483  
 w\_rho, 483  
 Wp, 483  
 wq\_complete, 480  
 write\_0grads, 480  
 write\_indicate, 483  
 write\_objective, 483  
 xct, 483  
 Xp, 483  
 forcebalance::gmxio::Moments\_GMX  
     \_\_init\_\_, 578  
     \_\_setattr\_\_, 578  
     absrd, 578  
     calc\_moments, 588  
     check\_files, 579  
     denoms, 588  
     engine, 588  
     engine\_, 588  
     FF, 588  
     gct, 588  
     get, 579  
     get\_G, 579  
     get\_H, 580  
     get\_X, 581  
     hct, 588  
     indicate, 582  
     link\_from\_tempdir, 582  
     maxrd, 583  
     meta\_get, 583  
     meta\_indicate, 584  
     mfnm, 588  
     na, 588  
     objective, 588  
     pgrad, 588  
     PrintOptionDict, 588  
     printcool\_table, 585  
     rd, 588  
     read, 585  
     read\_0grads, 586  
     read\_indicate, 588  
     read\_objective, 588  
     read\_reference\_data, 586  
     ref\_eigvals, 589  
     ref\_eigvecs, 589  
     ref\_moments, 589  
     refresh\_temp\_directory, 586  
     rundir, 589  
     set\_option, 586  
     stage, 586  
     submit\_jobs, 587  
     tempbase, 589  
     tempdir, 589  
     unpack\_moments, 587  
     verbose\_options, 589  
     wq\_complete, 587  
     write\_0grads, 587  
     write\_indicate, 589  
     write\_objective, 589  
     xct, 589  
 forcebalance::gmxio::Thermo\_GMX  
     \_\_init\_\_, 751  
     \_\_setattr\_\_, 751  
     absrd, 751  
     check\_files, 752  
     denoms, 762  
     engine\_, 762  
     engname, 762  
     FF, 762  
     gct, 763  
     get, 752  
     get\_G, 753  
     get\_H, 754  
     get\_X, 755  
     Gp, 763  
     hct, 763  
     indicate, 756  
     link\_from\_tempdir, 756  
     maxrd, 757  
     mdpx, 763  
     meta\_get, 757

meta\_indicate, 758  
 Objective, 763  
 objective\_term, 759  
 pgrad, 763  
 points, 763  
 Pp, 763  
 PrintOptionDict, 763  
 printcool\_table, 759  
 rd, 763  
 read, 760  
 read\_0grads, 760  
 read\_indicate, 763  
 read\_objective, 763  
 refresh\_temp\_directory, 760  
 retrieve, 760  
 rundir, 763  
 scripts, 763  
 set\_option, 761  
 simpfx, 764  
 stage, 761  
 submit\_jobs, 761  
 tempbase, 764  
 tempdir, 764  
 verbose\_options, 764  
 weights, 764  
 Wp, 764  
 wq\_complete, 762  
 write\_0grads, 762  
 write\_indicate, 764  
 write\_objective, 764  
 xct, 764  
 Xp, 764  
**forcebalance::gmgio::Vibration\_GMX**  
 \_\_init\_\_, 795  
 \_\_setattr\_\_, 795  
 absrd, 795  
 c2r, 805  
 calc\_eigvals, 805  
 check\_files, 796  
 engine, 805  
 engine., 805  
 FF, 805  
 gct, 805  
 get, 796  
 get\_G, 796  
 get\_H, 797  
 get\_X, 798  
 hct, 805  
 indicate, 799  
 link\_from\_tempdir, 799  
 maxrd, 800  
 meta\_get, 800  
 meta\_indicate, 801  
 na, 805  
 objective, 805  
 overlaps, 805  
 pgrad, 805  
 PrintOptionDict, 806  
 printcool\_table, 802  
 process\_vectors, 802  
 rd, 806  
 read, 803  
 read\_0grads, 803  
 read\_indicate, 806  
 read\_objective, 806  
 read\_reference\_data, 803  
 reassign, 806  
 ref\_eigvals, 806  
 ref\_eigvecs, 806  
 ref\_eigvecs\_nrm\_mw, 806  
 refresh\_temp\_directory, 803  
 rundir, 806  
 set\_option, 803  
 stage, 803  
 submit\_jobs, 804  
 tempbase, 806  
 tempdir, 806  
 verbose\_options, 806  
 vfnm, 806  
 vibration\_driver, 804  
 wq\_complete, 804  
 write\_0grads, 805  
 write\_indicate, 806  
 write\_objective, 807  
 xct, 807  
**forcebalance::interaction**  
 logger, 34  
**forcebalance::interaction::Interaction**  
 \_\_init\_\_, 333  
 \_\_setattr\_\_, 333  
 absrd, 333  
 check\_files, 334  
 divisor, 343  
 e\_err, 343  
 e\_err\_pct, 343  
 emm, 343  
 engine, 343  
 eqm, 343  
 FF, 343  
 gct, 343  
 get, 334  
 get\_G, 334  
 get\_H, 335  
 get\_X, 336  
 hct, 343  
 indicate, 337  
 label, 343  
 link\_from\_tempdir, 337

maxrd, 338  
 meta\_get, 338  
 meta\_indicate, 339  
 mol, 343  
 ns, 343  
 objective, 343  
 pgrad, 343  
 prefactor, 343  
 PrintOptionDict, 343  
 printcool\_table, 340  
 qfnm, 343  
 rd, 343  
 read, 340  
 read\_0grads, 341  
 read\_indicate, 344  
 read\_objective, 344  
 read\_reference\_data, 341  
 refresh\_temp\_directory, 341  
 rundir, 344  
 select1, 344  
 select2, 344  
 set\_option, 341  
 stage, 341  
 submit\_jobs, 342  
 tempbase, 344  
 tempdir, 344  
 verbose\_options, 344  
 weight, 344  
 wq\_complete, 342  
 write\_0grads, 342  
 write\_indicate, 344  
 write\_objective, 344  
 xct, 344  
 forcebalance::leastsq  
     CHECK\_BASIS, 34  
     CheckBasis, 34  
     LAST\_MVALS, 34  
     LastMvals, 34  
     logger, 34  
 forcebalance::leastsq::LeastSquares  
     \_\_init\_\_, 397  
     \_\_setattr\_\_, 397  
     absrd, 397  
     check\_files, 397  
     D, 407  
     FF, 407  
     gct, 407  
     get, 397  
     get\_G, 398  
     get\_H, 399  
     get\_X, 400  
     hct, 407  
     indicate, 401  
     link\_from\_tempdir, 401  
     MAQ, 407  
     maxrd, 401  
     meta\_get, 402  
     meta\_indicate, 403  
     objective, 407  
     pgrad, 407  
     PrintOptionDict, 407  
     printcool\_table, 404  
     rd, 407  
     read, 404  
     read\_0grads, 405  
     read\_indicate, 407  
     read\_objective, 407  
     refresh\_temp\_directory, 405  
     rundir, 407  
     set\_option, 405  
     stage, 405  
     submit\_jobs, 406  
     tempbase, 407  
     tempdir, 408  
     verbose\_options, 408  
     wq\_complete, 406  
     write\_0grads, 406  
     write\_indicate, 408  
     write\_objective, 408  
     xct, 408  
     forcebalance::lipid  
         logger, 35  
         weight\_info, 35  
     forcebalance::lipid::Lipid  
         \_\_init\_\_, 414  
         \_\_setattr\_\_, 415  
         absrd, 415  
         check\_files, 415  
         do\_self\_pol, 426  
         engname, 426  
         extra\_output, 426  
         FF, 426  
         gas\_engine, 426  
         gct, 426  
         get, 415  
         get\_G, 416  
         get\_H, 417  
         get\_X, 418  
         Gp, 426  
         hct, 426  
         indicate, 419  
         Labels, 426  
         last\_traj, 426  
         link\_from\_tempdir, 419  
         lipid\_mol, 426  
         MBarEnergy, 427  
         maxrd, 420  
         meta\_get, 420

meta\_indicate, 421  
 npt\_simulation, 422  
 Objective, 427  
 objective\_term, 422  
 pgrad, 427  
 PhasePoints, 427  
 polarization\_correction, 422  
 Pp, 427  
 prepare\_temp\_directory, 423  
 PrintOptionDict, 427  
 printcool\_table, 423  
 rd, 427  
 read, 423  
 read\_0grads, 424  
 read\_data, 424  
 read\_indicate, 427  
 read\_objective, 427  
 RefData, 427  
 refresh\_temp\_directory, 424  
 rundir, 427  
 SavedMVal, 427  
 SavedTraj, 427  
 set\_option, 424  
 stage, 424  
 submit\_jobs, 425  
 tempbase, 427  
 tempdir, 427  
 verbose\_options, 428  
 w\_al, 428  
 w\_alpha, 428  
 w\_cp, 428  
 w\_eps0, 428  
 w\_kappa, 428  
 w\_rho, 428  
 w\_scd, 428  
 Wp, 428  
 wq\_complete, 425  
 write\_0grads, 426  
 write\_indicate, 428  
 write\_objective, 428  
 xct, 428  
 Xp, 428  
 forcebalance::liquid  
     logger, 35  
     weight\_info, 35  
 forcebalance::liquid::Liquid  
     \_\_init\_\_, 450  
     \_\_setattr\_\_, 450  
     absrd, 450  
     AllResults, 462  
     check\_files, 450  
     do\_self\_pol, 462  
     engname, 462  
     extra\_output, 462  
     FF, 462  
     gas\_engine, 462  
     gas\_mol, 462  
     gct, 462  
     get, 450  
     get\_G, 451  
     get\_H, 452  
     get\_X, 453  
     Gp, 462  
     hct, 462  
     indicate, 454  
     Labels, 462  
     last\_traj, 462  
     link\_from\_tempdir, 454  
     liquid\_mol, 462  
     MBarEnergy, 462  
     maxrd, 455  
     meta\_get, 455  
     meta\_indicate, 456  
     npt\_simulation, 457  
     Objective, 462  
     objective\_term, 457  
     pgrad, 462  
     PhasePoints, 462  
     polarization\_correction, 457  
     Pp, 462  
     prepare\_temp\_directory, 458  
     PrintOptionDict, 462  
     printcool\_table, 458  
     rd, 462  
     read, 458  
     read\_0grads, 459  
     read\_data, 459  
     read\_indicate, 463  
     read\_objective, 463  
     RefData, 463  
     refresh\_temp\_directory, 459  
     rundir, 463  
     SavedTraj, 463  
     set\_option, 459  
     stage, 459  
     submit\_jobs, 460  
     tempbase, 463  
     tempdir, 463  
     verbose\_options, 463  
     w\_alpha, 463  
     w\_cp, 463  
     w\_eps0, 463  
     w\_hvap, 463  
     w\_kappa, 463  
     w\_rho, 463  
     Wp, 463  
     wq\_complete, 461  
     write\_0grads, 461

```

write_indicate, 464
write_objective, 464
xct, 464
Xp, 464
forcebalance::mol2io
    mol2_pdict, 36
forcebalance::mol2io::Mol2_Reader
    __init__, 534
    adict, 534
    atom, 534
    AtomTypes, 535
    build_pid, 534
    feed, 534
    itype, 535
    In, 535
    molatom, 535
    Molecules, 535
    pdict, 535
    Split, 534
    suffix, 535
    Whites, 534
forcebalance::molecule
    add_strip_to_mat, 38
    AlignToDensity, 38
    AlignToMoments, 38
    AllVariableNames, 42
    AtomVariableNames, 42
    bohrang, 42
    both, 39
    Box, 42
    BuildLatticeFromLengthsAngles, 39
    BuildLatticeFromVectors, 39
    cartesian_product2, 39
    ComputeOverlap, 39
    CubicLattice, 39
    diff, 39
    either, 39
    elem_from_atomname, 39
    Elements, 42
    EulerMatrix, 40
    even_list, 40
    format_gro_box, 40
    format_gro_coord, 40
    format_xyz_coord, 40
    format_xyzgen_coord, 40
    FrameVariableNames, 42
    get_rotate_translate, 40
    getElement, 40
    grouper, 40
    have_contact, 43
    is_charmm_coord, 40
    is_gro_box, 41
    is_gro_coord, 41
    isfloat, 42
    isint, 42
    main, 42
    MetaVariableNames, 43
    nodematch, 42
    PeriodicTable, 43
    pvec, 42
    QuantumVariableNames, 43
    radian, 43
    Radii, 43
    splitter, 43
forcebalance::molecule::Molecule
    __add__, 543
    __delitem__, 543
    __getattr__, 543
    __getitem__, 543
    __iadd__, 544
    __init__, 543
    __iter__, 544
    __len__, 544
    __setattr__, 544
    add_quantum, 544
    add_virtual_site, 544
    align, 544
    align_by_moments, 545
    align_center, 545
    all_pairwise_rmsd, 545
    append, 546
    atom_select, 546
    atom_stack, 546
    boxes, 558
    build_topology, 546
    built_bonds, 558
    center_of_mass, 546
    comms, 558
    Data, 558
    edit_qcrems, 546
    find_angles, 546
    find_dihedrals, 547
    foul, 558
    Funnel, 558
    load_frames, 547
    measure_dihedrals, 547
    molecules, 558
    openmm_boxes, 547
    openmm_positions, 547
    pathwise_rmsd, 547
    positive_resid, 558
    radius_of_gyration, 548
    Read_Tab, 559
    read_arc, 548
    read_charmm, 549
    read_com, 549
    read_dcd, 550
    read_gro, 550

```

read\_mdcrd, 550  
 read\_mol2, 551  
 read\_pdb, 551  
 read\_qcesp, 551  
 read\_qcin, 551  
 read\_qcout, 552  
 read\_qdata, 552  
 read\_xyz, 552  
 read\_xyz0, 553  
 ref\_rmsd, 553  
 repair, 553  
 replace\_peratom, 554  
 replace\_peratom\_conditional, 554  
 require, 554  
 require\_boxes, 554  
 require\_resid, 554  
 require\_resname, 554  
 resid, 559  
 resname, 559  
 rigid\_water, 554  
 split, 555  
 topology, 559  
 write, 555  
 Write\_Tab, 559  
 write\_arc, 555  
 write\_dcd, 555  
 write\_gro, 555  
 write\_mdcrd, 556  
 write\_molproq, 556  
 write\_pdb, 556  
 write\_qcin, 557  
 write\_qdata, 557  
 write\_xyz, 558  
 forcebalance::molecule::MyG  
   \_\_eq\_\_, 621  
   \_\_hash\_\_, 621  
   \_\_init\_\_, 621  
   AStr, 621  
   Alive, 622  
   e, 621  
   ef, 621  
   L, 622  
   x, 622  
 forcebalance::moments  
   logger, 44  
 forcebalance::moments::Moments  
   \_\_init\_\_, 563  
   \_\_setattr\_\_, 563  
   absrd, 563  
   calc\_moments, 573  
   check\_files, 564  
   denoms, 573  
   engine, 573  
   FF, 573  
     gct, 573  
     get, 564  
     get\_G, 564  
     get\_H, 565  
     get\_X, 566  
     hct, 573  
     indicate, 567  
     link\_from\_tempdir, 567  
     maxrd, 568  
     meta\_get, 568  
     meta\_indicate, 569  
     mfnm, 573  
     na, 573  
     objective, 573  
     pgrad, 573  
     PrintOptionDict, 573  
     printcool\_table, 570  
     rd, 573  
     read, 570  
     read\_0grads, 571  
     read\_indicate, 573  
     read\_objective, 573  
     read\_reference\_data, 571  
     ref\_eigvals, 574  
     ref\_eigvecs, 574  
     ref\_moments, 574  
     refresh\_temp\_directory, 571  
     rundir, 574  
     set\_option, 571  
     stage, 571  
     submit\_jobs, 572  
     tempbase, 574  
     tempdir, 574  
     unpack\_moments, 572  
     verbose\_options, 574  
     wq\_complete, 572  
     write\_0grads, 572  
     write\_indicate, 574  
     write\_objective, 574  
     xct, 574  
   forcebalance::nifty  
     allsplit, 47  
     astr, 47  
     bak, 47  
     bohrang, 56  
     click, 47  
     col, 47  
     commadash, 47  
     concurrent\_map, 47  
     CopyFile, 47  
     createWorkQueue, 48  
     destroyWorkQueue, 48  
     encode, 48  
     eqcgm, 56

extract\_int, 48  
 flat, 48  
 floatornan, 48  
 fqcgmx, 56  
 get\_least\_squares, 49  
 getWQIds, 49  
 getWorkQueue, 49  
 GoInto, 50  
 grouper, 50  
 invert\_svd, 50  
 isdecimal, 50  
 isfloat, 50  
 isint, 50  
 kb, 56  
 Leave, 51  
 link\_dir\_contents, 51  
 LinkFile, 51  
 logger, 56  
 lp\_dump, 51  
 lp\_load, 51  
 MissingFileInspection, 51  
 monotonic, 51  
 multiD\_statisticalInefficiency, 51  
 onefile, 52  
 orthogonalize, 52  
 pmat2d, 52  
 printcool, 52  
 printcool\_dictionary, 53  
 pvec1d, 54  
 queue\_up, 54  
 queue\_up\_src\_dest, 54  
 remove\_if\_exists, 54  
 row, 54  
 segments, 54  
 specific\_dct, 56  
 specific\_lst, 56  
 statisticalInefficiency, 55  
 uncommadash, 55  
 WORK\_QUEUE, 56  
 WQIDS, 57  
 warn\_once, 55  
 warn\_press\_key, 55  
 which, 55  
 wopen, 55  
 wq\_wait, 55  
 wq\_wait1, 56  
 XMLFILE, 57  
 forcebalance::nifty::LineChunker  
   \_\_enter\_\_, 409  
   \_\_exit\_\_, 409  
   \_\_init\_\_, 409  
   buf, 410  
   callback, 410  
   close, 409  
     nomnom, 410  
     push, 410  
   forcebalance::nifty::Pickler\_LP  
     \_\_init\_\_, 657  
   forcebalance::nifty::Unpickler\_LP  
     \_\_init\_\_, 777  
   forcebalance::objective  
     Implemented\_Targets, 57  
     Letters, 58  
     logger, 58  
   forcebalance::objective::Objective  
     \_\_init\_\_, 624  
     \_\_setattr\_\_, 624  
     FF, 625  
     Full, 624  
     Indicate, 625  
     ObjDict, 625  
     ObjDict\_Last, 625  
     Penalty, 625  
     PrintOptionDict, 626  
     set\_option, 625  
     Target\_Terms, 625  
     Targets, 626  
     verbose\_options, 626  
     WTot, 626  
   forcebalance::objective::Penalty  
     \_\_init\_\_, 654  
     a, 656  
     b, 656  
     compute, 654  
     FF, 656  
     FUSE, 654  
     FUSE\_BARRIER, 655  
     FUSE\_L0, 655  
     fadd, 656  
     fmul, 656  
     HYP, 655  
     L2\_norm, 655  
     Pen\_Names, 656  
     Pen\_Tab, 656  
     ptyp, 656  
     spacings, 656  
 forcebalance::openmmio  
   CopyAmoebaAngleParameters, 59  
   CopyAmoebaBondParameters, 59  
   CopyAmoebaInPlaneAngleParameters, 59  
   CopyAmoebaMultipoleParameters, 59  
   CopyAmoebaOutOfPlaneBendParameters, 59  
   CopyAmoebaVdwParameters, 59  
   CopyHarmonicAngleParameters, 59  
   CopyHarmonicBondParameters, 60  
   CopyNonbondedParameters, 60  
   CopyPeriodicTorsionParameters, 60  
   CopySystemParameters, 60

do\_nothing, 60  
 energy\_components, 60  
 get\_dipole, 60  
 get\_multipoles, 60  
 GetVirtualSiteParameters, 60  
 logger, 61  
 MTSVVVRIntegrator, 60  
 pdict, 61  
 PrepareVirtualSites, 61  
 ResetVirtualSites, 61  
 ResetVirtualSites\_fast, 61  
 SetAmoebaVirtualExclusions, 61  
 suffix\_dict, 62  
 UpdateSimulationParameters, 61  
 forcebalance::openmmio::AbInitio\_OpenMM  
     \_\_init\_\_, 168  
     \_\_setattr\_\_, 168  
     absrd, 168  
     AtomLists, 180  
     AtomMask, 180  
     boltz\_wts, 180  
     build\_invdist, 168  
     check\_files, 168  
     compute\_netforce\_torque, 168  
     e\_err, 180  
     e\_err\_pct, 180  
     e\_ref, 180  
     emd0, 181  
     energy\_all, 169  
     energy\_force\_all, 169  
     energy\_force\_one, 169  
     energy\_force\_transform, 169  
     energy\_force\_transform\_one, 169  
     energy\_one, 170  
     engine, 181  
     engine\_, 181  
     eqm, 181  
     esp\_err, 181  
     espval, 181  
     espxyz, 181  
     f\_err, 181  
     f\_err\_pct, 181  
     f\_ref, 181  
     FF, 181  
     fitatoms, 181  
     force, 181  
     force\_map, 181  
     fqm, 181  
     fref, 181  
     gct, 181  
     get, 170  
     get\_G, 171  
     get\_H, 172  
     get\_X, 173  
     get\_energy\_force, 170  
     get\_resp, 173  
     hct, 181  
     indicate, 174  
     invdists, 182  
     link\_from\_tempdir, 174  
     maxrd, 175  
     meta\_get, 175  
     meta\_indicate, 176  
     mol, 182  
     nesp, 182  
     new\_vsites, 182  
     nf\_err, 182  
     nf\_err\_pct, 182  
     nf\_ref, 182  
     nftqm, 182  
     nnf, 182  
     nparticles, 182  
     ns, 182  
     ntq, 182  
     objective, 182  
     pgrad, 182  
     PrintOptionDict, 182  
     printcool\_table, 177  
     qfnm, 182  
     qmatoms, 182  
     qmboltz\_wts, 182  
     rd, 182  
     read, 177  
     read\_0grads, 178  
     read\_indicate, 183  
     read\_objective, 183  
     read\_reference\_data, 178  
     refresh\_temp\_directory, 179  
     respterm, 183  
     rundir, 183  
     save\_vmvalls, 183  
     set\_option, 179  
     stage, 179  
     submit\_jobs, 179  
     tempbase, 183  
     tempdir, 183  
     tq\_err, 183  
     tq\_err\_pct, 183  
     tq\_ref, 183  
     use\_nft, 183  
     verbose\_options, 183  
     w\_energy, 183  
     w\_force, 183  
     w\_netforce, 184  
     w\_resp, 184  
     w\_torque, 184  
     whamboltz, 184  
     wq\_complete, 180

```

write_0grads, 180
write_indicate, 184
write_objective, 184
xct, 184
forcebalance::openmmio::BindingEnergy_OpenMM
    __init__, 245
    __setattr__, 245
    absrd, 245
    check_files, 246
    energy_part, 255
    engine_, 255
    engines, 255
    FF, 255
    gct, 255
    get, 246
    get_G, 246
    get_H, 247
    get_X, 248
    hct, 255
    indicate, 249
    inter_opts, 255
    link_from_tempdir, 249
    maxrd, 250
    meta_get, 250
    meta_indicate, 251
    objective, 255
    pgrad, 255
    PrintDict, 255
    PrintOptionDict, 255
    printcool_table, 252
    RMSDDict, 256
    rd, 255
    read, 252
    read_0grads, 253
    read_indicate, 255
    read_objective, 255
    refresh_temp_directory, 253
    rmsd_part, 255
    rundir, 256
    set_option, 253
    stage, 253
    submit_jobs, 254
    system_driver, 254
    tempbase, 256
    tempdir, 256
    verbose_options, 256
    wq_complete, 254
    write_0grads, 254
    write_indicate, 256
    write_objective, 256
    xct, 256
forcebalance::openmmio::Interaction_OpenMM
    __init__, 363
    __setattr__, 364
absrd, 364
check_files, 364
divisor, 373
e_err, 373
e_err_pct, 373
emm, 373
engine, 373
engine_, 373
eqm, 373
FF, 373
gct, 373
get, 364
get_G, 364
get_H, 365
get_X, 366
hct, 373
indicate, 367
label, 373
link_from_tempdir, 367
maxrd, 368
meta_get, 368
meta_indicate, 369
mol, 373
ns, 373
objective, 373
pgrad, 373
prefactor, 373
PrintOptionDict, 373
printcool_table, 370
qfnm, 373
rd, 374
read, 370
read_0grads, 371
read_indicate, 374
read_objective, 374
read_reference_data, 371
refresh_temp_directory, 371
rundir, 374
select1, 374
select2, 374
set_option, 371
stage, 371
submit_jobs, 372
tempbase, 374
tempdir, 374
verbose_options, 374
weight, 374
wq_complete, 372
write_0grads, 372
write_indicate, 374
write_objective, 374
xct, 375
forcebalance::openmmio::Liquid_OpenMM
    __init__, 488

```

\_\_setattr\_\_, 488  
 absrd, 488  
 AllResults, 500  
 check\_files, 488  
 do\_self\_pol, 500  
 engine\_, 500  
 engname, 500  
 extra\_output, 500  
 FF, 500  
 gas\_engine, 500  
 gas\_engine\_args, 500  
 gas\_mol, 500  
 gct, 500  
 get, 488  
 get\_G, 489  
 get\_H, 490  
 get\_X, 491  
 Gp, 500  
 hct, 500  
 indicate, 492  
 Labels, 500  
 last\_traj, 500  
 link\_from\_tempdir, 492  
 liquid\_mol, 500  
 MBarEnergy, 500  
 maxrd, 493  
 meta\_get, 493  
 meta\_indicate, 494  
 npt\_simulation, 495  
 nptfiles, 500  
 nptpx, 500  
 Objective, 500  
 objective\_term, 495  
 pgrad, 500  
 PhasePoints, 500  
 polarization\_correction, 495  
 Pp, 501  
 prepare\_temp\_directory, 496  
 PrintOptionDict, 501  
 printcool\_table, 496  
 rd, 501  
 read, 496  
 read\_0grads, 497  
 read\_data, 497  
 read\_indicate, 501  
 read\_objective, 501  
 RefData, 501  
 refresh\_temp\_directory, 497  
 rundir, 501  
 SavedTraj, 501  
 scripts, 501  
 set\_option, 497  
 stage, 497  
 submit\_jobs, 498  
 tempbase, 501  
 tempdir, 501  
 verbose\_options, 501  
 w\_alpha, 501  
 w\_cp, 501  
 w\_eps0, 501  
 w\_hvap, 501  
 w\_kappa, 502  
 w\_rho, 502  
 Wp, 502  
 wq\_complete, 499  
 write\_0grads, 499  
 write\_indicate, 502  
 write\_objective, 502  
 xct, 502  
 Xp, 502  
**forcebalance::openmmio::Moments\_OpenMM**  
 \_\_init\_\_, 593  
 \_\_setattr\_\_, 593  
 absrd, 593  
 calc\_moments, 603  
 check\_files, 594  
 denoms, 603  
 engine, 603  
 engine\_, 603  
 FF, 603  
 gct, 603  
 get, 594  
 get\_G, 594  
 get\_H, 595  
 get\_X, 596  
 hct, 603  
 indicate, 597  
 link\_from\_tempdir, 597  
 maxrd, 598  
 meta\_get, 598  
 meta\_indicate, 599  
 mfnm, 603  
 na, 603  
 objective, 603  
 pgrad, 603  
 PrintOptionDict, 603  
 printcool\_table, 600  
 rd, 603  
 read, 600  
 read\_0grads, 601  
 read\_indicate, 603  
 read\_objective, 603  
 read\_reference\_data, 601  
 ref\_eigvals, 604  
 ref\_eigvecs, 604  
 ref\_moments, 604  
 refresh\_temp\_directory, 601  
 rundir, 604

```

set_option, 601
stage, 601
submit_jobs, 602
tempbase, 604
tempdir, 604
unpack_moments, 602
verbose_options, 604
wq_complete, 602
write_0grads, 602
write_indicate, 604
write_objective, 604
xct, 604

forcebalance::openmmio::OpenMM
    __init__, 629
    __setattr__, 629
    A, 635
    AMOEBA, 635
    AtomLists, 635
    AtomMask, 635
    B, 635
    compute_mass, 629
    compute_volume, 629
    create_simulation, 629
    energy, 629
    energy_dipole, 630
    energy_force, 630
    energy_force_one, 630
    energy_one, 631
    energy_rmsd, 631
    evaluate_, 631
    evaluate_one_, 632
    FF, 635
    fFXML, 635
    forcefield, 635
    interaction_energy, 632
    mass, 635
    mmopts, 635
    mod, 635
    mol, 635
    molecular_dynamics, 632
    multipole_moments, 633
    name, 635
    nbcharges, 635
    ndof, 635
    normal_modes, 633
    optimize, 633
    pbc, 635
    pdb, 635
    platform, 635
    platname, 635
    precision, 635
    prepare, 634
    PrintOptionDict, 636
    readsrc, 634

                    root, 636
                    set_option, 634
                    set_positions, 634
                    setopts, 634
                    simkwargs, 636
                    simulation, 636
                    srccdir, 636
                    system, 636
                    target, 636
                    tdiv, 636
                    tempdir, 636
                    update_simulation, 634
                    valkwd, 636
                    verbose, 636
                    verbose_options, 636
                    vsinfo, 636
                    vsprm, 636
                    xyz_omm, 636

forcebalance::openmmio::OpenMM_Reader
    __init__, 638
    adict, 639
    AtomTypes, 639
    build_pid, 638
    feed, 638
    itype, 639
    In, 639
    molatom, 639
    Molecules, 639
    pdict, 639
    Split, 638
    suffix, 639
    Whites, 638

forcebalance::optimizer
    Counter, 62
    First, 62
    GOODSTEP, 63
    GoodStep, 63
    ITERATION, 63
    ITERINIT, 63
    logger, 63

forcebalance::optimizer::Optimizer
    __init__, 642
    __setattr__, 642
    adjh, 642
    Anneal, 642
    BFGS, 643
    bakdir, 651
    BasinHopping, 643
    bhyp, 651
    chk, 651
    ConjugateGradient, 643
    dx, 652
    excision, 652
    FDCheckG, 644

```

```

FDCheckH, 644
FF, 652
failmsg, 652
GeneticAlgorithm, 645
Grad, 652
Gradient, 645
H, 652
h, 652
Hess, 652
Hessian, 645
MainOptimizer, 645
mvals0, 652
mvals_bak, 652
NewtonCG, 646
NewtonRaphson, 646
np, 652
Objective, 652
OptTab, 653
Penalty, 653
Powell, 647
prev_bad, 653
PrintOptionDict, 653
readchk, 647
recover, 647
resdir, 653
Run, 647
save_mvals_to_input, 648
Scan_Values, 648
ScanMVals, 648
ScanPVals, 649
Scipy_BFGS, 649
ScipyOptimizer, 649
set_option, 650
Simplex, 650
SinglePoint, 650
step, 650
TruncatedNewton, 651
uncert, 653
Val, 653
verbose_options, 653
writechk, 651
x_best, 653
x_prev, 653
xk_prev, 653
forcebalance::output::CleanFileHandler
emit, 272
forcebalance::output::CleanStreamHandler
__init__, 273
emit, 274
forcebalance::output::ForceBalanceLogger
__init__, 307
addHandler, 307
defaultHandler, 307
removeHandler, 307
forcebalance::output::ModLogger
error, 523
forcebalance::output::RawFileHandler
emit, 669
forcebalance::output::RawStreamHandler
__init__, 670
emit, 671
forcebalance::parser
all_opts_names, 66
blkwd, 66
gen_opts_defaults, 66
gen_opts_types, 66
iocc, 66
logger, 66
mainsections, 67
ParsTab, 67
parse_inputs, 65
printsection, 65
read_internals, 66
read_mvals, 66
read_priors, 66
read_pvals, 66
subdict, 67
tgt_opts_defaults, 67
tgt_opts_types, 67
forcebalance::psi4io
logger, 68
forcebalance::psi4io::GBS_Reader
__init__, 312
adict, 313
amom, 313
AtomTypes, 313
basis_number, 313
build_pid, 312
contraction_number, 313
destroy, 313
element, 313
feed, 312
isdata, 313
itype, 313
last_amom, 313
In, 313
molatom, 313
Molecules, 313
pdct, 313
Split, 313
suffix, 313
Whites, 313
forcebalance::psi4io::Grid_Reader
__init__, 329
adict, 329
AtomTypes, 329
build_pid, 329
element, 329

```

feed, 329  
 isdata, 329  
 itype, 329  
 In, 329  
 molatom, 329  
 Molecules, 329  
 pdict, 329  
 point, 329  
 radii, 329  
 Split, 329  
 suffix, 330  
 Whites, 329  
**forcebalance::psi4io::RDVR3\_Psi4**  
 \_\_init\_\_, 674  
 \_\_setattr\_\_, 674  
 absrd, 674  
 bidirect, 685  
 callderivs, 685  
 check\_files, 675  
 driver, 675  
 elements, 685  
 FF, 685  
 factor, 685  
 gct, 685  
 get, 675  
 get.G, 676  
 get.H, 677  
 get.X, 678  
 gradd, 685  
 hct, 685  
 hdiagd, 685  
 indicate, 679  
 link\_from\_tempdir, 679  
 maxrd, 680  
 meta\_get, 680  
 meta\_indicate, 681  
 molecules, 685  
 objd, 685  
 objective, 685  
 objfiles, 685  
 objvals, 685  
 pgrad, 685  
 PrintOptionDict, 685  
 printcool\_table, 682  
 rd, 685  
 read, 682  
 read\_0grads, 683  
 read\_indicate, 685  
 read\_objective, 686  
 refresh\_temp\_directory, 683  
 rundir, 686  
 set\_option, 683  
 stage, 683  
 submit\_jobs, 684  
 tdir, 686  
 tempbase, 686  
 tempdir, 686  
 verbose\_options, 686  
 wq\_complete, 684  
 write\_0grads, 684  
 write\_indicate, 686  
 write\_objective, 686  
 xct, 686  
**forcebalance::psi4io::THCDF\_Psi4**  
 \_\_init\_\_, 718  
 \_\_setattr\_\_, 718  
 absrd, 718  
 check\_files, 719  
 D, 730  
 DATfnm, 730  
 DF\_Energy, 730  
 driver, 719  
 Elements, 730  
 FF, 730  
 GBSfnm, 730  
 gct, 730  
 get, 719  
 get\_G, 721  
 get\_H, 722  
 get\_X, 723  
 hct, 730  
 indicate, 724  
 link\_from\_tempdir, 724  
 MAQ, 730  
 MP2\_Energy, 730  
 maxrd, 724  
 meta\_get, 725  
 meta\_indicate, 726  
 Molecules, 730  
 objective, 730  
 pgrad, 730  
 prepare\_temp\_directory, 727  
 PrintOptionDict, 731  
 printcool\_table, 727  
 rd, 731  
 read, 728  
 read\_0grads, 728  
 read\_indicate, 731  
 read\_objective, 731  
 refresh\_temp\_directory, 728  
 rundir, 731  
 set\_option, 728  
 stage, 728  
 submit\_jobs, 729  
 tempbase, 731  
 tempdir, 731  
 throw\_outs, 731  
 verbose\_options, 731

```

wq_complete, 729
write_0grads, 729
write_indicate, 731
write_nested_destroy, 730
write_objective, 731
xct, 731
forcebalance::qchemio
    logger, 69
    ndtypes, 69
    pdict, 69
    QChem_Dielectric_Energy, 69
forcebalance::qchemio::QCIn_Reader
    __init__, 661
    adict, 662
    atom, 662
    AtomTypes, 662
    build_pid, 661
    cnum, 662
    feed, 661
    itype, 662
    In, 662
    molatom, 662
    Molecules, 662
    pdict, 662
    sec, 662
    shell, 662
    snum, 662
    Split, 661
    suffix, 662
    Whites, 661
forcebalance::quantity
    energy_derivatives, 70
    logger, 71
    mean_stderr, 70
forcebalance::quantity::Quantity
    __init__, 664
    __str__, 664
    engname, 664
    extract, 664
    name, 664
    pressure, 664
    temperature, 664
forcebalance::quantity::Quantity_Density
    __init__, 666
    __str__, 666
    engname, 666
    extract, 666
    name, 666
    pressure, 666
    temperature, 666
forcebalance::quantity::Quantity_H_vap
    __init__, 668
    __str__, 668
    engname, 668
extract, 668
name, 668
pressure, 668
temperature, 668
forcebalance::target::RemoteTarget
    __init__, 690
    __setattr__, 690
    absrd, 690
    check_files, 690
    FF, 699
    gct, 699
    get, 690
    get_G, 691
    get_H, 691
    get_X, 692
    hct, 699
    indicate, 693
    link_from_tempdir, 693
    maxrd, 693
    meta_get, 694
    meta_indicate, 695
    pgrad, 699
    PrintOptionDict, 699
    printcool_table, 696
    r_options, 699
    r_tgt_opts, 699
    rd, 699
    read, 696
    read_0grads, 697
    read_indicate, 699
    read_objective, 700
    refresh_temp_directory, 697
    remote_indicate, 700
    rundir, 700
    set_option, 697
    stage, 697
    submit_jobs, 698
    tempbase, 700
    tempdir, 700
    verbose_options, 700
    wq_complete, 698
    write_0grads, 699
    write_indicate, 700
    write_objective, 700
    xct, 700
forcebalance::target::Target
    __init__, 704
    __setattr__, 704
    absrd, 704
    check_files, 705
    FF, 714
    gct, 714

```

get, 705  
 get\_G, 705  
 get\_H, 706  
 get\_X, 707  
 hct, 714  
 link\_from\_tempdir, 708  
 maxrd, 708  
 meta\_get, 709  
 meta\_indicate, 710  
 pgrad, 714  
 PrintOptionDict, 714  
 printcool\_table, 711  
 rd, 714  
 read, 711  
 read\_0grads, 712  
 read\_indicate, 714  
 read\_objective, 714  
 refresh\_temp\_directory, 712  
 rundir, 714  
 set\_option, 712  
 stage, 712  
 submit\_jobs, 713  
 tempbase, 714  
 tempdir, 714  
 verbose\_options, 714  
 wq\_complete, 713  
 write\_0grads, 713  
 write\_indicate, 714  
 write\_objective, 715  
 xct, 715  
 forcebalance::thermo  
 logger, 71  
 forcebalance::thermo::Point  
 \_\_init\_\_, 659  
 \_\_str\_\_, 659  
 data, 659  
 idnr, 659  
 pressure, 659  
 ref, 659  
 temperature, 659  
 forcebalance::thermo::Thermo  
 \_\_init\_\_, 735  
 \_\_setattr\_\_, 735  
 absrd, 735  
 check\_files, 736  
 denoms, 745  
 FF, 745  
 gct, 745  
 get, 736  
 get\_G, 736  
 get\_H, 737  
 get\_X, 738  
 Gp, 745  
 hct, 746  
 indicate, 739  
 link\_from\_tempdir, 739  
 maxrd, 740  
 meta\_get, 740  
 meta\_indicate, 741  
 Objective, 746  
 objective\_term, 742  
 pgrad, 746  
 points, 746  
 Pp, 746  
 PrintOptionDict, 746  
 printcool\_table, 742  
 rd, 746  
 read, 743  
 read\_0grads, 743  
 read\_indicate, 746  
 read\_objective, 746  
 refresh\_temp\_directory, 743  
 retrieve, 743  
 rundir, 746  
 set\_option, 744  
 simpfx, 746  
 stage, 744  
 submit\_jobs, 744  
 tempbase, 746  
 tempdir, 746  
 verbose\_options, 747  
 weights, 747  
 Wp, 747  
 wq\_complete, 745  
 write\_0grads, 745  
 write\_indicate, 747  
 write\_objective, 747  
 xct, 747  
 Xp, 747  
 forcebalance::tinkerio  
 alp, 73  
 eckeys, 73  
 logger, 73  
 pdict, 73  
 write\_key, 72  
 forcebalance::tinkerio::AbInitio\_TINKER  
 \_\_init\_\_, 190  
 \_\_setattr\_\_, 190  
 absrd, 190  
 AtomLists, 202  
 AtomMask, 202  
 boltz\_wts, 202  
 build\_invdist, 190  
 check\_files, 190  
 compute\_netforce\_torque, 190  
 e\_err, 202  
 e\_err\_pct, 202  
 e\_ref, 202

emd0, 203  
 energy\_all, 191  
 energy\_force\_all, 191  
 energy\_force\_one, 191  
 energy\_force\_transform, 191  
 energy\_force\_transform\_one, 191  
 energy\_one, 192  
 engine, 203  
 engine\_, 203  
 eqm, 203  
 esp\_err, 203  
 espval, 203  
 espxyz, 203  
 f\_err, 203  
 f\_err\_pct, 203  
 f\_ref, 203  
 FF, 203  
 fitatoms, 203  
 force, 203  
 force\_map, 203  
 fqm, 203  
 fref, 203  
 gct, 203  
 get, 192  
 get\_G, 193  
 get\_H, 194  
 get\_X, 195  
 get\_energy\_force, 192  
 get\_resp, 195  
 hct, 203  
 indicate, 196  
 invdists, 204  
 link\_from\_tempdir, 196  
 maxrd, 197  
 meta\_get, 197  
 meta\_indicate, 198  
 mol, 204  
 nesp, 204  
 new\_vsites, 204  
 nf\_err, 204  
 nf\_err\_pct, 204  
 nf\_ref, 204  
 nftqm, 204  
 nnf, 204  
 nparticles, 204  
 ns, 204  
 ntq, 204  
 objective, 204  
 pgrad, 204  
 PrintOptionDict, 204  
 printcool\_table, 199  
 qfnm, 204  
 qmatoms, 204  
 qmboltz\_wts, 204  
 rd, 204  
 read, 199  
 read\_0grads, 200  
 read\_indicate, 205  
 read\_objective, 205  
 read\_reference\_data, 200  
 refresh\_temp\_directory, 201  
 respterm, 205  
 rundir, 205  
 save\_vmvalls, 205  
 set\_option, 201  
 stage, 201  
 submit\_jobs, 201  
 tempbase, 205  
 tempdir, 205  
 tq\_err, 205  
 tq\_err\_pct, 205  
 tq\_ref, 205  
 use\_nft, 205  
 verbose\_options, 205  
 w\_energy, 205  
 w\_force, 205  
 w\_netforce, 206  
 w\_resp, 206  
 w\_torque, 206  
 whamboltz, 206  
 wq\_complete, 202  
 write\_0grads, 202  
 write\_indicate, 206  
 write\_objective, 206  
 xct, 206  
**forcebalance::tinkerio::BindingEnergy\_TINKER**  
 \_\_init\_\_, 260  
 \_\_setattr\_\_, 260  
 absrd, 260  
 check\_files, 261  
 energy\_part, 270  
 engine\_, 270  
 engines, 270  
 FF, 270  
 gct, 270  
 get, 261  
 get\_G, 261  
 get\_H, 262  
 get\_X, 263  
 hct, 270  
 indicate, 264  
 inter\_opts, 270  
 link\_from\_tempdir, 264  
 maxrd, 265  
 meta\_get, 265  
 meta\_indicate, 266  
 objective, 270  
 pgrad, 270

PrintDict, 270  
 PrintOptionDict, 270  
 printcool\_table, 267  
 RMSDDict, 271  
 rd, 270  
 read, 267  
 read\_0grads, 268  
 read\_indicate, 270  
 read\_objective, 270  
 refresh\_temp\_directory, 268  
 rmsd\_part, 270  
 rundir, 271  
 set\_option, 268  
 stage, 268  
 submit\_jobs, 269  
 system\_driver, 269  
 tempbase, 271  
 tempdir, 271  
 verbose\_options, 271  
 wq\_complete, 269  
 write\_0grads, 269  
 write\_indicate, 271  
 write\_objective, 271  
 xct, 271  
**forcebalance::tinkerio::Interaction\_TINKER**  
 \_\_init\_\_, 378  
 \_\_setattr\_\_, 379  
 absrd, 379  
 check\_files, 379  
 divisor, 388  
 e\_err, 388  
 e\_err\_pct, 388  
 emm, 388  
 engine, 388  
 engine\_, 388  
 eqm, 388  
 FF, 388  
 gct, 388  
 get, 379  
 get\_G, 379  
 get\_H, 380  
 get\_X, 381  
 hct, 388  
 indicate, 382  
 label, 388  
 link\_from\_tempdir, 382  
 maxrd, 383  
 meta\_get, 383  
 meta\_indicate, 384  
 mol, 388  
 ns, 388  
 objective, 388  
 pgrad, 388  
 prefactor, 388  
 PrintOptionDict, 388  
 printcool\_table, 385  
 qfnm, 388  
 rd, 389  
 read, 385  
 read\_0grads, 386  
 read\_indicate, 389  
 read\_objective, 389  
 read\_reference\_data, 386  
 refresh\_temp\_directory, 386  
 rundir, 389  
 select1, 389  
 select2, 389  
 set\_option, 386  
 stage, 386  
 submit\_jobs, 387  
 tempbase, 389  
 tempdir, 389  
 verbose\_options, 389  
 weight, 389  
 wq\_complete, 387  
 write\_0grads, 387  
 write\_indicate, 389  
 write\_objective, 389  
 xct, 390  
**forcebalance::tinkerio::Liquid\_TINKER**  
 \_\_init\_\_, 507  
 \_\_setattr\_\_, 507  
 absrd, 507  
 AllResults, 520  
 check\_files, 507  
 do\_self\_pol, 520  
 DynDict, 520  
 DynDict\_New, 520  
 engine\_, 520  
 engname, 520  
 extra\_output, 520  
 FF, 520  
 gas\_engine, 520  
 gas\_engine\_args, 520  
 gas\_mol, 520  
 gct, 520  
 get, 507  
 get\_G, 509  
 get\_H, 510  
 get\_X, 511  
 Gp, 520  
 hct, 520  
 indicate, 512  
 Labels, 520  
 last\_traj, 520  
 link\_from\_tempdir, 512  
 liquid\_mol, 520  
 MBarEnergy, 520

maxrd, 513  
 meta\_get, 513  
 meta\_indicate, 514  
 npt\_simulation, 515  
 nptfiles, 520  
 nptpx, 520  
 Objective, 520  
 objective\_term, 515  
 pgrad, 521  
 PhasePoints, 521  
 polarization\_correction, 515  
 Pp, 521  
 prepare\_temp\_directory, 516  
 PrintOptionDict, 521  
 printcool\_table, 516  
 rd, 521  
 read, 516  
 read\_0grads, 517  
 read\_data, 517  
 read\_indicate, 521  
 read\_objective, 521  
 RefData, 521  
 refresh\_temp\_directory, 517  
 rundir, 521  
 SavedTraj, 521  
 scripts, 521  
 set\_option, 517  
 stage, 517  
 submit\_jobs, 518  
 tempbase, 521  
 tempdir, 521  
 verbose\_options, 521  
 w\_alpha, 521  
 w\_cp, 522  
 w\_eps0, 522  
 w\_hvap, 522  
 w\_kappa, 522  
 w\_rho, 522  
 Wp, 522  
 wq\_complete, 519  
 write\_0grads, 519  
 write\_indicate, 522  
 write\_objective, 522  
 xct, 522  
 Xp, 522  
 forcebalance::tinkerio::Moments\_TINKER  
     \_\_init\_\_, 608  
     \_\_setattr\_\_, 608  
     absrd, 608  
     calc\_moments, 618  
     check\_files, 609  
     denoms, 618  
     engine, 618  
     engine\_, 618  
             FF, 618  
             gct, 618  
             get, 609  
             get\_G, 609  
             get\_H, 610  
             get\_X, 611  
             hct, 618  
             indicate, 612  
             link\_from\_tempdir, 612  
             maxrd, 613  
             meta\_get, 613  
             meta\_indicate, 614  
             mfnm, 618  
             na, 618  
             objective, 618  
             pgrad, 618  
             PrintOptionDict, 618  
             printcool\_table, 615  
             rd, 618  
             read, 615  
             read\_0grads, 616  
             read\_indicate, 618  
             read\_objective, 618  
             read\_reference\_data, 616  
             ref\_eigvals, 619  
             ref\_eigvecs, 619  
             ref\_moments, 619  
             refresh\_temp\_directory, 616  
             rundir, 619  
             set\_option, 616  
             stage, 616  
             submit\_jobs, 617  
             tempbase, 619  
             tempdir, 619  
             unpack\_moments, 617  
             verbose\_options, 619  
             wq\_complete, 617  
             write\_0grads, 617  
             write\_indicate, 619  
             write\_objective, 619  
             xct, 619  
         forcebalance::tinkerio::TINKER  
             \_\_init\_\_, 767  
             \_\_setattr\_\_, 767  
             A, 772  
             abskey, 772  
             AtomLists, 772  
             AtomMask, 772  
             B, 772  
             calltinker, 767  
             energy, 767  
             energy\_dipole, 768  
             energy\_force, 768  
             energy\_force\_one, 768

```

energy_rmsd, 769
evaluate_, 769
FF, 772
interaction_energy, 769
key, 772
mdtraj, 772
mol, 772
molecular_dynamics, 769
multipole_moments, 770
name, 772
normal_modes, 770
optimize, 771
pbc, 772
prepare, 771
PrintOptionDict, 772
prm, 772
readsrc, 771
rigid, 772
root, 772
set_option, 771
setopts, 771
srcdir, 772
target, 772
tempdir, 772
tinkerpath, 772
valkwrd, 772
verbose, 772
verbose_options, 773
warn_vn, 773
forcebalance::tinkerio::Tinker_Reader
__init__, 775
adict, 776
atom, 776
AtomTypes, 776
build_pid, 775
feed, 775
itype, 776
In, 776
molatom, 776
Molecules, 776
pdict, 776
Split, 775
suffix, 776
Whites, 775
forcebalance::tinkerio::Vibration_TINKER
__init__, 810
__setattr__, 810
absrd, 810
c2r, 820
calc_eigvals, 820
check_files, 811
engine, 820
engine_, 820
FF, 820
gct, 820
get, 811
get_G, 811
get_H, 812
get_X, 813
hct, 820
indicate, 814
link_from_tempdir, 814
maxrd, 815
meta_get, 815
meta_indicate, 816
na, 820
objective, 820
overlaps, 820
pgrad, 820
PrintOptionDict, 821
printcool_table, 817
process_vectors, 817
rd, 821
read, 818
read_Ograds, 818
read_indicate, 821
read_objective, 821
read_reference_data, 818
reassign, 821
ref_eigvals, 821
ref_eigvecs, 821
ref_eigvecs_nrm_mw, 821
refresh_temp_directory, 818
rundir, 821
set_option, 818
stage, 818
submit_jobs, 819
tempbase, 821
tempdir, 821
verbose_options, 821
vfnm, 821
vibration_driver, 819
wq_complete, 819
write_Ograds, 820
write_indicate, 821
write_objective, 822
xct, 822
forcebalance::vibration
count_assignment, 74
logger, 74
forcebalance::vibration::Vibration
__init__, 781
__setattr__, 781
absrd, 781
c2r, 790
calc_eigvals, 790
check_files, 781
engine, 790

```

FF, 790  
 gct, 790  
 get, 781  
 get.G, 781  
 get.H, 782  
 get.X, 783  
 hct, 790  
 indicate, 784  
 link\_from\_tempdir, 784  
 maxrd, 785  
 meta\_get, 785  
 meta\_indicate, 786  
 na, 790  
 objective, 790  
 overlaps, 790  
 pgrad, 790  
 PrintOptionDict, 790  
 printcool\_table, 787  
 process\_vectors, 787  
 rd, 791  
 read, 788  
 read\_0grads, 788  
 read\_indicate, 791  
 read\_objective, 791  
 read\_reference\_data, 788  
 reassign, 791  
 ref\_eigvals, 791  
 ref\_eigvecs, 791  
 ref\_eigvecs\_nrm\_mw, 791  
 refresh\_temp\_directory, 788  
 rundir, 791  
 set\_option, 788  
 stage, 788  
 submit\_jobs, 789  
 tempbase, 791  
 tempdir, 791  
 verbose\_options, 791  
 vfnm, 791  
 vibration\_driver, 789  
 wq\_complete, 789  
 write\_0grads, 790  
 write\_indicate, 791  
 write\_objective, 791  
 xct, 792  
**forcefield**  
 forcebalance::openmmio::OpenMM, 635  
**forcefield.py**, 826  
**format\_gro\_box**  
 forcebalance::molecule, 40  
**format\_gro\_coord**  
 forcebalance::molecule, 40  
**format\_xyz\_coord**  
 forcebalance::molecule, 40  
**format\_xyzgen\_coord**

forcebalance::molecule, 40  
**fout**  
 forcebalance::molecule::Molecule, 558  
**fqcgmx**  
 forcebalance::nifty, 56  
**fqm**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::openmmio::AbInitio\_OpenMM, 181  
 forcebalance::tinkerio::AbInitio\_TINKER, 203  
**FrameVariableNames**  
 forcebalance::molecule, 42  
**frcmod\_pdict**  
 forcebalance::amberio, 18  
**fref**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 115  
 forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::openmmio::AbInitio\_OpenMM, 181  
 forcebalance::tinkerio::AbInitio\_TINKER, 203  
**Full**  
 forcebalance::objective::Objective, 624  
**Funnel**  
 forcebalance::molecule::Molecule, 558  
**GBSfnm**  
 forcebalance::psi4io::THCDF\_Psi4, 730  
**GOODSTEP**  
 forcebalance::optimizer, 63  
**gas\_engine**  
 forcebalance::gmxio::Lipid\_GMX, 444  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::lipid::Lipid, 426  
 forcebalance::liquid::Liquid, 462  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**gas\_engine\_args**  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**gas\_mol**  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::liquid::Liquid, 462  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**gct**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 159  
 forcebalance::amberio::AbInitio\_AMBER, 116  
 forcebalance::binding::BindingEnergy, 225  
 forcebalance::counterpoise::Counterpoise, 289

forcebalance::gmxio::AbInitio\_GMX, 137  
 forcebalance::gmxio::BindingEnergy\_GMX, 240  
 forcebalance::gmxio::Interaction\_GMX, 358  
 forcebalance::gmxio::Lipid\_GMX, 444  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::gmxio::Moments\_GMX, 588  
 forcebalance::gmxio::Thermo\_GMX, 763  
 forcebalance::gmxio::Vibration\_GMX, 805  
 forcebalance::interaction::Interaction, 343  
 forcebalance::leastsq::LeastSquares, 407  
 forcebalance::lipid::Lipid, 426  
 forcebalance::liquid::Liquid, 462  
 forcebalance::moments::Moments, 573  
 forcebalance::openmmio::AbInitio\_OpenMM, 181  
 forcebalance::openmmio::BindingEnergy\_OpenMM, 255  
 forcebalance::openmmio::Interaction\_OpenMM, 373  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::openmmio::Moments\_OpenMM, 603  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
 forcebalance::psi4io::THCDF\_Psi4, 730  
 forcebalance::target::RemoteTarget, 699  
 forcebalance::target::Target, 714  
 forcebalance::thermo::Thermo, 745  
 forcebalance::tinkerio::AbInitio\_TINKER, 203  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 270  
 forcebalance::tinkerio::Interaction\_TINKER, 388  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
 forcebalance::tinkerio::Moments\_TINKER, 618  
 forcebalance::tinkerio::Vibration\_TINKER, 820  
 forcebalance::vibration::Vibration, 790  
 gen\_opts\_defaults  
     forcebalance::parser, 66  
 gen\_opts\_types  
     forcebalance::parser, 66  
 generate\_vsites\_positions  
     forcebalance::gmxio::GMX, 322  
 GeneticAlgorithm  
     forcebalance::optimizer::Optimizer, 645  
 get  
     forcebalance::abinitio::AbInitio, 81  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 148  
     forcebalance::amberio::AbInitio\_AMBER, 104  
     forcebalance::binding::BindingEnergy, 216  
     forcebalance::counterpoise::Counterpoise, 278  
     forcebalance::gmxio::AbInitio\_GMX, 126  
     forcebalance::gmxio::BindingEnergy\_GMX, 231  
     forcebalance::gmxio::Interaction\_GMX, 349  
     forcebalance::gmxio::Lipid\_GMX, 433  
     forcebalance::gmxio::Liquid\_GMX, 468  
     forcebalance::gmxio::Moments\_GMX, 579  
     forcebalance::gmxio::Thermo\_GMX, 752  
     forcebalance::gmxio::Vibration\_GMX, 796  
     forcebalance::interaction::Interaction, 334  
 forcebalance::leastsq::LeastSquares, 397  
 forcebalance::lipid::Lipid, 415  
 forcebalance::liquid::Liquid, 450  
 forcebalance::moments::Moments, 564  
 forcebalance::openmmio::AbInitio\_OpenMM, 170  
 forcebalance::openmmio::BindingEnergy\_OpenMM, 246  
 forcebalance::openmmio::Interaction\_OpenMM, 364  
 forcebalance::openmmio::Liquid\_OpenMM, 488  
 forcebalance::openmmio::Moments\_OpenMM, 594  
 forcebalance::psi4io::RDVR3\_Psi4, 675  
 forcebalance::psi4io::THCDF\_Psi4, 719  
 forcebalance::target::RemoteTarget, 690  
 forcebalance::target::Target, 705  
 forcebalance::thermo::Thermo, 736  
 forcebalance::tinkerio::AbInitio\_TINKER, 192  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 261  
 forcebalance::tinkerio::Interaction\_TINKER, 379  
 forcebalance::tinkerio::Liquid\_TINKER, 507  
 forcebalance::tinkerio::Moments\_TINKER, 609  
 forcebalance::tinkerio::Vibration\_TINKER, 811  
 forcebalance::vibration::Vibration, 781  
 get\_G  
     forcebalance::abinitio::AbInitio, 83  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 149  
     forcebalance::amberio::AbInitio\_AMBER, 105  
     forcebalance::binding::BindingEnergy, 216  
     forcebalance::counterpoise::Counterpoise, 279  
     forcebalance::gmxio::AbInitio\_GMX, 127  
     forcebalance::gmxio::BindingEnergy\_GMX, 231  
     forcebalance::gmxio::Interaction\_GMX, 349  
     forcebalance::gmxio::Lipid\_GMX, 434  
     forcebalance::gmxio::Liquid\_GMX, 470  
     forcebalance::gmxio::Moments\_GMX, 579  
     forcebalance::gmxio::Thermo\_GMX, 753  
     forcebalance::gmxio::Vibration\_GMX, 796  
     forcebalance::interaction::Interaction, 334  
     forcebalance::leastsq::LeastSquares, 398  
     forcebalance::lipid::Lipid, 416  
     forcebalance::liquid::Liquid, 451  
     forcebalance::moments::Moments, 564  
     forcebalance::openmmio::AbInitio\_OpenMM, 171  
     forcebalance::openmmio::BindingEnergy\_OpenMM, 246  
     forcebalance::openmmio::Interaction\_OpenMM, 364  
     forcebalance::openmmio::Liquid\_OpenMM, 489  
     forcebalance::openmmio::Moments\_OpenMM, 594  
     forcebalance::psi4io::RDVR3\_Psi4, 676  
     forcebalance::psi4io::THCDF\_Psi4, 721  
     forcebalance::target::RemoteTarget, 691  
     forcebalance::target::Target, 705  
     forcebalance::thermo::Thermo, 736  
     forcebalance::tinkerio::AbInitio\_TINKER, 193  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 261

forcebalance::tinkerio::Interaction\_TINKER, 379  
 forcebalance::tinkerio::Liquid\_TINKER, 509  
 forcebalance::tinkerio::Moments\_TINKER, 609  
 forcebalance::tinkerio::Vibration\_TINKER, 811  
 forcebalance::vibration::Vibration, 781  
**get\_H**  
     forcebalance::abinitio::AbInitio, 84  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 150  
     forcebalance::amberio::AbInitio\_AMBER, 106  
     forcebalance::binding::BindingEnergy, 217  
     forcebalance::counterpoise::Counterpoise, 280  
     forcebalance::gmxio::AbInitio\_GMX, 128  
     forcebalance::gmxio::BindingEnergy\_GMX, 232  
     forcebalance::gmxio::Interaction\_GMX, 350  
     forcebalance::gmxio::Lipid\_GMX, 435  
     forcebalance::gmxio::Liquid\_GMX, 471  
     forcebalance::gmxio::Moments\_GMX, 580  
     forcebalance::gmxio::Thermo\_GMX, 754  
     forcebalance::gmxio::Vibration\_GMX, 797  
     forcebalance::interaction::Interaction, 335  
     forcebalance::leastsq::LeastSquares, 399  
     forcebalance::lipid::Lipid, 417  
     forcebalance::liquid::Liquid, 452  
     forcebalance::moments::Moments, 565  
     forcebalance::openmmio::AbInitio\_OpenMM, 172  
     forcebalance::openmmio::BindingEnergy\_OpenMM, 247  
     forcebalance::openmmio::Interaction\_OpenMM, 365  
     forcebalance::openmmio::Liquid\_OpenMM, 490  
     forcebalance::openmmio::Moments\_OpenMM, 595  
     forcebalance::psi4io::RDVR3\_Psi4, 677  
     forcebalance::psi4io::THCDF\_Psi4, 722  
     forcebalance::target::RemoteTarget, 691  
     forcebalance::target::Target, 706  
     forcebalance::thermo::Thermo, 737  
     forcebalance::tinkerio::AbInitio\_TINKER, 194  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 262  
     forcebalance::tinkerio::Interaction\_TINKER, 380  
     forcebalance::tinkerio::Liquid\_TINKER, 510  
     forcebalance::tinkerio::Moments\_TINKER, 610  
     forcebalance::tinkerio::Vibration\_TINKER, 812  
     forcebalance::vibration::Vibration, 782  
**get\_X**  
     forcebalance::abinitio::AbInitio, 85  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 151  
     forcebalance::amberio::AbInitio\_AMBER, 107  
     forcebalance::binding::BindingEnergy, 218  
     forcebalance::counterpoise::Counterpoise, 281  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::gmxio::BindingEnergy\_GMX, 233  
     forcebalance::gmxio::Interaction\_GMX, 351  
     forcebalance::gmxio::Lipid\_GMX, 436  
     forcebalance::gmxio::Liquid\_GMX, 472  
     forcebalance::gmxio::Moments\_GMX, 581  
  
 forcebalance::gmxio::Thermo\_GMX, 755  
 forcebalance::gmxio::Vibration\_GMX, 798  
 forcebalance::interaction::Interaction, 336  
 forcebalance::leastsq::LeastSquares, 400  
 forcebalance::lipid::Lipid, 418  
 forcebalance::liquid::Liquid, 453  
 forcebalance::moments::Moments, 566  
 forcebalance::openmmio::AbInitio\_OpenMM, 173  
 forcebalance::openmmio::BindingEnergy\_OpenMM, 248  
 forcebalance::openmmio::Interaction\_OpenMM, 366  
 forcebalance::openmmio::Liquid\_OpenMM, 491  
 forcebalance::openmmio::Moments\_OpenMM, 596  
 forcebalance::psi4io::RDVR3\_Psi4, 678  
 forcebalance::psi4io::THCDF\_Psi4, 723  
 forcebalance::target::RemoteTarget, 692  
 forcebalance::target::Target, 707  
 forcebalance::thermo::Thermo, 738  
 forcebalance::tinkerio::AbInitio\_TINKER, 195  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 263  
 forcebalance::tinkerio::Interaction\_TINKER, 381  
 forcebalance::tinkerio::Liquid\_TINKER, 511  
 forcebalance::tinkerio::Moments\_TINKER, 611  
 forcebalance::tinkerio::Vibration\_TINKER, 813  
 forcebalance::vibration::Vibration, 783  
**get\_atom**  
     forcebalance::Mol2::mol2, 524  
**get\_bonded\_atoms**  
     forcebalance::Mol2::mol2, 525  
**get\_dipole**  
     forcebalance::openmmio, 60  
**get\_energy\_force**  
     forcebalance::abinitio::AbInitio, 82  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 148  
     forcebalance::amberio::AbInitio\_AMBER, 104  
     forcebalance::gmxio::AbInitio\_GMX, 126  
     forcebalance::openmmio::AbInitio\_OpenMM, 170  
     forcebalance::tinkerio::AbInitio\_TINKER, 192  
**get\_least\_squares**  
     forcebalance::nifty, 49  
**get\_multipoles**  
     forcebalance::openmmio, 60  
**get\_resp**  
     forcebalance::abinitio::AbInitio, 85  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 151  
     forcebalance::amberio::AbInitio\_AMBER, 107  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmmio::AbInitio\_OpenMM, 173  
     forcebalance::tinkerio::AbInitio\_TINKER, 195  
**get\_rotate\_translate**  
     forcebalance::molecule, 40  
**getElement**  
     forcebalance::molecule, 40  
**GetVirtualSiteParameters**

```

    forcebalance::openmmio, 60
getWQIds
    forcebalance::nifty, 49
getWorkQueue
    forcebalance::nifty, 49
gmx_defs
    forcebalance::gmlio::GMX, 326
gmlio.py, 827
gmxpath
    forcebalance::gmlio::GMX, 326
gmxsuffix
    forcebalance::gmlio::GMX, 326
GolTo
    forcebalance::nifty, 50
GoodStep
    forcebalance::optimizer, 63
Gp
    forcebalance::gmlio::Lipid_GMX, 444
    forcebalance::gmlio::Liquid_GMX, 481
    forcebalance::gmlio::Thermo_GMX, 763
    forcebalance::lipid::Lipid, 426
    forcebalance::liquid::Liquid, 462
    forcebalance::openmmio::Liquid_OpenMM, 500
    forcebalance::thermo::Thermo, 745
    forcebalance::tinkerio::Liquid_TINKER, 520
Grad
    forcebalance::optimizer::Optimizer, 652
gradd
    forcebalance::psi4io::RDVR3_Psi4, 685
Gradient
    forcebalance::optimizer::Optimizer, 645
group
    forcebalance::molecule, 40
    forcebalance::nifty, 50

H
    forcebalance::optimizer::Optimizer, 652
h
    forcebalance::optimizer::Optimizer, 652
HYP
    forcebalance::objective::Penalty, 655
have_contact
    forcebalance::molecule, 43
hct
    forcebalance::abinitio::AbInitio, 93
    forcebalance::abinitio_internal::AbInitio_Internal, 160
    forcebalance::amber::AbInitio_AMBER, 116
    forcebalance::binding::BindingEnergy, 225
    forcebalance::counterpoise::Counterpoise, 289
    forcebalance::gmlio::AbInitio_GMX, 137
    forcebalance::gmlio::BindingEnergy_GMX, 240
    forcebalance::gmlio::Interaction_GMX, 358
    forcebalance::gmlio::Lipid_GMX, 444
    forcebalance::gmlio::Liquid_GMX, 481
forcebalance::gmlio::Moments_GMX, 588
forcebalance::gmlio::Thermo_GMX, 763
forcebalance::gmlio::Vibration_GMX, 805
forcebalance::interaction::Interaction, 343
forcebalance::leastsq::LeastSquares, 407
forcebalance::lipid::Lipid, 426
forcebalance::liquid::Liquid, 462
forcebalance::moments::Moments, 573
forcebalance::openmmio::AbInitio_OpenMM, 181
forcebalance::openmmio::BindingEnergy_OpenMM,
    255
forcebalance::openmmio::Interaction_OpenMM, 373
forcebalance::openmmio::Liquid_OpenMM, 500
forcebalance::openmmio::Moments_OpenMM, 603
forcebalance::psi4io::RDVR3_Psi4, 685
forcebalance::psi4io::THCDF_Psi4, 730
forcebalance::target::RemoteTarget, 699
forcebalance::target::Target, 714
forcebalance::thermo::Thermo, 746
forcebalance::tinkerio::AbInitio_TINKER, 203
forcebalance::tinkerio::BindingEnergy_TINKER, 270
forcebalance::tinkerio::Interaction_TINKER, 388
forcebalance::tinkerio::Liquid_TINKER, 520
forcebalance::tinkerio::Moments_TINKER, 618
forcebalance::tinkerio::Vibration_TINKER, 820
forcebalance::vibration::Vibration, 790
hdiagd
    forcebalance::psi4io::RDVR3_Psi4, 685
Hess
    forcebalance::optimizer::Optimizer, 652
Hessian
    forcebalance::optimizer::Optimizer, 645

ITERATION
    forcebalance::optimizer, 63
ITERINIT
    forcebalance::optimizer, 63
idnr
    forcebalance::thermo::Point, 659
Implemented.Targets
    forcebalance::objective, 57
in_fd
    forcebalance::finite_difference, 26
in_fd_src
    forcebalance::finite_difference, 27
Indicate
    forcebalance::objective::Objective, 625
indicate
    forcebalance::abinitio::AbInitio, 86
    forcebalance::abinitio_internal::AbInitio_Internal, 152
    forcebalance::amber::AbInitio_AMBER, 108
    forcebalance::binding::BindingEnergy, 219
    forcebalance::gmlio::AbInitio_GMX, 130
    forcebalance::gmlio::BindingEnergy_GMX, 234

```

forcebalance::gmxio::Interaction\_GMX, 352  
 forcebalance::gmxio::Lipid\_GMX, 437  
 forcebalance::gmxio::Liquid\_GMX, 473  
 forcebalance::gmxio::Moments\_GMX, 582  
 forcebalance::gmxio::Thermo\_GMX, 756  
 forcebalance::gmxio::Vibration\_GMX, 799  
 forcebalance::interaction::Interaction, 337  
 forcebalance::leastsq::LeastSquares, 401  
 forcebalance::lipid::Lipid, 419  
 forcebalance::liquid::Liquid, 454  
 forcebalance::moments::Moments, 567  
 forcebalance::openmmio::AbInitio\_OpenMM, 174  
 forcebalance::openmmio::BindingEnergy\_OpenMM, 249  
 forcebalance::openmmio::Interaction\_OpenMM, 367  
 forcebalance::openmmio::Liquid\_OpenMM, 492  
 forcebalance::openmmio::Moments\_OpenMM, 597  
 forcebalance::psi4io::RDVR3\_Psi4, 679  
 forcebalance::psi4io::THCDF\_Psi4, 724  
 forcebalance::target::RemoteTarget, 693  
 forcebalance::thermo::Thermo, 739  
 forcebalance::tinkerio::AbInitio\_TINKER, 196  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 264  
 forcebalance::tinkerio::Interaction\_TINKER, 382  
 forcebalance::tinkerio::Liquid\_TINKER, 512  
 forcebalance::tinkerio::Moments\_TINKER, 612  
 forcebalance::tinkerio::Vibration\_TINKER, 814  
 forcebalance::vibration::Vibration, 784  
 inter\_opts  
     forcebalance::binding::BindingEnergy, 225  
     forcebalance::gmxio::BindingEnergy\_GMX, 240  
     forcebalance::openmmio::BindingEnergy\_OpenMM, 255  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 270  
 interaction.py, 828  
 interaction\_energy  
     forcebalance::gmxio::GMX, 322  
     forcebalance::openmmio::OpenMM, 632  
     forcebalance::tinkerio::TINKER, 769  
 invdists  
     forcebalance::abinitio::AbInitio, 93  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
     forcebalance::amberio::AbInitio\_AMBER, 116  
     forcebalance::gmxio::AbInitio\_GMX, 138  
     forcebalance::openmmio::AbInitio\_OpenMM, 182  
     forcebalance::tinkerio::AbInitio\_TINKER, 204  
 invert\_svd  
     forcebalance::nifty, 50  
 iocc  
     forcebalance::parser, 66  
 is\_charmm\_coord  
     forcebalance::molecule, 40  
 is\_gro\_box  
     forcebalance::molecule, 41  
 is\_gro\_coord  
     forcebalance::molecule, 41  
 is\_mol2\_atom  
     forcebalance::amberio, 18  
 isdata  
     forcebalance::psi4io::GBS\_Reader, 313  
     forcebalance::psi4io::Grid\_Reader, 329  
 isdecimal  
     forcebalance::nifty, 50  
 isfloat  
     forcebalance::molecule, 42  
     forcebalance::nifty, 50  
 isint  
     forcebalance::molecule, 42  
     forcebalance::nifty, 50  
 itype  
     forcebalance::amberio::FrcMod\_Reader, 310  
     forcebalance::amberio::Mol2\_Reader, 537  
     forcebalance::BaseReader, 212  
     forcebalance::custom\_io::Gen\_Reader, 316  
     forcebalance::gmxio::ITP\_Reader, 393  
     forcebalance::mol2io::Mol2\_Reader, 535  
     forcebalance::openmmio::OpenMM\_Reader, 639  
     forcebalance::psi4io::GBS\_Reader, 313  
     forcebalance::psi4io::Grid\_Reader, 329  
     forcebalance::qchemio::QCIn\_Reader, 662  
     forcebalance::tinkerio::Tinker\_Reader, 776  
 kb  
     forcebalance::nifty, 56  
 key  
     forcebalance::tinkerio::TINKER, 772  
 L  
     forcebalance::chemistry, 20  
     forcebalance::molecule::MyG, 622  
 L2\_norm  
     forcebalance::objective::Penalty, 655  
 LAST\_MVALS  
     forcebalance::leastsq, 34  
 label  
     forcebalance::gmxio::Interaction\_GMX, 358  
     forcebalance::interaction::Interaction, 343  
     forcebalance::openmmio::Interaction\_OpenMM, 373  
     forcebalance::tinkerio::Interaction\_TINKER, 388  
 Labels  
     forcebalance::gmxio::Lipid\_GMX, 444  
     forcebalance::gmxio::Liquid\_GMX, 481  
     forcebalance::lipid::Lipid, 426  
     forcebalance::liquid::Liquid, 462  
     forcebalance::openmmio::Liquid\_OpenMM, 500  
     forcebalance::tinkerio::Liquid\_TINKER, 520  
 last\_amom  
     forcebalance::psi4io::GBS\_Reader, 313

**last\_traj**  
 forcebalance::gmxio::Lipid\_GMX, 444  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::lipid::Lipid, 426  
 forcebalance::liquid::Liquid, 462  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**LastMvals**  
 forcebalance::leastsq, 34  
**leastsq.py**, 828  
**Leave**  
 forcebalance::nifty, 51  
**Letters**  
 forcebalance::objective, 58  
**LfDict**  
 forcebalance::gmxio::Lipid\_GMX, 445  
 forcebalance::gmxio::Liquid\_GMX, 481  
**LfDict\_New**  
 forcebalance::gmxio::Lipid\_GMX, 445  
 forcebalance::gmxio::Liquid\_GMX, 481  
**line**  
 forcebalance::chemistry, 20  
**linedestroy\_save**  
 forcebalance::forcefield::FF, 304  
**linedestroy\_this**  
 forcebalance::forcefield::FF, 304  
**link\_dir\_contents**  
 forcebalance::nifty, 51  
**link\_from\_tempdir**  
 forcebalance::abinitio::AbInitio, 86  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 152  
 forcebalance::amberio::AbInitio\_AMBER, 108  
 forcebalance::binding::BindingEnergy, 219  
 forcebalance::counterpoise::Counterpoise, 282  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::gmxio::BindingEnergy\_GMX, 234  
 forcebalance::gmxio::Interaction\_GMX, 352  
 forcebalance::gmxio::Lipid\_GMX, 437  
 forcebalance::gmxio::Liquid\_GMX, 473  
 forcebalance::gmxio::Moments\_GMX, 582  
 forcebalance::gmxio::Thermo\_GMX, 756  
 forcebalance::gmxio::Vibration\_GMX, 799  
 forcebalance::interaction::Interaction, 337  
 forcebalance::leastsq::LeastSquares, 401  
 forcebalance::lipid::Lipid, 419  
 forcebalance::liquid::Liquid, 454  
 forcebalance::moments::Moments, 567  
 forcebalance::openmmio::AbInitio\_OpenMM, 174  
 forcebalance::openmmio::BindingEnergy\_OpenMM, 249  
 forcebalance::openmmio::Interaction\_OpenMM, 367  
 forcebalance::openmmio::Liquid\_OpenMM, 492  
 forcebalance::openmmio::Moments\_OpenMM, 597  
 forcebalance::psi4io::RDVR3\_Psi4, 679  
 forcebalance::psi4io::THCDF\_Psi4, 724  
 forcebalance::target::RemoteTarget, 693  
 forcebalance::target::Target, 708  
 forcebalance::thermo::Thermo, 739  
 forcebalance::tinkerio::AbInitio\_TINKER, 196  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 264  
 forcebalance::tinkerio::Interaction\_TINKER, 382  
 forcebalance::tinkerio::Liquid\_TINKER, 512  
 forcebalance::tinkerio::Moments\_TINKER, 612  
 forcebalance::tinkerio::Vibration\_TINKER, 814  
 forcebalance::vibration::Vibration, 784  
**LinkFile**  
 forcebalance::nifty, 51  
**links**  
 forcebalance::gmxio::GMX, 323  
**lipid.py**, 829  
**lipid\_mol**  
 forcebalance::gmxio::Lipid\_GMX, 445  
 forcebalance::lipid::Lipid, 426  
**liquid.py**, 829  
**liquid\_mol**  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::liquid::Liquid, 462  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**list\_map**  
 forcebalance::forcefield::FF, 301  
**In**  
 forcebalance::amberio::FrcMod\_Reader, 310  
 forcebalance::amberio::Mol2\_Reader, 537  
 forcebalance::BaseReader, 212  
 forcebalance::custom\_io::Gen\_Reader, 316  
 forcebalance::gmxio::ITP\_Reader, 393  
 forcebalance::mol2io::Mol2\_Reader, 535  
 forcebalance::openmmio::OpenMM\_Reader, 639  
 forcebalance::psi4io::GBS\_Reader, 313  
 forcebalance::psi4io::Grid\_Reader, 329  
 forcebalance::qcchemio::QCIn\_Reader, 662  
 forcebalance::tinkerio::Tinker\_Reader, 776  
**load\_cp**  
 forcebalance::counterpoise::Counterpoise, 282  
**load\_frames**  
 forcebalance::molecule::Molecule, 547  
**loadxyz**  
 forcebalance::counterpoise::Counterpoise, 282  
**logger**  
 forcebalance::abinitio, 16  
 forcebalance::amberio, 18  
 forcebalance::binding, 19  
 forcebalance::counterpoise, 22  
 forcebalance::engine, 24  
 forcebalance::finite\_difference, 27  
 forcebalance::forcefield, 29  
 forcebalance::gmxio, 33

forcebalance::interaction, 34  
 forcebalance::leastsq, 34  
 forcebalance::lipid, 35  
 forcebalance::liquid, 35  
 forcebalance::moments, 44  
 forcebalance::nifty, 56  
 forcebalance::objective, 58  
 forcebalance::openmmio, 61  
 forcebalance::optimizer, 63  
 forcebalance::parser, 66  
 forcebalance::psi4io, 68  
 forcebalance::qcchemio, 69  
 forcebalance::quantity, 71  
 forcebalance::target, 71  
 forcebalance::thermo, 71  
 forcebalance::tinkerio, 73  
 forcebalance::vibration, 74  
**LookupByMass**  
 forcebalance::chemistry, 20  
**lp\_dump**  
 forcebalance::nifty, 51  
**lp\_load**  
 forcebalance::nifty, 51  
**MAQ**  
 forcebalance::leastsq::LeastSquares, 407  
 forcebalance::psi4io::THCDF\_Psi4, 730  
**MBarEnergy**  
 forcebalance::gmxio::Lipid\_GMX, 445  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::lipid::Lipid, 427  
 forcebalance::liquid::Liquid, 462  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**MP2\_Energy**  
 forcebalance::psi4io::THCDF\_Psi4, 730  
**MTSVVRIntegrator**  
 forcebalance::openmmio, 60  
**main**  
 forcebalance::molecule, 42  
**MainOptimizer**  
 forcebalance::optimizer::Optimizer, 645  
**mainsections**  
 forcebalance::parser, 67  
**make**  
 forcebalance::forcefield::FF, 301  
**make\_redirect**  
 forcebalance::forcefield::FF, 302  
**map**  
 forcebalance::forcefield::FF, 304  
**mass**  
 forcebalance::openmmio::OpenMM, 635  
**maxrd**  
 forcebalance::abinitio::AbInitio, 87  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 153  
 forcebalance::amberio::AbInitio\_AMBER, 109  
 forcebalance::binding::BindingEnergy, 220  
 forcebalance::counterpoise::Counterpoise, 284  
 forcebalance::gmxio::AbInitio\_GMX, 131  
 forcebalance::gmxio::BindingEnergy\_GMX, 235  
 forcebalance::gmxio::Interaction\_GMX, 353  
 forcebalance::gmxio::Lipid\_GMX, 438  
 forcebalance::gmxio::Liquid\_GMX, 474  
 forcebalance::gmxio::Moments\_GMX, 583  
 forcebalance::gmxio::Thermo\_GMX, 757  
 forcebalance::gmxio::Vibration\_GMX, 800  
 forcebalance::interaction::Interaction, 338  
 forcebalance::leastsq::LeastSquares, 401  
 forcebalance::lipid::Lipid, 420  
 forcebalance::liquid::Liquid, 455  
 forcebalance::moments::Moments, 568  
 forcebalance::openmmio::AbInitio\_OpenMM, 175  
 forcebalance::openmmio::BindingEnergy\_OpenMM,  
     250  
 forcebalance::openmmio::Interaction\_OpenMM, 368  
 forcebalance::openmmio::Liquid\_OpenMM, 493  
 forcebalance::openmmio::Moments\_OpenMM, 598  
 forcebalance::psi4io::RDVR3\_Psi4, 680  
 forcebalance::psi4io::THCDF\_Psi4, 724  
 forcebalance::target::RemoteTarget, 693  
 forcebalance::target::Target, 708  
 forcebalance::thermo::Thermo, 740  
 forcebalance::tinkerio::AbInitio\_TINKER, 197  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 265  
 forcebalance::tinkerio::Interaction\_TINKER, 383  
 forcebalance::tinkerio::Liquid\_TINKER, 513  
 forcebalance::tinkerio::Moments\_TINKER, 613  
 forcebalance::tinkerio::Vibration\_TINKER, 815  
 forcebalance::vibration::Vibration, 785  
**md**  
 forcebalance::gmxio::GMX, 323  
**mdene**  
 forcebalance::gmxio::GMX, 326  
**mdp**  
 forcebalance::gmxio::GMX, 326  
**mdpx**  
 forcebalance::gmxio::Thermo\_GMX, 763  
**mdtraj**  
 forcebalance::gmxio::GMX, 326  
 forcebalance::tinkerio::TINKER, 772  
**mean\_stderr**  
 forcebalance::quantity, 70  
**measure\_dihedrals**  
 forcebalance::molecule::Molecule, 547  
**meta\_get**  
 forcebalance::abinitio::AbInitio, 87  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 153  
 forcebalance::amberio::AbInitio\_AMBER, 109

forcebalance::binding::BindingEnergy, 220  
 forcebalance::counterpoise::Counterpoise, 284  
 forcebalance::gmxio::AbInitio\_GMX, 131  
 forcebalance::gmxio::BindingEnergy\_GMX, 235  
 forcebalance::gmxio::Interaction\_GMX, 353  
 forcebalance::gmxio::Lipid\_GMX, 438  
 forcebalance::gmxio::Liquid\_GMX, 474  
 forcebalance::gmxio::Moments\_GMX, 583  
 forcebalance::gmxio::Thermo\_GMX, 757  
 forcebalance::gmxio::Vibration\_GMX, 800  
 forcebalance::interaction::Interaction, 338  
 forcebalance::leastsq::LeastSquares, 402  
 forcebalance::lipid::Lipid, 420  
 forcebalance::liquid::Liquid, 455  
 forcebalance::moments::Moments, 568  
 forcebalance::openmmio::AbInitio\_OpenMM, 175  
 forcebalance::openmmio::BindingEnergy\_OpenMM, 250  
 forcebalance::openmmio::Interaction\_OpenMM, 368  
 forcebalance::openmmio::Liquid\_OpenMM, 493  
 forcebalance::openmmio::Moments\_OpenMM, 598  
 forcebalance::psi4io::RDVR3\_Psi4, 680  
 forcebalance::psi4io::THCDF\_Psi4, 725  
 forcebalance::target::RemoteTarget, 694  
 forcebalance::target::Target, 709  
 forcebalance::thermo::Thermo, 740  
 forcebalance::tinkerio::AbInitio\_TINKER, 197  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 265  
 forcebalance::tinkerio::Interaction\_TINKER, 383  
 forcebalance::tinkerio::Liquid\_TINKER, 513  
 forcebalance::tinkerio::Moments\_TINKER, 613  
 forcebalance::tinkerio::Vibration\_TINKER, 815  
 forcebalance::vibration::Vibration, 785  
 meta\_indicate  
     forcebalance::abinitio::AbInitio, 88  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 154  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::binding::BindingEnergy, 221  
     forcebalance::counterpoise::Counterpoise, 285  
     forcebalance::gmxio::AbInitio\_GMX, 132  
     forcebalance::gmxio::BindingEnergy\_GMX, 236  
     forcebalance::gmxio::Interaction\_GMX, 354  
     forcebalance::gmxio::Lipid\_GMX, 439  
     forcebalance::gmxio::Liquid\_GMX, 475  
     forcebalance::gmxio::Moments\_GMX, 584  
     forcebalance::gmxio::Thermo\_GMX, 758  
     forcebalance::gmxio::Vibration\_GMX, 801  
     forcebalance::interaction::Interaction, 339  
     forcebalance::leastsq::LeastSquares, 403  
     forcebalance::lipid::Lipid, 421  
     forcebalance::liquid::Liquid, 456  
     forcebalance::moments::Moments, 569  
     forcebalance::openmmio::AbInitio\_OpenMM, 176  
 forcebalance::openmmio::BindingEnergy\_OpenMM, 251  
 forcebalance::openmmio::Interaction\_OpenMM, 369  
 forcebalance::openmmio::Liquid\_OpenMM, 494  
 forcebalance::openmmio::Moments\_OpenMM, 599  
 forcebalance::psi4io::RDVR3\_Psi4, 681  
 forcebalance::psi4io::THCDF\_Psi4, 726  
 forcebalance::target::RemoteTarget, 695  
 forcebalance::target::Target, 710  
 forcebalance::thermo::Thermo, 741  
 forcebalance::tinkerio::AbInitio\_TINKER, 198  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 266  
 forcebalance::tinkerio::Interaction\_TINKER, 384  
 forcebalance::tinkerio::Liquid\_TINKER, 514  
 forcebalance::tinkerio::Moments\_TINKER, 614  
 forcebalance::tinkerio::Vibration\_TINKER, 816  
 forcebalance::vibration::Vibration, 786  
 MetaVariableNames  
     forcebalance::molecule, 43  
 mfnm  
     forcebalance::gmxio::Moments\_GMX, 588  
     forcebalance::moments::Moments, 573  
     forcebalance::openmmio::Moments\_OpenMM, 603  
     forcebalance::tinkerio::Moments\_TINKER, 618  
 MissingFileInspection  
     forcebalance::nifty, 51  
 mktransmat  
     forcebalance::forcefield::FF, 302  
 mmopts  
     forcebalance::openmmio::OpenMM, 635  
 mod  
     forcebalance::openmmio::OpenMM, 635  
 mol  
     forcebalance::abinitio::AbInitio, 93  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
     forcebalance::amberio::AbInitio\_AMBER, 116  
     forcebalance::amberio::Mol2\_Reader, 537  
     forcebalance::gmxio::AbInitio\_GMX, 138  
     forcebalance::gmxio::GMX, 326  
     forcebalance::gmxio::Interaction\_GMX, 358  
     forcebalance::gmxio::ITP\_Reader, 393  
     forcebalance::interaction::Interaction, 343  
     forcebalance::openmmio::AbInitio\_OpenMM, 182  
     forcebalance::openmmio::Interaction\_OpenMM, 373  
     forcebalance::openmmio::OpenMM, 635  
     forcebalance::tinkerio::AbInitio\_TINKER, 204  
     forcebalance::tinkerio::Interaction\_TINKER, 388  
     forcebalance::tinkerio::TINKER, 772  
 Mol2.py, 829  
 mol2\_pdct  
     forcebalance::amberio, 18  
     forcebalance::mol2io, 36  
 mol2io.py, 830  
 mol\_name

forcebalance::Mol2::mol2, 527  
 mol\_type  
     forcebalance::Mol2::mol2, 527  
 molatom  
     forcebalance::amberio::FrcMod\_Reader, 310  
     forcebalance::amberio::Mol2\_Reader, 537  
     forcebalance::BaseReader, 212  
     forcebalance::custom\_io::Gen\_Reader, 316  
     forcebalance::gmxio::ITP\_Reader, 393  
     forcebalance::mol2io::Mol2\_Reader, 535  
     forcebalance::openmmio::OpenMM\_Reader, 639  
     forcebalance::psi4io::GBS\_Reader, 313  
     forcebalance::psi4io::Grid\_Reader, 329  
     forcebalance::qchemio::QCIn\_Reader, 662  
     forcebalance::tinkerio::Tinker\_Reader, 776  
 molecular\_dynamics  
     forcebalance::gmxio::GMX, 323  
     forcebalance::openmmio::OpenMM, 632  
     forcebalance::tinkerio::TINKER, 769  
 molecule.py, 830  
 Molecules  
     forcebalance::amberio::FrcMod\_Reader, 310  
     forcebalance::amberio::Mol2\_Reader, 537  
     forcebalance::BaseReader, 212  
     forcebalance::custom\_io::Gen\_Reader, 316  
     forcebalance::gmxio::ITP\_Reader, 393  
     forcebalance::mol2io::Mol2\_Reader, 535  
     forcebalance::openmmio::OpenMM\_Reader, 639  
     forcebalance::psi4io::GBS\_Reader, 313  
     forcebalance::psi4io::Grid\_Reader, 329  
     forcebalance::psi4io::THCDF\_Psi4, 730  
     forcebalance::qchemio::QCIn\_Reader, 662  
     forcebalance::tinkerio::Tinker\_Reader, 776  
 molecules  
     forcebalance::molecule::Molecule, 558  
     forcebalance::psi4io::RDVR3\_Psi4, 685  
 moments.py, 832  
 monotonic  
     forcebalance::nifty, 51  
 multiD\_statisticalInefficiency  
     forcebalance::nifty, 51  
 multipole\_moments  
     forcebalance::gmxio::GMX, 324  
     forcebalance::openmmio::OpenMM, 633  
     forcebalance::tinkerio::TINKER, 770  
 mvals0  
     forcebalance::optimizer::Optimizer, 652  
 mvals\_bak  
     forcebalance::optimizer::Optimizer, 652  
 n\_snaps  
     forcebalance::gmxio::GMX, 324  
 na  
     forcebalance::counterpoise::Counterpoise, 289  
 forcebalance::gmxio::Moments\_GMX, 588  
 forcebalance::gmxio::Vibration\_GMX, 805  
 forcebalance::moments::Moments, 573  
 forcebalance::openmmio::Moments\_OpenMM, 603  
 forcebalance::tinkerio::Moments\_TINKER, 618  
 forcebalance::tinkerio::Vibration\_TINKER, 820  
 forcebalance::vibration::Vibration, 790  
 name  
     forcebalance::engine::Engine, 292  
     forcebalance::gmxio::GMX, 326  
     forcebalance::openmmio::OpenMM, 635  
     forcebalance::quantity::Quantity, 664  
     forcebalance::quantity::Quantity\_Density, 666  
     forcebalance::quantity::Quantity\_H\_vap, 668  
     forcebalance::tinkerio::TINKER, 772  
 nbcharges  
     forcebalance::openmmio::OpenMM, 635  
 nbtype  
     forcebalance::gmxio::ITP\_Reader, 393  
 ndof  
     forcebalance::openmmio::OpenMM, 635  
 ndtypes  
     forcebalance::custom\_io, 23  
     forcebalance::qchemio, 69  
 nesp  
     forcebalance::abinitio::AbInitio, 94  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
     forcebalance::amberio::AbInitio\_AMBER, 116  
     forcebalance::gmxio::AbInitio\_GMX, 138  
     forcebalance::openmmio::AbInitio\_OpenMM, 182  
     forcebalance::tinkerio::AbInitio\_TINKER, 204  
 new\_vsites  
     forcebalance::abinitio::AbInitio, 94  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
     forcebalance::amberio::AbInitio\_AMBER, 116  
     forcebalance::gmxio::AbInitio\_GMX, 138  
     forcebalance::openmmio::AbInitio\_OpenMM, 182  
     forcebalance::tinkerio::AbInitio\_TINKER, 204  
 NewtonCG  
     forcebalance::optimizer::Optimizer, 646  
 NewtonRaphson  
     forcebalance::optimizer::Optimizer, 646  
 nf\_err  
     forcebalance::abinitio::AbInitio, 94  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
     forcebalance::amberio::AbInitio\_AMBER, 116  
     forcebalance::gmxio::AbInitio\_GMX, 138  
     forcebalance::openmmio::AbInitio\_OpenMM, 182  
     forcebalance::tinkerio::AbInitio\_TINKER, 204  
 nf\_err\_pct  
     forcebalance::abinitio::AbInitio, 94  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
     forcebalance::amberio::AbInitio\_AMBER, 116  
     forcebalance::gmxio::AbInitio\_GMX, 138

forcebalance::openmmio::AbInitio\_OpenMM, 182  
 forcebalance::tinkerio::AbInitio\_TINKER, 204  
**nf\_ref**  
 forcebalance::abinitio::AbInitio, 94  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
 forcebalance::amberio::AbInitio\_AMBER, 116  
 forcebalance::gmxio::AbInitio\_GMX, 138  
 forcebalance::openmmio::AbInitio\_OpenMM, 182  
 forcebalance::tinkerio::AbInitio\_TINKER, 204  
**nftqm**  
 forcebalance::abinitio::AbInitio, 94  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
 forcebalance::amberio::AbInitio\_AMBER, 116  
 forcebalance::gmxio::AbInitio\_GMX, 138  
 forcebalance::openmmio::AbInitio\_OpenMM, 182  
 forcebalance::tinkerio::AbInitio\_TINKER, 204  
**nftytes**  
 forcebalance::gmxio, 33  
**nifty.py**, 832  
**nnf**  
 forcebalance::abinitio::AbInitio, 94  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
 forcebalance::amberio::AbInitio\_AMBER, 116  
 forcebalance::gmxio::AbInitio\_GMX, 138  
 forcebalance::openmmio::AbInitio\_OpenMM, 182  
 forcebalance::tinkerio::AbInitio\_TINKER, 204  
**nodematch**  
 forcebalance::molecule, 42  
**nomnom**  
 forcebalance::nifty::LineChunker, 410  
**normal\_modes**  
 forcebalance::gmxio::GMX, 324  
 forcebalance::openmmio::OpenMM, 633  
 forcebalance::tinkerio::TINKER, 770  
**np**  
 forcebalance::forcefield::FF, 304  
 forcebalance::optimizer::Optimizer, 652  
**nparticles**  
 forcebalance::abinitio::AbInitio, 94  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
 forcebalance::amberio::AbInitio\_AMBER, 116  
 forcebalance::gmxio::AbInitio\_GMX, 138  
 forcebalance::openmmio::AbInitio\_OpenMM, 182  
 forcebalance::tinkerio::AbInitio\_TINKER, 204  
**npt\_simulation**  
 forcebalance::gmxio::Lipid\_GMX, 440  
 forcebalance::gmxio::Liquid\_GMX, 476  
 forcebalance::lipid::Lipid, 422  
 forcebalance::liquid::Liquid, 457  
 forcebalance::openmmio::Liquid\_OpenMM, 495  
 forcebalance::tinkerio::Liquid\_TINKER, 515  
**nptfiles**  
 forcebalance::gmxio::Lipid\_GMX, 445  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**nptpx**  
 forcebalance::gmxio::Lipid\_GMX, 445  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::tinkerio::Liquid\_TINKER, 520  
**ns**  
 forcebalance::abinitio::AbInitio, 94  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
 forcebalance::amberio::AbInitio\_AMBER, 116  
 forcebalance::counterpoise::Counterpoise, 289  
 forcebalance::gmxio::AbInitio\_GMX, 138  
 forcebalance::gmxio::Interaction\_GMX, 358  
 forcebalance::interaction::Interaction, 343  
 forcebalance::openmmio::AbInitio\_OpenMM, 182  
 forcebalance::openmmio::Interaction\_OpenMM, 373  
 forcebalance::tinkerio::AbInitio\_TINKER, 204  
 forcebalance::tinkerio::Interaction\_TINKER, 388  
**ntq**  
 forcebalance::abinitio::AbInitio, 94  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
 forcebalance::amberio::AbInitio\_AMBER, 116  
 forcebalance::gmxio::AbInitio\_GMX, 138  
 forcebalance::openmmio::AbInitio\_OpenMM, 182  
 forcebalance::tinkerio::AbInitio\_TINKER, 204  
**num\_atoms**  
 forcebalance::Mol2::mol2, 527  
**num\_bonds**  
 forcebalance::Mol2::mol2, 527  
**num\_compounds**  
 forcebalance::Mol2::mol2\_set, 538  
**num\_feat**  
 forcebalance::Mol2::mol2, 527  
**num\_sets**  
 forcebalance::Mol2::mol2, 527  
**num\_subst**  
 forcebalance::Mol2::mol2, 527  
**ObjDict**  
 forcebalance::objective::Objective, 625  
**ObjDict\_Last**  
 forcebalance::objective::Objective, 625  
**objd**  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
**Objective**  
 forcebalance::gmxio::Lipid\_GMX, 445  
 forcebalance::gmxio::Liquid\_GMX, 481  
 forcebalance::gmxio::Thermo\_GMX, 763  
 forcebalance::lipid::Lipid, 427  
 forcebalance::liquid::Liquid, 462  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::optimizer::Optimizer, 652  
 forcebalance::thermo::Thermo, 746

forcebalance::tinkerio::Liquid\_TINKER, 520  
**objective**  
 forcebalance::abinitio::AbInitio, 94  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
 forcebalance::amberio::AbInitio\_AMBER, 116  
 forcebalance::binding::BindingEnergy, 225  
 forcebalance::gmxio::AbInitio\_GMX, 138  
 forcebalance::gmxio::BindingEnergy\_GMX, 240  
 forcebalance::gmxio::Interaction\_GMX, 358  
 forcebalance::gmxio::Moments\_GMX, 588  
 forcebalance::gmxio::Vibration\_GMX, 805  
 forcebalance::interaction::Interaction, 343  
 forcebalance::leastsq::LeastSquares, 407  
 forcebalance::moments::Moments, 573  
 forcebalance::openmmio::AbInitio\_OpenMM, 182  
 forcebalance::openmmio::BindingEnergy\_OpenMM, 255  
 forcebalance::openmmio::Interaction\_OpenMM, 373  
 forcebalance::openmmio::Moments\_OpenMM, 603  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
 forcebalance::psi4io::THCDF\_Psi4, 730  
 forcebalance::tinkerio::AbInitio\_TINKER, 204  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 270  
 forcebalance::tinkerio::Interaction\_TINKER, 388  
 forcebalance::tinkerio::Moments\_TINKER, 618  
 forcebalance::tinkerio::Vibration\_TINKER, 820  
 forcebalance::vibration::Vibration, 790  
**objective.py**, 834  
**objective\_term**  
 forcebalance::gmxio::Lipid\_GMX, 440  
 forcebalance::gmxio::Liquid\_GMX, 476  
 forcebalance::gmxio::Thermo\_GMX, 759  
 forcebalance::lipid::Lipid, 422  
 forcebalance::liquid::Liquid, 457  
 forcebalance::openmmio::Liquid\_OpenMM, 495  
 forcebalance::thermo::Thermo, 742  
 forcebalance::tinkerio::Liquid\_TINKER, 515  
**objfiles**  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
**objvals**  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
**onefile**  
 forcebalance::nifty, 52  
**openmm\_boxes**  
 forcebalance::molecule::Molecule, 547  
**openmm\_positions**  
 forcebalance::molecule::Molecule, 547  
**openmmio.py**, 835  
**openmmxml**  
 forcebalance::forcefield::FF, 304  
**OptTab**  
 forcebalance::optimizer::Optimizer, 653  
**optimize**  
 forcebalance::gmxio::GMX, 324  
 forcebalance::openmmio::OpenMM, 633  
 forcebalance::tinkerio::TINKER, 771  
**optimizer.py**, 836  
**origin\_atom\_id**  
 forcebalance::Mol2::mol2\_bond, 532  
**orthogonalize**  
 forcebalance::nifty, 52  
**out**  
 forcebalance::Mol2::mol2, 525  
**output.py**, 837  
**overlaps**  
 forcebalance::gmxio::Vibration\_GMX, 805  
 forcebalance::tinkerio::Vibration\_TINKER, 820  
 forcebalance::vibration::Vibration, 790  
**PT.py**, 838  
**ParsTab**  
 forcebalance::parser, 67  
**parse**  
 forcebalance::Mol2::mol2, 525  
 forcebalance::Mol2::mol2\_atom, 528  
 forcebalance::Mol2::mol2\_bond, 531  
 forcebalance::Mol2::mol2\_set, 538  
**parse\_atomtype.line**  
 forcebalance::gmxio, 31  
**parse\_inputs**  
 forcebalance::parser, 65  
**parse\_interactions**  
 forcebalance::binding, 19  
**parser.py**, 837  
**pathwise\_rmsd**  
 forcebalance::molecule::Molecule, 547  
**patoms**  
 forcebalance::forcefield::FF, 304  
**pbc**  
 forcebalance::gmxio::GMX, 326  
 forcebalance::openmmio::OpenMM, 635  
 forcebalance::tinkerio::TINKER, 772  
**pdb**  
 forcebalance::openmmio::OpenMM, 635  
**pdict**  
 forcebalance::amberio::FrcMod\_Reader, 310  
 forcebalance::amberio::Mol2\_Reader, 537  
 forcebalance::BaseReader, 212  
 forcebalance::custom\_io, 23  
 forcebalance::custom\_io::Gen\_Reader, 316  
 forcebalance::gmxio, 33  
 forcebalance::gmxio::ITP\_Reader, 394  
 forcebalance::mol2io::Mol2\_Reader, 535  
 forcebalance::openmmio, 61  
 forcebalance::openmmio::OpenMM\_Reader, 639  
 forcebalance::psi4io::GBS\_Reader, 313  
 forcebalance::psi4io::Grid\_Reader, 329  
 forcebalance::qchemio, 69

forcebalance::qchemio::QCIn\_Reader, 662  
 forcebalance::tinkerio, 73  
 forcebalance::tinkerio::Tinker\_Reader, 776  
**Pen\_Names**  
 forcebalance::objective::Penalty, 656  
**Pen\_Tab**  
 forcebalance::objective::Penalty, 656  
**Penalty**  
 forcebalance::objective::Objective, 625  
 forcebalance::optimizer::Optimizer, 653  
**PeriodicTable**  
 forcebalance::chemistry, 20  
 forcebalance::molecule, 43  
 forcebalance::PT, 68  
**pfields**  
 forcebalance::forcefield::FF, 305  
**ptypes**  
 forcebalance::gmxio, 33  
**pgrad**  
 forcebalance::abinitio::AbInitio, 94  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
 forcebalance::amber::AbInitio\_AMBER, 116  
 forcebalance::binding::BindingEnergy, 225  
 forcebalance::counterpoise::Counterpoise, 289  
 forcebalance::gmxio::AbInitio\_GMX, 138  
 forcebalance::gmxio::BindingEnergy\_GMX, 240  
 forcebalance::gmxio::Interaction\_GMX, 358  
 forcebalance::gmxio::Lipid\_GMX, 445  
 forcebalance::gmxio::Liquid\_GMX, 482  
 forcebalance::gmxio::Moments\_GMX, 588  
 forcebalance::gmxio::Thermo\_GMX, 763  
 forcebalance::gmxio::Vibration\_GMX, 805  
 forcebalance::interaction::Interaction, 343  
 forcebalance::leastsq::LeastSquares, 407  
 forcebalance::lipid::Lipid, 427  
 forcebalance::liquid::Liquid, 462  
 forcebalance::moments::Moments, 573  
 forcebalance::openmmio::AbInitio\_OpenMM, 182  
 forcebalance::openmmio::BindingEnergy\_OpenMM, 255  
 forcebalance::openmmio::Interaction\_OpenMM, 373  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::openmmio::Moments\_OpenMM, 603  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
 forcebalance::psi4io::THCDF\_Psi4, 730  
 forcebalance::target::RemoteTarget, 699  
 forcebalance::target::Target, 714  
 forcebalance::thermo::Thermo, 746  
 forcebalance::tinkerio::AbInitio\_TINKER, 204  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 270  
 forcebalance::tinkerio::Interaction\_TINKER, 388  
 forcebalance::tinkerio::Liquid\_TINKER, 521  
 forcebalance::tinkerio::Moments\_TINKER, 618  
 forcebalance::tinkerio::Vibration\_TINKER, 820  
 forcebalance::vibration::Vibration, 790  
**PhasePoints**  
 forcebalance::gmxio::Lipid\_GMX, 445  
 forcebalance::gmxio::Liquid\_GMX, 482  
 forcebalance::lipid::Lipid, 427  
 forcebalance::liquid::Liquid, 462  
 forcebalance::openmmio::Liquid\_OpenMM, 500  
 forcebalance::tinkerio::Liquid\_TINKER, 521  
**platform**  
 forcebalance::openmmio::OpenMM, 635  
**platname**  
 forcebalance::openmmio::OpenMM, 635  
**plist**  
 forcebalance::forcefield::FF, 305  
**pmat2d**  
 forcebalance::nifty, 52  
**point**  
 forcebalance::psi4io::Grid\_Reader, 329  
**points**  
 forcebalance::gmxio::Thermo\_GMX, 763  
 forcebalance::thermo::Thermo, 746  
**polarization\_correction**  
 forcebalance::gmxio::Lipid\_GMX, 440  
 forcebalance::gmxio::Liquid\_GMX, 476  
 forcebalance::lipid::Lipid, 422  
 forcebalance::liquid::Liquid, 457  
 forcebalance::openmmio::Liquid\_OpenMM, 495  
 forcebalance::tinkerio::Liquid\_TINKER, 515  
**positive\_resid**  
 forcebalance::molecule::Molecule, 558  
**Powell**  
 forcebalance::optimizer::Optimizer, 647  
**Pp**  
 forcebalance::gmxio::Lipid\_GMX, 445  
 forcebalance::gmxio::Liquid\_GMX, 482  
 forcebalance::gmxio::Thermo\_GMX, 763  
 forcebalance::lipid::Lipid, 427  
 forcebalance::liquid::Liquid, 462  
 forcebalance::openmmio::Liquid\_OpenMM, 501  
 forcebalance::thermo::Thermo, 746  
 forcebalance::tinkerio::Liquid\_TINKER, 521  
**precision**  
 forcebalance::openmmio::OpenMM, 635  
**prefactor**  
 forcebalance::gmxio::Interaction\_GMX, 358  
 forcebalance::interaction::Interaction, 343  
 forcebalance::openmmio::Interaction\_OpenMM, 373  
 forcebalance::tinkerio::Interaction\_TINKER, 388  
**prepare**  
 forcebalance::engine::Engine, 292  
 forcebalance::gmxio::GMX, 325  
 forcebalance::openmmio::OpenMM, 634  
 forcebalance::tinkerio::TINKER, 771  
**prepare\_temp\_directory**

forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::gmxio::Lipid\_GMX, 441  
 forcebalance::gmxio::Liquid\_GMX, 477  
 forcebalance::lipid::Lipid, 423  
 forcebalance::liquid::Liquid, 458  
 forcebalance::openmmio::Liquid\_OpenMM, 496  
 forcebalance::psi4io::THCDF\_Psi4, 727  
 forcebalance::tinkerio::Liquid\_TINKER, 516  
**PrepareVirtualSites**  
     forcebalance::openmmio, 61  
**pressure**  
     forcebalance::quantity::Quantity, 664  
     forcebalance::quantity::Quantity\_Density, 666  
     forcebalance::quantity::Quantity\_H\_vap, 668  
     forcebalance::thermo::Point, 659  
**prev\_bad**  
     forcebalance::optimizer::Optimizer, 653  
**print\_map**  
     forcebalance::forcefield::FF, 302  
**PrintDict**  
     forcebalance::binding::BindingEnergy, 225  
     forcebalance::gmxio::BindingEnergy\_GMX, 240  
     forcebalance::openmmio::BindingEnergy\_OpenMM,  
         255  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 270  
**PrintOptionDict**  
     forcebalance::abinitio::AbInitio, 94  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
     forcebalance::amberio::AbInitio\_AMBER, 116  
     forcebalance::BaseClass, 209  
     forcebalance::binding::BindingEnergy, 225  
     forcebalance::counterpoise::Counterpoise, 289  
     forcebalance::engine::Engine, 292  
     forcebalance::forcefield::FF, 305  
     forcebalance::gmxio::AbInitio\_GMX, 138  
     forcebalance::gmxio::BindingEnergy\_GMX, 240  
     forcebalance::gmxio::GMX, 326  
     forcebalance::gmxio::Interaction\_GMX, 358  
     forcebalance::gmxio::Lipid\_GMX, 445  
     forcebalance::gmxio::Liquid\_GMX, 482  
     forcebalance::gmxio::Moments\_GMX, 588  
     forcebalance::gmxio::Thermo\_GMX, 763  
     forcebalance::gmxio::Vibration\_GMX, 806  
     forcebalance::interaction::Interaction, 343  
     forcebalance::leastsq::LeastSquares, 407  
     forcebalance::lipid::Lipid, 427  
     forcebalance::liquid::Liquid, 462  
     forcebalance::moments::Moments, 573  
     forcebalance::objective::Objective, 626  
     forcebalance::openmmio::AbInitio\_OpenMM, 182  
     forcebalance::openmmio::BindingEnergy\_OpenMM,  
         255  
     forcebalance::openmmio::Interaction\_OpenMM, 373  
     forcebalance::openmmio::Liquid\_OpenMM, 501

forcebalance::openmmio::Moments\_OpenMM, 603  
 forcebalance::openmmio::OpenMM, 636  
 forcebalance::optimizer::Optimizer, 653  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
 forcebalance::psi4io::THCDF\_Psi4, 731  
 forcebalance::target::RemoteTarget, 699  
 forcebalance::target::Target, 714  
 forcebalance::thermo::Thermo, 746  
 forcebalance::tinkerio::AbInitio\_TINKER, 204  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 270  
 forcebalance::tinkerio::Interaction\_TINKER, 388  
 forcebalance::tinkerio::Liquid\_TINKER, 521  
 forcebalance::tinkerio::Moments\_TINKER, 618  
 forcebalance::tinkerio::TINKER, 772  
 forcebalance::tinkerio::Vibration\_TINKER, 821  
 forcebalance::vibration::Vibration, 790  
**printcool**  
     forcebalance::nifty, 52  
**printcool\_dictionary**  
     forcebalance::nifty, 53  
**printcool\_table**  
     forcebalance::abinitio::AbInitio, 89  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 155  
     forcebalance::amberio::AbInitio\_AMBER, 111  
     forcebalance::binding::BindingEnergy, 222  
     forcebalance::counterpoise::Counterpoise, 286  
     forcebalance::gmxio::AbInitio\_GMX, 133  
     forcebalance::gmxio::BindingEnergy\_GMX, 237  
     forcebalance::gmxio::Interaction\_GMX, 355  
     forcebalance::gmxio::Lipid\_GMX, 441  
     forcebalance::gmxio::Liquid\_GMX, 477  
     forcebalance::gmxio::Moments\_GMX, 585  
     forcebalance::gmxio::Thermo\_GMX, 759  
     forcebalance::gmxio::Vibration\_GMX, 802  
     forcebalance::interaction::Interaction, 340  
     forcebalance::leastsq::LeastSquares, 404  
     forcebalance::lipid::Lipid, 423  
     forcebalance::liquid::Liquid, 458  
     forcebalance::moments::Moments, 570  
     forcebalance::openmmio::AbInitio\_OpenMM, 177  
     forcebalance::openmmio::BindingEnergy\_OpenMM,  
         252  
     forcebalance::openmmio::Interaction\_OpenMM, 370  
     forcebalance::openmmio::Liquid\_OpenMM, 496  
     forcebalance::openmmio::Moments\_OpenMM, 600  
     forcebalance::psi4io::RDVR3\_Psi4, 682  
     forcebalance::psi4io::THCDF\_Psi4, 727  
     forcebalance::target::RemoteTarget, 696  
     forcebalance::target::Target, 711  
     forcebalance::thermo::Thermo, 742  
 forcebalance::tinkerio::AbInitio\_TINKER, 199  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 267  
 forcebalance::tinkerio::Interaction\_TINKER, 385  
 forcebalance::tinkerio::Liquid\_TINKER, 516

forcebalance::tinkerio::Moments\_TINKER, 615  
 forcebalance::tinkerio::Vibration\_TINKER, 817  
 forcebalance::vibration::Vibration, 787  
 printsection  
     forcebalance::parser, 65  
 prm  
     forcebalance::tinkerio::TINKER, 772  
 prmdestroy\_save  
     forcebalance::forcefield::FF, 305  
 prmdestroy\_this  
     forcebalance::forcefield::FF, 305  
 process\_vectors  
     forcebalance::gmxio::Vibration\_GMX, 802  
     forcebalance::tinkerio::Vibration\_TINKER, 817  
     forcebalance::vibration::Vibration, 787  
 psi4io.py, 838  
 ptyp  
     forcebalance::objective::Penalty, 656  
 push  
     forcebalance::nifty::LineChunker, 410  
 pvals0  
     forcebalance::forcefield::FF, 305  
 pvec  
     forcebalance::molecule, 42  
 pvec1d  
     forcebalance::nifty, 54  
  
 QChem\_Dielectric\_Energy  
     forcebalance::qchemio, 69  
 qchemio.py, 838  
 qfnm  
     forcebalance::abinitio::AbInitio, 94  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
     forcebalance::amberio::AbInitio\_AMBER, 116  
     forcebalance::gmxio::AbInitio\_GMX, 138  
     forcebalance::gmxio::Interaction\_GMX, 358  
     forcebalance::interaction::Interaction, 343  
     forcebalance::openmmio::AbInitio\_OpenMM, 182  
     forcebalance::openmmio::Interaction\_OpenMM, 373  
     forcebalance::tinkerio::AbInitio\_TINKER, 204  
     forcebalance::tinkerio::Interaction\_TINKER, 388  
 qid  
     forcebalance::forcefield::FF, 305  
 qid2  
     forcebalance::forcefield::FF, 305  
 qmap  
     forcebalance::forcefield::FF, 305  
 qmatoms  
     forcebalance::abinitio::AbInitio, 94  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
     forcebalance::amberio::AbInitio\_AMBER, 116  
     forcebalance::gmxio::AbInitio\_GMX, 138  
     forcebalance::openmmio::AbInitio\_OpenMM, 182  
     forcebalance::tinkerio::AbInitio\_TINKER, 204  
  
 qmboltz\_wts  
     forcebalance::abinitio::AbInitio, 94  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 160  
     forcebalance::amberio::AbInitio\_AMBER, 117  
     forcebalance::gmxio::AbInitio\_GMX, 138  
     forcebalance::openmmio::AbInitio\_OpenMM, 182  
     forcebalance::tinkerio::AbInitio\_TINKER, 204  
 quantity.py, 839  
 QuantumVariableNames  
     forcebalance::molecule, 43  
 queue\_up  
     forcebalance::nifty, 54  
 queue\_up\_src\_dest  
     forcebalance::nifty, 54  
  
 r\_options  
     forcebalance::target::RemoteTarget, 699  
 r\_tgt\_opts  
     forcebalance::target::RemoteTarget, 699  
 RMSDDict  
     forcebalance::binding::BindingEnergy, 225  
     forcebalance::gmxio::BindingEnergy\_GMX, 241  
     forcebalance::openmmio::BindingEnergy\_OpenMM,  
         256  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 271  
 radian  
     forcebalance::molecule, 43  
 Radii  
     forcebalance::chemistry, 21  
     forcebalance::molecule, 43  
 radii  
     forcebalance::psi4io::Grid\_Reader, 329  
 radius\_of\_gyration  
     forcebalance::molecule::Molecule, 548  
 rd  
     forcebalance::abinitio::AbInitio, 94  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 161  
     forcebalance::amberio::AbInitio\_AMBER, 117  
     forcebalance::binding::BindingEnergy, 225  
     forcebalance::counterpoise::Counterpoise, 289  
     forcebalance::gmxio::AbInitio\_GMX, 138  
     forcebalance::gmxio::BindingEnergy\_GMX, 240  
     forcebalance::gmxio::Interaction\_GMX, 359  
     forcebalance::gmxio::Lipid\_GMX, 445  
     forcebalance::gmxio::Liquid\_GMX, 482  
     forcebalance::gmxio::Moments\_GMX, 588  
     forcebalance::gmxio::Thermo\_GMX, 763  
     forcebalance::gmxio::Vibration\_GMX, 806  
     forcebalance::interaction::Interaction, 343  
     forcebalance::leastsq::LeastSquares, 407  
     forcebalance::lipid::Lipid, 427  
     forcebalance::liquid::Liquid, 462  
     forcebalance::moments::Moments, 573  
     forcebalance::openmmio::AbInitio\_OpenMM, 182

forcebalance::openmmio::BindingEnergy\_OpenMM,  
     255  
 forcebalance::openmmio::Interaction\_OpenMM, 374  
 forcebalance::openmmio::Liquid\_OpenMM, 501  
 forcebalance::openmmio::Moments\_OpenMM, 603  
 forcebalance::psi4io::RDVR3\_Psi4, 685  
 forcebalance::psi4io::THCDF\_Psi4, 731  
 forcebalance::target::RemoteTarget, 699  
 forcebalance::target::Target, 714  
 forcebalance::thermo::Thermo, 746  
 forcebalance::tinkerio::AbInitio\_TINKER, 204  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 270  
 forcebalance::tinkerio::Interaction\_TINKER, 389  
 forcebalance::tinkerio::Liquid\_TINKER, 521  
 forcebalance::tinkerio::Moments\_TINKER, 618  
 forcebalance::tinkerio::Vibration\_TINKER, 821  
 forcebalance::vibration::Vibration, 791  
**read**  
 forcebalance::abinitio::AbInitio, 89  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 155  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::binding::BindingEnergy, 222  
 forcebalance::counterpoise::Counterpoise, 286  
 forcebalance::gmxio::AbInitio\_GMX, 133  
 forcebalance::gmxio::BindingEnergy\_GMX, 237  
 forcebalance::gmxio::Interaction\_GMX, 355  
 forcebalance::gmxio::Lipid\_GMX, 441  
 forcebalance::gmxio::Liquid\_GMX, 477  
 forcebalance::gmxio::Moments\_GMX, 585  
 forcebalance::gmxio::Thermo\_GMX, 760  
 forcebalance::gmxio::Vibration\_GMX, 803  
 forcebalance::interaction::Interaction, 340  
 forcebalance::leastsq::LeastSquares, 404  
 forcebalance::lipid::Lipid, 423  
 forcebalance::liquid::Liquid, 458  
 forcebalance::moments::Moments, 570  
 forcebalance::openmmio::AbInitio\_OpenMM, 177  
 forcebalance::openmmio::BindingEnergy\_OpenMM,  
     252  
 forcebalance::openmmio::Interaction\_OpenMM, 370  
 forcebalance::openmmio::Liquid\_OpenMM, 496  
 forcebalance::openmmio::Moments\_OpenMM, 600  
 forcebalance::psi4io::RDVR3\_Psi4, 682  
 forcebalance::psi4io::THCDF\_Psi4, 728  
 forcebalance::target::RemoteTarget, 699  
 forcebalance::target::Target, 711  
 forcebalance::thermo::Thermo, 743  
 forcebalance::tinkerio::AbInitio\_TINKER, 199  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 267  
 forcebalance::tinkerio::Interaction\_TINKER, 385  
 forcebalance::tinkerio::Liquid\_TINKER, 516  
 forcebalance::tinkerio::Moments\_TINKER, 615  
 forcebalance::tinkerio::Vibration\_TINKER, 818  
 forcebalance::vibration::Vibration, 788  
**read\_Ograds**  
 forcebalance::abinitio::AbInitio, 90  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 156  
 forcebalance::amberio::AbInitio\_AMBER, 112  
 forcebalance::binding::BindingEnergy, 223  
 forcebalance::counterpoise::Counterpoise, 287  
 forcebalance::gmxio::AbInitio\_GMX, 134  
 forcebalance::gmxio::BindingEnergy\_GMX, 238  
 forcebalance::gmxio::Interaction\_GMX, 356  
 forcebalance::gmxio::Lipid\_GMX, 442  
 forcebalance::gmxio::Liquid\_GMX, 478  
 forcebalance::gmxio::Moments\_GMX, 586  
 forcebalance::gmxio::Thermo\_GMX, 760  
 forcebalance::gmxio::Vibration\_GMX, 803  
 forcebalance::interaction::Interaction, 341  
 forcebalance::leastsq::LeastSquares, 405  
 forcebalance::lipid::Lipid, 424  
 forcebalance::liquid::Liquid, 459  
 forcebalance::moments::Moments, 571  
 forcebalance::openmmio::AbInitio\_OpenMM, 178  
 forcebalance::openmmio::BindingEnergy\_OpenMM,  
     253  
 forcebalance::openmmio::Interaction\_OpenMM, 371  
 forcebalance::openmmio::Liquid\_OpenMM, 497  
 forcebalance::openmmio::Moments\_OpenMM, 601  
 forcebalance::psi4io::RDVR3\_Psi4, 683  
 forcebalance::psi4io::THCDF\_Psi4, 728  
 forcebalance::target::RemoteTarget, 697  
 forcebalance::target::Target, 712  
 forcebalance::thermo::Thermo, 743  
 forcebalance::tinkerio::AbInitio\_TINKER, 200  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 268  
 forcebalance::tinkerio::Interaction\_TINKER, 386  
 forcebalance::tinkerio::Liquid\_TINKER, 517  
 forcebalance::tinkerio::Moments\_TINKER, 616  
 forcebalance::tinkerio::Vibration\_TINKER, 818  
 forcebalance::vibration::Vibration, 788  
**Read\_Tab**  
 forcebalance::molecule::Molecule, 559  
**read\_arc**  
 forcebalance::molecule::Molecule, 548  
**read\_charmm**  
 forcebalance::molecule::Molecule, 549  
**read\_com**  
 forcebalance::molecule::Molecule, 549  
**read\_data**  
 forcebalance::gmxio::Lipid\_GMX, 442  
 forcebalance::gmxio::Liquid\_GMX, 478  
 forcebalance::lipid::Lipid, 424  
 forcebalance::liquid::Liquid, 459  
 forcebalance::openmmio::Liquid\_OpenMM, 497  
 forcebalance::tinkerio::Liquid\_TINKER, 517  
**read\_dcd**  
 forcebalance::molecule::Molecule, 550

```

read_gro
    forcebalance::molecule::Molecule, 550
read_indicate
    forcebalance::abinitio::AbInitio, 95
    forcebalance::abinitio_internal::AbInitio_Internal, 161
    forcebalance::amberio::AbInitio_AMBER, 117
    forcebalance::binding::BindingEnergy, 225
    forcebalance::counterpoise::Counterpoise, 289
    forcebalance::gmxio::AbInitio_GMX, 139
    forcebalance::gmxio::BindingEnergy_GMX, 240
    forcebalance::gmxio::Interaction_GMX, 359
    forcebalance::gmxio::Lipid_GMX, 445
    forcebalance::gmxio::Liquid_GMX, 482
    forcebalance::gmxio::Moments_GMX, 588
    forcebalance::gmxio::Thermo_GMX, 763
    forcebalance::gmxio::Vibration_GMX, 806
    forcebalance::interaction::Interaction, 344
    forcebalance::leastsq::LeastSquares, 407
    forcebalance::lipid::Lipid, 427
    forcebalance::liquid::Liquid, 463
    forcebalance::moments::Moments, 573
    forcebalance::openmmio::AbInitio_OpenMM, 183
    forcebalance::openmmio::BindingEnergy_OpenMM,
        255
    forcebalance::openmmio::Interaction_OpenMM, 374
    forcebalance::openmmio::Liquid_OpenMM, 501
    forcebalance::openmmio::Moments_OpenMM, 603
    forcebalance::psi4io::RDVR3_Psi4, 685
    forcebalance::psi4io::THCDF_Psi4, 731
    forcebalance::target::RemoteTarget, 699
    forcebalance::target::Target, 714
    forcebalance::thermo::Thermo, 746
    forcebalance::tinkerio::AbInitio_TINKER, 205
    forcebalance::tinkerio::BindingEnergy_TINKER, 270
    forcebalance::tinkerio::Interaction_TINKER, 389
    forcebalance::tinkerio::Liquid_TINKER, 521
    forcebalance::tinkerio::Moments_TINKER, 618
    forcebalance::tinkerio::Vibration_TINKER, 821
    forcebalance::vibration::Vibration, 791
read_internals
    forcebalance::parser, 66
read_mdcrd
    forcebalance::molecule::Molecule, 550
read_mol2
    forcebalance::molecule::Molecule, 551
read_mvals
    forcebalance::parser, 66
read_objective
    forcebalance::abinitio::AbInitio, 95
    forcebalance::abinitio_internal::AbInitio_Internal, 161
    forcebalance::amberio::AbInitio_AMBER, 117
    forcebalance::binding::BindingEnergy, 225
    forcebalance::counterpoise::Counterpoise, 289
    forcebalance::gmxio::AbInitio_GMX, 139
    forcebalance::gmxio::BindingEnergy_GMX, 240
    forcebalance::gmxio::Interaction_GMX, 359
    forcebalance::gmxio::Lipid_GMX, 445
    forcebalance::gmxio::Liquid_GMX, 482
    forcebalance::gmxio::Moments_GMX, 588
    forcebalance::gmxio::Thermo_GMX, 763
    forcebalance::gmxio::Vibration_GMX, 806
    forcebalance::interaction::Interaction, 344
    forcebalance::leastsq::LeastSquares, 407
    forcebalance::lipid::Lipid, 427
    forcebalance::liquid::Liquid, 463
    forcebalance::moments::Moments, 573
    forcebalance::openmmio::AbInitio_OpenMM, 183
    forcebalance::openmmio::BindingEnergy_OpenMM,
        255
    forcebalance::openmmio::Interaction_OpenMM, 374
    forcebalance::openmmio::Liquid_OpenMM, 501
    forcebalance::openmmio::Moments_OpenMM, 603
    forcebalance::psi4io::RDVR3_Psi4, 686
    forcebalance::psi4io::THCDF_Psi4, 731
    forcebalance::target::RemoteTarget, 700
    forcebalance::target::Target, 714
    forcebalance::thermo::Thermo, 746
    forcebalance::tinkerio::AbInitio_TINKER, 205
    forcebalance::tinkerio::BindingEnergy_TINKER, 270
    forcebalance::tinkerio::Interaction_TINKER, 389
    forcebalance::tinkerio::Liquid_TINKER, 521
    forcebalance::tinkerio::Moments_TINKER, 618
    forcebalance::tinkerio::Vibration_TINKER, 821
    forcebalance::vibration::Vibration, 791
read_pdb
    forcebalance::molecule::Molecule, 551
read_priors
    forcebalance::parser, 66
read_pvals
    forcebalance::parser, 66
read_qcesp
    forcebalance::molecule::Molecule, 551
read_qcin
    forcebalance::molecule::Molecule, 551
read_qcout
    forcebalance::molecule::Molecule, 552
read_qdata
    forcebalance::molecule::Molecule, 552
read_reference_data
    forcebalance::abinitio::AbInitio, 90
    forcebalance::abinitio_internal::AbInitio_Internal, 156
    forcebalance::amberio::AbInitio_AMBER, 112
    forcebalance::gmxio::AbInitio_GMX, 134
    forcebalance::gmxio::Interaction_GMX, 356
    forcebalance::gmxio::Moments_GMX, 586
    forcebalance::gmxio::Vibration_GMX, 803
    forcebalance::interaction::Interaction, 341
    forcebalance::moments::Moments, 571

```

```

forcebalance::openmmio::AbInitio_OpenMM, 178
forcebalance::openmmio::Interaction_OpenMM, 371
forcebalance::openmmio::Moments_OpenMM, 601
forcebalance::tinkerio::AbInitio_TINKER, 200
forcebalance::tinkerio::Interaction_TINKER, 386
forcebalance::tinkerio::Moments_TINKER, 616
forcebalance::tinkerio::Vibration_TINKER, 818
forcebalance::vibration::Vibration, 788
read_xyz
    forcebalance::molecule::Molecule, 552
read_xyz0
    forcebalance::molecule::Molecule, 553
readchk
    forcebalance::optimizer::Optimizer, 647
Readers
    forcebalance::forcefield::FF, 305
readsrc
    forcebalance::engine::Engine, 292
    forcebalance::gmxio::GMX, 325
    forcebalance::openmmio::OpenMM, 634
    forcebalance::tinkerio::TINKER, 771
reassign
    forcebalance::gmxio::Vibration_GMX, 806
    forcebalance::tinkerio::Vibration_TINKER, 821
    forcebalance::vibration::Vibration, 791
recover
    forcebalance::optimizer::Optimizer, 647
redirect
    forcebalance::forcefield::FF, 305
ref
    forcebalance::thermo::Point, 659
ref_eigvals
    forcebalance::gmxio::Moments_GMX, 589
    forcebalance::gmxio::Vibration_GMX, 806
    forcebalance::moments::Moments, 574
    forcebalance::openmmio::Moments_OpenMM, 604
    forcebalance::tinkerio::Moments_TINKER, 619
    forcebalance::tinkerio::Vibration_TINKER, 821
    forcebalance::vibration::Vibration, 791
ref_eigvecs
    forcebalance::gmxio::Moments_GMX, 589
    forcebalance::gmxio::Vibration_GMX, 806
    forcebalance::moments::Moments, 574
    forcebalance::openmmio::Moments_OpenMM, 604
    forcebalance::tinkerio::Moments_TINKER, 619
    forcebalance::tinkerio::Vibration_TINKER, 821
    forcebalance::vibration::Vibration, 791
ref_eigvecs_nrm_mw
    forcebalance::gmxio::Vibration_GMX, 806
    forcebalance::tinkerio::Vibration_TINKER, 821
    forcebalance::vibration::Vibration, 791
ref_moments
    forcebalance::gmxio::Moments_GMX, 589
    forcebalance::moments::Moments, 574
forcebalance::openmmio::Moments_OpenMM, 604
forcebalance::tinkerio::Moments_TINKER, 616
forcebalance::tinkerio::Liquid_TINKER, 517
forcebalance::tinkerio::Vibration_TINKER, 818
forcebalance::vibration::Vibration, 788
forcebalance::openmmio::Moments_OpenMM, 604
forcebalance::tinkerio::Moments_TINKER, 619
ref_rmsd
    forcebalance::molecule::Molecule, 553
RefData
    forcebalance::gmxio::Lipid_GMX, 445
    forcebalance::gmxio::Liquid_GMX, 482
    forcebalance::lipid::Lipid, 427
    forcebalance::liquid::Liquid, 463
    forcebalance::openmmio::Liquid_OpenMM, 501
    forcebalance::tinkerio::Liquid_TINKER, 521
refresh_temp_directory
    forcebalance::abinitio::AbInitio, 91
    forcebalance::abinitio.internal::AbInitio_Internal, 157
    forcebalance::amberio::AbInitio_AMBER, 113
    forcebalance::binding::BindingEnergy, 223
    forcebalance::counterpoise::Counterpoise, 287
    forcebalance::gmxio::AbInitio_GMX, 135
    forcebalance::gmxio::BindingEnergy_GMX, 238
    forcebalance::gmxio::Interaction_GMX, 356
    forcebalance::gmxio::Lipid_GMX, 442
    forcebalance::gmxio::Liquid_GMX, 478
    forcebalance::gmxio::Moments_GMX, 586
    forcebalance::gmxio::Thermo_GMX, 760
    forcebalance::gmxio::Vibration_GMX, 803
    forcebalance::interaction::Interaction, 341
    forcebalance::leastsq::LeastSquares, 405
    forcebalance::lipid::Lipid, 424
    forcebalance::liquid::Liquid, 459
    forcebalance::moments::Moments, 571
    forcebalance::openmmio::AbInitio_OpenMM, 179
    forcebalance::openmmio::BindingEnergy_OpenMM,
        253
    forcebalance::openmmio::Interaction_OpenMM, 371
    forcebalance::openmmio::Liquid_OpenMM, 497
    forcebalance::openmmio::Moments_OpenMM, 601
    forcebalance::psi4io::RDVR3_Psi4, 683
    forcebalance::psi4io::THCDF_Psi4, 728
    forcebalance::target::RemoteTarget, 697
    forcebalance::target::Target, 712
    forcebalance::thermo::Thermo, 743
    forcebalance::tinkerio::AbInitio_TINKER, 201
    forcebalance::tinkerio::BindingEnergy_TINKER, 268
    forcebalance::tinkerio::Interaction_TINKER, 386
    forcebalance::tinkerio::Liquid_TINKER, 517
    forcebalance::tinkerio::Moments_TINKER, 616
    forcebalance::tinkerio::Vibration_TINKER, 818
    forcebalance::vibration::Vibration, 788
remote_indicate
    forcebalance::target::RemoteTarget, 700
remove_if_exists
    forcebalance::nifty, 54
removeHandler
    forcebalance::output::ForceBalanceLogger, 307

```

```

repair
    forcebalance::molecule::Molecule, 553
replace_peratom
    forcebalance::molecule::Molecule, 554
replace_peratom_conditional
    forcebalance::molecule::Molecule, 554
require
    forcebalance::molecule::Molecule, 554
require_boxes
    forcebalance::molecule::Molecule, 554
require_resid
    forcebalance::molecule::Molecule, 554
require_resname
    forcebalance::molecule::Molecule, 554
resdir
    forcebalance::optimizer::Optimizer, 653
ResetVirtualSites
    forcebalance::openmmio, 61
ResetVirtualSites_fast
    forcebalance::openmmio, 61
resid
    forcebalance::molecule::Molecule, 559
residue_distances
    forcebalance::contact, 22
resname
    forcebalance::molecule::Molecule, 559
respterm
    forcebalance::abinitio::AbInitio, 95
    forcebalance::abinitio_internal::AbInitio_Internal, 161
    forcebalance::amberio::AbInitio_AMBER, 117
    forcebalance::gmxio::AbInitio_GMX, 139
    forcebalance::openmmio::AbInitio_OpenMM, 183
    forcebalance::tinkerio::AbInitio_TINKER, 205
retrieve
    forcebalance::gmxio::Thermo_GMX, 760
    forcebalance::thermo::Thermo, 743
rigid
    forcebalance::tinkerio::TINKER, 772
rigid_water
    forcebalance::molecule::Molecule, 554
rm_gmx_baks
    forcebalance::gmxio, 31
rmsd_part
    forcebalance::binding::BindingEnergy, 225
    forcebalance::gmxio::BindingEnergy_GMX, 240
    forcebalance::openmmio::BindingEnergy_OpenMM,
        255
    forcebalance::tinkerio::BindingEnergy_TINKER, 270
root
    forcebalance::engine::Engine, 292
    forcebalance::gmxio::GMX, 326
    forcebalance::openmmio::OpenMM, 636
    forcebalance::tinkerio::TINKER, 772
row
    forcebalance::nifty, 54
rs
    forcebalance::forcefield::FF, 305
rs_override
    forcebalance::forcefield, 28
rsmake
    forcebalance::forcefield::FF, 303
Run
    forcebalance::optimizer::Optimizer, 647
rundir
    forcebalance::abinitio::AbInitio, 95
    forcebalance::abinitio_internal::AbInitio_Internal, 161
    forcebalance::amberio::AbInitio_AMBER, 117
    forcebalance::binding::BindingEnergy, 226
    forcebalance::counterpoise::Counterpoise, 289
    forcebalance::gmxio::AbInitio_GMX, 139
    forcebalance::gmxio::BindingEnergy_GMX, 241
    forcebalance::gmxio::Interaction_GMX, 359
    forcebalance::gmxio::Lipid_GMX, 445
    forcebalance::gmxio::Liquid_GMX, 482
    forcebalance::gmxio::Moments_GMX, 589
    forcebalance::gmxio::Thermo_GMX, 763
    forcebalance::gmxio::Vibration_GMX, 806
    forcebalance::interaction::Interaction, 344
    forcebalance::leastsq::LeastSquares, 407
    forcebalance::lipid::Lipid, 427
    forcebalance::liquid::Liquid, 463
    forcebalance::moments::Moments, 574
    forcebalance::openmmio::AbInitio_OpenMM, 183
    forcebalance::openmmio::BindingEnergy_OpenMM,
        256
    forcebalance::openmmio::Interaction_OpenMM, 374
    forcebalance::openmmio::Liquid_OpenMM, 501
    forcebalance::openmmio::Moments_OpenMM, 604
    forcebalance::psi4io::RDVR3_Psi4, 686
    forcebalance::psi4io::THCDF_Psi4, 731
    forcebalance::target::RemoteTarget, 700
    forcebalance::target::Target, 714
    forcebalance::thermo::Thermo, 746
    forcebalance::tinkerio::AbInitio_TINKER, 205
    forcebalance::tinkerio::BindingEnergy_TINKER, 271
    forcebalance::tinkerio::Interaction_TINKER, 389
    forcebalance::tinkerio::Liquid_TINKER, 521
    forcebalance::tinkerio::Moments_TINKER, 619
    forcebalance::tinkerio::Vibration_TINKER, 821
    forcebalance::vibration::Vibration, 791
save_mvals_to_input
    forcebalance::optimizer::Optimizer, 648
save_vmvalls
    forcebalance::abinitio::AbInitio, 95
    forcebalance::abinitio_internal::AbInitio_Internal, 161
    forcebalance::amberio::AbInitio_AMBER, 117
    forcebalance::gmxio::AbInitio_GMX, 139

```

forcebalance::openmmio::AbInitio\_OpenMM, 183  
 forcebalance::tinkerio::AbInitio\_TINKER, 205  
**SavedMVal**  
 forcebalance::gmxio::Lipid\_GMX, 445  
 forcebalance::lipid::Lipid, 427  
**SavedTraj**  
 forcebalance::gmxio::Lipid\_GMX, 446  
 forcebalance::gmxio::Liquid\_GMX, 482  
 forcebalance::lipid::Lipid, 427  
 forcebalance::liquid::Liquid, 463  
 forcebalance::openmmio::Liquid\_OpenMM, 501  
 forcebalance::tinkerio::Liquid\_TINKER, 521  
**Scan\_Values**  
 forcebalance::optimizer::Optimizer, 648  
**ScanMVals**  
 forcebalance::optimizer::Optimizer, 648  
**ScanPVals**  
 forcebalance::optimizer::Optimizer, 649  
**scd\_persnap**  
 forcebalance::gmxio::GMX, 325  
**Scipy\_BFGS**  
 forcebalance::optimizer::Optimizer, 649  
**ScipyOptimizer**  
 forcebalance::optimizer::Optimizer, 649  
**scripts**  
 forcebalance::gmxio::Lipid\_GMX, 446  
 forcebalance::gmxio::Liquid\_GMX, 482  
 forcebalance::gmxio::Thermo\_GMX, 763  
 forcebalance::openmmio::Liquid\_OpenMM, 501  
 forcebalance::tinkerio::Liquid\_TINKER, 521  
**sec**  
 forcebalance::custom\_io::Gen\_Reader, 316  
 forcebalance::gmxio::ITP\_Reader, 394  
 forcebalance::qchemio::QCIn\_Reader, 662  
**section**  
 forcebalance::amberio::Mol2\_Reader, 538  
**segments**  
 forcebalance::nifty, 54  
**select1**  
 forcebalance::gmxio::Interaction\_GMX, 359  
 forcebalance::interaction::Interaction, 344  
 forcebalance::openmmio::Interaction\_OpenMM, 374  
 forcebalance::tinkerio::Interaction\_TINKER, 389  
**select2**  
 forcebalance::gmxio::Interaction\_GMX, 359  
 forcebalance::interaction::Interaction, 344  
 forcebalance::openmmio::Interaction\_OpenMM, 374  
 forcebalance::tinkerio::Interaction\_TINKER, 389  
**set\_atom\_id**  
 forcebalance::Mol2::mol2\_atom, 529  
**set\_atom\_name**  
 forcebalance::Mol2::mol2\_atom, 529  
**set\_atom\_type**  
 forcebalance::Mol2::mol2\_atom, 529  
**set\_bond\_id**  
 forcebalance::Mol2::mol2\_bond, 532  
**set\_bond\_type**  
 forcebalance::Mol2::mol2\_bond, 532  
**set\_charge**  
 forcebalance::Mol2::mol2\_atom, 529  
**set\_charge\_type**  
 forcebalance::Mol2::mol2, 526  
**set\_crds**  
 forcebalance::Mol2::mol2\_atom, 530  
**set\_donor\_acceptor\_atoms**  
 forcebalance::Mol2::mol2, 526  
**set\_mol\_name**  
 forcebalance::Mol2::mol2, 526  
**set\_mol\_type**  
 forcebalance::Mol2::mol2, 527  
**set\_num\_atoms**  
 forcebalance::Mol2::mol2, 527  
**set\_num\_bonds**  
 forcebalance::Mol2::mol2, 527  
**set\_num\_feat**  
 forcebalance::Mol2::mol2, 527  
**set\_num\_sets**  
 forcebalance::Mol2::mol2, 527  
**set\_num\_subst**  
 forcebalance::Mol2::mol2, 527  
**set\_option**  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio.internal::AbInitio\_Internal, 157  
 forcebalance::amberio::AbInitio\_AMBER, 113  
 forcebalance::BaseClass, 209  
 forcebalance::binding::BindingEnergy, 223  
 forcebalance::counterpoise::Counterpoise, 287  
 forcebalance::engine::Engine, 292  
 forcebalance::forcefield::FF, 303  
 forcebalance::gmxio::AbInitio\_GMX, 135  
 forcebalance::gmxio::BindingEnergy\_GMX, 238  
 forcebalance::gmxio::GMX, 325  
 forcebalance::gmxio::Interaction\_GMX, 356  
 forcebalance::gmxio::Lipid\_GMX, 442  
 forcebalance::gmxio::Liquid\_GMX, 478  
 forcebalance::gmxio::Moments\_GMX, 586  
 forcebalance::gmxio::Thermo\_GMX, 761  
 forcebalance::gmxio::Vibration\_GMX, 803  
 forcebalance::interaction::Interaction, 341  
 forcebalance::leastsq::LeastSquares, 405  
 forcebalance::lipid::Lipid, 424  
 forcebalance::liquid::Liquid, 459  
 forcebalance::moments::Moments, 571  
 forcebalance::objective::Objective, 625  
 forcebalance::openmmio::AbInitio\_OpenMM, 179  
 forcebalance::openmmio::BindingEnergy\_OpenMM,  
 253  
 forcebalance::openmmio::Interaction\_OpenMM, 371

forcebalance::openmmio::Liquid\_OpenMM, 497  
 forcebalance::openmmio::Moments\_OpenMM, 601  
 forcebalance::openmmio::OpenMM, 634  
 forcebalance::optimizer::Optimizer, 650  
 forcebalance::psi4io::RDVR3\_Psi4, 683  
 forcebalance::psi4io::THCDF\_Psi4, 728  
 forcebalance::target::RemoteTarget, 697  
 forcebalance::target::Target, 712  
 forcebalance::thermo::Thermo, 744  
 forcebalance::tinkerio::AbInitio\_TINKER, 201  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 268  
 forcebalance::tinkerio::Interaction\_TINKER, 386  
 forcebalance::tinkerio::Liquid\_TINKER, 517  
 forcebalance::tinkerio::Moments\_TINKER, 616  
 forcebalance::tinkerio::TINKER, 771  
 forcebalance::tinkerio::Vibration\_TINKER, 818  
 forcebalance::vibration::Vibration, 788  
 set\_origin\_atom\_id  
     forcebalance::Mol2::mol2\_bond, 532  
 set\_positions  
     forcebalance::openmmio::OpenMM, 634  
 set\_status\_bit  
     forcebalance::Mol2::mol2\_atom, 530  
     forcebalance::Mol2::mol2\_bond, 532  
 set\_subst\_id  
     forcebalance::Mol2::mol2\_atom, 530  
 set\_subst\_name  
     forcebalance::Mol2::mol2\_atom, 530  
 set\_target\_atom\_id  
     forcebalance::Mol2::mol2\_bond, 532  
 SetAmoebaVirtualExclusions  
     forcebalance::openmmio, 61  
 setopts  
     forcebalance::engine::Engine, 292  
     forcebalance::gmxio::GMX, 325  
     forcebalance::openmmio::OpenMM, 634  
     forcebalance::tinkerio::TINKER, 771  
 shell  
     forcebalance::qchemio::QCIn\_Reader, 662  
 simkwargs  
     forcebalance::openmmio::OpenMM, 636  
 simpfx  
     forcebalance::gmxio::Thermo\_GMX, 764  
     forcebalance::thermo::Thermo, 746  
 Simplex  
     forcebalance::optimizer::Optimizer, 650  
 simulation  
     forcebalance::openmmio::OpenMM, 636  
 SinglePoint  
     forcebalance::optimizer::Optimizer, 650  
 snum  
     forcebalance::qchemio::QCIn\_Reader, 662  
 spacings  
     forcebalance::objective::Penalty, 656  
 specific\_dct  
     forcebalance::nifty, 56  
 specific\_lst  
     forcebalance::nifty, 56  
 Split  
     forcebalance::amberio::FrcMod\_Reader, 309  
     forcebalance::amberio::Mol2\_Reader, 537  
     forcebalance::BaseReader, 211  
     forcebalance::custom\_io::Gen\_Reader, 316  
     forcebalance::gmxio::ITP\_Reader, 393  
     forcebalance::mol2io::Mol2\_Reader, 534  
     forcebalance::openmmio::OpenMM\_Reader, 638  
     forcebalance::psi4io::GBS\_Reader, 313  
     forcebalance::psi4io::Grid\_Reader, 329  
     forcebalance::qchemio::QCIn\_Reader, 661  
     forcebalance::tinkerio::Tinker\_Reader, 775  
 split  
     forcebalance::molecule::Molecule, 555  
 splitter  
     forcebalance::molecule, 43  
 sprint\_map  
     forcebalance::forcefield::FF, 303  
 srkdir  
     forcebalance::engine::Engine, 292  
     forcebalance::gmxio::GMX, 326  
     forcebalance::openmmio::OpenMM, 636  
     forcebalance::tinkerio::TINKER, 772  
 stage  
     forcebalance::abinitio::AbInitio, 91  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 157  
     forcebalance::amberio::AbInitio\_AMBER, 113  
     forcebalance::binding::BindingEnergy, 223  
     forcebalance::counterpoise::Counterpoise, 287  
     forcebalance::gmxio::AbInitio\_GMX, 135  
     forcebalance::gmxio::BindingEnergy\_GMX, 238  
     forcebalance::gmxio::Interaction\_GMX, 356  
     forcebalance::gmxio::Lipid\_GMX, 442  
     forcebalance::gmxio::Liquid\_GMX, 478  
     forcebalance::gmxio::Moments\_GMX, 586  
     forcebalance::gmxio::Thermo\_GMX, 761  
     forcebalance::gmxio::Vibration\_GMX, 803  
     forcebalance::interaction::Interaction, 341  
     forcebalance::leastsq::LeastSquares, 405  
     forcebalance::lipid::Lipid, 424  
     forcebalance::liquid::Liquid, 459  
     forcebalance::moments::Moments, 571  
     forcebalance::openmmio::AbInitio\_OpenMM, 179  
     forcebalance::openmmio::BindingEnergy\_OpenMM,  
         253  
     forcebalance::openmmio::Interaction\_OpenMM, 371  
     forcebalance::openmmio::Liquid\_OpenMM, 497  
     forcebalance::openmmio::Moments\_OpenMM, 601  
     forcebalance::psi4io::RDVR3\_Psi4, 683  
     forcebalance::psi4io::THCDF\_Psi4, 728

forcebalance::target::RemoteTarget, 697  
 forcebalance::target::Target, 712  
 forcebalance::thermo::Thermo, 744  
 forcebalance::tinkerio::AbInitio\_TINKER, 201  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 268  
 forcebalance::tinkerio::Interaction\_TINKER, 386  
 forcebalance::tinkerio::Liquid\_TINKER, 517  
 forcebalance::tinkerio::Moments\_TINKER, 616  
 forcebalance::tinkerio::Vibration\_TINKER, 818  
 forcebalance::vibration::Vibration, 788  
 statisticallnefficiency  
     forcebalance::nifty, 55  
 status\_bit  
     forcebalance::Mol2::mol2\_atom, 530  
     forcebalance::Mol2::mol2\_bond, 532  
 step  
     forcebalance::optimizer::Optimizer, 650  
 subdict  
     forcebalance::parser, 67  
 submit\_jobs  
     forcebalance::abinitio::AbInitio, 91  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 157  
     forcebalance::amberio::AbInitio\_AMBER, 114  
     forcebalance::binding::BindingEnergy, 224  
     forcebalance::counterpoise::Counterpoise, 288  
     forcebalance::gmxio::AbInitio\_GMX, 135  
     forcebalance::gmxio::BindingEnergy\_GMX, 239  
     forcebalance::gmxio::Interaction\_GMX, 357  
     forcebalance::gmxio::Lipid\_GMX, 443  
     forcebalance::gmxio::Liquid\_GMX, 479  
     forcebalance::gmxio::Moments\_GMX, 587  
     forcebalance::gmxio::Thermo\_GMX, 761  
     forcebalance::gmxio::Vibration\_GMX, 804  
     forcebalance::interaction::Interaction, 342  
     forcebalance::leastsq::LeastSquares, 406  
     forcebalance::lipid::Lipid, 425  
     forcebalance::liquid::Liquid, 460  
     forcebalance::moments::Moments, 572  
     forcebalance::openmmio::AbInitio\_OpenMM, 179  
     forcebalance::openmmio::BindingEnergy\_OpenMM, 254  
     forcebalance::openmmio::Interaction\_OpenMM, 372  
     forcebalance::openmmio::Liquid\_OpenMM, 498  
     forcebalance::openmmio::Moments\_OpenMM, 602  
     forcebalance::psi4io::RDVR3\_Psi4, 684  
     forcebalance::psi4io::THCDF\_Psi4, 729  
     forcebalance::target::RemoteTarget, 698  
     forcebalance::target::Target, 713  
     forcebalance::thermo::Thermo, 744  
     forcebalance::tinkerio::AbInitio\_TINKER, 201  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 269  
     forcebalance::tinkerio::Interaction\_TINKER, 387  
     forcebalance::tinkerio::Liquid\_TINKER, 518  
     forcebalance::tinkerio::Moments\_TINKER, 617  
     forcebalance::tinkerio::Vibration\_TINKER, 819  
     forcebalance::vibration::Vibration, 789  
 subst\_id  
     forcebalance::Mol2::mol2\_atom, 530  
 subst\_name  
     forcebalance::Mol2::mol2\_atom, 530  
 suffix  
     forcebalance::amberio::FrcMod\_Reader, 310  
     forcebalance::amberio::Mol2\_Reader, 538  
     forcebalance::BaseReader, 212  
     forcebalance::custom\_io::Gen\_Reader, 316  
     forcebalance::gmxio::ITP\_Reader, 394  
     forcebalance::mol2io::Mol2\_Reader, 535  
     forcebalance::openmmio::OpenMM\_Reader, 639  
     forcebalance::psi4io::GBS\_Reader, 313  
     forcebalance::psi4io::Grid\_Reader, 330  
     forcebalance::qchemio::QCIn\_Reader, 662  
     forcebalance::tinkerio::Tinker\_Reader, 776  
 suffix\_dict  
     forcebalance::openmmio, 62  
 system  
     forcebalance::openmmio::OpenMM, 636  
 system\_driver  
     forcebalance::binding::BindingEnergy, 224  
     forcebalance::gmxio::BindingEnergy\_GMX, 239  
     forcebalance::openmmio::BindingEnergy\_OpenMM, 254  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 269  
 target  
     forcebalance::engine::Engine, 292  
     forcebalance::gmxio::GMX, 326  
     forcebalance::openmmio::OpenMM, 636  
     forcebalance::tinkerio::TINKER, 772  
 target.py, 839  
 Target.Terms  
     forcebalance::objective::Objective, 625  
 target\_atom\_id  
     forcebalance::Mol2::mol2\_bond, 532  
 Targets  
     forcebalance::objective::Objective, 626  
 tdir  
     forcebalance::psi4io::RDVR3\_Psi4, 686  
 tdiv  
     forcebalance::openmmio::OpenMM, 636  
 tempbase  
     forcebalance::abinitio::AbInitio, 95  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 161  
     forcebalance::amberio::AbInitio\_AMBER, 117  
     forcebalance::binding::BindingEnergy, 226  
     forcebalance::counterpoise::Counterpoise, 289  
     forcebalance::gmxio::AbInitio\_GMX, 139  
     forcebalance::gmxio::BindingEnergy\_GMX, 241  
     forcebalance::gmxio::Interaction\_GMX, 359

forcebalance::gmxio::Lipid\_GMX, 446  
 forcebalance::gmxio::Liquid\_GMX, 482  
 forcebalance::gmxio::Moments\_GMX, 589  
 forcebalance::gmxio::Thermo\_GMX, 764  
 forcebalance::gmxio::Vibration\_GMX, 806  
 forcebalance::interaction::Interaction, 344  
 forcebalance::leastsq::LeastSquares, 407  
 forcebalance::lipid::Lipid, 427  
 forcebalance::liquid::Liquid, 463  
 forcebalance::moments::Moments, 574  
 forcebalance::openmmio::AbInitio\_OpenMM, 183  
 forcebalance::openmmio::BindingEnergy\_OpenMM, 256  
 forcebalance::openmmio::Interaction\_OpenMM, 374  
 forcebalance::openmmio::Liquid\_OpenMM, 501  
 forcebalance::openmmio::Moments\_OpenMM, 604  
 forcebalance::psi4io::RDVR3\_Psi4, 686  
 forcebalance::psi4io::THCDF\_Psi4, 731  
 forcebalance::target::RemoteTarget, 700  
 forcebalance::target::Target, 714  
 forcebalance::thermo::Thermo, 746  
 forcebalance::tinkerio::AbInitio\_TINKER, 205  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 271  
 forcebalance::tinkerio::Interaction\_TINKER, 389  
 forcebalance::tinkerio::Liquid\_TINKER, 521  
 forcebalance::tinkerio::Moments\_TINKER, 619  
 forcebalance::tinkerio::Vibration\_TINKER, 821  
 forcebalance::vibration::Vibration, 791  
**tempdir**  
 forcebalance::abinitio::AbInitio, 95  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 161  
 forcebalance::amberio::AbInitio\_AMBER, 117  
 forcebalance::binding::BindingEnergy, 226  
 forcebalance::counterpoise::Counterpoise, 290  
 forcebalance::engine::Engine, 292  
 forcebalance::gmxio::AbInitio\_GMX, 139  
 forcebalance::gmxio::BindingEnergy\_GMX, 241  
 forcebalance::gmxio::GMX, 326  
 forcebalance::gmxio::Interaction\_GMX, 359  
 forcebalance::gmxio::Lipid\_GMX, 446  
 forcebalance::gmxio::Liquid\_GMX, 482  
 forcebalance::gmxio::Moments\_GMX, 589  
 forcebalance::gmxio::Thermo\_GMX, 764  
 forcebalance::gmxio::Vibration\_GMX, 806  
 forcebalance::interaction::Interaction, 344  
 forcebalance::leastsq::LeastSquares, 408  
 forcebalance::lipid::Lipid, 427  
 forcebalance::liquid::Liquid, 463  
 forcebalance::moments::Moments, 574  
 forcebalance::openmmio::AbInitio\_OpenMM, 183  
 forcebalance::openmmio::BindingEnergy\_OpenMM, 256  
 forcebalance::openmmio::Interaction\_OpenMM, 374  
 forcebalance::openmmio::Liquid\_OpenMM, 501  
 forcebalance::openmmio::Moments\_OpenMM, 604  
 forcebalance::openmmio::OpenMM, 636  
 forcebalance::psi4io::RDVR3\_Psi4, 686  
 forcebalance::psi4io::THCDF\_Psi4, 731  
 forcebalance::target::RemoteTarget, 700  
 forcebalance::target::Target, 714  
 forcebalance::thermo::Thermo, 746  
 forcebalance::tinkerio::AbInitio\_TINKER, 205  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 271  
 forcebalance::tinkerio::Interaction\_TINKER, 389  
 forcebalance::tinkerio::Liquid\_TINKER, 521  
 forcebalance::tinkerio::Moments\_TINKER, 619  
 forcebalance::tinkerio::TINKER, 772  
 forcebalance::tinkerio::Vibration\_TINKER, 821  
 forcebalance::vibration::Vibration, 791  
**temperature**  
 forcebalance::quantity::Quantity, 664  
 forcebalance::quantity::Quantity\_Density, 666  
 forcebalance::quantity::Quantity\_H\_vap, 668  
 forcebalance::thermo::Point, 659  
**tgt\_opts\_defaults**  
 forcebalance::parser, 67  
**tgt\_opts\_types**  
 forcebalance::parser, 67  
**thermo.py**, 840  
**throw\_outs**  
 forcebalance::psi4io::THCDF\_Psi4, 731  
**tinkerio.py**, 840  
**tinkerpath**  
 forcebalance::tinkerio::TINKER, 772  
**tinkerprm**  
 forcebalance::forcefield::FF, 305  
**tm**  
 forcebalance::forcefield::FF, 305  
**tml**  
 forcebalance::forcefield::FF, 305  
**top**  
 forcebalance::gmxio::GMX, 326  
**topology**  
 forcebalance::molecule::Molecule, 559  
**tq\_err**  
 forcebalance::abinitio::AbInitio, 95  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 161  
 forcebalance::amberio::AbInitio\_AMBER, 117  
 forcebalance::gmxio::AbInitio\_GMX, 139  
 forcebalance::openmmio::AbInitio\_OpenMM, 183  
 forcebalance::tinkerio::AbInitio\_TINKER, 205  
**tq\_err\_pct**  
 forcebalance::abinitio::AbInitio, 95  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 161  
 forcebalance::amberio::AbInitio\_AMBER, 117  
 forcebalance::gmxio::AbInitio\_GMX, 139  
 forcebalance::openmmio::AbInitio\_OpenMM, 183  
 forcebalance::tinkerio::AbInitio\_TINKER, 205

tq\_ref  
 forcebalance::abinitio::AbInitio, 95  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 161  
 forcebalance::amberio::AbInitio\_AMBER, 117  
 forcebalance::gmxio::AbInitio\_GMX, 139  
 forcebalance::openmmio::AbInitio\_OpenMM, 183  
 forcebalance::tinkerio::AbInitio\_TINKER, 205  
 TruncatedNewton  
 forcebalance::optimizer::Optimizer, 651  
  
 uncert  
 forcebalance::optimizer::Optimizer, 653  
 uncommadash  
 forcebalance::nifty, 55  
 unpack\_moments  
 forcebalance::gmxio::Moments\_GMX, 587  
 forcebalance::moments::Moments, 572  
 forcebalance::openmmio::Moments\_OpenMM, 602  
 forcebalance::tinkerio::Moments\_TINKER, 617  
 update\_simulation  
 forcebalance::openmmio::OpenMM, 634  
 UpdateSimulationParameters  
 forcebalance::openmmio, 61  
 use\_nft  
 forcebalance::abinitio::AbInitio, 95  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 161  
 forcebalance::amberio::AbInitio\_AMBER, 117  
 forcebalance::gmxio::AbInitio\_GMX, 139  
 forcebalance::openmmio::AbInitio\_OpenMM, 183  
 forcebalance::tinkerio::AbInitio\_TINKER, 205  
  
 Val  
 forcebalance::optimizer::Optimizer, 653  
 valkwrd  
 forcebalance::gmxio::GMX, 326  
 forcebalance::openmmio::OpenMM, 636  
 forcebalance::tinkerio::TINKER, 772  
 verbose  
 forcebalance::engine::Engine, 292  
 forcebalance::gmxio::GMX, 327  
 forcebalance::openmmio::OpenMM, 636  
 forcebalance::tinkerio::TINKER, 772  
 verbose\_options  
 forcebalance::abinitio::AbInitio, 95  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 161  
 forcebalance::amberio::AbInitio\_AMBER, 118  
 forcebalance::BaseClass, 209  
 forcebalance::binding::BindingEnergy, 226  
 forcebalance::counterpoise::Counterpoise, 290  
 forcebalance::engine::Engine, 292  
 forcebalance::forcefield::FF, 305  
 forcebalance::gmxio::AbInitio\_GMX, 139  
 forcebalance::gmxio::BindingEnergy\_GMX, 241  
 forcebalance::gmxio::GMX, 327  
  
 forcebalance::gmxio::Interaction\_GMX, 359  
 forcebalance::gmxio::Lipid\_GMX, 446  
 forcebalance::gmxio::Liquid\_GMX, 482  
 forcebalance::gmxio::Moments\_GMX, 589  
 forcebalance::gmxio::Thermo\_GMX, 764  
 forcebalance::gmxio::Vibration\_GMX, 806  
 forcebalance::interaction::Interaction, 344  
 forcebalance::leastsq::LeastSquares, 408  
 forcebalance::lipid::Lipid, 428  
 forcebalance::liquid::Liquid, 463  
 forcebalance::moments::Moments, 574  
 forcebalance::objective::Objective, 626  
 forcebalance::openmmio::AbInitio\_OpenMM, 183  
 forcebalance::openmmio::BindingEnergy\_OpenMM,  
 256  
 forcebalance::openmmio::Interaction\_OpenMM, 374  
 forcebalance::openmmio::Liquid\_OpenMM, 501  
 forcebalance::openmmio::Moments\_OpenMM, 604  
 forcebalance::openmmio::OpenMM, 636  
 forcebalance::optimizer::Optimizer, 653  
 forcebalance::psi4io::RDVR3\_Psi4, 686  
 forcebalance::psi4io::THCDF\_Psi4, 731  
 forcebalance::target::RemoteTarget, 700  
 forcebalance::target::Target, 714  
 forcebalance::thermo::Thermo, 747  
 forcebalance::tinkerio::AbInitio\_TINKER, 205  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 271  
 forcebalance::tinkerio::Interaction\_TINKER, 389  
 forcebalance::tinkerio::Liquid\_TINKER, 521  
 forcebalance::tinkerio::Moments\_TINKER, 619  
 forcebalance::tinkerio::TINKER, 773  
 forcebalance::tinkerio::Vibration\_TINKER, 821  
 forcebalance::vibration::Vibration, 791  
 vfnm  
 forcebalance::gmxio::Vibration\_GMX, 806  
 forcebalance::tinkerio::Vibration\_TINKER, 821  
 forcebalance::vibration::Vibration, 791  
 vibration.py, 841  
 vibration\_driver  
 forcebalance::gmxio::Vibration\_GMX, 804  
 forcebalance::tinkerio::Vibration\_TINKER, 819  
 forcebalance::vibration::Vibration, 789  
 vsinfo  
 forcebalance::openmmio::OpenMM, 636  
 vsprm  
 forcebalance::openmmio::OpenMM, 636  
  
 w\_al  
 forcebalance::gmxio::Lipid\_GMX, 446  
 forcebalance::lipid::Lipid, 428  
 w\_alpha  
 forcebalance::gmxio::Lipid\_GMX, 446  
 forcebalance::gmxio::Liquid\_GMX, 482  
 forcebalance::lipid::Lipid, 428

forcebalance::liquid::Liquid, 463  
 forcebalance::openmmio::Liquid\_OpenMM, 501  
 forcebalance::tinkerio::Liquid\_TINKER, 521  
**w\_cp**  
 forcebalance::gmxio::Lipid\_GMX, 446  
 forcebalance::gmxio::Liquid\_GMX, 483  
 forcebalance::lipid::Lipid, 428  
 forcebalance::liquid::Liquid, 463  
 forcebalance::openmmio::Liquid\_OpenMM, 501  
 forcebalance::tinkerio::Liquid\_TINKER, 522  
**w\_energy**  
 forcebalance::abinitio::AbInitio, 95  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 161  
 forcebalance::amberio::AbInitio\_AMBER, 118  
 forcebalance::gmxio::AbInitio\_GMX, 139  
 forcebalance::openmmio::AbInitio\_OpenMM, 183  
 forcebalance::tinkerio::AbInitio\_TINKER, 205  
**w\_eps0**  
 forcebalance::gmxio::Lipid\_GMX, 446  
 forcebalance::gmxio::Liquid\_GMX, 483  
 forcebalance::lipid::Lipid, 428  
 forcebalance::liquid::Liquid, 463  
 forcebalance::openmmio::Liquid\_OpenMM, 501  
 forcebalance::tinkerio::Liquid\_TINKER, 522  
**w\_force**  
 forcebalance::abinitio::AbInitio, 95  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 162  
 forcebalance::amberio::AbInitio\_AMBER, 118  
 forcebalance::gmxio::AbInitio\_GMX, 139  
 forcebalance::openmmio::AbInitio\_OpenMM, 183  
 forcebalance::tinkerio::AbInitio\_TINKER, 205  
**w\_hvap**  
 forcebalance::gmxio::Liquid\_GMX, 483  
 forcebalance::liquid::Liquid, 463  
 forcebalance::openmmio::Liquid\_OpenMM, 501  
 forcebalance::tinkerio::Liquid\_TINKER, 522  
**w\_kappa**  
 forcebalance::gmxio::Lipid\_GMX, 446  
 forcebalance::gmxio::Liquid\_GMX, 483  
 forcebalance::lipid::Lipid, 428  
 forcebalance::liquid::Liquid, 463  
 forcebalance::openmmio::Liquid\_OpenMM, 502  
 forcebalance::tinkerio::Liquid\_TINKER, 522  
**w\_netforce**  
 forcebalance::abinitio::AbInitio, 96  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 162  
 forcebalance::amberio::AbInitio\_AMBER, 118  
 forcebalance::gmxio::AbInitio\_GMX, 140  
 forcebalance::openmmio::AbInitio\_OpenMM, 184  
 forcebalance::tinkerio::AbInitio\_TINKER, 206  
**w\_resp**  
 forcebalance::abinitio::AbInitio, 96  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 162  
 forcebalance::amberio::AbInitio\_AMBER, 118  
 forcebalance::gmxio::AbInitio\_GMX, 140  
 forcebalance::openmmio::AbInitio\_OpenMM, 184  
 forcebalance::tinkerio::AbInitio\_TINKER, 206  
**w\_rho**  
 forcebalance::gmxio::Lipid\_GMX, 446  
 forcebalance::gmxio::Liquid\_GMX, 483  
 forcebalance::lipid::Lipid, 428  
 forcebalance::liquid::Liquid, 463  
 forcebalance::openmmio::Liquid\_OpenMM, 502  
 forcebalance::tinkerio::Liquid\_TINKER, 522  
**w\_scd**  
 forcebalance::gmxio::Lipid\_GMX, 446  
 forcebalance::lipid::Lipid, 428  
**w\_torque**  
 forcebalance::abinitio::AbInitio, 96  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 162  
 forcebalance::amberio::AbInitio\_AMBER, 118  
 forcebalance::gmxio::AbInitio\_GMX, 140  
 forcebalance::openmmio::AbInitio\_OpenMM, 184  
 forcebalance::tinkerio::AbInitio\_TINKER, 206  
**WORK\_QUEUE**  
 forcebalance::nifty, 56  
**WQIDS**  
 forcebalance::nifty, 57  
**WTot**  
 forcebalance::objective::Objective, 626  
**warn\_once**  
 forcebalance::nifty, 55  
**warn\_press\_key**  
 forcebalance::nifty, 55  
**warn\_vn**  
 forcebalance::tinkerio::TINKER, 773  
**warnngmx**  
 forcebalance::gmxio::GMX, 325  
**weight**  
 forcebalance::gmxio::Interaction\_GMX, 359  
 forcebalance::interaction::Interaction, 344  
 forcebalance::openmmio::Interaction\_OpenMM, 374  
 forcebalance::tinkerio::Interaction\_TINKER, 389  
**weight\_info**  
 forcebalance::lipid, 35  
 forcebalance::liquid, 35  
**weighted\_variance**  
 forcebalance::abinitio, 16  
**weighted\_variance2**  
 forcebalance::abinitio, 16  
**weights**  
 forcebalance::gmxio::Thermo\_GMX, 764  
 forcebalance::thermo::Thermo, 747  
**whamboltz**  
 forcebalance::abinitio::AbInitio, 96  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 162  
 forcebalance::amberio::AbInitio\_AMBER, 118  
 forcebalance::gmxio::AbInitio\_GMX, 140

forcebalance::openmmio::AbInitio\_OpenMM, 184  
 forcebalance::tinkerio::AbInitio\_TINKER, 206  
 which  
     forcebalance::nifty, 55  
 Whites  
     forcebalance::amberio::FrcMod\_Reader, 310  
     forcebalance::amberio::Mol2\_Reader, 537  
     forcebalance::BaseReader, 211  
     forcebalance::custom\_io::Gen\_Reader, 316  
     forcebalance::gmxio::ITP\_Reader, 393  
     forcebalance::mol2io::Mol2\_Reader, 534  
     forcebalance::openmmio::OpenMM\_Reader, 638  
     forcebalance::psi4io::GBS\_Reader, 313  
     forcebalance::psi4io::Grid\_Reader, 329  
     forcebalance::qchemio::QCIn\_Reader, 661  
     forcebalance::tinkerio::Tinker\_Reader, 775  
 wopen  
     forcebalance::nifty, 55  
 Wp  
     forcebalance::gmxio::Lipid\_GMX, 446  
     forcebalance::gmxio::Liquid\_GMX, 483  
     forcebalance::gmxio::Thermo\_GMX, 764  
     forcebalance::lipid::Lipid, 428  
     forcebalance::liquid::Liquid, 463  
     forcebalance::openmmio::Liquid\_OpenMM, 502  
     forcebalance::thermo::Thermo, 747  
     forcebalance::tinkerio::Liquid\_TINKER, 522  
 wq\_complete  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 158  
     forcebalance::amberio::AbInitio\_AMBER, 114  
     forcebalance::binding::BindingEnergy, 224  
     forcebalance::counterpoise::Counterpoise, 288  
     forcebalance::gmxio::AbInitio\_GMX, 136  
     forcebalance::gmxio::BindingEnergy\_GMX, 239  
     forcebalance::gmxio::Interaction\_GMX, 357  
     forcebalance::gmxio::Lipid\_GMX, 443  
     forcebalance::gmxio::Liquid\_GMX, 480  
     forcebalance::gmxio::Moments\_GMX, 587  
     forcebalance::gmxio::Thermo\_GMX, 762  
     forcebalance::gmxio::Vibration\_GMX, 804  
     forcebalance::interaction::Interaction, 342  
     forcebalance::leastsq::LeastSquares, 406  
     forcebalance::lipid::Lipid, 425  
     forcebalance::liquid::Liquid, 461  
     forcebalance::moments::Moments, 572  
     forcebalance::openmmio::AbInitio\_OpenMM, 180  
     forcebalance::openmmio::BindingEnergy\_OpenMM, 254  
     forcebalance::openmmio::Interaction\_OpenMM, 372  
     forcebalance::openmmio::Liquid\_OpenMM, 499  
     forcebalance::openmmio::Moments\_OpenMM, 602  
     forcebalance::psi4io::RDVR3\_Psi4, 684  
     forcebalance::psi4io::THCDF\_Psi4, 729  
     forcebalance::target::RemoteTarget, 699  
     forcebalance::target::Target, 713  
     forcebalance::thermo::Thermo, 745  
     forcebalance::tinkerio::AbInitio\_TINKER, 202  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 269  
     forcebalance::tinkerio::Interaction\_TINKER, 387  
     forcebalance::tinkerio::Liquid\_TINKER, 519  
     forcebalance::tinkerio::Moments\_TINKER, 617  
     forcebalance::tinkerio::Vibration\_TINKER, 820  
     forcebalance::vibration::Vibration, 790  
 Write\_Tab

```

    forcebalance::molecule::Molecule, 559
write_arc
    forcebalance::molecule::Molecule, 555
write_dcd
    forcebalance::molecule::Molecule, 555
write_gro
    forcebalance::molecule::Molecule, 555
write_indicate
    forcebalance::abinitio::AbInitio, 96
    forcebalance::abinitio_internal::AbInitio_Internal, 162
    forcebalance::amberio::AbInitio_AMBER, 118
    forcebalance::binding::BindingEnergy, 226
    forcebalance::counterpoise::Counterpoise, 290
    forcebalance::gmxio::AbInitio_GMX, 140
    forcebalance::gmxio::BindingEnergy_GMX, 241
    forcebalance::gmxio::Interaction_GMX, 359
    forcebalance::gmxio::Lipid_GMX, 446
    forcebalance::gmxio::Liquid_GMX, 483
    forcebalance::gmxio::Moments_GMX, 589
    forcebalance::gmxio::Thermo_GMX, 764
    forcebalance::gmxio::Vibration_GMX, 806
    forcebalance::interaction::Interaction, 344
    forcebalance::leastsq::LeastSquares, 408
    forcebalance::lipid::Lipid, 428
    forcebalance::liquid::Liquid, 464
    forcebalance::moments::Moments, 574
    forcebalance::openmmio::AbInitio_OpenMM, 184
    forcebalance::openmmio::BindingEnergy_OpenMM,
        256
    forcebalance::openmmio::Interaction_OpenMM, 374
    forcebalance::openmmio::Liquid_OpenMM, 502
    forcebalance::openmmio::Moments_OpenMM, 604
    forcebalance::psi4io::RDVR3_Psi4, 686
    forcebalance::psi4io::THCDF_Psi4, 731
    forcebalance::target::RemoteTarget, 700
    forcebalance::target::Target, 714
    forcebalance::thermo::Thermo, 747
    forcebalance::tinkerio::AbInitio_TINKER, 206
    forcebalance::tinkerio::BindingEnergy_TINKER, 271
    forcebalance::tinkerio::Interaction_TINKER, 389
    forcebalance::tinkerio::Liquid_TINKER, 522
    forcebalance::tinkerio::Moments_TINKER, 619
    forcebalance::tinkerio::Vibration_TINKER, 821
    forcebalance::vibration::Vibration, 791
write_key
    forcebalance::tinkerio, 72
write_mdcrd
    forcebalance::molecule::Molecule, 556
write_mdp
    forcebalance::gmxio, 31
write_molproq
    forcebalance::molecule::Molecule, 556
write_ndx
    forcebalance::gmxio, 32
write_nested_destroy
    forcebalance::psi4io::THCDF_Psi4, 730
write_objective
    forcebalance::abinitio::AbInitio, 96
    forcebalance::abinitio_internal::AbInitio_Internal, 162
    forcebalance::amberio::AbInitio_AMBER, 118
    forcebalance::binding::BindingEnergy, 226
    forcebalance::counterpoise::Counterpoise, 290
    forcebalance::gmxio::AbInitio_GMX, 140
    forcebalance::gmxio::BindingEnergy_GMX, 241
    forcebalance::gmxio::Interaction_GMX, 359
    forcebalance::gmxio::Lipid_GMX, 446
    forcebalance::gmxio::Liquid_GMX, 483
    forcebalance::gmxio::Moments_GMX, 589
    forcebalance::gmxio::Thermo_GMX, 764
    forcebalance::gmxio::Vibration_GMX, 807
    forcebalance::interaction::Interaction, 344
    forcebalance::leastsq::LeastSquares, 408
    forcebalance::lipid::Lipid, 428
    forcebalance::liquid::Liquid, 464
    forcebalance::moments::Moments, 574
    forcebalance::openmmio::AbInitio_OpenMM, 184
    forcebalance::openmmio::BindingEnergy_OpenMM,
        256
    forcebalance::openmmio::Interaction_OpenMM, 374
    forcebalance::openmmio::Liquid_OpenMM, 502
    forcebalance::openmmio::Moments_OpenMM, 604
    forcebalance::psi4io::RDVR3_Psi4, 686
    forcebalance::psi4io::THCDF_Psi4, 731
    forcebalance::target::RemoteTarget, 700
    forcebalance::target::Target, 715
    forcebalance::thermo::Thermo, 747
    forcebalance::tinkerio::AbInitio_TINKER, 206
    forcebalance::tinkerio::BindingEnergy_TINKER, 271
    forcebalance::tinkerio::Interaction_TINKER, 389
    forcebalance::tinkerio::Liquid_TINKER, 522
    forcebalance::tinkerio::Moments_TINKER, 619
    forcebalance::tinkerio::Vibration_TINKER, 822
    forcebalance::vibration::Vibration, 791
write_pdb
    forcebalance::molecule::Molecule, 556
write_qcin
    forcebalance::molecule::Molecule, 557
write_qdata
    forcebalance::molecule::Molecule, 557
write_xyz
    forcebalance::molecule::Molecule, 558
write_techk
    forcebalance::optimizer::Optimizer, 651
x
    forcebalance::Mol2::mol2_atom, 530
    forcebalance::molecule::MyG, 622
x.best

```

forcebalance::optimizer::Optimizer, 653  
 x\_prev  
     forcebalance::optimizer::Optimizer, 653  
 XMLFILE  
     forcebalance::nifty, 57  
 xct  
     forcebalance::abinitio::AbInitio, 96  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 162  
     forcebalance::amberio::AbInitio\_AMBER, 118  
     forcebalance::binding::BindingEnergy, 226  
     forcebalance::counterpoise::Counterpoise, 290  
     forcebalance::gmxi::AbInitio\_GMX, 140  
     forcebalance::gmxi::BindingEnergy\_GMX, 241  
     forcebalance::gmxi::Interaction\_GMX, 360  
     forcebalance::gmxi::Lipid\_GMX, 446  
     forcebalance::gmxi::Liquid\_GMX, 483  
     forcebalance::gmxi::Moments\_GMX, 589  
     forcebalance::gmxi::Thermo\_GMX, 764  
     forcebalance::gmxi::Vibration\_GMX, 807  
     forcebalance::interaction::Interaction, 344  
     forcebalance::leastsq::LeastSquares, 408  
     forcebalance::lipid::Lipid, 428  
     forcebalance::liquid::Liquid, 464  
     forcebalance::moments::Moments, 574  
     forcebalance::openmmio::AbInitio\_OpenMM, 184  
     forcebalance::openmmio::BindingEnergy\_OpenMM,  
         256  
     forcebalance::openmmio::Interaction\_OpenMM, 375  
     forcebalance::openmmio::Liquid\_OpenMM, 502  
     forcebalance::openmmio::Moments\_OpenMM, 604  
     forcebalance::psi4io::RDVR3\_Psi4, 686  
     forcebalance::psi4io::THCDF\_Psi4, 731  
     forcebalance::target::RemoteTarget, 700  
     forcebalance::target::Target, 715  
     forcebalance::thermo::Thermo, 747  
     forcebalance::tinkerio::AbInitio\_TINKER, 206  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 271  
     forcebalance::tinkerio::Interaction\_TINKER, 390  
     forcebalance::tinkerio::Liquid\_TINKER, 522  
     forcebalance::tinkerio::Moments\_TINKER, 619  
     forcebalance::tinkerio::Vibration\_TINKER, 822  
     forcebalance::vibration::Vibration, 792  
 xk\_prev  
     forcebalance::optimizer::Optimizer, 653  
 Xp  
     forcebalance::gmxi::Lipid\_GMX, 446  
     forcebalance::gmxi::Liquid\_GMX, 483  
     forcebalance::gmxi::Thermo\_GMX, 764  
     forcebalance::lipid::Lipid, 428  
     forcebalance::liquid::Liquid, 464  
     forcebalance::openmmio::Liquid\_OpenMM, 502  
     forcebalance::thermo::Thermo, 747  
     forcebalance::tinkerio::Liquid\_TINKER, 522  
 xyz\_omms