

ForceBalance Developer API Guide version 1.1

Generated by Doxygen 1.7.6.1



# Contents

<b>1</b>	<b>Project Roadmap</b>	<b>1</b>
1.1	Most Recently Implemented: . . . . .	1
1.2	Current Development Goals, for version 1.2.2: . . . . .	1
1.3	Longterm Development Ideas . . . . .	1
<b>2</b>	<b>Todo List</b>	<b>2</b>
<b>3</b>	<b>Namespace Index</b>	<b>3</b>
3.1	Packages . . . . .	3
<b>4</b>	<b>Hierarchical Index</b>	<b>5</b>
4.1	Class Hierarchy . . . . .	5
<b>5</b>	<b>Class Index</b>	<b>7</b>
5.1	Class List . . . . .	7
<b>6</b>	<b>File Index</b>	<b>10</b>
6.1	File List . . . . .	10
<b>7</b>	<b>Namespace Documentation</b>	<b>11</b>
7.1	forcebalance Namespace Reference . . . . .	11
7.1.1	Variable Documentation . . . . .	13
7.2	forcebalance.abinitio Namespace Reference . . . . .	13
7.2.1	Detailed Description . . . . .	13
7.2.2	Function Documentation . . . . .	13
7.2.3	Variable Documentation . . . . .	14
7.3	forcebalance.abinitio_internal Namespace Reference . . . . .	14
7.3.1	Detailed Description . . . . .	14
7.4	forcebalance.amberio Namespace Reference . . . . .	14
7.4.1	Detailed Description . . . . .	15
7.4.2	Function Documentation . . . . .	15
7.4.3	Variable Documentation . . . . .	15
7.5	forcebalance.binding Namespace Reference . . . . .	16
7.5.1	Detailed Description . . . . .	16
7.5.2	Function Documentation . . . . .	16
7.5.3	Variable Documentation . . . . .	16
7.6	forcebalance.chemistry Namespace Reference . . . . .	17
7.6.1	Function Documentation . . . . .	17
7.6.2	Variable Documentation . . . . .	17
7.7	forcebalance.contact Namespace Reference . . . . .	19
7.7.1	Function Documentation . . . . .	19
7.8	forcebalance.counterpoise Namespace Reference . . . . .	19
7.8.1	Detailed Description . . . . .	20
7.8.2	Variable Documentation . . . . .	20
7.9	forcebalance.custom_io Namespace Reference . . . . .	20
7.9.1	Detailed Description . . . . .	20
7.9.2	Variable Documentation . . . . .	21
7.10	forcebalance.engine Namespace Reference . . . . .	21
7.10.1	Variable Documentation . . . . .	21
7.11	forcebalance.finite_difference Namespace Reference . . . . .	21
7.11.1	Function Documentation . . . . .	22
7.11.2	Variable Documentation . . . . .	25
7.12	forcebalance.forcefield Namespace Reference . . . . .	25

7.12.1	Detailed Description	26
7.12.2	Function Documentation	27
7.12.3	Variable Documentation	27
7.13	forcebalance.gmxio Namespace Reference	28
7.13.1	Detailed Description	29
7.13.2	Function Documentation	29
7.13.3	Variable Documentation	30
7.14	forcebalance.gmxqpio Namespace Reference	31
7.14.1	Function Documentation	32
7.14.2	Variable Documentation	32
7.15	forcebalance.interaction Namespace Reference	32
7.15.1	Detailed Description	32
7.15.2	Variable Documentation	32
7.16	forcebalance.leastsq Namespace Reference	32
7.16.1	Function Documentation	33
7.16.2	Variable Documentation	33
7.17	forcebalance.liquid Namespace Reference	33
7.17.1	Detailed Description	33
7.17.2	Function Documentation	33
7.17.3	Variable Documentation	34
7.18	forcebalance.Mol2 Namespace Reference	34
7.18.1	Variable Documentation	34
7.19	forcebalance.mol2io Namespace Reference	34
7.19.1	Detailed Description	34
7.19.2	Variable Documentation	34
7.20	forcebalance.molecule Namespace Reference	35
7.20.1	Function Documentation	36
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	60
7.23	forcebalance.objective Namespace Reference	61
7.23.1	Detailed Description	61
7.23.2	Variable Documentation	61
7.24	forcebalance.openmmio Namespace Reference	62
7.24.1	Detailed Description	63
7.24.2	Function Documentation	63
7.24.3	Variable Documentation	66
7.25	forcebalance.optimizer Namespace Reference	67
7.25.1	Detailed Description	67
7.25.2	Function Documentation	67
7.25.3	Variable Documentation	67
7.26	forcebalance.output Namespace Reference	68
7.27	forcebalance.parser Namespace Reference	68
7.27.1	Detailed Description	69
7.27.2	Function Documentation	70
7.27.3	Variable Documentation	71
7.28	forcebalance.psi4io Namespace Reference	72
7.28.1	Detailed Description	73
7.28.2	Variable Documentation	73

7.29	forcebalance.PT Namespace Reference . . . . .	73
7.29.1	Variable Documentation . . . . .	73
7.30	forcebalance.qchemio Namespace Reference . . . . .	74
7.30.1	Detailed Description . . . . .	74
7.30.2	Function Documentation . . . . .	74
7.30.3	Variable Documentation . . . . .	74
7.31	forcebalance.target Namespace Reference . . . . .	75
7.31.1	Variable Documentation . . . . .	75
7.32	forcebalance.tinkerio Namespace Reference . . . . .	75
7.32.1	Detailed Description . . . . .	76
7.32.2	Function Documentation . . . . .	76
7.32.3	Variable Documentation . . . . .	76
7.33	forcebalance.vibration Namespace Reference . . . . .	77
7.33.1	Detailed Description . . . . .	77
7.33.2	Variable Documentation . . . . .	77
<b>8</b>	<b>Class Documentation</b> . . . . .	<b>77</b>
8.1	forcebalance.abinitio.AblInitio Class Reference . . . . .	77
8.1.1	Detailed Description . . . . .	81
8.1.2	Constructor & Destructor Documentation . . . . .	81
8.1.3	Member Function Documentation . . . . .	82
8.1.4	Member Data Documentation . . . . .	91
8.2	forcebalance.amberio.AblInitio_AMBER Class Reference . . . . .	94
8.2.1	Detailed Description . . . . .	99
8.2.2	Constructor & Destructor Documentation . . . . .	99
8.2.3	Member Function Documentation . . . . .	99
8.2.4	Member Data Documentation . . . . .	109
8.3	forcebalance.gmxio.AblInitio_GMX Class Reference . . . . .	112
8.3.1	Detailed Description . . . . .	117
8.3.2	Constructor & Destructor Documentation . . . . .	117
8.3.3	Member Function Documentation . . . . .	117
8.3.4	Member Data Documentation . . . . .	128
8.4	forcebalance.abinitio_internal.AblInitio_Internal Class Reference . . . . .	131
8.4.1	Detailed Description . . . . .	136
8.4.2	Constructor & Destructor Documentation . . . . .	136
8.4.3	Member Function Documentation . . . . .	136
8.4.4	Member Data Documentation . . . . .	145
8.5	forcebalance.openmmio.AblInitio_OpenMM Class Reference . . . . .	148
8.5.1	Detailed Description . . . . .	153
8.5.2	Constructor & Destructor Documentation . . . . .	153
8.5.3	Member Function Documentation . . . . .	153
8.5.4	Member Data Documentation . . . . .	164
8.6	forcebalance.tinkerio.AblInitio_TINKER Class Reference . . . . .	167
8.6.1	Detailed Description . . . . .	172
8.6.2	Constructor & Destructor Documentation . . . . .	172
8.6.3	Member Function Documentation . . . . .	172
8.6.4	Member Data Documentation . . . . .	183
8.7	forcebalance.forcefield.BackedUpDict Class Reference . . . . .	186
8.7.1	Detailed Description . . . . .	187
8.7.2	Constructor & Destructor Documentation . . . . .	187
8.7.3	Member Function Documentation . . . . .	187
8.7.4	Member Data Documentation . . . . .	188
8.8	forcebalance.BaseClass Class Reference . . . . .	188
8.8.1	Detailed Description . . . . .	189

8.8.2	Constructor & Destructor Documentation . . . . .	189
8.8.3	Member Function Documentation . . . . .	189
8.8.4	Member Data Documentation . . . . .	189
8.9	<b>forcebalance.BaseReader Class Reference . . . . .</b>	190
8.9.1	Detailed Description . . . . .	191
8.9.2	Constructor & Destructor Documentation . . . . .	191
8.9.3	Member Function Documentation . . . . .	192
8.9.4	Member Data Documentation . . . . .	192
8.10	<b>forcebalance.binding.BindingEnergy Class Reference . . . . .</b>	193
8.10.1	Detailed Description . . . . .	195
8.10.2	Constructor & Destructor Documentation . . . . .	195
8.10.3	Member Function Documentation . . . . .	195
8.10.4	Member Data Documentation . . . . .	201
8.11	<b>forcebalance.tinkerio.BindingEnergy_TINKER Class Reference . . . . .</b>	202
8.11.1	Detailed Description . . . . .	204
8.11.2	Constructor & Destructor Documentation . . . . .	204
8.11.3	Member Function Documentation . . . . .	205
8.11.4	Member Data Documentation . . . . .	210
8.12	<b>forcebalance.output.CleanFileHandler Class Reference . . . . .</b>	211
8.12.1	Detailed Description . . . . .	212
8.12.2	Member Function Documentation . . . . .	212
8.13	<b>forcebalance.output.CleanStreamHandler Class Reference . . . . .</b>	212
8.13.1	Detailed Description . . . . .	213
8.13.2	Constructor & Destructor Documentation . . . . .	213
8.13.3	Member Function Documentation . . . . .	214
8.14	<b>forcebalance.counterpoise.Counterpoise Class Reference . . . . .</b>	214
8.14.1	Detailed Description . . . . .	216
8.14.2	Constructor & Destructor Documentation . . . . .	216
8.14.3	Member Function Documentation . . . . .	217
8.14.4	Member Data Documentation . . . . .	223
8.15	<b>forcebalance.engine.Engine Class Reference . . . . .</b>	224
8.15.1	Detailed Description . . . . .	225
8.15.2	Constructor & Destructor Documentation . . . . .	226
8.15.3	Member Function Documentation . . . . .	226
8.15.4	Member Data Documentation . . . . .	227
8.16	<b>forcebalance.forcefield.FF Class Reference . . . . .</b>	227
8.16.1	Detailed Description . . . . .	230
8.16.2	Constructor & Destructor Documentation . . . . .	230
8.16.3	Member Function Documentation . . . . .	230
8.16.4	Member Data Documentation . . . . .	237
8.17	<b>forcebalance.output.ForceBalanceLogger Class Reference . . . . .</b>	239
8.17.1	Detailed Description . . . . .	239
8.17.2	Constructor & Destructor Documentation . . . . .	240
8.17.3	Member Function Documentation . . . . .	240
8.17.4	Member Data Documentation . . . . .	240
8.18	<b>forcebalance.amberio.FrcMod_Reader Class Reference . . . . .</b>	240
8.18.1	Detailed Description . . . . .	242
8.18.2	Constructor & Destructor Documentation . . . . .	242
8.18.3	Member Function Documentation . . . . .	242
8.18.4	Member Data Documentation . . . . .	243
8.19	<b>forcebalance.psi4io.GBS_Reader Class Reference . . . . .</b>	243
8.19.1	Detailed Description . . . . .	245
8.19.2	Constructor & Destructor Documentation . . . . .	245
8.19.3	Member Function Documentation . . . . .	245

8.19.4 Member Data Documentation . . . . .	246
8.20 forcebalance.custom_io.Gen_Reader Class Reference . . . . .	247
8.20.1 Detailed Description . . . . .	249
8.20.2 Constructor & Destructor Documentation . . . . .	249
8.20.3 Member Function Documentation . . . . .	249
8.20.4 Member Data Documentation . . . . .	250
8.21 forcebalance.gmxio.GMX Class Reference . . . . .	250
8.21.1 Detailed Description . . . . .	252
8.21.2 Constructor & Destructor Documentation . . . . .	253
8.21.3 Member Function Documentation . . . . .	253
8.21.4 Member Data Documentation . . . . .	255
8.22 forcebalance.psi4io.Grid_Reader Class Reference . . . . .	255
8.22.1 Detailed Description . . . . .	257
8.22.2 Constructor & Destructor Documentation . . . . .	257
8.22.3 Member Function Documentation . . . . .	257
8.22.4 Member Data Documentation . . . . .	258
8.23 forcebalance.interaction.Interaction Class Reference . . . . .	259
8.23.1 Detailed Description . . . . .	262
8.23.2 Constructor & Destructor Documentation . . . . .	262
8.23.3 Member Function Documentation . . . . .	262
8.23.4 Member Data Documentation . . . . .	269
8.24 forcebalance.gmxio.Interaction_GMX Class Reference . . . . .	270
8.24.1 Detailed Description . . . . .	274
8.24.2 Constructor & Destructor Documentation . . . . .	274
8.24.3 Member Function Documentation . . . . .	274
8.24.4 Member Data Documentation . . . . .	281
8.25 forcebalance.openmmio.Interaction_OpenMM Class Reference . . . . .	282
8.25.1 Detailed Description . . . . .	286
8.25.2 Constructor & Destructor Documentation . . . . .	286
8.25.3 Member Function Documentation . . . . .	286
8.25.4 Member Data Documentation . . . . .	293
8.26 forcebalance.tinkerio.Interaction_TINKER Class Reference . . . . .	295
8.26.1 Detailed Description . . . . .	298
8.26.2 Constructor & Destructor Documentation . . . . .	298
8.26.3 Member Function Documentation . . . . .	298
8.26.4 Member Data Documentation . . . . .	305
8.27 forcebalance.gmxio.ITP_Reader Class Reference . . . . .	306
8.27.1 Detailed Description . . . . .	308
8.27.2 Constructor & Destructor Documentation . . . . .	309
8.27.3 Member Function Documentation . . . . .	309
8.27.4 Member Data Documentation . . . . .	310
8.28 forcebalance.leastsq.LeastSquares Class Reference . . . . .	311
8.28.1 Detailed Description . . . . .	313
8.28.2 Constructor & Destructor Documentation . . . . .	313
8.28.3 Member Function Documentation . . . . .	313
8.28.4 Member Data Documentation . . . . .	320
8.29 forcebalance.nifty.LineChunker Class Reference . . . . .	321
8.29.1 Detailed Description . . . . .	322
8.29.2 Constructor & Destructor Documentation . . . . .	322
8.29.3 Member Function Documentation . . . . .	322
8.29.4 Member Data Documentation . . . . .	322
8.30 forcebalance.liquid.Liquid Class Reference . . . . .	322
8.30.1 Detailed Description . . . . .	325
8.30.2 Constructor & Destructor Documentation . . . . .	325

8.30.3 Member Function Documentation . . . . .	325
8.30.4 Member Data Documentation . . . . .	332
8.31 forcebalance.gmxio.Liquid_GMX Class Reference . . . . .	334
8.31.1 Detailed Description . . . . .	337
8.31.2 Constructor & Destructor Documentation . . . . .	337
8.31.3 Member Function Documentation . . . . .	337
8.31.4 Member Data Documentation . . . . .	344
8.32 forcebalance.openmmio.Liquid_OpenMM Class Reference . . . . .	346
8.32.1 Detailed Description . . . . .	349
8.32.2 Constructor & Destructor Documentation . . . . .	349
8.32.3 Member Function Documentation . . . . .	349
8.32.4 Member Data Documentation . . . . .	357
8.33 forcebalance.tinkerio.Liquid_TINKER Class Reference . . . . .	360
8.33.1 Detailed Description . . . . .	363
8.33.2 Constructor & Destructor Documentation . . . . .	363
8.33.3 Member Function Documentation . . . . .	363
8.33.4 Member Data Documentation . . . . .	370
8.34 forcebalance.Mol2.mol2 Class Reference . . . . .	372
8.34.1 Detailed Description . . . . .	373
8.34.2 Constructor & Destructor Documentation . . . . .	373
8.34.3 Member Function Documentation . . . . .	373
8.34.4 Member Data Documentation . . . . .	377
8.35 forcebalance.Mol2.mol2.atom Class Reference . . . . .	378
8.35.1 Detailed Description . . . . .	378
8.35.2 Constructor & Destructor Documentation . . . . .	378
8.35.3 Member Function Documentation . . . . .	379
8.35.4 Member Data Documentation . . . . .	381
8.36 forcebalance.Mol2.mol2.bond Class Reference . . . . .	382
8.36.1 Detailed Description . . . . .	382
8.36.2 Constructor & Destructor Documentation . . . . .	382
8.36.3 Member Function Documentation . . . . .	382
8.36.4 Member Data Documentation . . . . .	384
8.37 forcebalance.amberio.Mol2_Reader Class Reference . . . . .	384
8.37.1 Detailed Description . . . . .	386
8.37.2 Constructor & Destructor Documentation . . . . .	386
8.37.3 Member Function Documentation . . . . .	386
8.37.4 Member Data Documentation . . . . .	387
8.38 forcebalance.mol2io.Mol2_Reader Class Reference . . . . .	388
8.38.1 Detailed Description . . . . .	389
8.38.2 Constructor & Destructor Documentation . . . . .	389
8.38.3 Member Function Documentation . . . . .	389
8.38.4 Member Data Documentation . . . . .	390
8.39 forcebalance.Mol2.mol2.set Class Reference . . . . .	390
8.39.1 Detailed Description . . . . .	391
8.39.2 Constructor & Destructor Documentation . . . . .	391
8.39.3 Member Function Documentation . . . . .	391
8.39.4 Member Data Documentation . . . . .	391
8.40 forcebalance.molecule.Molecule Class Reference . . . . .	391
8.40.1 Detailed Description . . . . .	395
8.40.2 Constructor & Destructor Documentation . . . . .	395
8.40.3 Member Function Documentation . . . . .	395
8.40.4 Member Data Documentation . . . . .	410
8.41 forcebalance.molecule.MolfileTimestep Class Reference . . . . .	411
8.41.1 Detailed Description . . . . .	412

8.42	forcebalance.moments.Moments Class Reference . . . . .	412
8.42.1	Detailed Description . . . . .	415
8.42.2	Constructor & Destructor Documentation . . . . .	416
8.42.3	Member Function Documentation . . . . .	416
8.42.4	Member Data Documentation . . . . .	423
8.43	forcebalance.tinkerio.Moments_TINKER Class Reference . . . . .	424
8.43.1	Detailed Description . . . . .	427
8.43.2	Constructor & Destructor Documentation . . . . .	428
8.43.3	Member Function Documentation . . . . .	428
8.43.4	Member Data Documentation . . . . .	435
8.44	forcebalance.gmxqpio.Monomer_QTPIE Class Reference . . . . .	436
8.44.1	Detailed Description . . . . .	439
8.44.2	Constructor & Destructor Documentation . . . . .	439
8.44.3	Member Function Documentation . . . . .	439
8.44.4	Member Data Documentation . . . . .	446
8.45	forcebalance.objective.Objective Class Reference . . . . .	447
8.45.1	Detailed Description . . . . .	448
8.45.2	Constructor & Destructor Documentation . . . . .	449
8.45.3	Member Function Documentation . . . . .	449
8.45.4	Member Data Documentation . . . . .	450
8.46	forcebalance.openmmio.OpenMM_Reader Class Reference . . . . .	450
8.46.1	Detailed Description . . . . .	452
8.46.2	Constructor & Destructor Documentation . . . . .	452
8.46.3	Member Function Documentation . . . . .	452
8.46.4	Member Data Documentation . . . . .	453
8.47	forcebalance.optimizer.Optimizer Class Reference . . . . .	454
8.47.1	Detailed Description . . . . .	457
8.47.2	Constructor & Destructor Documentation . . . . .	457
8.47.3	Member Function Documentation . . . . .	457
8.47.4	Member Data Documentation . . . . .	462
8.48	forcebalance.objective.Penalty Class Reference . . . . .	464
8.48.1	Detailed Description . . . . .	464
8.48.2	Constructor & Destructor Documentation . . . . .	465
8.48.3	Member Function Documentation . . . . .	465
8.48.4	Member Data Documentation . . . . .	466
8.49	forcebalance.nifty.Pickler_LP Class Reference . . . . .	467
8.49.1	Detailed Description . . . . .	468
8.49.2	Constructor & Destructor Documentation . . . . .	468
8.50	forcebalance.qchemio.QCIn_Reader Class Reference . . . . .	468
8.50.1	Detailed Description . . . . .	470
8.50.2	Constructor & Destructor Documentation . . . . .	470
8.50.3	Member Function Documentation . . . . .	470
8.50.4	Member Data Documentation . . . . .	471
8.51	forcebalance.output.RawFileHandler Class Reference . . . . .	472
8.51.1	Detailed Description . . . . .	472
8.51.2	Member Function Documentation . . . . .	472
8.52	forcebalance.output.RawStreamHandler Class Reference . . . . .	473
8.52.1	Detailed Description . . . . .	473
8.52.2	Constructor & Destructor Documentation . . . . .	473
8.52.3	Member Function Documentation . . . . .	474
8.53	forcebalance.psi4io.RDVR3_Psi4 Class Reference . . . . .	474
8.53.1	Detailed Description . . . . .	476
8.53.2	Constructor & Destructor Documentation . . . . .	476
8.53.3	Member Function Documentation . . . . .	477

8.53.4 Member Data Documentation . . . . .	483
8.54 forcebalance.target.RemoteTarget Class Reference . . . . .	484
8.54.1 Detailed Description . . . . .	486
8.54.2 Constructor & Destructor Documentation . . . . .	486
8.54.3 Member Function Documentation . . . . .	486
8.54.4 Member Data Documentation . . . . .	492
8.55 forcebalance.target.Target Class Reference . . . . .	493
8.55.1 Detailed Description . . . . .	495
8.55.2 Constructor & Destructor Documentation . . . . .	495
8.55.3 Member Function Documentation . . . . .	496
8.55.4 Member Data Documentation . . . . .	502
8.56 forcebalance.psi4io.THCDF_Psi4 Class Reference . . . . .	503
8.56.1 Detailed Description . . . . .	505
8.56.2 Constructor & Destructor Documentation . . . . .	505
8.56.3 Member Function Documentation . . . . .	506
8.56.4 Member Data Documentation . . . . .	512
8.57 forcebalance.tinkerio.Tinker_Reader Class Reference . . . . .	513
8.57.1 Detailed Description . . . . .	515
8.57.2 Constructor & Destructor Documentation . . . . .	515
8.57.3 Member Function Documentation . . . . .	515
8.57.4 Member Data Documentation . . . . .	516
8.58 forcebalance.nifty.Unpickler_LP Class Reference . . . . .	517
8.58.1 Detailed Description . . . . .	518
8.58.2 Constructor & Destructor Documentation . . . . .	518
8.59 forcebalance.vibration.Vibration Class Reference . . . . .	518
8.59.1 Detailed Description . . . . .	520
8.59.2 Constructor & Destructor Documentation . . . . .	520
8.59.3 Member Function Documentation . . . . .	521
8.59.4 Member Data Documentation . . . . .	527
8.60 forcebalance.tinkerio.Vibration_TINKER Class Reference . . . . .	528
8.60.1 Detailed Description . . . . .	529
8.60.2 Constructor & Destructor Documentation . . . . .	529
8.60.3 Member Function Documentation . . . . .	529
<b>9 File Documentation</b>	<b>530</b>
9.1 __init__.py File Reference . . . . .	530
9.2 abinitio.py File Reference . . . . .	530
9.3 abinitio_internal.py File Reference . . . . .	530
9.4 amberio.py File Reference . . . . .	531
9.5 api.dox File Reference . . . . .	531
9.6 binding.py File Reference . . . . .	531
9.7 chemistry.py File Reference . . . . .	531
9.8 contact.py File Reference . . . . .	532
9.9 counterpoise.py File Reference . . . . .	532
9.10 custom_io.py File Reference . . . . .	533
9.11 engine.py File Reference . . . . .	533
9.12 finite_difference.py File Reference . . . . .	533
9.13 forcefield.py File Reference . . . . .	534
9.14 gmxio.py File Reference . . . . .	534
9.15 gmxqpio.py File Reference . . . . .	535
9.16 interaction.py File Reference . . . . .	536
9.17 leastsq.py File Reference . . . . .	536
9.18 liquid.py File Reference . . . . .	536
9.19 Mol2.py File Reference . . . . .	537

9.20 mol2io.py File Reference . . . . .	537
9.21 molecule.py File Reference . . . . .	538
9.22 moments.py File Reference . . . . .	539
9.23 nifty.py File Reference . . . . .	540
9.24 objective.py File Reference . . . . .	542
9.25 openmmio.py File Reference . . . . .	542
9.26 optimizer.py File Reference . . . . .	543
9.27 output.py File Reference . . . . .	544
9.28 parser.py File Reference . . . . .	544
9.29 psi4io.py File Reference . . . . .	545
9.30 PT.py File Reference . . . . .	545
9.31 qchemio.py File Reference . . . . .	545
9.32 target.py File Reference . . . . .	546
9.33 tinkerio.py File Reference . . . . .	546
9.34 vibration.py File Reference . . . . .	547

<b>Index</b>	<b>548</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]-123

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
- -123 stands for the number of commits that follow the manually specified 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:

- Implementing and expanding support for running target calculations in parallel across multiple nodes
- Development of a ForceBalance GUI interface to complement the current command line interface
- Expand unit test cases to provide more thorough coverage of forcebalance modules

### 1.2 Current Development Goals, for version 1.2.2:

- 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.

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.

**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>forcebalance</b>	11
<b>forcebalance.abinitio</b> Ab-initio fitting module (energies, forces, resp)	13
<b>forcebalance.abinitio_internal</b> Internal implementation of energy matching (for TIP3P water only)	14
<b>forcebalance.amberio</b> AMBER force field input/output	14
<b>forcebalance.binding</b> Binding energy fitting module	16
<b>forcebalance.chemistry</b>	17
<b>forcebalance.contact</b>	19
<b>forcebalance.counterpoise</b> Match an empirical potential to the counterpoise correction for basis set superposition error (BS-SE)	19
<b>forcebalance.custom_io</b> Custom force field parser	20

<b>forcebalance.engine</b>	21
<b>forcebalance.finite_difference</b>	21
<b>forcebalance.forcefield</b>	
<b>Force field module</b>	25
<b>forcebalance.gmxio</b>	
<b>GROMACS input/output</b>	28
<b>forcebalance.gmxqpio</b>	31
<b>forcebalance.interaction</b>	
<b>Interaction energy fitting module</b>	32
<b>forcebalance.leastsq</b>	32
<b>forcebalance.liquid</b>	
<b>Matching of liquid bulk properties</b>	33
<b>forcebalance.Mol2</b>	34
<b>forcebalance.mol2io</b>	
<b>Mol2 I/O</b>	34
<b>forcebalance.molecule</b>	35
<b>forcebalance.moments</b>	
<b>Multipole moment fitting module</b>	44
<b>forcebalance.nifty</b>	
<b>Nifty functions, intended to be imported by any module within ForceBalance</b>	44
<b>forcebalance.objective</b>	
<b>ForceBalance objective function</b>	61
<b>forcebalance.openmmio</b>	
<b>OpenMM input/output</b>	62
<b>forcebalance.optimizer</b>	
<b>Optimization algorithms</b>	67
<b>forcebalance.output</b>	68
<b>forcebalance.parser</b>	
<b>Input file parser for ForceBalance jobs</b>	68
<b>forcebalance.psi4io</b>	
<b>PSI4 force field input/output</b>	72
<b>forcebalance.PT</b>	73
<b>forcebalance.qchemio</b>	
<b>Q-Chem input file parser</b>	74
<b>forcebalance.target</b>	75

<b>forcebalance.tinkerio</b>	
TINKER input/output	75
<b>forcebalance.vibration</b>	
Vibrational mode fitting module	77

## 4 Hierarchical Index

### 4.1 Class Hierarchy

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

dict	
<b>forcebalance.forcefield.BackedUpDict</b>	186
FileHandler	
<b>forcebalance.output.CleanFileHandler</b>	211
<b>forcebalance.output.RawFileHandler</b>	472
Logger	
<b>forcebalance.output.ForceBalanceLogger</b>	239
<b>forcebalance.Mol2.mol2</b>	372
<b>forcebalance.Mol2.mol2_atom</b>	378
<b>forcebalance.Mol2.mol2_bond</b>	382
<b>forcebalance.Mol2.mol2_set</b>	390
object	
<b>forcebalance.BaseClass</b>	188
<b>forcebalance.engine.Engine</b>	224
<b>forcebalance.gmxio.GMX</b>	250
<b>forcebalance.forcefield.FF</b>	227
<b>forcebalance.objective.Objective</b>	447
<b>forcebalance.optimizer.Optimizer</b>	454
<b>forcebalance.target.Target</b>	493
<b>forcebalance.abinitio.ABInitio</b>	77
<b>forcebalance.abinitio_internal.ABInitio_Internal</b>	131
<b>forcebalance.amberio.ABInitio_AMBER</b>	94
<b>forcebalance.gmxio.ABInitio_GMX</b>	112
<b>forcebalance.openmmio.ABInitio_OpenMM</b>	148
<b>forcebalance.tinkerio.ABInitio_TINKER</b>	167

<b>forcebalance.binding.BindingEnergy</b>	<b>193</b>
<b>forcebalance.tinkerio.BindingEnergy_TINKER</b>	<b>202</b>
<b>forcebalance.counterpoise.Counterpoise</b>	<b>214</b>
<b>forcebalance.gmxqpio.Monomer_QTPIE</b>	<b>436</b>
<b>forcebalance.interaction.Interaction</b>	<b>259</b>
<b>forcebalance.gmxio.Interaction_GMX</b>	<b>270</b>
<b>forcebalance.openmmio.Interaction_OpenMM</b>	<b>282</b>
<b>forcebalance.tinkerio.Interaction_TINKER</b>	<b>295</b>
<b>forcebalance.leastsq.LeastSquares</b>	<b>311</b>
<b>forcebalance.psi4io.THCDF_Psi4</b>	<b>503</b>
<b>forcebalance.liquid.Liquid</b>	<b>322</b>
<b>forcebalance.gmxio.Liquid_GMX</b>	<b>334</b>
<b>forcebalance.openmmio.Liquid_OpenMM</b>	<b>346</b>
<b>forcebalance.tinkerio.Liquid_TINKER</b>	<b>360</b>
<b>forcebalance.moments.Moments</b>	<b>412</b>
<b>forcebalance.tinkerio.Moments_TINKER</b>	<b>424</b>
<b>forcebalance.psi4io.RDVR3_Psi4</b>	<b>474</b>
<b>forcebalance.target.RemoteTarget</b>	<b>484</b>
<b>forcebalance.vibration.Vibration</b>	<b>518</b>
<b>forcebalance.BaseReader</b>	<b>190</b>
<b>forcebalance.amberio.FrcMod_Reader</b>	<b>240</b>
<b>forcebalance.amberio.Mol2_Reader</b>	<b>384</b>
<b>forcebalance.custom_io.Gen_Reader</b>	<b>247</b>
<b>forcebalance.gmxio.ITP_Reader</b>	<b>306</b>
<b>forcebalance.mol2io.Mol2_Reader</b>	<b>388</b>
<b>forcebalance.openmmio.OpenMM_Reader</b>	<b>450</b>
<b>forcebalance.psi4io.GBS_Reader</b>	<b>243</b>
<b>forcebalance.psi4io.Grid_Reader</b>	<b>255</b>
<b>forcebalance.qchemio.QCIn_Reader</b>	<b>468</b>
<b>forcebalance.tinkerio.Tinker_Reader</b>	<b>513</b>

<b>forcebalance.molecule.Molecule</b>	391
<b>forcebalance.nifty.LineChunker</b>	321
<b>forcebalance.objective.Penalty</b>	464
Pickler	
<b>forcebalance.nifty.Pickler_LP</b>	467
StreamHandler	
<b>forcebalance.output.CleanStreamHandler</b>	212
<b>forcebalance.output.RawStreamHandler</b>	473
Structure	
<b>forcebalance.molecule.MolfileTimestep</b>	411
Unpickler	
<b>forcebalance.nifty.Unpickler_LP</b>	517
Vibration	
<b>forcebalance.tinkerio.Vibration_TINKER</b>	528

## 5 Class Index

### 5.1 Class List

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

<b>forcebalance.abinitio.AbInitio</b>	77
Subclass of Target for fitting force fields to ab initio data	
<b>forcebalance.amberio.AbInitio_AMBER</b>	94
Subclass of Target for force and energy matching using AMBER	
<b>forcebalance.gmxio.AbInitio_GMX</b>	112
Subclass of AbInitio for force and energy matching using normal GROMACS	
<b>forcebalance.abinitio_internal.AbInitio_Internal</b>	131
Subclass of Target for force and energy matching using an internal implementation	
<b>forcebalance.openmmio.AbInitio_OpenMM</b>	148
Subclass of AbInitio for force and energy matching using OpenMM	
<b>forcebalance.tinkerio.AbInitio_TINKER</b>	167
Subclass of Target for force and energy matching using TINKER	
<b>forcebalance.forcefield.BackedUpDict</b>	186
<b>forcebalance.BaseClass</b>	188
Provides some nifty functions that are common to all ForceBalance classes	
<b>forcebalance.BaseReader</b>	190
The 'reader' class	

<b>forcebalance.binding.BindingEnergy</b>	Improved subclass of Target for fitting force fields to binding energies	193
<b>forcebalance.tinkerio.BindingEnergy_TINKER</b>	Subclass of BindingEnergy for binding energy matching using TINKER	202
<b>forcebalance.output.CleanFileHandler</b>	File handler that does not write terminal escape codes and carriage returns to files	211
<b>forcebalance.output.CleanStreamHandler</b>	Similar to <code>RawStreamHandler</code> except it does not write terminal escape codes	212
<b>forcebalance.counterpoise.Counterpoise</b>	Target subclass for matching the counterpoise correction	214
<b>forcebalance.engine.Engine</b>	Base class for all engines	224
<b>forcebalance.forcefield.FF</b>	Force field class	227
<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	239
<b>forcebalance.amberio.FrcMod_Reader</b>	Finite state machine for parsing FrcMod force field file	240
<b>forcebalance.psi4io.GBS_Reader</b>	Interaction type -> Parameter Dictionary	243
<b>forcebalance.custom_io.Gen_Reader</b>	Finite state machine for parsing custom GROMACS force field files	247
<b>forcebalance.gmxio.GMX</b>	Derived from Engine object for carrying out general purpose GROMACS calculations	250
<b>forcebalance.psi4io.Grid_Reader</b>	Finite state machine for parsing DVR grid files	255
<b>forcebalance.interaction.Interaction</b>	Subclass of Target for fitting force fields to interaction energies	259
<b>forcebalance.gmxio.Interaction_GMX</b>	Subclass of Interaction for interaction energy matching using GROMACS	270
<b>forcebalance.openmmio.Interaction_OpenMM</b>	Subclass of Target for interaction matching using OpenMM	282
<b>forcebalance.tinkerio.Interaction_TINKER</b>	Subclass of Target for interaction matching using TINKER	295
<b>forcebalance.gmxio.ITP_Reader</b>	Finite state machine for parsing GROMACS force field files	306
<b>forcebalance.leastsq.LeastSquares</b>	Subclass of Target for general least squares fitting	311

<b>forcebalance.nifty.LineChunker</b>	<b>321</b>
<b>forcebalance.liquid.Liquid</b>	
Subclass of Target for liquid property matching	322
<b>forcebalance.gmxio.Liquid_GMX</b>	<b>334</b>
<b>forcebalance.openmmio.Liquid_OpenMM</b>	<b>346</b>
<b>forcebalance.tinkerio.Liquid_TINKER</b>	<b>360</b>
<b>forcebalance.Mol2.mol2</b>	
This is to manage one <b>Mol2</b> series of lines on the form:	372
<b>forcebalance.Mol2.mol2_atom</b>	
This is to manage <b>Mol2</b> atomic lines on the form: 1 C1 5.4790 42.2880 49.5910 C.ar 1 <1> 0.0424	378
<b>forcebalance.Mol2.mol2_bond</b>	
This is to manage <b>Mol2</b> bond lines on the form: 1 1 2 ar	382
<b>forcebalance.amberio.Mol2_Reader</b>	
Finite state machine for parsing <b>Mol2</b> force field file	384
<b>forcebalance.mol2io.Mol2_Reader</b>	
Finite state machine for parsing <b>Mol2</b> force field file	388
<b>forcebalance.Mol2.mol2_set</b>	<b>390</b>
<b>forcebalance.molecule.Molecule</b>	
Lee-Ping's general file format conversion class	391
<b>forcebalance.molecule.MolfileTimestep</b>	
Wrapper for the timestep C structure used in molfile plugins	411
<b>forcebalance.moments.Moments</b>	
Subclass of Target for fitting force fields to multipole moments (from experiment or theory)	412
<b>forcebalance.tinkerio.Moments_TINKER</b>	
Subclass of Target for multipole moment matching using TINKER	424
<b>forcebalance.gmxqpio.Monomer_QTPIE</b>	
Subclass of Target for monomer properties of QTPIE (implemented within gromacs WCV branch)	436
<b>forcebalance.objective.Objective</b>	
Objective function	447
<b>forcebalance.openmmio.OpenMM_Reader</b>	
Class for parsing OpenMM force field files	450
<b>forcebalance.optimizer.Optimizer</b>	
Optimizer class	454
<b>forcebalance.objective.Penalty</b>	
Penalty functions for regularizing the force field optimizer	464
<b>forcebalance.nifty.Pickler_LP</b>	
A subclass of the python Pickler that implements pickling of <b>ElementTree</b> types	467

<b>forcebalance.qchemio.QCIn.Reader</b>	Finite state machine for parsing Q-Chem input files	468
<b>forcebalance.output.RawFileHandler</b>	Exactly like output.FileHandler except it does no extra formatting before sending logging messages to the file	472
<b>forcebalance.output.RawStreamHandler</b>	Exactly like output.StreamHandler except it does no extra formatting before sending logging messages to the stream	473
<b>forcebalance.psi4io.RDVR3_Psi4</b>	Subclass of Target for R-DVR3 grid fitting	474
<b>forcebalance.target.RemoteTarget</b>		484
<b>forcebalance.target.Target</b>	Base class for all fitting targets	493
<b>forcebalance.psi4io.THCDF_Psi4</b>		503
<b>forcebalance.tinkerio.Tinker.Reader</b>	Finite state machine for parsing TINKER force field files	513
<b>forcebalance.nifty.Unpickler_LP</b>	A subclass of the python Unpickler that implements unpickling of _ElementTree types	517
<b>forcebalance.vibration.Vibration</b>	Subclass of Target for fitting force fields to vibrational spectra (from experiment or theory)	518
<b>forcebalance.tinkerio.Vibration_TINKER</b>	Subclass of Target for vibrational frequency matching using TINKER	528

## 6 File Index

### 6.1 File List

Here is a list of all files with brief descriptions:

<b>_init__.py</b>	530
<b>abinitio.py</b>	530
<b>abinitio_internal.py</b>	530
<b>amberio.py</b>	531
<b>binding.py</b>	531
<b>chemistry.py</b>	531
<b>contact.py</b>	532
<b>counterpoise.py</b>	532
<b>custom.io.py</b>	533

<code>engine.py</code>	533
<code>finite_difference.py</code>	533
<code>forcefield.py</code>	534
<code>gmxio.py</code>	534
<code>gmxqpio.py</code>	535
<code>interaction.py</code>	536
<code>leastsq.py</code>	536
<code>liquid.py</code>	536
<code>Mol2.py</code>	537
<code>mol2io.py</code>	537
<code>molecule.py</code>	538
<code>moments.py</code>	539
<code>nifty.py</code>	540
<code>objective.py</code>	542
<code>openmmio.py</code>	542
<code>optimizer.py</code>	543
<code>output.py</code>	544
<code>parser.py</code>	544
<code>psi4io.py</code>	545
<code>PT.py</code>	545
<code>qchemio.py</code>	545
<code>target.py</code>	546
<code>tinkerio.py</code>	546
<code>vibration.py</code>	547

## 7 Namespace Documentation

### 7.1 forcebalance Namespace Reference

#### Namespaces

- `abinitio`  
 $Ab\text{-}initio$  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.*
- [gmxqpio](#)
- [interaction](#)  
*Interaction energy fitting module.*
- [leastsq](#)
- [liquid](#)  
*Matching of liquid bulk properties.*
- [Mol2](#)
- [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.*
- [target](#)
- [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 )`

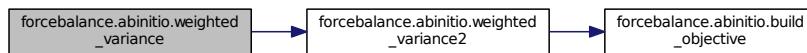
This function builds an objective function (number) from the complicated polytensor and covariance matrices.

Definition at line 1156 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 1126 of file abinitio.py.

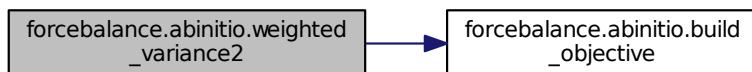
Here is the call graph for this function:



```
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 1140 of file abinitio.py.

Here is the call graph for this function:



### 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

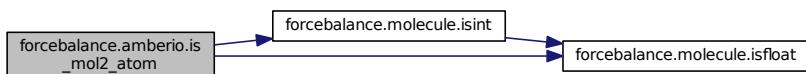
Date

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_pdict = {'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

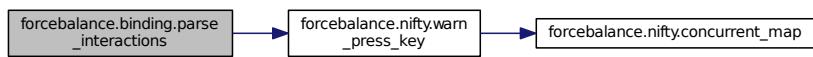
```
def forcebalance.binding.parse_interactions( input_file ) Parse through the interactions input file.
```

Parameters

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

Definition at line 31 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 23 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.*
- dictionary `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.****dictionary forcebalance.chemistry.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,
14     '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, 1.89, 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 )** 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

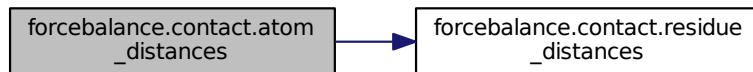
contacts should be a num\_contacts x 2 array where each row gives the indices of 2 atoms whos distance you care to monitor.

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 27 of file contact.py.

Here is the call graph for this function:



**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 86 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 29 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.*
- 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.ctypes = [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' : cotypes }
```

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 `in_fd`  
*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.

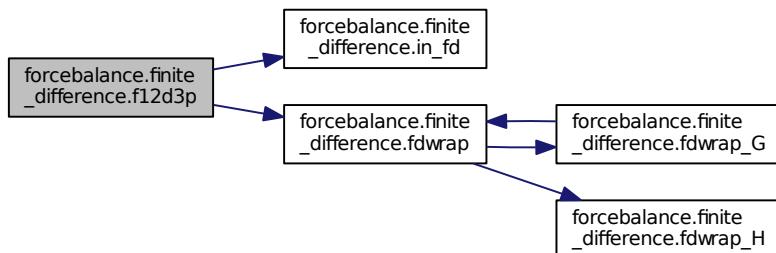
How to use: use fdwrap or something similar to generate a one-variable function from the (usually) much more complicated function that we wish to differentiate. Then pass it to this function.

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.

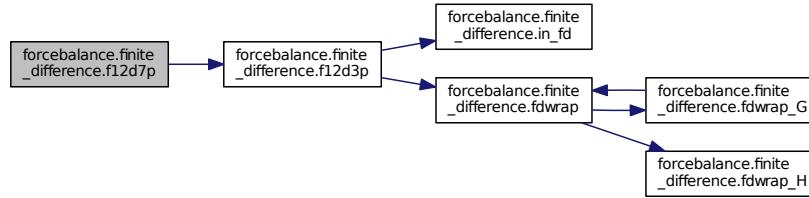
Definition at line 109 of file finite\_difference.py.

Here is the call graph for this function:



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

Here is the call graph for this function:



**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.

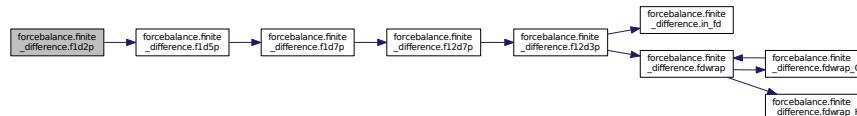
How to use: use fdwrap or something similar to generate a one-variable function from the (usually) much more complicated function that we wish to differentiate. Then pass it to this function.

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.

Here is the call graph for this function:



**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/>

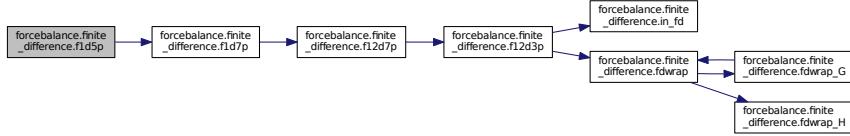
How to use: use fdwrap or something similar to generate a one-variable function from the (usually) much more complicated function that we wish to differentiate. Then pass it to this function.

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.

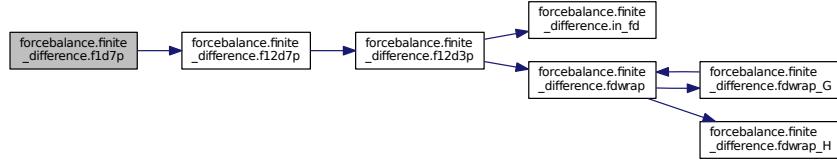
Here is the call graph for this function:



**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.

Here is the call graph for this function:



**def forcebalance.finite\_difference.fdwrap ( func, mvals0, pidx, key = None, kwargs )** A function wrapper for finite difference designed for differentiating 'get'-type functions.

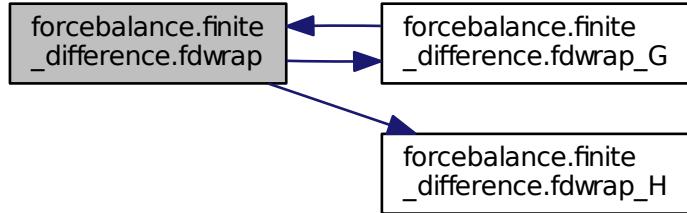
Since our finite difference stencils take single-variable functions and differentiate them around zero, and our objective function is quite a complicated function, we need a wrapper to serve as a middleman. The alternative would be to copy the finite difference formula to wherever we're taking the derivative, 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 147 of file finite\_difference.py.

Here is the call graph for this function:



**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 166 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 177 of file finite\_difference.py.

**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 121 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 145 of file forcefield.py.

**def forcebalance.forcefield.rs\_override ( *rsfactors*, *termttype*, *Temperature = 298.15* )** This function takes in a dictionary (rsfactors) and a string (termttype).

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

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

Parameters

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

Definition at line 1172 of file forcefield.py.

### 7.12.3 Variable Documentation

**dictionary forcebalance.forcefield.FF\_Extensions Initial value:**

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

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": openmmino.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 129 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 [AbInitio\\_GMX](#)  
*Subclass of AbInitio for force and energy matching using normal GROMACS.*
- class [Liquid\\_GMX](#)
- class [Interaction\\_GMX](#)  
*Subclass of Interaction for interaction energy matching using GROMACS.*

### Functions

- def [edit\\_mdp](#)  
*Create or edit a Gromacs MDP 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.*
  - string **shot.mdp**

### **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

**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.edit_mdp ( fin, fout, options, verbose = False ) Create or edit a Gromacs MDP file
```

### Parameters

in	<i>fin</i>	Input file name.
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.

Definition at line 37 of file qmxio.py.

Here is the call graph for this function:



**def forcebalance.qmxio.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 :

NG 11 22.5697 -0.0000 H 0.000120070229705 2.0002002500102001 0.00000100 ,

PARM 5 6

Look at all the variety!

## Parameters

in	line	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 183 of file gmxio.py.

Here is the call graph for this function:



**def forcebalance.gmxio.rm\_gmx\_baks ( dir )** Definition at line 430 of file gmxio.py.

### 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 93 of file gmxio.py.

**list forcebalance.gmxio.bftypes = [None, 'BONDS', 'G96BONDS', 'MORSE']** Bonded interaction function types.

Definition at line 91 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 96 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'          : aftypes,
7     'angletypes'      : aftypes,
8     'dihedrals'        : dftypes,
9     'dihedraltypes'   : dftypes,
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 104 of file gmxio.py.

**tuple forcebalance.gmxio.logger = getLogger(\_\_name\_\_)** Definition at line 28 of file gmxio.py.

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

Definition at line 87 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 127 of file gmxio.py.

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

Definition at line 89 of file gmxio.py.

**string forcebalance.gmxio.shot\_mdp** Initial value:

```
1 = """integrator = md
2 dt          = 0.001
3 nsteps      = 0
4 nstxout     = 0
5 nstfout     = 1
6 nstenergy   = 1
7 nstxtcout   = 0
8 xtc_grps    = System
9 energygrps  = System
10
11 nstlist     = 0
12 ns.type     = simple
13 rlist        = 0.0
14 vdwtype     = cut-off
15 coulombtype = cut-off
16 rcoulomb    = 0.0
17 rvdw        = 0.0
18 constraints = none
19 pbc         = no
20 """
```

Definition at line 438 of file gmxio.py.

## 7.14 forcebalance.gmxqpio Namespace Reference

### Classes

- class [Monomer\\_QTPIE](#)

*Subclass of Target for monomer properties of QTPIE (implemented within gromacs WCV branch).*

### Functions

- def [get\\_monomer\\_properties](#)

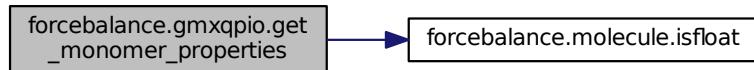
## Variables

- tuple `logger = getLogger(__name__)`

### 7.14.1 Function Documentation

`def forcebalance.gmxqpio.get_monomer_properties( print_stuff = 0 )` Definition at line 28 of file gmxqpio.py.

Here is the call graph for this function:



### 7.14.2 Variable Documentation

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

## 7.15 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__)`

### 7.15.1 Detailed Description

[Interaction](#) energy fitting module.

Author

Lee-Ping Wang

Date

05/2012

### 7.15.2 Variable Documentation

tuple `forcebalance.interaction.logger = getLogger(__name__)` Definition at line 21 of file interaction.py.

## 7.16 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`

### 7.16.1 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`.

### 7.16.2 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`.

## 7.17 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 [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 [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 [group](#)

*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', '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
- `Alive`

### 7.20.1 Function Documentation

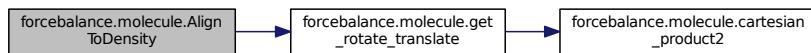
`def forcebalance.molecule.add_strip_to_mat( mat, strip )` Definition at line 441 of file molecule.py.

`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 544 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.

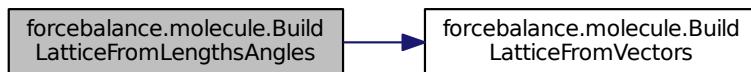
Definition at line 556 of file molecule.py.

**def forcebalance.molecule.both ( A, B, key )** Definition at line 479 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 261 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.BuildLatticeFromVectors ( v1, v2, v3 )** This function takes in three lattice vectors and tries to return a complete box specification.

The hash function is something we can use to discard two things that are obviously not equal. Here we neglect the hash. Return a list of the sorted atom numbers in this graph. Return a string of atoms, which serves as a rudimentary 'fingerprint' : '99,100,103,151' . Return an array of the elements. For instance ['H' 'C' 'C' 'H']. Create an Empirical Formula Get a list of the coordinates.

Definition at line 276 of file molecule.py.

**def forcebalance.molecule.cartesian\_product2 ( arrays )** Form a Cartesian product of two NumPy arrays.

Definition at line 617 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 527 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.diff ( A, B, key )** Definition at line 482 of file molecule.py.

**def forcebalance.molecule.either ( A, B, key )** Definition at line 490 of file molecule.py.

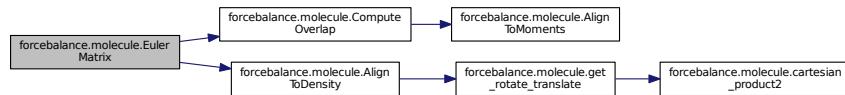
**def forcebalance.molecule.elem\_from\_atomname ( atomname )** Given an atom name, attempt to get the element in most cases.

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 499 of file molecule.py.

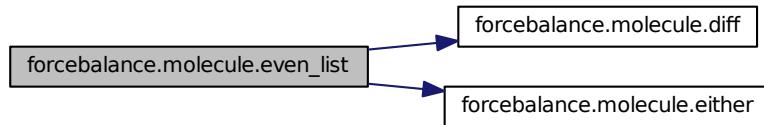
Here is the call graph for this function:



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

Definition at line 461 of file molecule.py.

Here is the call graph for this function:



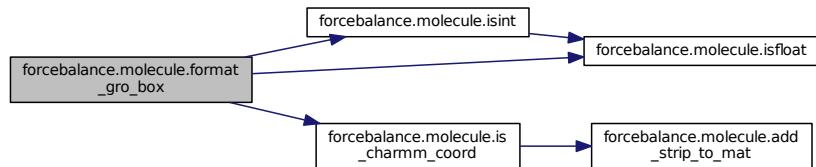
**def forcebalance.molecule.format\_gro\_box ( box )** Print a line corresponding to the box vector in accordance with .gro file format.

Parameters

in	box	Box NamedTuple
----	-----	----------------

Definition at line 392 of file molecule.py.

Here is the call graph for this function:



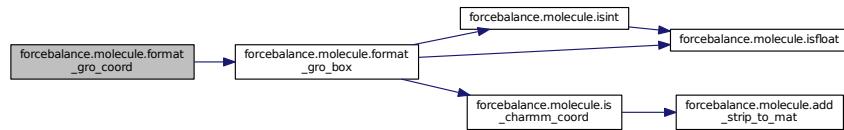
**def forcebalance.molecule.format\_gro\_coord ( resid, resname, aname, seqno, xyz )** Print a line in accordance with .gro file format, with six decimal points of precision.

## 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 371 of file molecule.py.

Here is the call graph for this function:



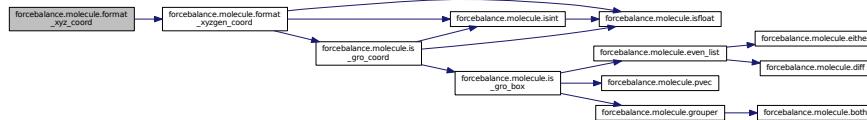
**def forcebalance.molecule.format\_xyz\_coord ( element, xyz, tinker = False )** Print a line consisting of (element, x, y, z) in accordance with .xyz file format.

## 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 355 of file molecule.py.

Here is the call graph for this function:



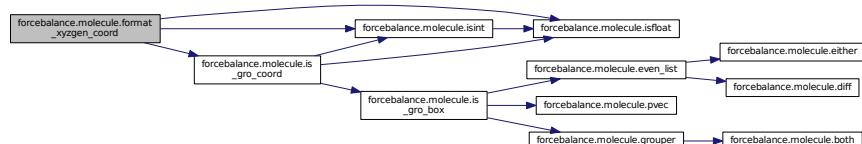
**def forcebalance.molecule.format\_xyzgen\_coord ( element, xyzgen )** Print a line consisting of (element, p, q, r, s, t, ...) where (p, q, r) are arbitrary atom-wise data (this might happen, for instance, with atomic charges)

## 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

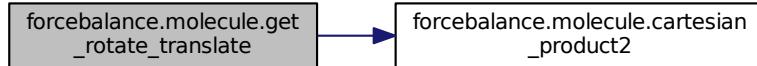
Definition at line 383 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.get\_rotate\_translate ( *matrix1*, *matrix2* )** Definition at line 579 of file molecule.py.

Here is the call graph for this function:



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

Here is the call graph for this function:



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

Definition at line 455 of file molecule.py.

Here is the call graph for this function:



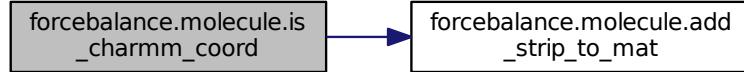
**def forcebalance.molecule.is\_charmm\_coord ( *line* )** Determines whether a line contains CHARMM data or not.

Parameters

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

Definition at line 419 of file molecule.py.

Here is the call graph for this function:

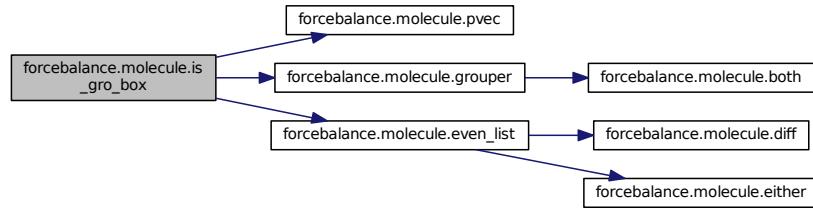


**def forcebalance.molecule.is\_gro\_box ( *line* )** Determines whether a line contains a GROMACS box vector or not.  
Parameters

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

Definition at line 432 of file molecule.py.

Here is the call graph for this function:

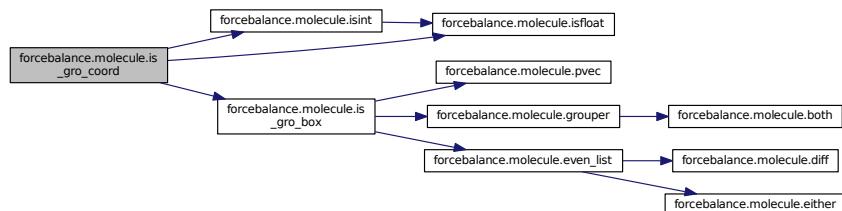


**def forcebalance.molecule.is\_gro\_coord ( *line* )** Determines whether a line contains GROMACS data or not.  
Parameters

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

Definition at line 404 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 250 of file molecule.py.

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

Definition at line 245 of file molecule.py.

Here is the call graph for this function:



```
def forcebalance.molecule.main ( ) Definition at line 2648 of file molecule.py.
```

```
def forcebalance.molecule.nodematch ( node1, node2 ) Definition at line 239 of file molecule.py.
```

Here is the call graph for this function:



```
def forcebalance.molecule.pvec ( vec ) Definition at line 450 of file molecule.py.
```

## 7.20.2 Variable Documentation

```
forcebalance.molecule.Alive Definition at line 305 of file molecule.py.
```

```
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 237 of file molecule.py.

```
tuple forcebalance.molecule.Box = namedtuple('Box',[a,b,c,alpha,beta,gamma,A,B,C,V]) Definition at line 257 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_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.

**tuple forcebalance.molecule.MetaVariableNames = set(['fnm', 'ftype', 'qcrems', 'qctemplate', '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                 .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                 .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                .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
                 , 140.9077), ('Nd', 144.24), ('Pm', 145), ('Sm', 150.36),
                  ('Eu', 151.964), ('Gd', 157.25), ('Tb', 158.9253), ('Dy', 162.5), ('Ho',
15                164.9303), ('Er', 167.259), ('Tm', 168.9342), ('Yb', 173.04),
16                ('Lu', 174.967), ('Hf', 178.49), ('Ta', 180.9479), ('W', 183.84), ('Re',
17                186.207), ('Os', 190.23), ('Ir', 192.217), ('Pt', 195.078),
18                ('Au', 196.9665), ('Hg', 200.59), ('Tl', 204.3833), ('Pb', 207.2), ('Bi',
19                208.9804), ('Po', 209), ('At', 210), ('Rn', 222),
20                ('Fr', 223), ('Ra', 226), ('Ac', 227), ('Th', 232.0381), ('Pa', 231.0359)
21                , ('U', 238.0289), ('Np', 237), ('Pu', 244),
22                ('Am', 243), ('Cm', 247), ('Bk', 247), ('Cf', 251), ('Es', 252), ('Fm',
23                257), ('Md', 258), ('No', 259),
24                ('Lr', 262), ('Rf', 261), ('Db', 262), ('Sg', 266), ('Bh', 264), ('Hs',
25                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 258 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, 1.22, 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, 2.03, 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 254 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 `pmat2d`  
*Printout of a 2-D matrix.*
- def `encode`
- def `segments`
- def `commadash`
- def `uncommadash`
- 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 `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 `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 `onefile`
- def `Golnto`
- def `allsplit`
- def `Leave`
- def `MissingFileInspection`
- 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().*
- def `multiopen`

*This function be given any of several variable types (single file name, file object, or list of lines, or a list of the above) and give a list of files:*

## 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 767 of file nifty.py.

Here is the call graph for this function:



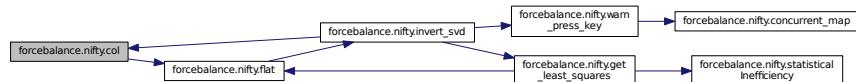
**def forcebalance.nifty.col ( vec )** Given any list, array, or matrix, return a 1-column matrix.

Input: vec = The input vector that is to be made into a column

Output: A column matrix

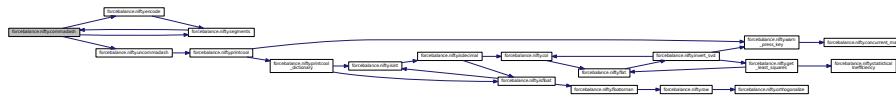
Definition at line 271 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.commadash ( / )** Definition at line 82 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 1022 of file nifty.py.

**def forcebalance.nifty.CopyFile( src, dest )** Definition at line 817 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.createWorkQueue( wq\_port, debug = True )** Definition at line 580 of file nifty.py.

**def forcebalance.nifty.destroyWorkQueue ( )** Definition at line 590 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.encode( / )** Definition at line 73 of file nifty.py.

Here is the call graph for this function:



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

## Parameters

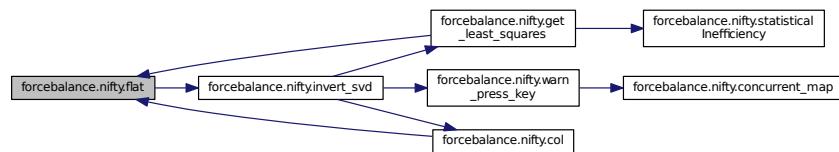
`in`      `vec` The data to be flattened

Returns

answer The flattened data

Definition at line 290 of file nifty.py.

Here is the call graph for this function:



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

```
@param[in] word The string to be converted
@return 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 253 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 | |
3 | | 1 (x0) (x0)^2 (x0)^3 |
4 | | 1 (x1) (x1)^2 (x1)^3 |
5 | | 1 (x2) (x2)^2 (x2)^3 |
6 | | 1 (x3) (x3)^2 (x3)^3 |
7 | | 1 (x4) (x4)^2 (x4)^3 |
8 | | -- -- |
```

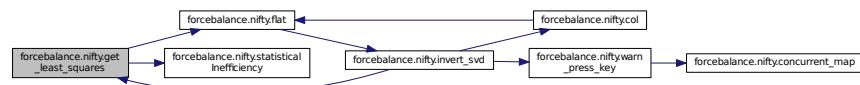
#### Parameters

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)

<code>out</code>	<code>yfit</code>	The fitted y-values
<code>out</code>	<code>MPPI</code>	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 358 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.getWorkQueue( )** Definition at line 572 of file nifty.py.

**def forcebalance.nifty.getWQIds( )** Definition at line 576 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.GolInt( Dir )** Definition at line 759 of file nifty.py.

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

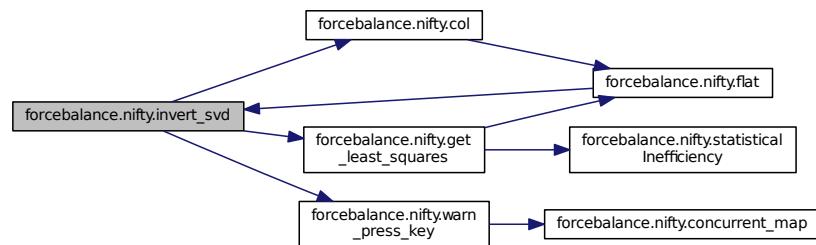
<code>in</code>	<code>X</code>	The matrix to be inverted
<code>in</code>	<code>thresh</code>	The SVD threshold; eigenvalues below this are not inverted but set to zero

Returns

`Xt` The inverted matrix

Definition at line 317 of file nifty.py.

Here is the call graph for this function:



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

## Parameters

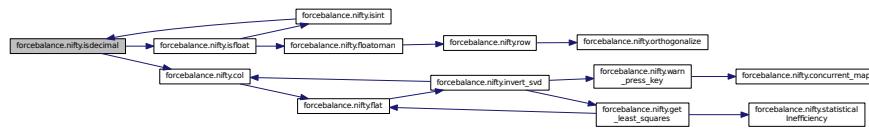
in	word	String (for instance, '123', '153.0', '2.', '-354')
----	------	---

## Returns

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

Definition at line 243 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

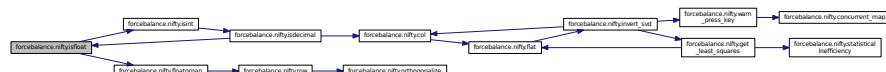
in	word	String (for instance, '123', '153.0', '2.', '-354')
----	------	---

## Returns

answer Boolean which specifies whether the string is any number

Definition at line 233 of file nifty.py.

Here is the call graph for this function:



**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 222 of file nifty.py.

Here is the call graph for this function:

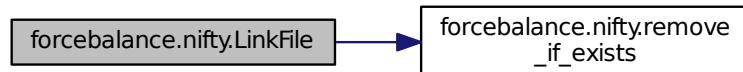


```
def forcebalance.nifty.Leave ( Dir ) Definition at line 773 of file nifty.py.
```

```
def forcebalance.nifty.link_dir_contents ( abssrcdir, absdestdir ) Definition at line 827 of file nifty.py.
```

```
def forcebalance.nifty.LinkFile ( src, dest, nosrcok = False ) Definition at line 804 of file nifty.py.
```

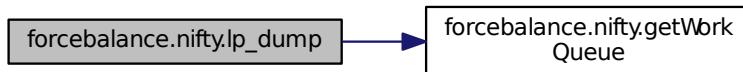
Here is the call graph for this function:



```
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 550 of file nifty.py.

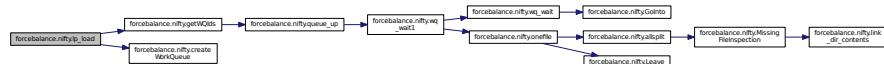
Here is the call graph for this function:



```
def forcebalance.nifty.lp_load ( file ) Use this instead of pickle.load for unpickling anything that contains _ElementTree types.
```

Definition at line 555 of file nifty.py.

Here is the call graph for this function:



```
def forcebalance.nifty.MissingFileInspection ( fnm ) Definition at line 794 of file nifty.py.
```

Here is the call graph for this function:



**def forcebalance.nifty.multiopen( arg )** This function be given any of several variable types (single file name, file object, or list of lines, or a list of the above) and give a list of files:

[file1, file2, file3 ... ]

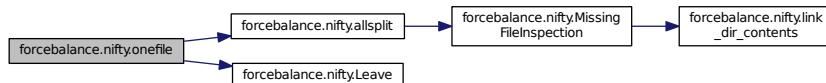
each of which can then be iterated over:

[[file1\_line1, file1\_line2 ... ], [file2\_line1, file2\_line2 ... ]]

Definition at line 1052 of file nifty.py.

**def forcebalance.nifty.onefile( ext, arg = None )** Definition at line 741 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 304 of file nifty.py.

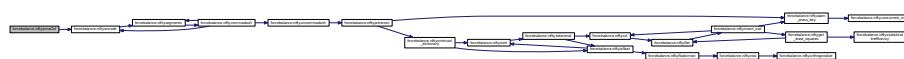
**def forcebalance.nifty.pmat2d( mat2d, precision = 1, loglevel = INFO )** Printout of a 2-D matrix.

Parameters

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

Definition at line 66 of file nifty.py.

Here is the call graph for this function:



```
def forcebalance.nifty.printcool( text, sym = "#", bold = False, color = 2, ansi = None, bottom = '--', minwidth = 50, center = True )  
    Cool-looking printout for slick formatting of output.
```

## Parameters

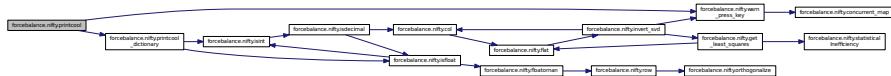
in	<i>text</i>	The string that the printout is based upon. This function will print out the string, ANSI-colored and enclosed in the symbol for example: <pre>##### ### I am cool ### #####</pre>
in	<i>sym</i>	The surrounding symbol
in	<i>bold</i>	Whether to use bold print
in	<i>color</i>	The ANSI color: 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

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

Definition at line 155 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.
```

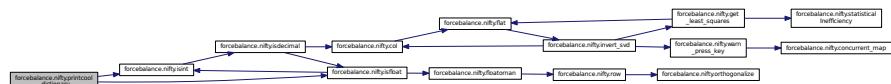
The keys in the dictionary are sorted before printing out.

## Parameters

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

Definition at line 198 of file nifty.py.

Here is the call graph for this function:



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

Parameters

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

Definition at line 55 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.queue\_up ( *wq*, *command*, *input\_files*, *output\_files*, *tgt* = *None*, *verbose* = *True* )**

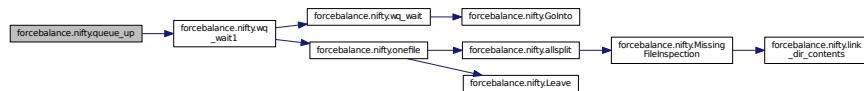
Submit a job to the Work Queue.

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 605 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.queue\_up\_src\_dest ( *wq*, *command*, *input\_files*, *output\_files*, *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 637 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 838 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 281 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.segments( e )** Definition at line 76 of file nifty.py.

Here is the call graph for this function:



```
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) q = statisticalnefficiency(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)`

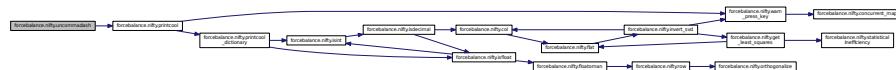
`@param[in] mintime (optional, int) - minimum amount of correlation function to compute (default: 3) The algorithm terminates after computing the correlation time out to mintime when the correlation function first goes negative. Note that this time may need to be increased if there is a strong initial 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 434 of file nifty.py

**def forcebalance.nifty.uncommadash( s )** Definition at line 92 of file nifty.py.

Here is the call graph for this function:



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

Definition at line 998 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.warn\_press\_key ( warning, timeout = 10 )** Definition at line 983 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.which( fnm )** Definition at line 842 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 730 of file nifty.py.

Here is the call graph for this function:

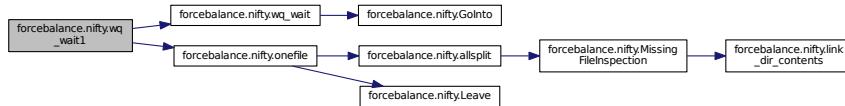


```
def forcebalance.nifty.wq_wait1 ( wq, wait_time = 10, wait_intvl = 1, print_time = 60, verbose = False )
```

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

Definition at line 658 of file nifty.py.

Here is the call graph for this function:



### 7.22.3 Variable Documentation

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

Definition at line 45 of file nifty.py.

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

Definition at line 41 of file nifty.py.

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

Definition at line 43 of file nifty.py.

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

Definition at line 39 of file nifty.py.

**tuple forcebalance.nifty.logger = getLogger(\_\_name\_\_)** Definition at line 34 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 792 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
     (or set the gmxpath option)",],
2      ([['force.mdin', 'stage.leap'], "This file is needed for setting up AMBER force matching
       targets"],),
3      ([['conf.pdb', 'mono.pdb'], "This file is needed for setting up OpenMM condensed phase
       property targets"],),
4      ([['liquid.xyz', 'liquid.key', 'mono.xyz', 'mono.key'], "This file is needed for setting up
       OpenMM condensed phase property targets"],),
5      ([['dynamic', 'analyze', 'minimize', 'testgrad', 'vibrate', 'optimize', 'polarize', '
       superpose'], "Make sure to install TINKER and add it to your path (or set the tinkerpath option")],
6      ([['runcuda.sh', 'npt.py', 'npt.tinker.py'], "This file belongs in the ForceBalance source
       directory, not sure why it is missing"],
7      ([['input.xyz']], "This file is needed for TINKER molecular property targets"),
8      ([['*.key$', '.*xyz$'], "I am guessing this file is probably needed by TINKER"]),
9      ([['*.gro$', '.*top$', '.*itp$', '.*mdp$', '.*ndx$'], "I am guessing this file is probably
       needed by GROMACS"])
10     ]
  
```

Definition at line 780 of file nifty.py.

**forcebalance.nifty.WORK\_QUEUE = None** Definition at line 567 of file nifty.py.

**tuple forcebalance.nifty.WQIDS = defaultdict(list)** Definition at line 570 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 496 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.objective.Implemented\_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     'LIQUID_OPENMM':Liquid_OpenMM,
9     'LIQUID_TINKER':Liquid_TINKER,
10    'LIQUID_GMX':Liquid_GMX,
11    'COUNTERPOISE':Counterpoise,
12    'THCDF_PSI4':THCDF_Psi4,
13    'RDVR3_PSI4':RDVR3_Psi4,
14    'INTERACTION_TINKER':Interaction_TINKER,
15    'INTERACTION_OPENMM':Interaction_OpenMM,
16    'BINDINGENERGY_TINKER':BindingEnergy_TINKER,
17    'MOMENTS_TINKER':Moments_TINKER,
18    'MONOMER_QTPIE':Monomer_QTPIE,
19    'REMOTE_TARGET':RemoteTarget,
20 }
```

The table of implemented Targets.

Definition at line 74 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 96 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 [Liquid\\_OpenMM](#)
- class [AbInitio\\_OpenMM](#)  
*Subclass of AbInitio for force and energy matching using OpenMM.*
- class [Interaction\\_OpenMM](#)  
*Subclass of Target for interaction matching using OpenMM.*

### Functions

- def [get\\_dipole](#)  
*Return the current dipole moment in Debye.*
- def [ResetVirtualSites](#)  
*Given a set of OpenMM-compatible positions and a System object, compute the correct virtual site positions according to 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 111 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.CopyAmoebaBondParameters ( *src*, *dest* )** Definition at line 97 of file openmmio.py.

Here is the call graph for this function:



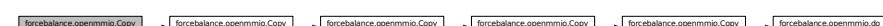
**def forcebalance.openmmio.CopyAmoebaInPlaneAngleParameters ( *src*, *dest* )** Definition at line 120 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.CopyAmoebaMultipoleParameters ( *src*, *dest* )** Definition at line 133 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.CopyAmoebaOutOfPlaneBendParameters ( *src*, *dest* )** Definition at line 103 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.CopyAmoebaVdwParameters ( src, dest )** Definition at line 129 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.CopyHarmonicAngleParameters ( src, dest )** Definition at line 141 of file openmmio.py.

Here is the call graph for this function:



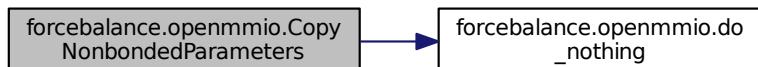
**def forcebalance.openmmio.CopyHarmonicBondParameters ( src, dest )** Definition at line 137 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.CopyNonbondedParameters ( src, dest )** Definition at line 149 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.CopyPeriodicTorsionParameters ( src, dest )** Definition at line 145 of file openmmio.py.

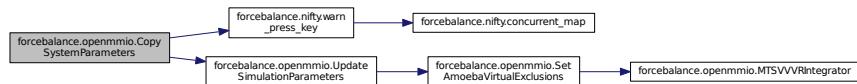
Here is the call graph for this function:



**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 163 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.do\_nothing ( *src*, *dest* )** Definition at line 156 of file openmmio.py.

**def forcebalance.openmmio.get\_dipole ( *simulation*, *q = 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 36 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.MTSVVVRIntegrator ( *temperature*, *collision\_rate*, *timestep*, *system*, *ninnersteps = 4* )** Create a multiple timestep velocity verlet with velocity randomization (VVVR) integrator.

#### ARGUMENTS

*temperature* (numpy.unit.Quantity compatible with kelvin) - the temperature  
*collision\_rate* (numpy.unit.Quantity compatible with 1/picoseconds) - the collision rate  
*timestep* (numpy.unit.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 241 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 66 of file openmmio.py.

Here is the call graph for this function:



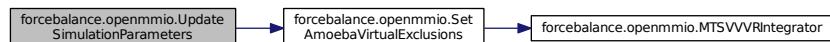
```
def forcebalance.openmmio.SetAmoebaVirtualExclusions ( system ) Definition at line 188 of file openmmio.py.
```

Here is the call graph for this function:



```
def forcebalance.openmmio.UpdateSimulationParameters ( src_system, dest_simulation ) Definition at line 182 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 24 of file openmmio.py.
```

```
string forcebalance.openmmio.pdict = "XML_Override" pdict is a useless variable if the force field is XML.  
Definition at line 320 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 306 of file openmmio.py.

## 7.25 forcebalance.optimizer Namespace Reference

Optimization algorithms.

### Classes

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

### Functions

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

### Variables

- tuple [logger](#) = getLogger(\_\_name\_\_)
- int [ITERATION\\_NUMBER](#) = 0
- int [GOODSTEP](#) = 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 29 of file optimizer.py.

Here is the call graph for this function:



**def forcebalance.optimizer.GoodStep ( )** Definition at line 33 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\_NUMBER = 0** Definition at line 25 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.*

## 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](#) = {"simtype": "type", "masterfile": "binding\_txt"}

*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.

```
@param[in] input_file The name of the input file.
@return options General options.
@return tgt_opts List of fitting target options.
```

**Todo** Implement internal coordinates.

Implement sampling correction.

Implement charge groups.

Definition at line 396 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**

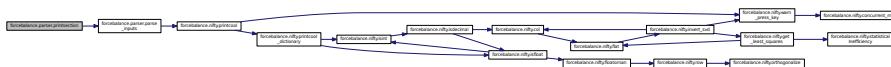
in	<i>heading</i>	Heading, either \$options or \$target
in	<i>optdict</i>	Options dictionary or None.
in	<i>typedict</i>	Option type dictionary, either gen_opts_types or tgt_opts_types specified in this file.

**Returns**

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

Definition at line 299 of file parser.py.

Here is the call graph for this function:



```
def forcebalance.parser.read_internals( fobj ) Definition at line 273 of file parser.py.
```

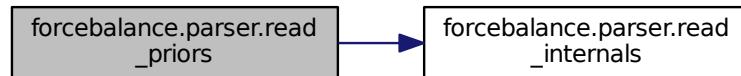
```
def forcebalance.parser.read_mvals( fobj ) Definition at line 248 of file parser.py.
```

Here is the call graph for this function:



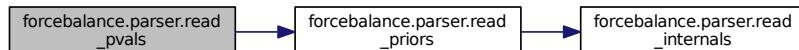
```
def forcebalance.parser.read_priors( fobj ) Definition at line 264 of file parser.py.
```

Here is the call graph for this function:



```
def forcebalance.parser.read_pvals( fobj ) Definition at line 256 of file parser.py.
```

Here is the call graph for this function:



### 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 213 of file parser.py.
```

```
dictionary forcebalance.parser.bkwd = {"simtype": "type", "masterfile": "binding_txt"} Option maps for maintaining backward compatibility.
```

Definition at line 243 of file parser.py.

```
dictionary forcebalance.parser.gen_opts_defaults = {} Default general options - basically a collapsed veresion of gen_opts_types.
```

Definition at line 227 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 216 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 246 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 277 of file parser.py.

**dictionary forcebalance.parser.subdict = {}** Definition at line 229 of file parser.py.

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

Definition at line 235 of file parser.py.

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

Definition at line 126 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,
     '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,
14    '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 cotypes = [None, 'BASS', 'BASSP'] Types of NDDO correction.*

- dictionary [pdict](#)

*Section -> Interaction type dictionary.*

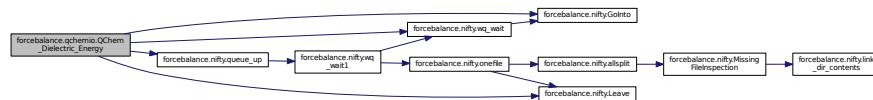
#### 7.30.1 Detailed Description

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 cotypes = [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.target Namespace Reference

### Classes

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

### Variables

- tuple [logger](#) = getLogger(`_name_`)

#### 7.31.1 Variable Documentation

**tuple [forcebalance.target.logger](#) = getLogger(`_name_`)** Definition at line 17 of file target.py.

## 7.32 forcebalance.tinkerio Namespace Reference

TINKER input/output.

### Classes

- class [Tinker\\_Reader](#)  
*Finite state machine for parsing TINKER force field files.*
- class [Liquid\\_TINKER](#)
- class [AbInitio\\_TINKER](#)  
*Subclass of Target for force and energy matching using TINKER.*
- class [Vibration\\_TINKER](#)  
*Subclass of Target for vibrational frequency matching using TINKER.*
- class [Moments\\_TINKER](#)  
*Subclass of Target for multipole moment matching using TINKER.*
- class [BindingEnergy\\_TINKER](#)  
*Subclass of BindingEnergy for binding energy matching using TINKER.*
- class [Interaction\\_TINKER](#)  
*Subclass of Target for interaction matching using TINKER.*

### Functions

- def [write\\_key\\_with\\_prm](#)  
*Copies a TINKER .key file but changes the parameter keyword as necessary to reflect the ForceBalance settings.*
- def [modify\\_key](#)  
*Performs in-place modification of a TINKER .key file.*

### Variables

- tuple [logger](#) = getLogger(`_name_`)
- dictionary [pdict](#)

### 7.32.1 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

### 7.32.2 Function Documentation

**def forcebalance.tinkerio.modify\_key( src, in\_dict )** Performs in-place modification of a TINKER .key file.

The input dictionary contains key:value pairs such as "polarization direct". If the key exists in the TINKER file, then that line is modified such that it contains the value in the dictionary. Note that this "key" is not to be confused with the .key extension in the TINKER file that we're modifying.

Sometimes keys like 'archive' do not have a value, in which case the dictionary should contain a None value or a blank space.

If the key doesn't exist in the TINKER file, then the key:value pair will be printed at the end.

Parameters

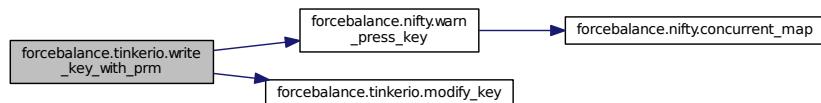
in	src	Name of the TINKER file to be modified.
in	in_dict	Dictionary containing key-value pairs used to modify the TINKER file.

Definition at line 200 of file tinkerio.py.

**def forcebalance.tinkerio.write\_key\_with\_prm( src, dest, prmfnm = None, ffobj = None )** Copies a TINKER .key file but changes the parameter keyword as necessary to reflect the ForceBalance settings.

Definition at line 144 of file tinkerio.py.

Here is the call graph for this function:



### 7.32.3 Variable Documentation

**tuple forcebalance.tinkerio.logger = getLogger(\_\_name\_\_)** Definition at line 30 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
3     ('Angstrom')
4     'ANGLE' : {'Atom':[1,2,3], 4:'K',5:'B'}, # Angle force constant and equilibrium angle
5     'UREYBRAD' : {'Atom':[1,2,3], 4:'K',5:'B'}, # Urey-Bradley force constant and equilibrium
6     distance (Angstrom)
7     'MCHARGE' : {'Atom':[1,2,3], 4:''}, # Atomic charge
8     'DIPOLE' : {0:'X',1:'Y',2:'Z'}, # Dipole moment in local frame
9     'QUADX' : {0:'X'}, # Quadrupole moment, X component
10    'QUADY' : {0:'X',1:'Y'}, # Quadrupole moment, Y component
11    'QUADZ' : {0:'X',1:'Y',2:'Z'}, # Quadrupole moment, Z component
```

```

10      'POLARIZE'      : {'Atom':[1], 2:'A',3:'T'},           # Atomic dipole polarizability
11      'BOND-CUBIC'    : {'Atom':[], 0:''},   # Below are global parameters.
12      'BOND-QUARTIC'  : {'Atom':[], 0:''},
13      'ANGLE-CUBIC'   : {'Atom':[], 0:''},
14      'ANGLE-QUARTIC' : {'Atom':[], 0:''},
15      'ANGLE-PENTIC'  : {'Atom':[], 0:''},
16      'ANGLE-SEXTIC'  : {'Atom':[], 0:''},
17      'DIELECTRIC'    : {'Atom':[], 0:''},
18      'POLAR-SOR'      : {'Atom':[], 0:''}
19          bending,                                # Ignored for now: stretch/bend coupling, out-of-plane
20          }                                     # torsional parameters, pi-torsion, torsion-torsion
21

```

Definition at line 32 of file tinkerio.py.

## 7.33 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).*

### Variables

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

#### 7.33.1 Detailed Description

Vibrational mode fitting module.

Author

Lee-Ping Wang

Date

08/2012

#### 7.33.2 Variable Documentation

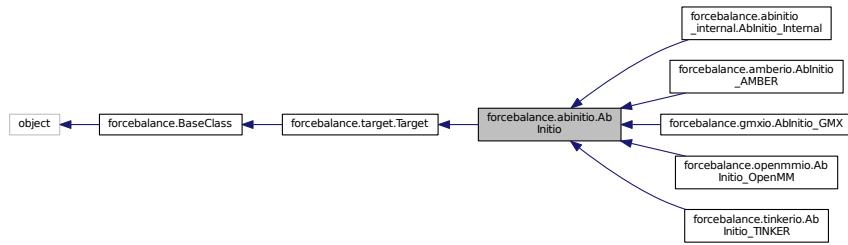
**tuple [forcebalance.vibration.logger](#) = getLogger([\\_\\_name\\_\\_](#))** Definition at line 22 of file vibration.py.

## 8 Class Documentation

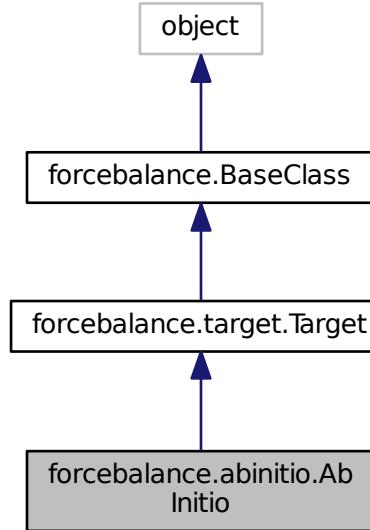
### 8.1 forcebalance.abinitio.ABInitio Class Reference

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 `read_topology`
- 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 `prepare_temp_directory`  
*Prepare the temporary directory, by default does nothing.*

- def `indicate`
- def `energy_force.transformer.all`
- def `energy_force.transformer`
- 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 `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 `sget`  
*Stages the directory for the target, and then calls 'get'.*
- 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 `set_option`

## Public Attributes

- `whamboltz_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.`
- `traj`
- `nparticles`  
`The number of (atoms + drude particles + virtual sites)`
- `AtomLists`  
`This is a default-dict containing a number of atom-wise lists, such as the residue number of each atom, the mass of each atom, and so on.`
- `new_vsites`  
`Read in the topology.`
- `save_vmvabs`  
`Save the mvabs from the last time we updated the vsites.`
- `topology_flag`
- `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`
- `tempdir`

- **rundir**  
*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.*
- **PrintOptionDict**
- **verbose\_options**

### 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.

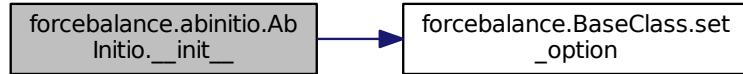
### 8.1.2 Constructor & Destructor Documentation

```
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.

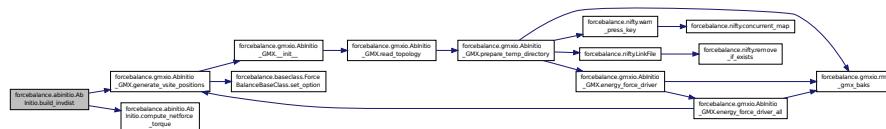
Here is the call graph for this function:



### 8.1.3 Member Function Documentation

**def forcebalance.abinitio.AbInitio.build\_invdist ( self, mvals )** Definition at line 168 of file abinitio.py.

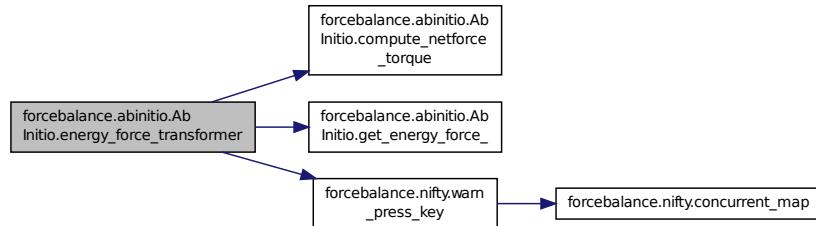
Here is the call graph for this function:



**def forcebalance.abinitio.AbInitio.compute\_netforce\_torque ( self, xyz, force, QM = False )** Definition at line 204 of file abinitio.py.

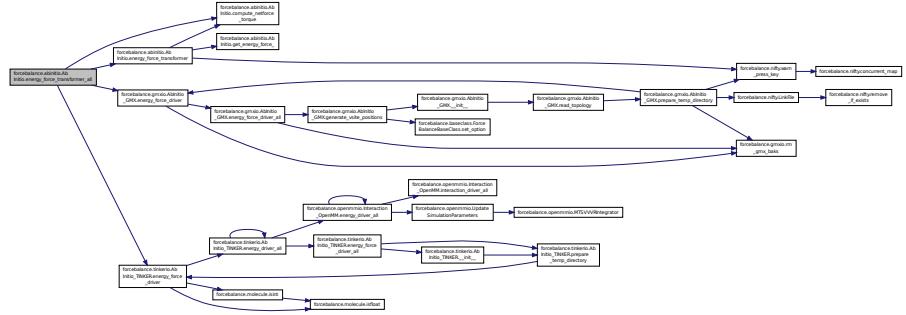
**def forcebalance.abinitio.AbInitio.energy\_force\_transformer ( self, i )** Definition at line 459 of file abinitio.py.

Here is the call graph for this function:



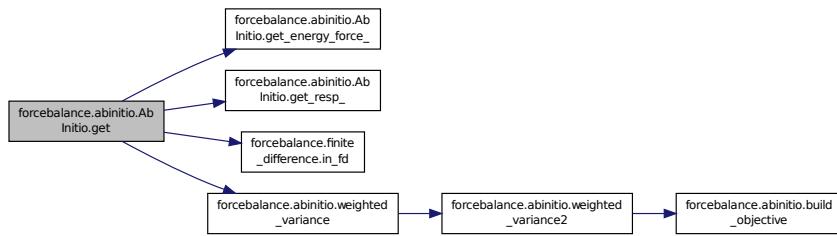
**def forcebalance.abinitio.AbInitio.energy\_force\_transformer\_all ( self )** Definition at line 442 of file abinitio.py.

Here is the call graph for this function:



**def forcebalance.abinitio.AbInitio.get ( self, mvals, AGrad = False, AHess = False )** Definition at line 1100 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:

- 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
```

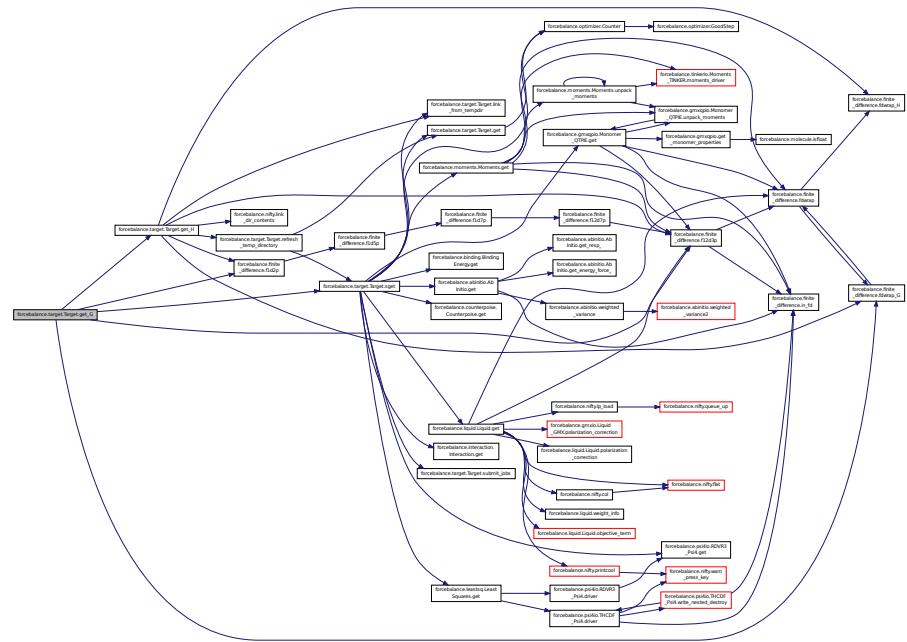
Definition at line 534 of file abinitio.py.

**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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



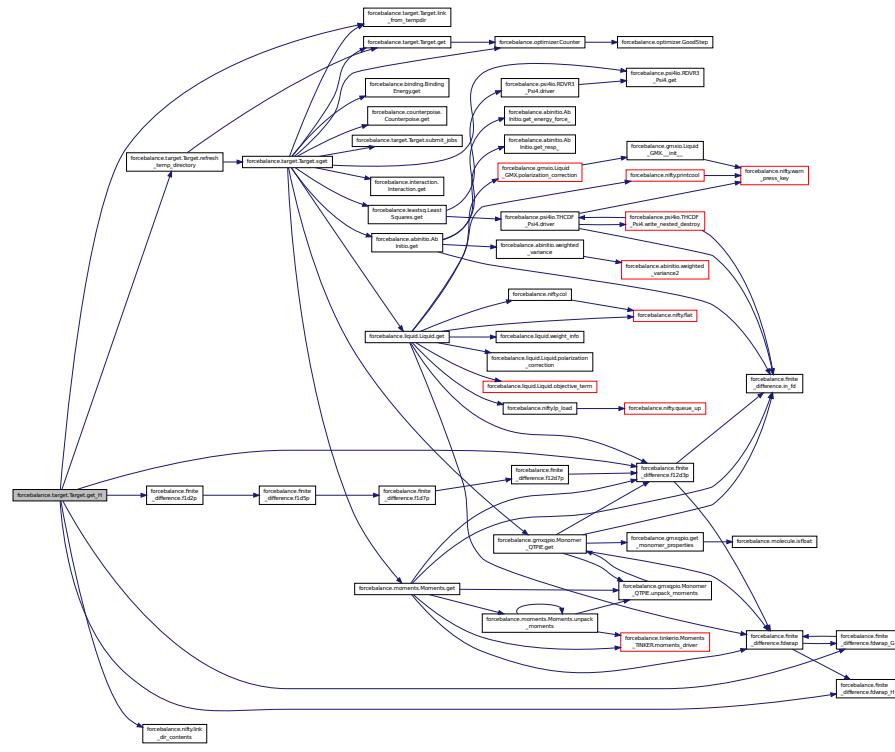
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 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.

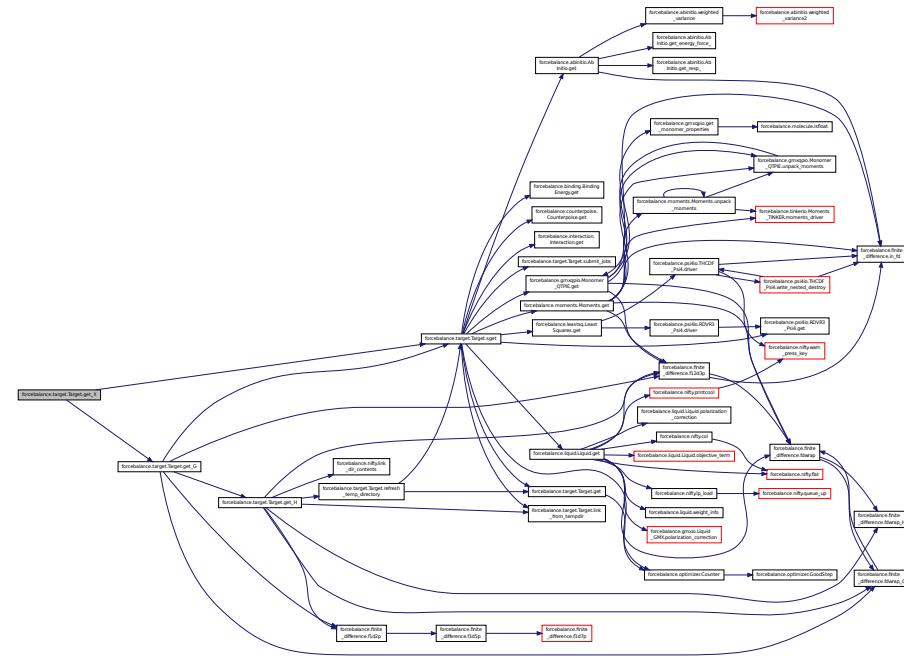
Implements the RESP objective function. (In Python so obviously not optimized.) This function takes the mathematical parameter values and returns the charges on the ATOMS (fancy mapping going on)

Definition at line 1001 of file abinitio.py.

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

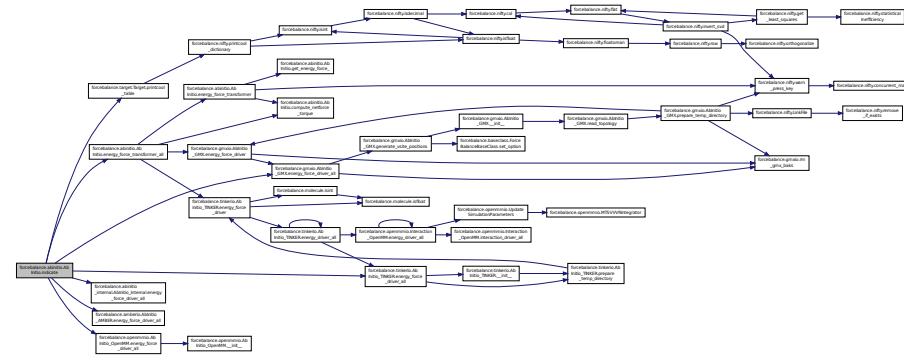
Definition at line 155 of file target.py.

Here is the call graph for this function:



**def** forcebalance.abinitio.ABInitio.indicate ( self ) Definition at line 422 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 213 of file target.py.

**def forcebalance.abinitio.AblInitio.prepare\_temp\_directory ( self, options, tgt\_opts )** Prepare the temporary directory, by default does nothing.

Definition at line 419 of file abinitio.py.

```
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.

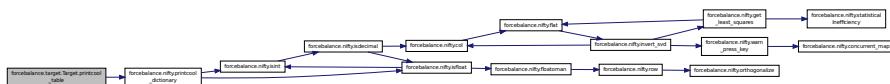
The input should be something like:

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 367 of file target.py.

Here is the call graph for this function:



```
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:

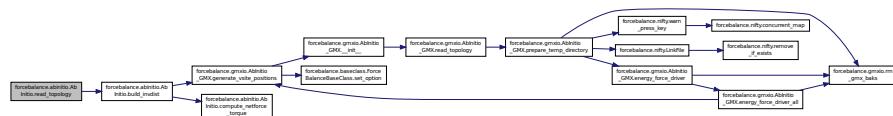
```
S = -1*Sum_i(P_i*log(P_i))
InfoContent = exp(-S)
```

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

Definition at line 317 of file abinitio.py.

**def forcebalance.abinitio.AbInitio.read\_topology ( self )** Definition at line 164 of file abinitio.py.

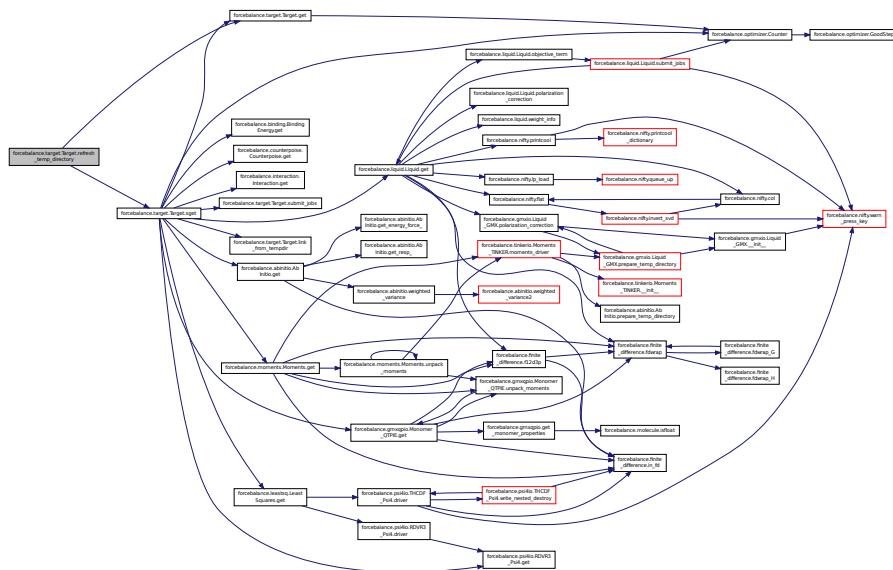
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



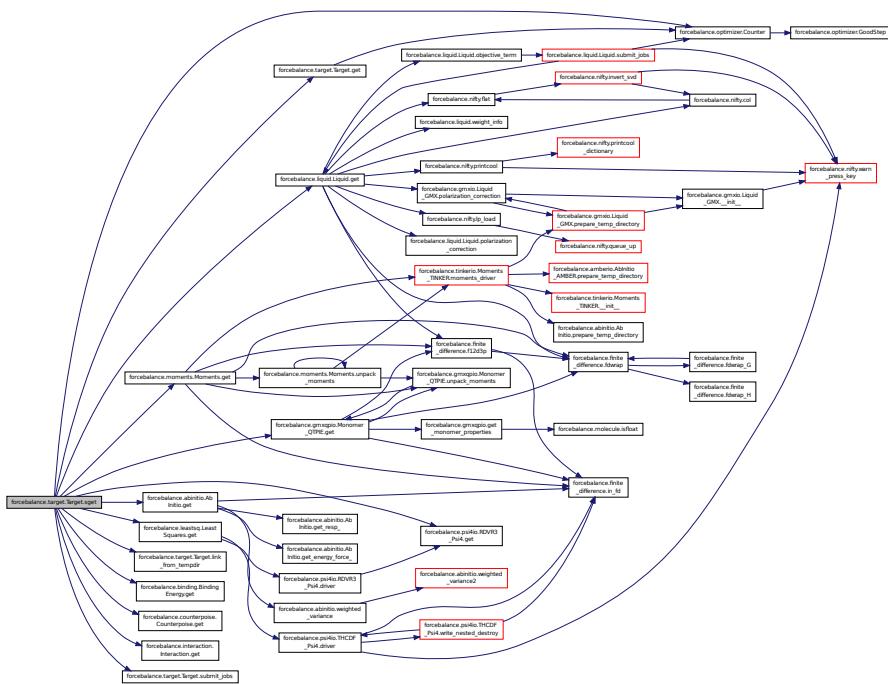
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

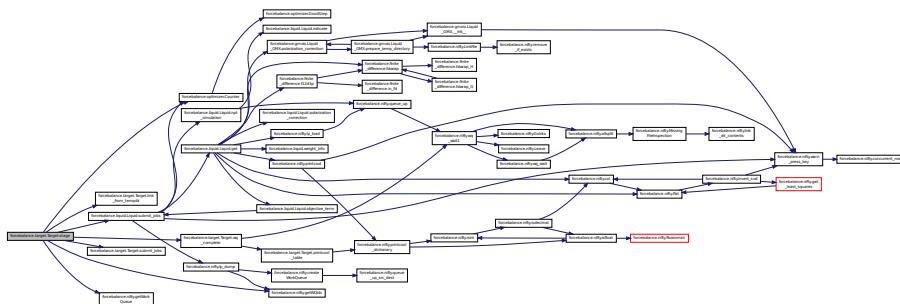


**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

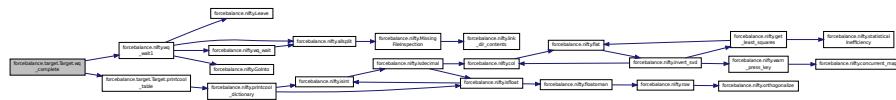


```
def forcebalance.target.Target.submit.jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### **8.1.4 Member Data Documentation**

**forcebalance.abinitio.AblInitio.AtomLists** This is a default-dict containing a number of atom-wise lists, such as the residue number of each atom, the mass of each atom, and so on.

Definition at line 150 of file abinitio.py.

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

Definition at line 128 of file abinitio.py.

**forcebalance.abinitio.ABInitio.e\_err\_pct** Definition at line 129 of file abinitio.py.

**forcebalance.abinitio.AblInitio.e\_ref** Definition at line 978 of file abinitio.py.

**forcebalance.abinitio.AbInitio.emd0** Energies of the sampling simulation.

Definition at line 116 of file abinitio.py.

**forcebalance.abinitio.ABInitio.eqm** Reference (QM) energies.

Definition at line 114 of file abinitio.py.

**forcebalance.abinitio.Ablnitiob.esp\_err** Qualitative Indicator: "relative RMS" for electrostatic potential.

Definition at line 134 of file abinitio.py.

## **forcebalance.abinitio.AbInitio.espval** ESP values.

Definition at line 122 of file abinitio.py.

**forcebalance.abinitio.AblInitio.espxyz** ESP grid points.

Definition at line 120 of file abinitio.py.

**forcebalance.abinitio.AblInitio.f\_err** Qualitative Indicator: average force error (fractional)

Definition at line 131 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err\_pct** Definition at line 132 of file abinitio.py.

**forcebalance.abinitio.AblInitio.f\_ref** Definition at line 982 of file abinitio.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.abinitio.AbInitio.fitatoms** Definition at line 354 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force** Definition at line 349 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force\_map** Definition at line 212 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fqm** Reference (QM) forces.

Definition at line 118 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fref** Definition at line 413 of file abinitio.py.

**forcebalance.target.Target.gct** [**inherited**] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [**inherited**] Counts how often the Hessian was computed.

Definition at line 145 of file target.py.

**forcebalance.abinitio.AbInitio.invdists** Definition at line 1009 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nesp** Definition at line 351 of file abinitio.py.

**forcebalance.abinitio.AbInitio.new\_vsites** Read in the topology.

Read in the reference data Prepare the temporary directory 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 159 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_err** Definition at line 135 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_err\_pct** Definition at line 136 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_ref** Definition at line 986 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nftqm** Definition at line 409 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nnf** Definition at line 255 of file abinitio.py.

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

Definition at line 147 of file abinitio.py.

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

Definition at line 141 of file abinitio.py.

**forcebalance.abinitio.AbInitio.ntq** Definition at line 256 of file abinitio.py.

**forcebalance.abinitio.AbInitio.objective** Definition at line 1120 of file abinitio.py.

**forcebalance.BaseClass.PrintOptionDict** [**inherited**] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.abinitio.AbInitio.qfnm** The qdata.txt file that contains the QM energies and forces.  
Definition at line 124 of file abinitio.py.

**forcebalance.abinitio.AbInitio.qmatoms** The number of atoms in the QM calculation (Irrelevant if not fitting forces)  
Definition at line 126 of file abinitio.py.

**forcebalance.abinitio.AbInitio.qmboltz\_wts** QM Boltzmann weights.  
Definition at line 112 of file abinitio.py.

**forcebalance.abinitio.AbInitio.respterm** Definition at line 1088 of file abinitio.py.

**forcebalance.target.Target.rundir [inherited]** 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.  
Definition at line 137 of file target.py.

**forcebalance.abinitio.AbInitio.save.vmvals** Save the mvals from the last time we updated the vsites.  
Definition at line 161 of file abinitio.py.

**forcebalance.target.Target.tempdir [inherited]** Root directory of the whole project.  
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 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 135 of file target.py.

**forcebalance.abinitio.AbInitio.topology\_flag** Definition at line 166 of file abinitio.py.

**forcebalance.abinitio.AbInitio.tq\_err** Definition at line 990 of file abinitio.py.

**forcebalance.abinitio.AbInitio.tq\_err\_pct** Definition at line 137 of file abinitio.py.

**forcebalance.abinitio.AbInitio.tq\_ref** Definition at line 989 of file abinitio.py.

**forcebalance.abinitio.AbInitio.traj** Definition at line 142 of file abinitio.py.

**forcebalance.abinitio.AbInitio.use\_nft** Whether to compute net forces and torques, or not.  
Definition at line 139 of file abinitio.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.abinitio.AbInitio.w\_energy** Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.AbInitio.w\_force** Definition at line 350 of file abinitio.py.

**forcebalance.abinitio.AbInitio.w\_netforce** Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.AbInitio.w\_resp** Definition at line 1002 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_torque** Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz** Definition at line 370 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz.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) WHAM Boltzmann weights

Definition at line 110 of file abinitio.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 141 of file target.py.

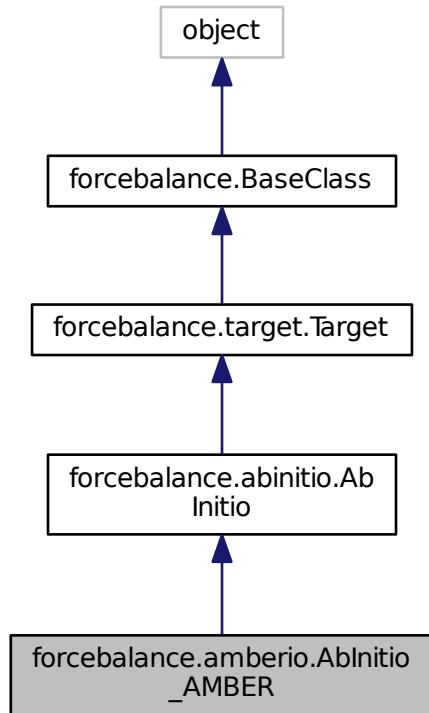
The documentation for this class was generated from the following file:

- [abinitio.py](#)

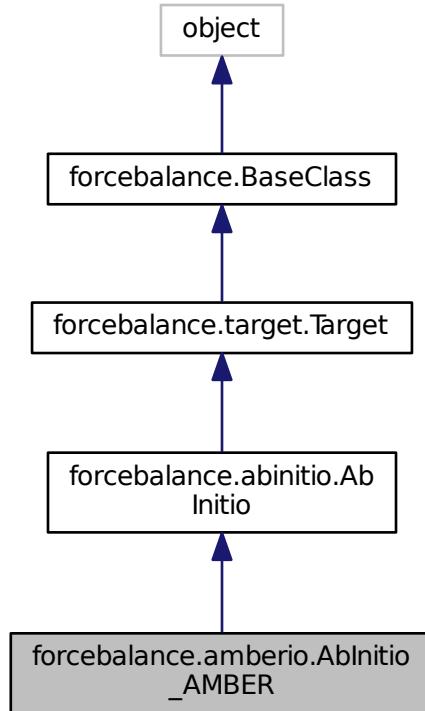
## 8.2 forcebalance.amberio.ABInitio\_AMBER Class Reference

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 `read_topology`
- 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_force_transformer_all`
- def `energy_force_transformer`
- 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 `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 `sget`  
*Stages the directory for the target, and then calls 'get'.*
- 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 `set_option`

## Public Attributes

- `trajfnm`  
*Name of the trajectory, we need this BEFORE initializing the SuperClass.*
- `all_at_once`  
*all\_at\_once is not implemented.*
- `whamboltz_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.*

- `traj`
- `nparticles`

*The number of (atoms + drude particles + virtual sites)*

- `AtomLists`

*This is a default-dict containing a number of atom-wise lists, such as the residue number of each atom, the mass of each atom, and so on.*

- `new_vsites`

*Read in the topology.*

- `save_vmvalls`

*Save the mvalls from the last time we updated the vsites.*

- `topology_flag`
- `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`
- `tempdir`

*Root directory of the whole project.*

- `rundir`

*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.*
- **PrintOptionDict**
- **verbose\_options**

### 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.

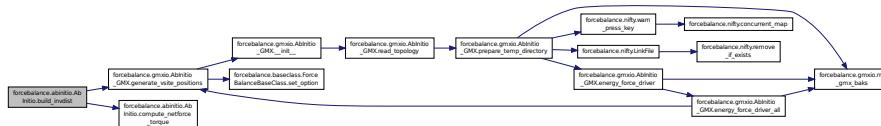
### 8.2.2 Constructor & Destructor Documentation

**def forcebalance.amberio.AblInitio\_AMBER.\_\_init\_\_ ( self, options, tgt\_opts, forcefield )** Definition at line 174 of file amberio.py.

### 8.2.3 Member Function Documentation

**def forcebalance.abinitio.AblInitio.build\_invdist ( self, mvals ) [inherited]** Definition at line 168 of file abinitio.py.

Here is the call graph for this function:

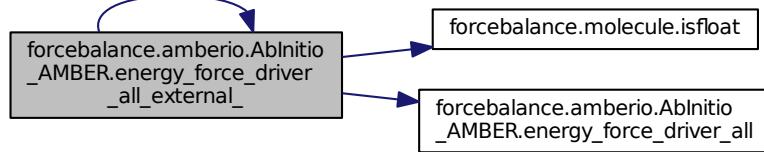


**def forcebalance.abinitio.AblInitio.compute\_netforce\_torque ( self, xyz, force, QM = False ) [inherited]** Definition at line 204 of file abinitio.py.

**def forcebalance.amberio.AblInitio\_AMBER.energy\_force\_driver\_all ( self )** Definition at line 228 of file amberio.py.

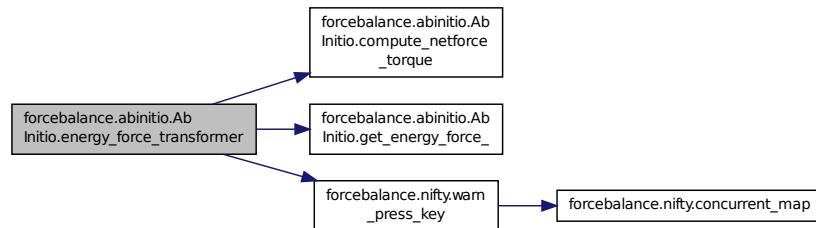
**def forcebalance.amberio.AblInitio\_AMBER.energy\_force\_driver\_all\_external\_ ( self )** Definition at line 190 of file amberio.py.

Here is the call graph for this function:



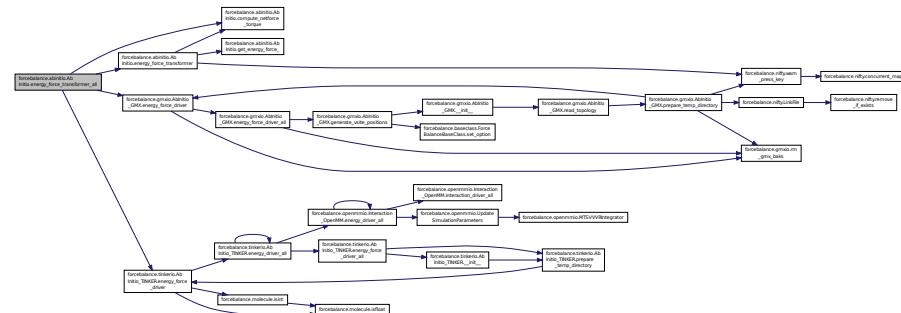
**def forcebalance.abinitio.AblInitio.energy\_force\_transformer ( self, i ) [inherited]** Definition at line 459 of file abinitio.py.

Here is the call graph for this function:



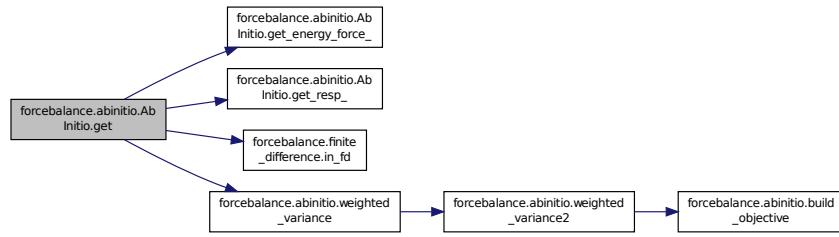
**def forcebalance.abinitio.AblInitio.energy\_force\_transformer\_all ( self ) [inherited]** Definition at line 442 of file abinitio.py.

Here is the call graph for this function:



**def forcebalance.abinitio.AblInitio.get ( self, mvals, AGrad = False, AHess = False ) [inherited]** Definition at line 1100 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
```

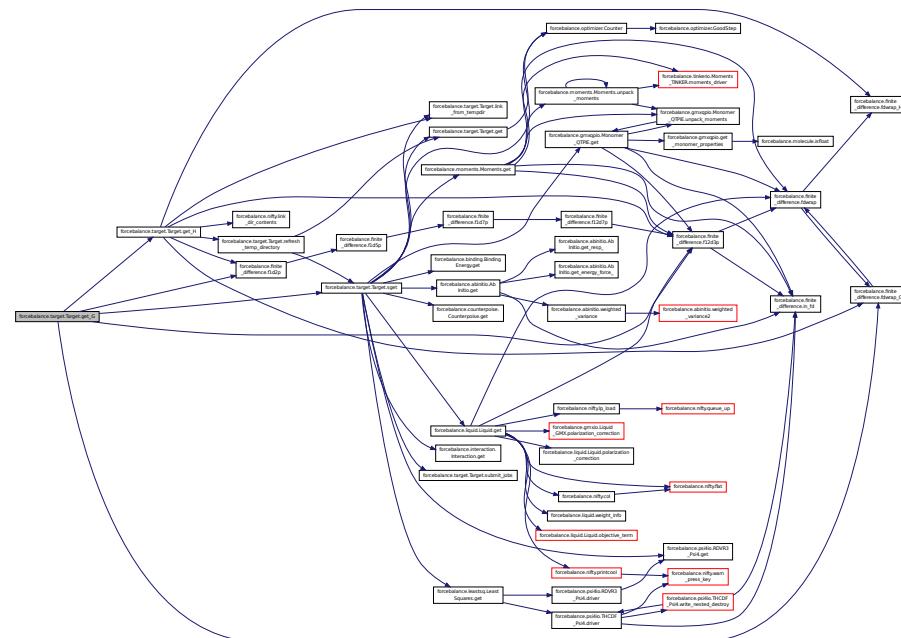
Definition at line 534 of file abinitio.py.

**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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



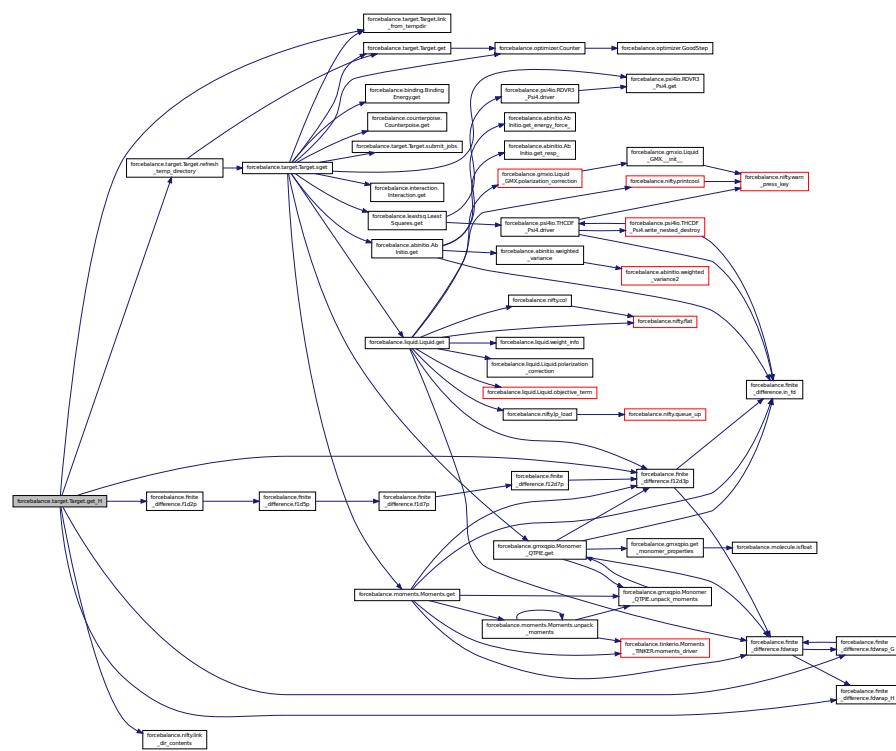
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 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.
```

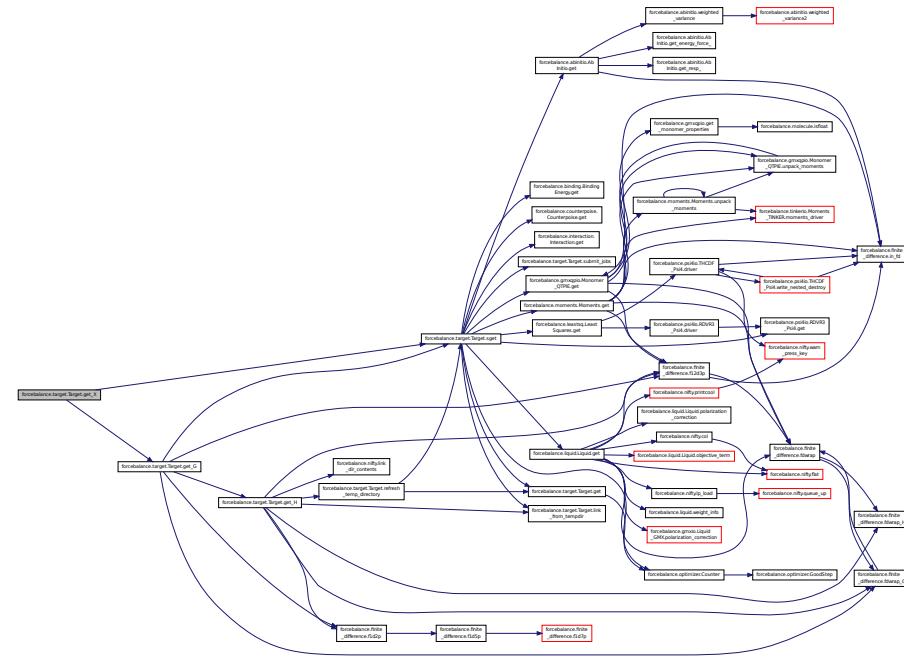
Implements the RESP objective function. (In Python so obviously not optimized.) This function takes the mathematical parameter values and returns the charges on the ATOMS (fancy mapping going on)

Definition at line 1001 of file abinitio.py.

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

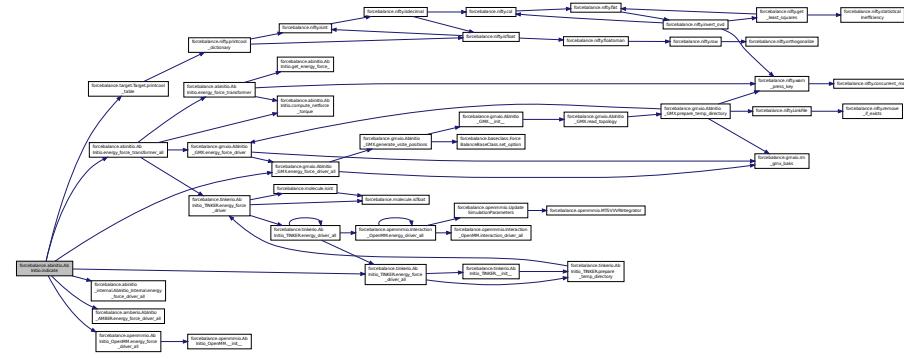
Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.abinitio.AblInitio.indicate ( self ) [inherited]** Definition at line 422 of file abinitio.py.

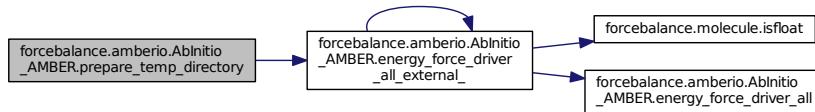
Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

**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.

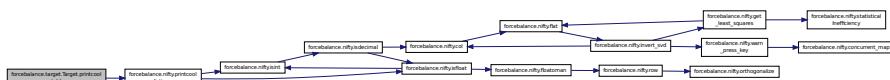
The input should be something like:

## 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 367 of file target.py.

Here is the call graph for this function:



```
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.

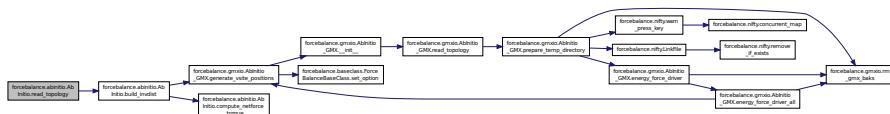
Here, we have a measure for the information content of our snapshots, which comes easily from the definition of information entropy:

```
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 317 of file abinitio.py.

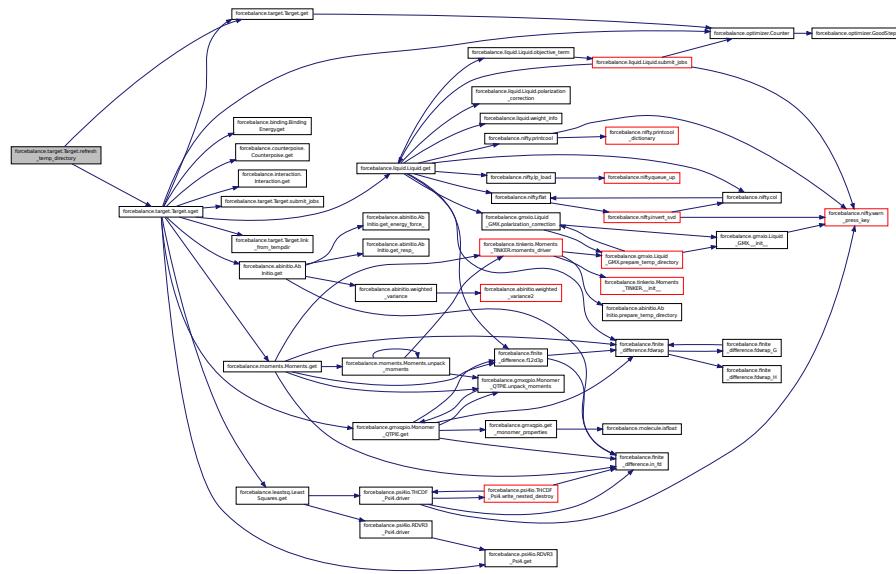
**def forcebalance.abinitio.Ablinitio.read\_topology( self ) [inherited]** Definition at line 164 of file abinitio.py.  
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



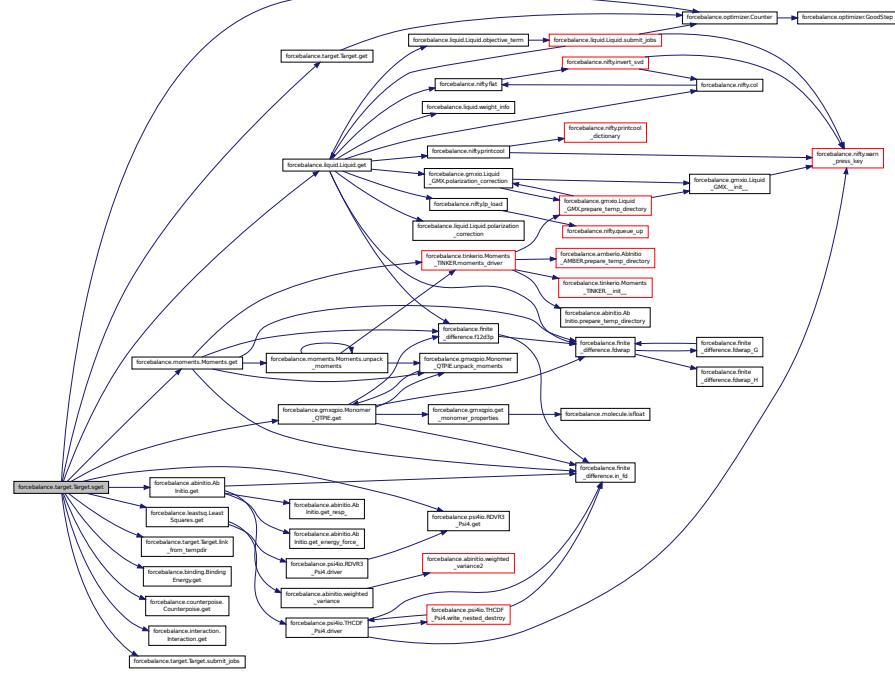
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

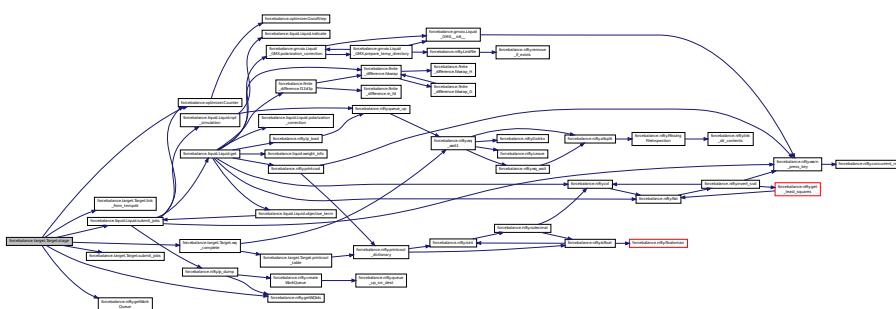


```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

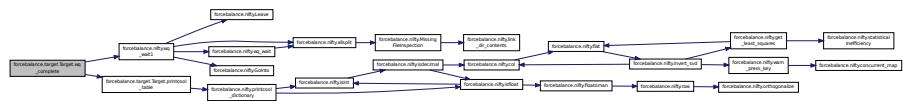


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### **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.AblInitio.AtomLists** [inherited] This is a default-dict containing a number of atom-wise lists, such as the residue number of each atom, the mass of each atom, and so on.

Definition at line 150 of file abinitio.py.

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

Definition at line 128 of file abinitio.py.

**forcebalance.abinitio.ABInitio.e\_err\_pct** [inherited] Definition at line 129 of file abinitio.py.

**forcebalance.abinitio.ABInitio.e\_ref** [inherited] Definition at line 978 of file abinitio.py.

**forcebalance.abinitio.ABInitio.emd0** [inherited] Energies of the sampling simulation.

Definition at line 116 of file abinitio.py.

Definition at line 114 of file abinitio.py.

Definition at line 114 of file `asmlibio.py`.

**forcebalance.ab initio.ABinitio.espm** [Inherited] Qualitative indicator: relative RMS for electrostatic potential.

Definition at line 134 of file abinfllo.py.

**forcebalance.abinitio.AblInitio.espval** [inherited] ESP values.

Definition at line 122 of file abinitio.py.

**forcebalance.abinitio.AblInitio.espxyz** [inherited] ESP grid points.

Definition at line 120 of file abinitio.py.

**forcebalance.abinitio.ABInitio.f\_err** [inherited] Qualitative Indicator: average force error (fractional)

Definition at line 131 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err\_pct** [inherited] Definition at line 132 of file abinitio.py.

**forcebalance.abinitio.AblInitio.f\_ref** [inherited] Definition at line 982 of file abinitio.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.abinitio.AbInitio.fitatoms** [inherited] Definition at line 354 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force** [inherited] Definition at line 349 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force\_map** [inherited] Definition at line 212 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fqm** [inherited] Reference (QM) forces.  
Definition at line 118 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fref** [inherited] Definition at line 413 of file abinitio.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.  
Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.  
Definition at line 145 of file target.py.

**forcebalance.abinitio.AbInitio.invdists** [inherited] Definition at line 1009 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nesp** [inherited] Definition at line 351 of file abinitio.py.

**forcebalance.abinitio.AbInitio.new\_vsites** [inherited] Read in the topology.

Read in the reference data Prepare the temporary directory 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 159 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_err** [inherited] Definition at line 135 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_err\_pct** [inherited] Definition at line 136 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_ref** [inherited] Definition at line 986 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nftqm** [inherited] Definition at line 409 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nnf** [inherited] Definition at line 255 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nparticles** [inherited] The number of (atoms + drude particles + virtual sites)  
Definition at line 147 of file abinitio.py.

**forcebalance.abinitio.AbInitio.ns** [inherited] Read in the trajectory file.  
Definition at line 141 of file abinitio.py.

**forcebalance.abinitio.AbInitio.ntq** [inherited] Definition at line 256 of file abinitio.py.

**forcebalance.abinitio.AbInitio.objective** [inherited] Definition at line 1120 of file abinitio.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.qfnm** [inherited] The qdata.txt file that contains the QM energies and forces.  
Definition at line 124 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 126 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmboltz\_wts** [inherited] QM Boltzmann weights.  
Definition at line 112 of file abinitio.py.

**forcebalance.abinitio.ABInitio.respterm** [inherited] Definition at line 1088 of file abinitio.py.

**forcebalance.target.Target.rundir** [inherited] 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.

Definition at line 137 of file target.py.

**forcebalance.abinitio.ABInitio.save\_vvals** [inherited] Save the mvals from the last time we updated the vsites.

Definition at line 161 of file abinitio.py.

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

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 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 135 of file target.py.

**forcebalance.abinitio.ABInitio.topology\_flag** [inherited] Definition at line 166 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_err** [inherited] Definition at line 990 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_err\_pct** [inherited] Definition at line 137 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_ref** [inherited] Definition at line 989 of file abinitio.py.

**forcebalance.abinitio.ABInitio.traj** [inherited] Definition at line 142 of file abinitio.py.

**forcebalance.amberio.ABInitio\_AMBER.trajfnm** Name of the trajectory, we need this BEFORE initializing the SuperClass.

Definition at line 176 of file amberio.py.

**forcebalance.abinitio.ABInitio.use\_nft** [inherited] Whether to compute net forces and torques, or not.  
Definition at line 139 of file abinitio.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.w\_energy** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.AbInitio.w\_force** [inherited] Definition at line 350 of file abinitio.py.

**forcebalance.abinitio.AbInitio.w\_netforce** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.AbInitio.w\_resp** [inherited] Definition at line 1002 of file abinitio.py.

**forcebalance.abinitio.AbInitio.w\_torque** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.AbInitio.whamboltz** [inherited] Definition at line 370 of file abinitio.py.

**forcebalance.abinitio.AbInitio.whamboltz\_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) WHAM Boltzmann weights

Definition at line 110 of file abinitio.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 141 of file target.py.

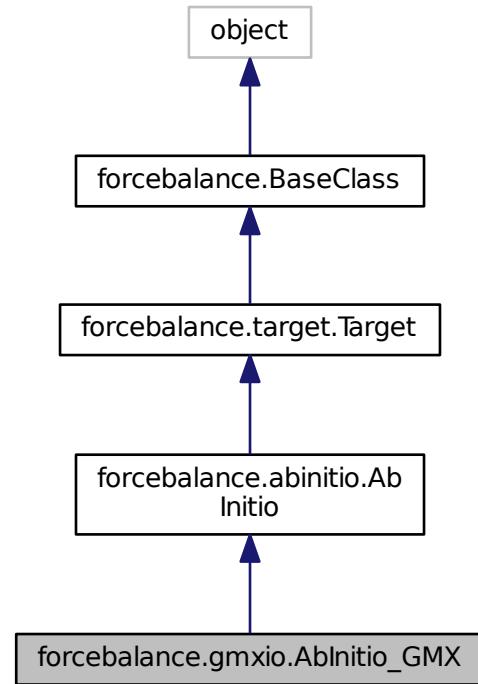
The documentation for this class was generated from the following file:

- [amberio.py](#)

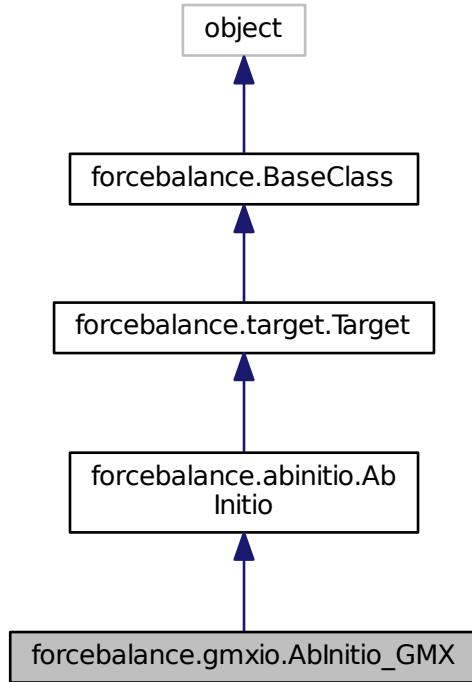
### 8.3 forcebalance.gmxio.AbInitio\_GMX Class Reference

Subclass of AbInitio for force and energy matching using normal GROMACS.

Inheritance diagram for forcebalance.gmxio.AbInitio\_GMX:



Collaboration diagram for forcebalance.gmxio.AbInitio\_GMX:



### Public Member Functions

- def `_init_`
- def `prepare_temp_directory`
- def `read_topology`
- def `energy_force_driver`

*Computes the energy and force using GROMACS for a single snapshot.*

- def `energy_force_driver_all`

*Computes the energy and force using GROMACS for a trajectory.*

- def `generate_vsites_positions`

*Call mdrun in order to update the virtual site positions.*

- 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_force_transformer_all`
- def `energy_force_transformer`
- 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 `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 `sget`  
*Stages the directory for the target, and then calls 'get'.*
- 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 `set_option`

## Public Attributes

- `engine`  
*Default file names for coordinates, top and mdp files.*
- `AtomMask`
- `AtomLists`
- `topology_flag`
- `whamboltz_wts`  
*Initialize the base class.*
- `qm boltz_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.*
- **traj**
- **nparticles**  
*The number of (atoms + drude particles + virtual sites)*
- **new\_vsites**  
*Read in the topology.*
- **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**
- **tempdir**  
*Root directory of the whole project.*
- **rundir**  
*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.*
  - `PrintOptionDict`
  - `verbose_options`

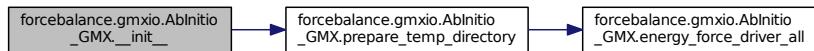
### **8.3.1 Detailed Description**

Subclass of `AblInitio` for force and energy matching using normal GROMACS.  
Implements the `prepare_temp_directory` and `energy_force_driver` methods.  
Definition at line 638 of file `gmxio.py`.

### 8.3.2 Constructor & Destructor Documentation

**def forcebalance.gmxio.ABInitio\_GMX.\_init\_ ( self, options, tgt\_opts, forcefield )** Definition at line 640 of file gmxio.py.

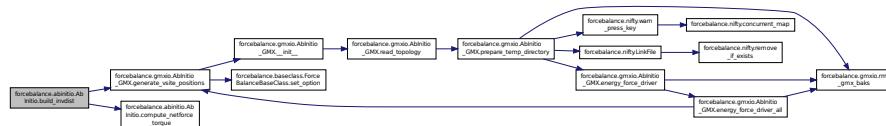
Here is the call graph for this function:



### 8.3.3 Member Function Documentation

**def forcebalance.abinitio.AblInitio.build\_invdist ( self, mvals ) [inherited]** Definition at line 168 of file abinitio.py.

Here is the call graph for this function:

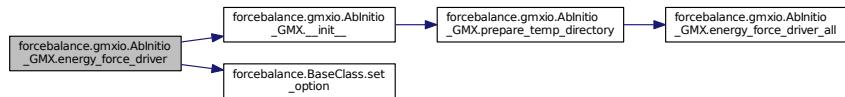


```
def forcebalance.abinitio.AbInitio.compute_netforce_torque ( self, xyz, force, QM = False )  
[inherited] Definition at line 204 of file abinitio.py.
```

**def forcebalance.gmxio.ABInitio\_GMX.energy\_force\_driver ( self, shot )** Computes the energy and force using GROMACS for a single snapshot.

This does not require GROMACS-X2.  
Definition at line 668 of file gmxio.py.

Here is the call graph for this function:



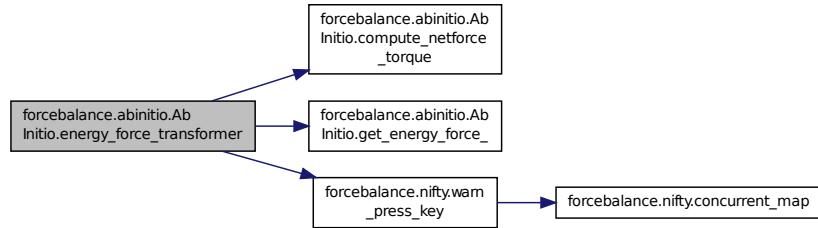
**def forcebalance.gmxio.AbInitio.\_GMX.energy\_force\_driver\_all ( self )** Computes the energy and force using GROMACS for a trajectory.

This does not require GROMACS-X2.

Definition at line 673 of file gmxio.py.

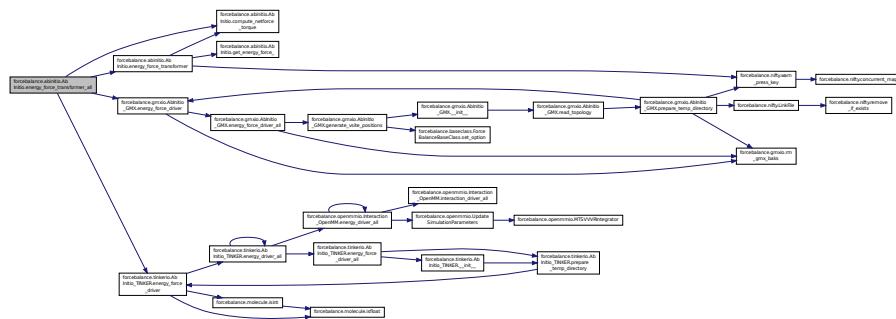
**def forcebalance.abinitio.AbInitio.energy\_force\_transformer ( self, i ) [inherited]** Definition at line 459 of file abinitio.py.

Here is the call graph for this function:



**def forcebalance.abinitio.AbInitio.energy\_force\_transformer\_all ( self ) [inherited]** Definition at line 442 of file abinitio.py.

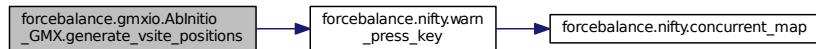
Here is the call graph for this function:



**def forcebalance.gmxio.AbInitio\_GMX.generate\_vsites\_positions ( self )** Call mdrun in order to update the virtual site positions.

Definition at line 678 of file gmxio.py.

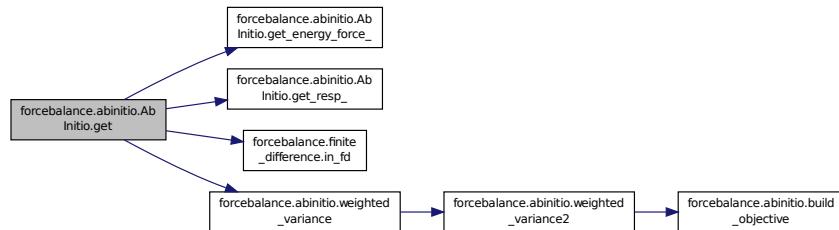
Here is the call graph for this function:



**def forcebalance.abinitio.AbInitio.get ( self, mvals, AGrad = False, AHess = False ) [inherited]**

Definition at line 1100 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
```

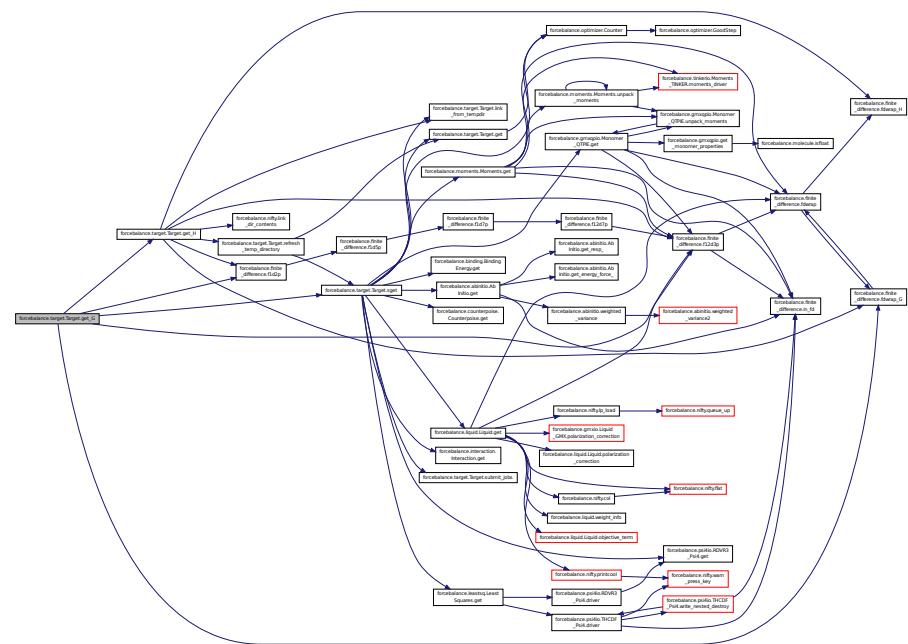
Definition at line 534 of file abinitio.py.

**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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



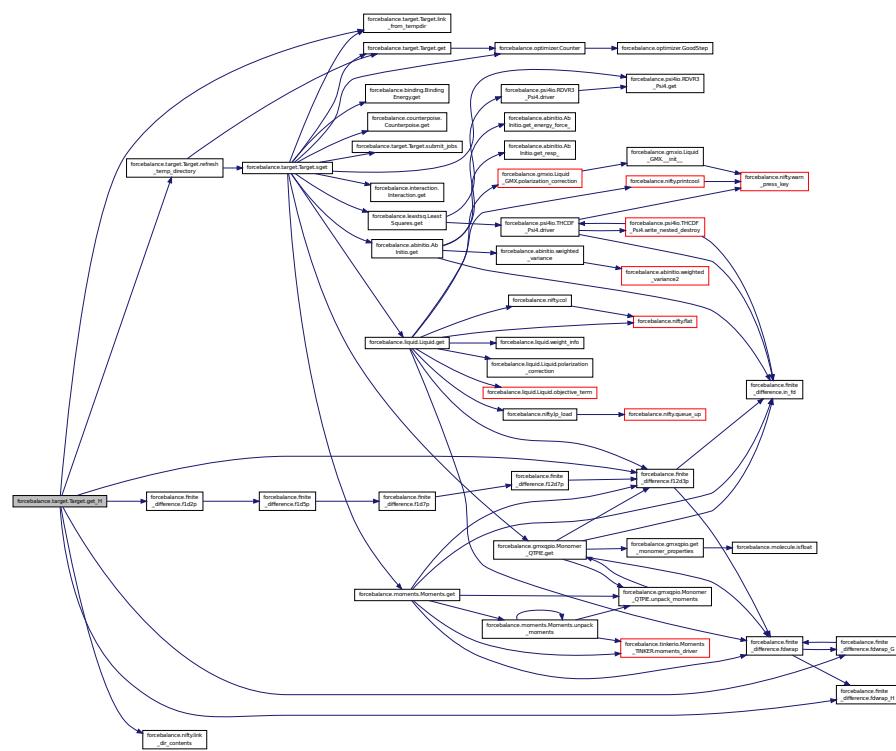
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 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.
```

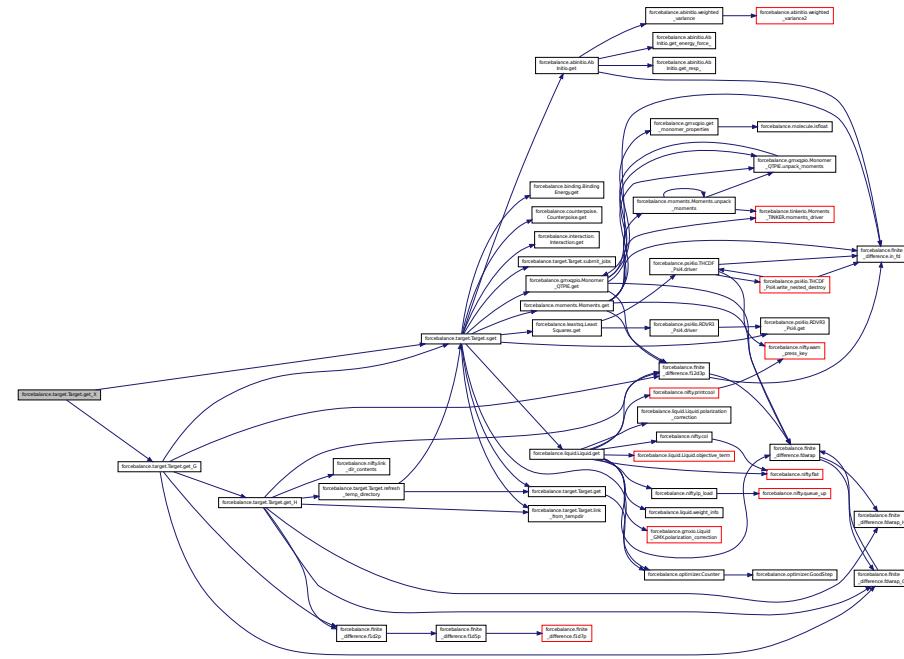
Implements the RESP objective function. (In Python so obviously not optimized.) This function takes the mathematical parameter values and returns the charges on the ATOMS (fancy mapping going on)

Definition at line 1001 of file abinitio.py.

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

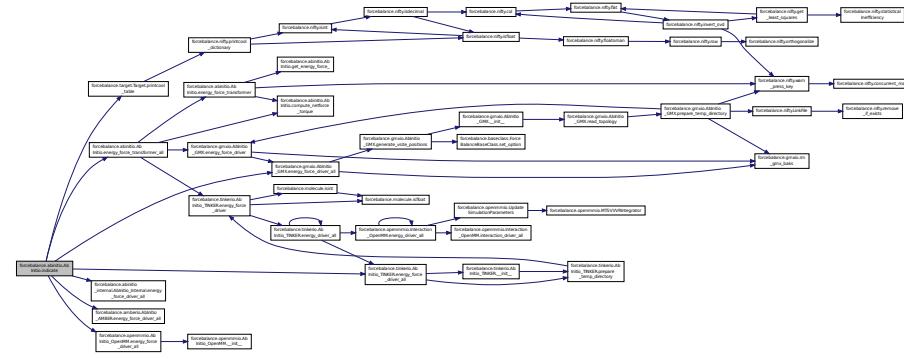
Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.abinitio.AblInitio.indicate ( self ) [inherited]** Definition at line 422 of file abinitio.py.

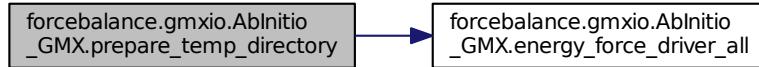
Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

**def forcebalance.gmxio.AbInitio\_GMX.prepare\_temp\_directory ( self, options, tgt\_opts )** Definition at line 659 of file gmxio.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.

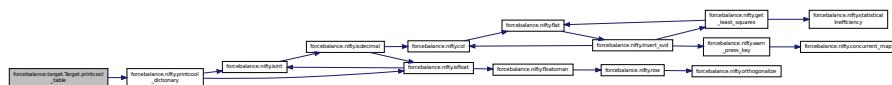
The input should be something like:

#### Parameters

<i>data</i>	Column contents in the form of an <i>OrderedDict</i> , 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 <i>OrderedDict</i> , 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 367 of file target.py.

Here is the call graph for this function:



**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.

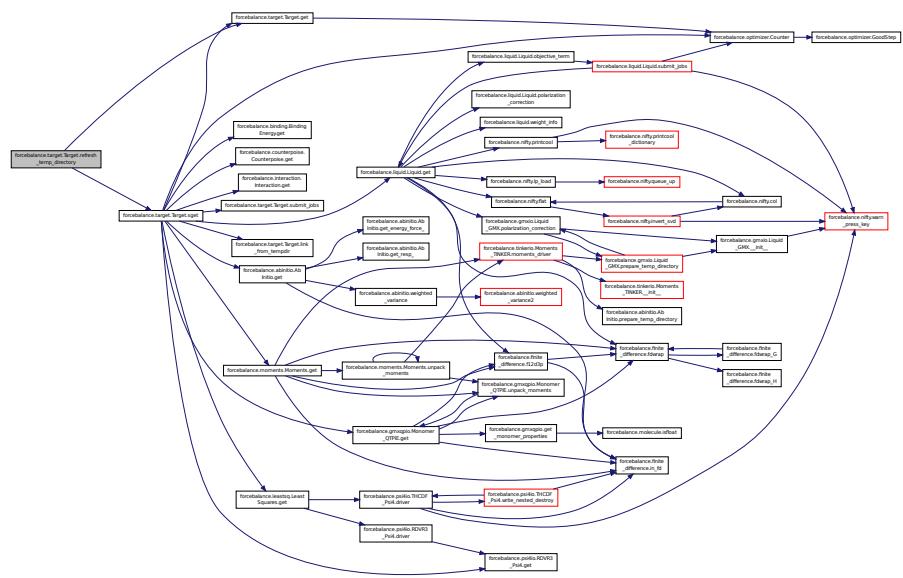
Definition at line 317 of file abinitio.py.

**def forcebalance.gmxio.AbInitio\_GMX.read\_topology( self )** Definition at line 662 of file gmxio.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 219 of file target.py.

Here is the call graph for this function:



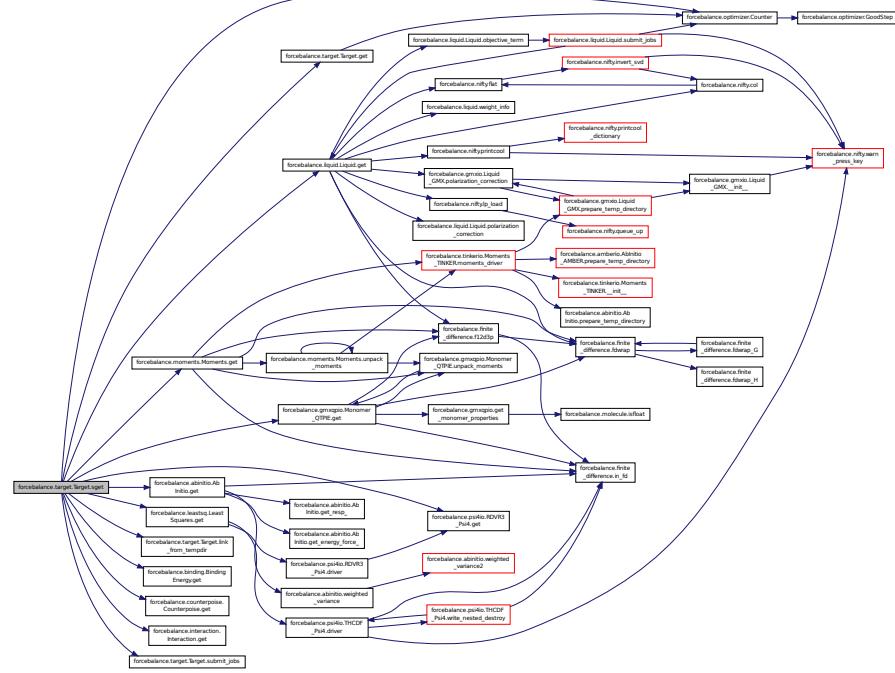
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

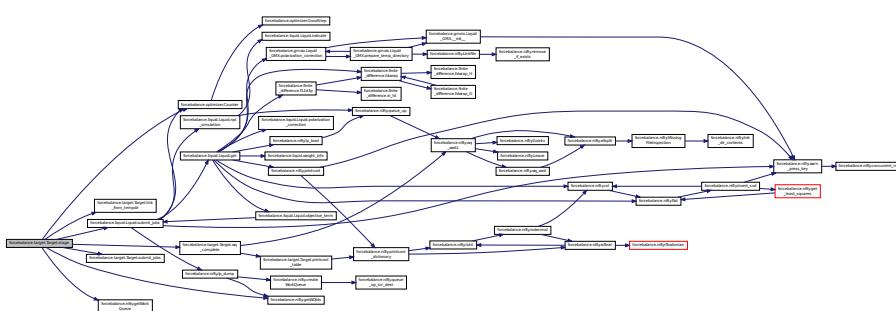


**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

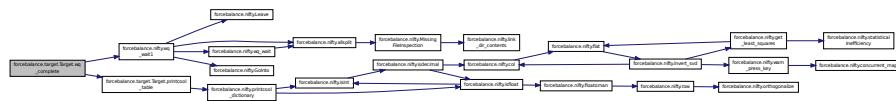


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### **8.3.4 Member Data Documentation**

**forcebalance.gmxio.ABInitio\_GMX.AtomLists** Definition at line 657 of file gmxio.py.

**forcebalance.gmxio.ABInitio\_GMX.AtomMask** Definition at line 656 of file gmxio.py.

**forcebalance.abinitio.AblInitio.e\_err** [inherited] Qualitative Indicator: average energy error (in kJ/mol)  
Definition at line 128 of file abinitio.py.

**forcebalance.abinitio.ABInitio.e\_err\_pct** [inherited] Definition at line 129 of file abinitio.py.

**forcebalance.abinitio.AbInitio.e\_ref** [inherited] Definition at line 978 of file abinitio.py.

**forcebalance.abinitio.AbInitio.emd0** [inherited] Energies of the sampling simulation.  
Definition at line 116 of file abinitio.py.

**forcebalance.gmxio.ABInitio\_GMX.engine** Default file names for coordinates, top and mdp files.  
Initialize base class. Build keyword dictionaries to pass to engine. Create engine object.  
Definition at line 654 of file gmxio.py.

**forcebalance.abinitio.AblInitio.eqm** [inherited] Reference (QM) energies.  
Definition at line 114 of file abinitio.py.

**forcebalance.abinitio.ABInitio.esp\_err** [inherited] Qualitative Indicator: "relative RMS" for electrostatic potential.

**forcebalance.abinitio.AbInitio.espval** [inherited] ESP values.  
Definition at line 122 of file abinitio.py.

**forcebalance.abinitio.AblInitio.espxyz** [inherited] ESP grid points.  
Definition at line 120 of file abinitio.py.

**forcebalance.abinitio.ABInitio.f\_err** [inherited] Qualitative Indicator  
Definition at line 121 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err\_pct** [inherited] Definition at line 132 of file abinitio.py.

**forcebalance.abinitio.AblInitio.f\_ref** [inherited] Definition at line 982 of file abinitio.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 189 of file target.py.

**forcebalance.abinitio.AbInitio.fitatoms** [inherited] Definition at line 354 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force** [inherited] Definition at line 349 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force\_map** [inherited] Definition at line 212 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fqm** [inherited] Reference (QM) forces.  
Definition at line 118 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fref** [inherited] Definition at line 413 of file abinitio.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.  
Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.  
Definition at line 145 of file target.py.

**forcebalance.abinitio.AbInitio.invdists** [inherited] Definition at line 1009 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nesp** [inherited] Definition at line 351 of file abinitio.py.

**forcebalance.abinitio.AbInitio.new\_vsites** [inherited] Read in the topology.

Read in the reference data Prepare the temporary directory 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 159 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_err** [inherited] Definition at line 135 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_err\_pct** [inherited] Definition at line 136 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_ref** [inherited] Definition at line 986 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nftqm** [inherited] Definition at line 409 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nnf** [inherited] Definition at line 255 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nparticles** [inherited] The number of (atoms + drude particles + virtual sites)  
Definition at line 147 of file abinitio.py.

**forcebalance.abinitio.AbInitio.ns** [inherited] Read in the trajectory file.  
Definition at line 141 of file abinitio.py.

**forcebalance.abinitio.AbInitio.ntq** [inherited] Definition at line 256 of file abinitio.py.

**forcebalance.abinitio.AbInitio.objective** [inherited] Definition at line 1120 of file abinitio.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.qfnm** [inherited] The qdata.txt file that contains the QM energies and forces.  
Definition at line 124 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 126 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmboltz\_wts** [inherited] QM Boltzmann weights.  
Definition at line 112 of file abinitio.py.

**forcebalance.abinitio.ABInitio.respterm** [inherited] Definition at line 1088 of file abinitio.py.

**forcebalance.target.Target.rundir** [inherited] 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.

Definition at line 137 of file target.py.

**forcebalance.abinitio.ABInitio.save\_vvals** [inherited] Save the mvals from the last time we updated the vsites.

Definition at line 161 of file abinitio.py.

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

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 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 135 of file target.py.

**forcebalance.gmxio.ABInitio\_GMX.topology\_flag** Definition at line 663 of file gmxio.py.

**forcebalance.abinitio.ABInitio.tq\_err** [inherited] Definition at line 990 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_err\_pct** [inherited] Definition at line 137 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_ref** [inherited] Definition at line 989 of file abinitio.py.

**forcebalance.abinitio.ABInitio.traj** [inherited] Definition at line 142 of file abinitio.py.

**forcebalance.abinitio.ABInitio.use\_nft** [inherited] Whether to compute net forces and torques, or not.  
Definition at line 139 of file abinitio.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.w\_energy** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_force** [inherited] Definition at line 350 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_netforce** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_resp** [inherited] Definition at line 1002 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_torque** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz** [inherited] Definition at line 370 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz.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) WHAM Boltzmann weights

Definition at line 110 of file abinitio.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 141 of file target.py.

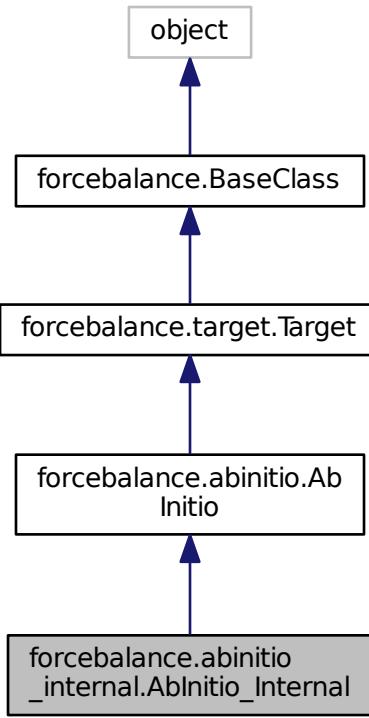
The documentation for this class was generated from the following file:

- [gmxio.py](#)

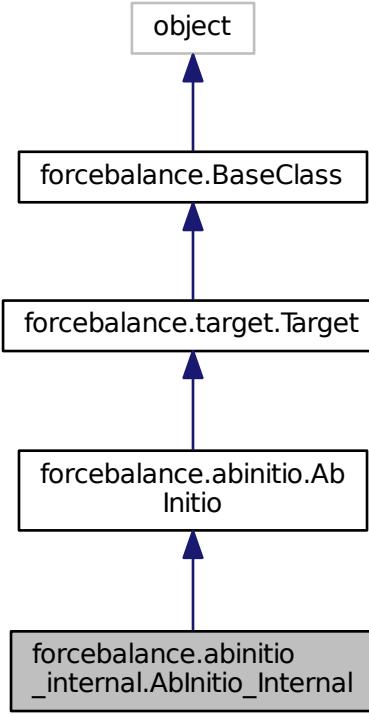
## 8.4 forcebalance.abinitio\_internal.ABInitio\_Internal Class Reference

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 `read_topology`
- 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 `prepare_temp_directory`

*Prepare the temporary directory, by default does nothing.*

- def `indicate`
- def `energy_force_transformer_all`
- def `energy_force_transformer`
- 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.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 `sget`  
 Stages the directory for the target, and then calls 'get'.
- 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 `set_option`

## Public Attributes

- `trajfnm`  
 Name of the trajectory, we need this BEFORE initializing the SuperClass.
- `whamboltz_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.*
- **traj**
- **nparticles**  
*The number of (atoms + drude particles + virtual sites)*
- **AtomLists**  
*This is a default-dict containing a number of atom-wise lists, such as the residue number of each atom, the mass of each atom, and so on.*
- **new\_vsites**  
*Read in the topology.*
- **save\_vmvabs**  
*Save the mvabs from the last time we updated the vsites.*
- **topology\_flag**
- **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**
- **tempdir**  
*Root directory of the whole project.*
- **rundir**  
*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.
- **PrintOptionDict**
- **verbose\_options**

#### 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_ab initio.AblInitio_Internal.__init__ ( self, options, tgt_opts, forcefield )
```

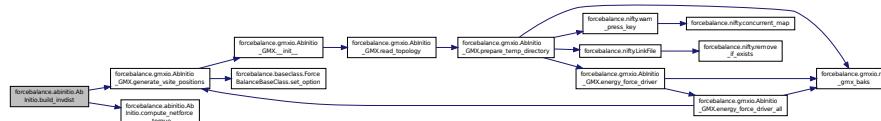
Definition at line 40 of file abinitio\_internal.py.

#### 8.4.3 Member Function Documentation

```
def forcebalance.abinitio.AblInitio.build_invdist ( self, mvals ) [inherited]
```

Definition at line 168 of file abinitio.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.AblInitio.compute_netforce_torque ( self, xyz, force, QM = False ) [inherited]
```

Definition at line 204 of file abinitio.py.

```
def forcebalance.abinitio_ab initio.AblInitio_Internal.energy_force_driver_all ( self )
```

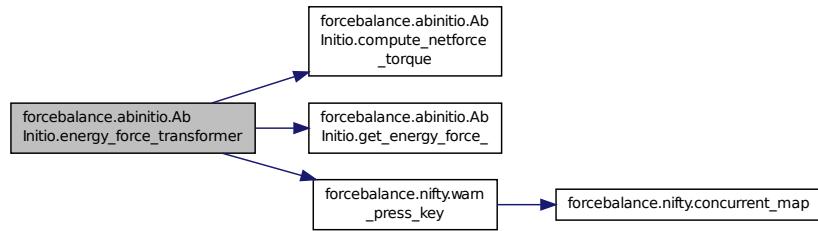
Here we actually compute the interactions and return the energies and forces.

I verified this to give the same answer as GROMACS.

Definition at line 50 of file abinitio\_internal.py.

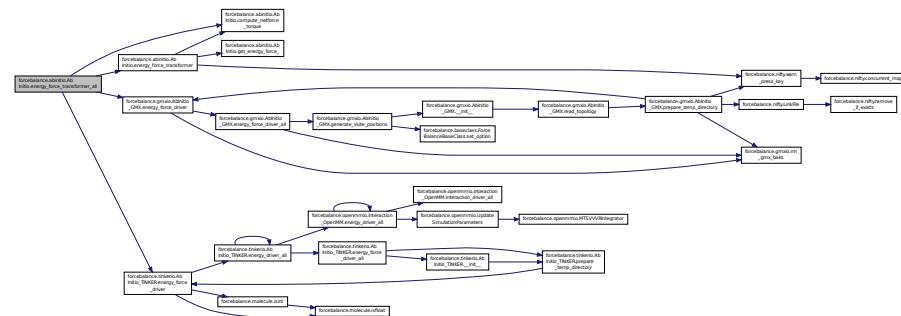
**def forcebalance.abinitio.ABInitio.energy\_force\_transformer( self, i ) [inherited]** Definition at line 459 of file abinitio.py.

Here is the call graph for this function:



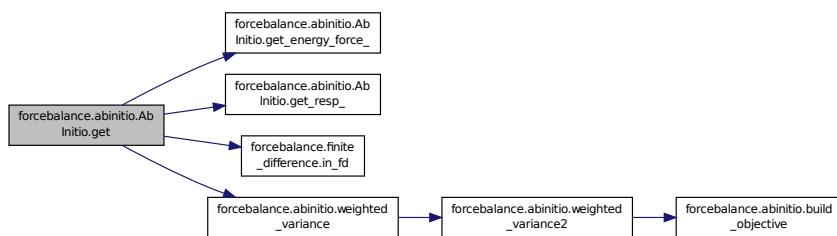
**def forcebalance.abinitio.ABInitio.energy\_force\_transformer\_all ( self ) [inherited]** Definition at line 442 of file abinitio.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.ABInitio.get ( self, mvals, AGrad = False, AHess = False ) [inherited]
Definition at line 1100 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
```

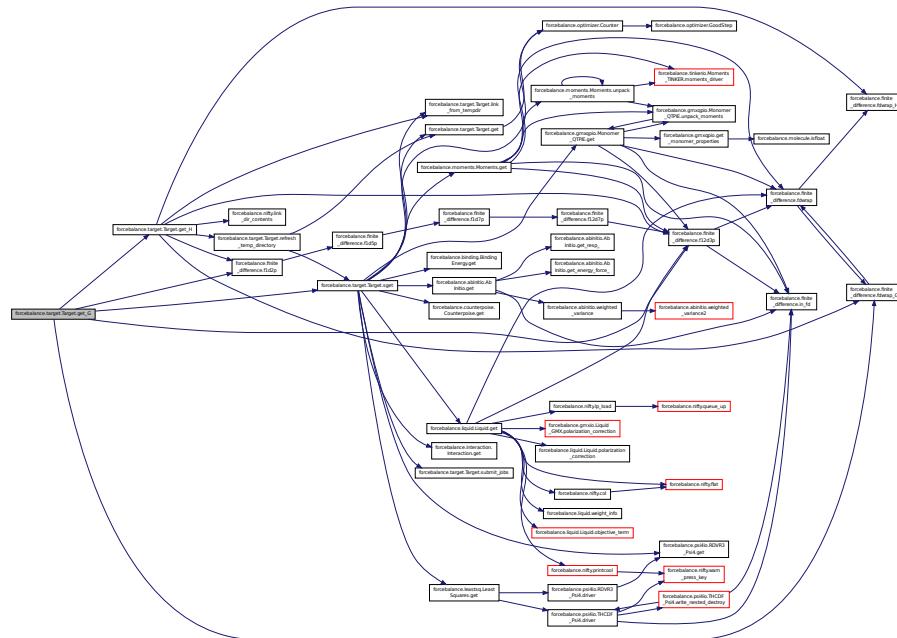
Definition at line 534 of file abinitio.py.

```
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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



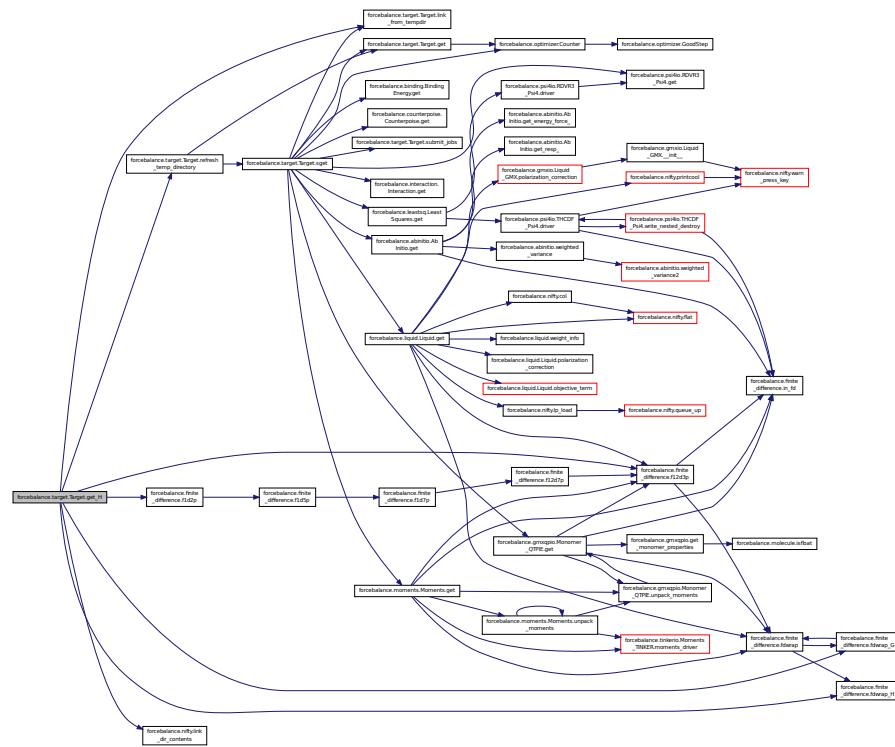
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 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.
```

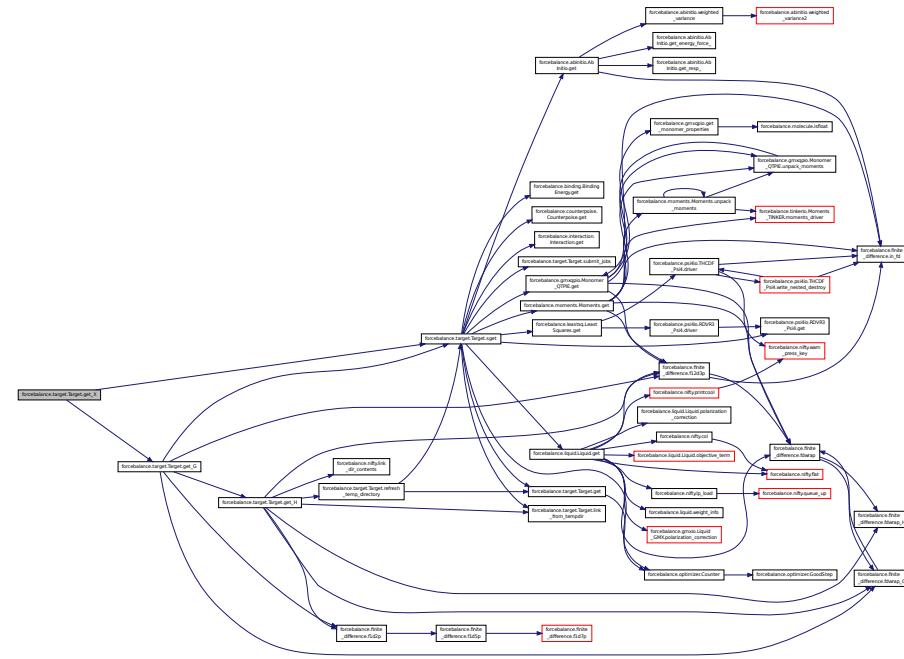
Implements the RESP objective function. (In Python so obviously not optimized.) This function takes the mathematical parameter values and returns the charges on the ATOMS (fancy mapping going on)

Definition at line 1001 of file abinitio.py.

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

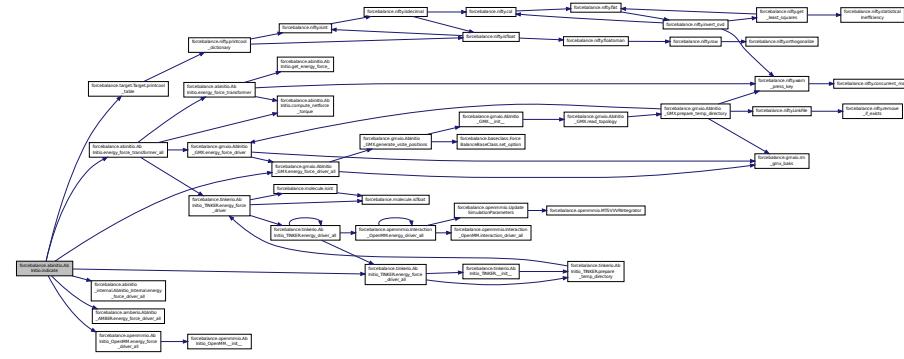
Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.abinitio.AblInitio.indicate ( self ) [inherited]** Definition at line 422 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 213 of file target.py.

```
def forcebalance.abinitio.AblInitio.prepare_temp_directory ( self, options, tgt_opts ) [inherited] Prepare the temporary directory, by default does nothing.
```

Definition at line 419 of file abinitio.py.

```
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.

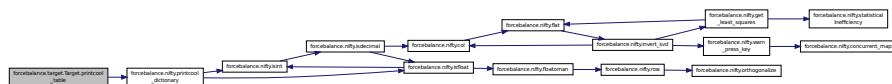
The input should be something like:

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 367 of file target.py.

Here is the call graph for this function:



```
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 317 of file abinitio.py.

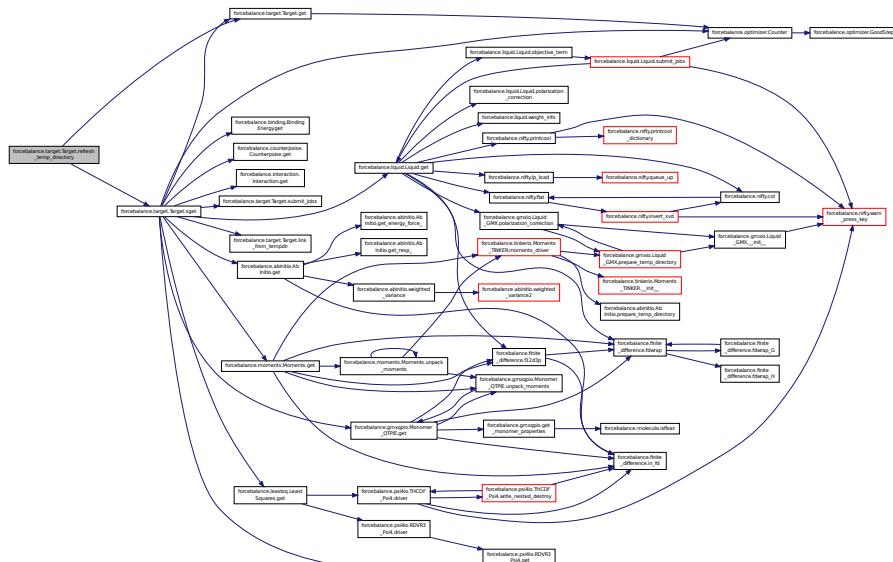
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



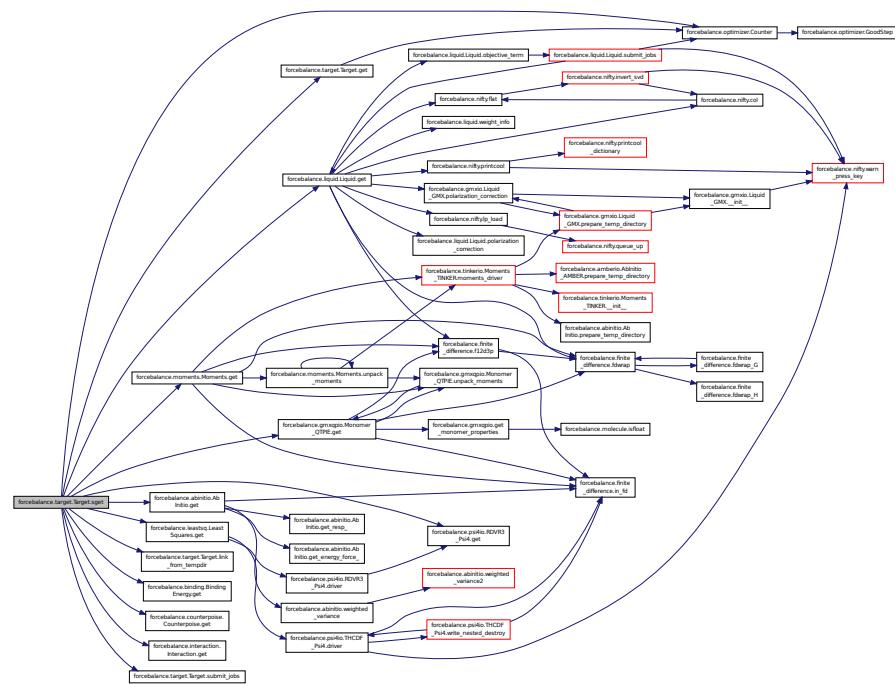
```
def forcebalance.BaseClass.set_option ( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

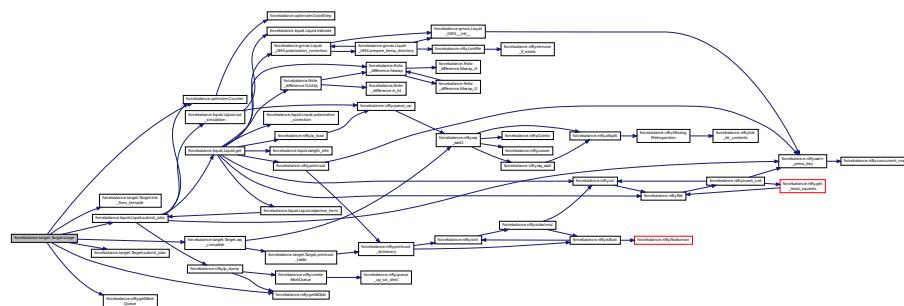


```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

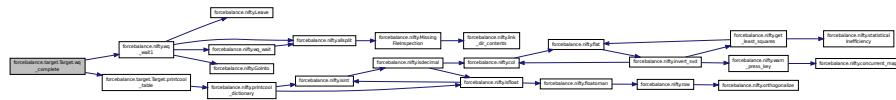


```
def forcebalance.target.Target.submit.jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### **8.4.4 Member Data Documentation**

**forcebalance.abinitio.AblInitio.AtomLists** [inherited] This is a default-dict containing a number of atom-wise lists, such as the residue number of each atom, the mass of each atom, and so on.

Definition at line 150 of file abinitio.py.

**forcebalance.abinitio.AblInitio.e\_err** [inherited] Qualitative Indicator: average energy error (in kJ/mol)

Definition at line 128 of file abinitio.py.

**forcebalance.abinitio.AblInitio.e\_err\_pct** [inherited] Definition at line 129 of file abinitio.py.

**forcebalance.abinitio.ABInitio.e\_ref** [inherited] Definition at line 978 of file abinitio.py.

**forcebalance.abinitio.AbInitio.emd0** [inherited] Energies of the sampling simulation.

Definition at line 116 of file abinitio.py.

**forcebalance.abinitio.AblInitio.eqm** [inherited] Reference (QM) energies.

Definition at line 114 of file abinitio.py.

**forcebalance.abinitio.AblInitio.esp\_err** [inherited] Qualitative Indicator: "relative RMS" for electrostatic potential.

Definition at line 134 of file abinitio.py.

**forcebalance.abinitio.AblInitio.espval** [inherited] ESP values.

Definition at line 122 of file abinitio.py.

**forcebalance.abinitio.AblInitio.espxyz** [inherited] ESP grid points.

Definition at line 120 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err** [inherited] Qualitative Indicator: average force error (fractional)  
Definition at line 131 of file abinitio.py.

**forcebalance.abinitio.AblInitio.f\_err\_pct** [inherited] Definition at line 132 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_ref** [inherited] Definition at line 982 of file abinitio.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)  
Definition at line 139 of file target.py.

**forcebalance.abinitio.ABInitio.fitatoms [inherited]** Definition at line 354 of file abinitio.py.

**forcebalance.abinitio.ABInitio.force [inherited]** Definition at line 349 of file abinitio.py.

**forcebalance.abinitio.ABInitio.force\_map [inherited]** Definition at line 212 of file abinitio.py.

**forcebalance.abinitio.ABInitio.fqm [inherited]** Reference (QM) forces.  
Definition at line 118 of file abinitio.py.

**forcebalance.abinitio.ABInitio.fref [inherited]** Definition at line 413 of file abinitio.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.  
Definition at line 143 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.  
Definition at line 145 of file target.py.

**forcebalance.abinitio.ABInitio.invdists [inherited]** Definition at line 1009 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nesp [inherited]** Definition at line 351 of file abinitio.py.

**forcebalance.abinitio.ABInitio.new\_vsites [inherited]** Read in the topology.  
Read in the reference data Prepare the temporary directory 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 159 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_err [inherited]** Definition at line 135 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_err\_pct [inherited]** Definition at line 136 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nf\_ref [inherited]** Definition at line 986 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nftqm [inherited]** Definition at line 409 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nnf [inherited]** Definition at line 255 of file abinitio.py.

**forcebalance.abinitio.ABInitio.nparticles [inherited]** The number of (atoms + drude particles + virtual sites)  
Definition at line 147 of file abinitio.py.

**forcebalance.abinitio.ABInitio.ns [inherited]** Read in the trajectory file.  
Definition at line 141 of file abinitio.py.

**forcebalance.abinitio.ABInitio.ntq [inherited]** Definition at line 256 of file abinitio.py.

**forcebalance.abinitio.ABInitio.objective [inherited]** Definition at line 1120 of file abinitio.py.

**forcebalance.BaseClass.PrintOptionDict** [**inherited**] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.abinitio.AblInitio.qfnm** [**inherited**] The qdata.txt file that contains the QM energies and forces.  
Definition at line 124 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 126 of file abinitio.py.

**forcebalance.abinitio.AblInitio.qmboltz\_wts** [**inherited**] QM Boltzmann weights.  
Definition at line 112 of file abinitio.py.

**forcebalance.abinitio.AblInitio.respterm** [**inherited**] Definition at line 1088 of file abinitio.py.

**forcebalance.target.Target.rundir** [**inherited**] 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.

Definition at line 137 of file target.py.

**forcebalance.abinitio.AblInitio.save\_vmvales** [**inherited**] Save the mvales from the last time we updated the vsites.

Definition at line 161 of file abinitio.py.

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

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 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 135 of file target.py.

**forcebalance.abinitio.AblInitio.topology\_flag** [**inherited**] Definition at line 166 of file abinitio.py.

**forcebalance.abinitio.AblInitio.tq\_err** [**inherited**] Definition at line 990 of file abinitio.py.

**forcebalance.abinitio.AblInitio.tq\_err\_pct** [**inherited**] Definition at line 137 of file abinitio.py.

**forcebalance.abinitio.AblInitio.tq\_ref** [**inherited**] Definition at line 989 of file abinitio.py.

**forcebalance.abinitio.AblInitio.traj** [**inherited**] Definition at line 142 of file abinitio.py.

**forcebalance.abinitio\_internal.AblInitio\_Internal.trajfnm** Name of the trajectory, we need this BEFORE initializing the SuperClass.

Definition at line 42 of file abinitio\_internal.py.

**forcebalance.abinitio.AblInitio.use\_nft** [**inherited**] Whether to compute net forces and torques, or not.  
Definition at line 139 of file abinitio.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.abinitio.AbInitio.w.energy** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.AbInitio.w.force** [inherited] Definition at line 350 of file abinitio.py.

**forcebalance.abinitio.AbInitio.w.netforce** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.AbInitio.w.resp** [inherited] Definition at line 1002 of file abinitio.py.

**forcebalance.abinitio.AbInitio.w.torque** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.AbInitio.whamboltz** [inherited] Definition at line 370 of file abinitio.py.

**forcebalance.abinitio.AbInitio.whamboltz.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) WHAM Boltzmann weights

Definition at line 110 of file abinitio.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 141 of file target.py.

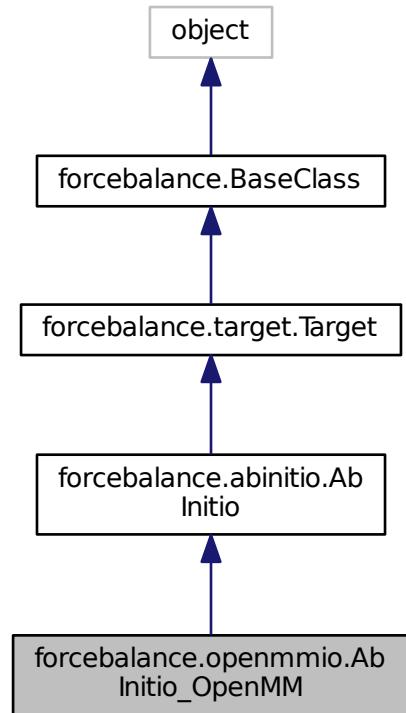
The documentation for this class was generated from the following file:

- [abinitio.internal.py](#)

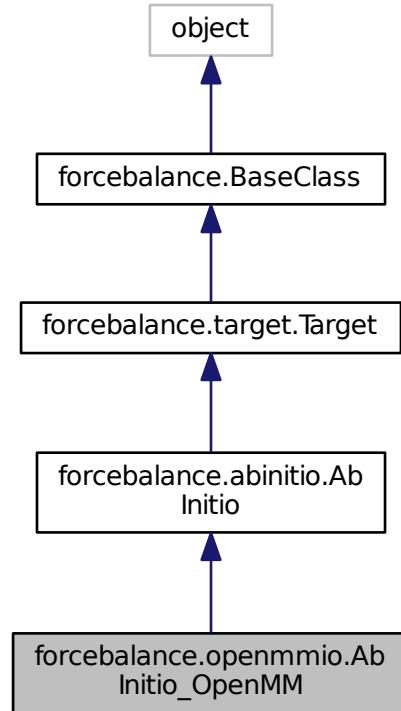
## 8.5 forcebalance.openmmio.AbInitio\_OpenMM Class Reference

Subclass of AbInitio for 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 `read_topology`
- def `prepare_temp_directory`
- def `energy_force_driver_all_external_`
- def `energy_force_driver_all_internal_`

*Loop through the snapshots and compute the energies and forces using OpenMM.*

- 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_force_transformer_all`
- def `energy_force_transformer`
- def `get_energy_force_`

*LPW 06-30-2013.*

- def `get_resp_`

- def [get](#)  
*Computes the objective function contribution without any parametric derivatives.*
- 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 [sget](#)  
*Stages the directory for the target, and then calls 'get'.*
- 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 [set\\_option](#)

## Public Attributes

- [trajfnm](#)  
*Name of the trajectory, we need this BEFORE initializing the SuperClass.*
- [platform](#)  
*Initialize the SuperClass!*
- [simulation](#)  
*Create the simulation object within this class itself.*
- [xyz\\_omms](#)
- [topology\\_flag](#)
- [whamboltz\\_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.*
- **traj**
- **nparticles**  
*The number of (atoms + drude particles + virtual sites)*
- **AtomLists**  
*This is a default-dict containing a number of atom-wise lists, such as the residue number of each atom, the mass of each atom, and so on.*
- **new\_vsites**  
*Read in the topology.*
- **save\_vmvvals**  
*Save the mvvals 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**

- **tempdir**  
Root directory of the whole project.
- **rundir**  
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.
- **PrintOptionDict**
- **verbose\_options**

### 8.5.1 Detailed Description

Subclass of AbInitio for force and energy matching using OpenMM.

Implements the prepare and energy\_force\_driver methods. The get method is in the superclass.

Definition at line 422 of file openmmio.py.

### 8.5.2 Constructor & Destructor Documentation

```
def forcebalance.openmmio.AbInitio_OpenMM.__init__( self, options, tgt_opts, forcefield )
```

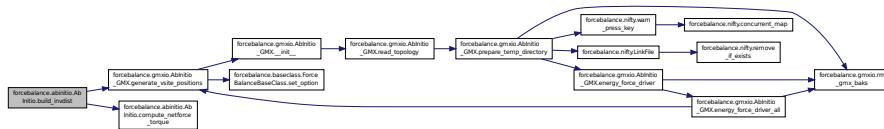
Definition at line 425 of file openmmio.py.

### 8.5.3 Member Function Documentation

```
def forcebalance.abinitio.AbInitio.build_invdist( self, mvals ) [inherited]
```

Definition at line 168 of file abinitio.py.

Here is the call graph for this function:



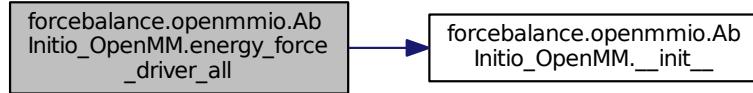
```
def forcebalance.abinitio.AbInitio.compute_netforce_torque( self, xyz, force, QM = False ) [inherited]
```

Definition at line 204 of file abinitio.py.

```
def forcebalance.openmmio.AbInitio_OpenMM.energy_force_driver_all( self )
```

Definition at line 551 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.ABIInitio\_OpenMM.energy\_force\_driver\_all\_external\_( self )** Definition at line 494 of file openmmio.py.

```
def forcebalance.openmmio.ABInitio_OpenMM.energy_force_driver_all_internal_( self ) Loop through the snapshots and compute the energies and forces using OpenMM.
```

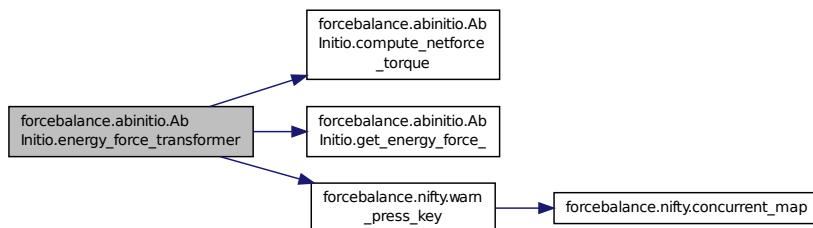
Definition at line 503 of file openmmio.py.

Here is the call graph for this function:



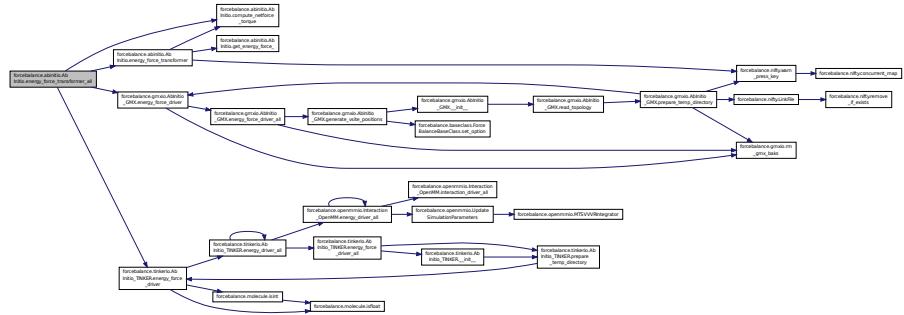
**def forcebalance.abinitio.AblInitio.energy\_force\_transformer ( self, i ) [inherited]** Definition at line 459 of file abinitio.py.

Here is the call graph for this function:



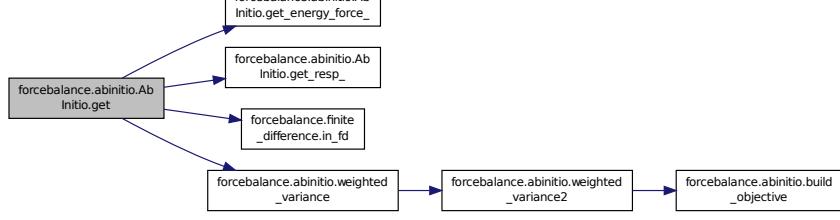
**def forcebalance.abinitio.AblInitio.energy\_force\_transformer\_all ( self ) [inherited]** Definition at line 442 of file abinitio.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.ABInitio.get ( self, mvals, AGrad = False, AHess = False ) [inherited]
Definition at line 1100 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
```

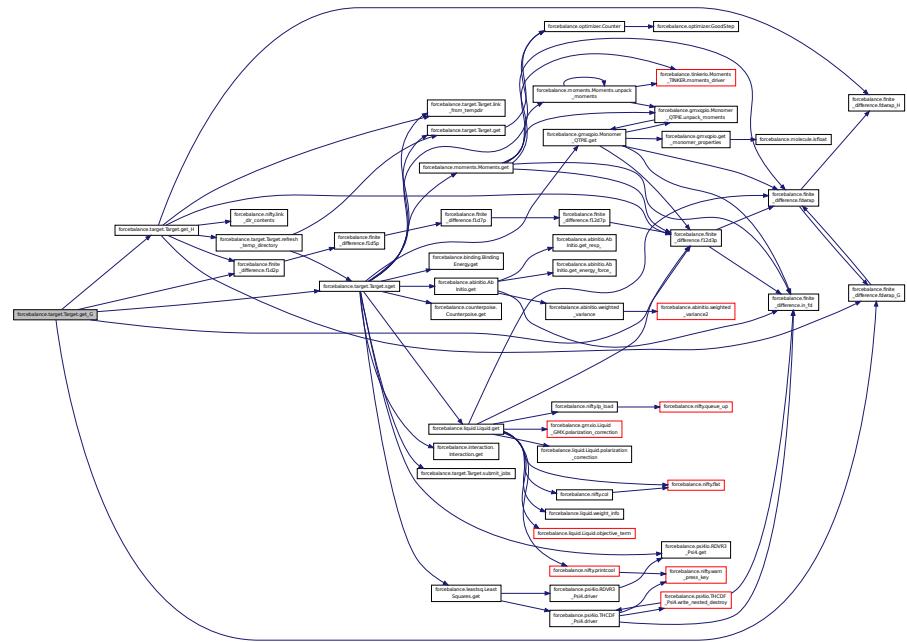
Definition at line 534 of file abinitio.py.

**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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



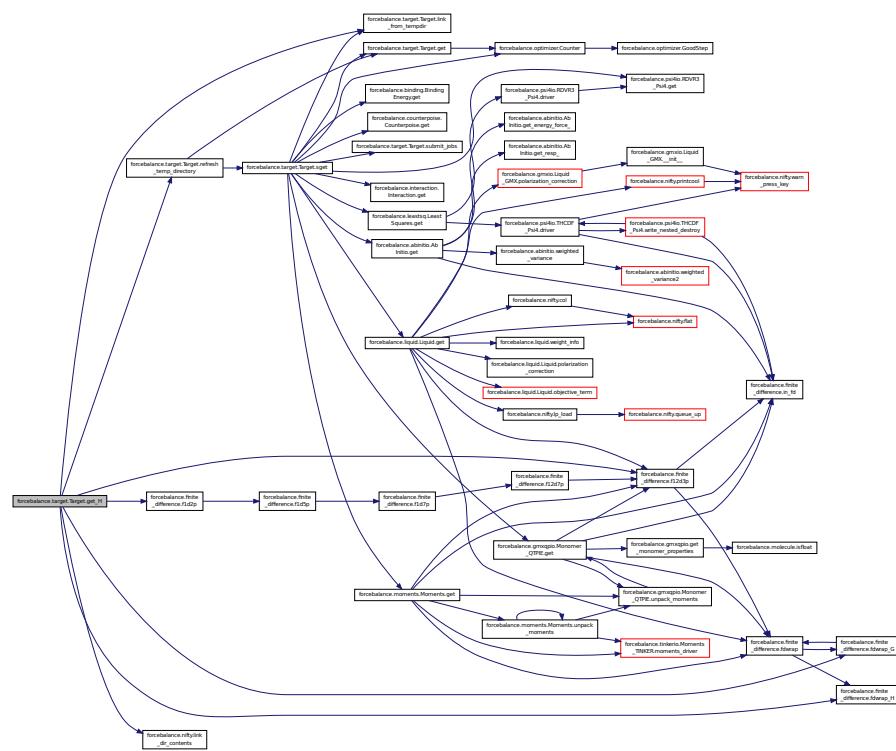
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 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.
```

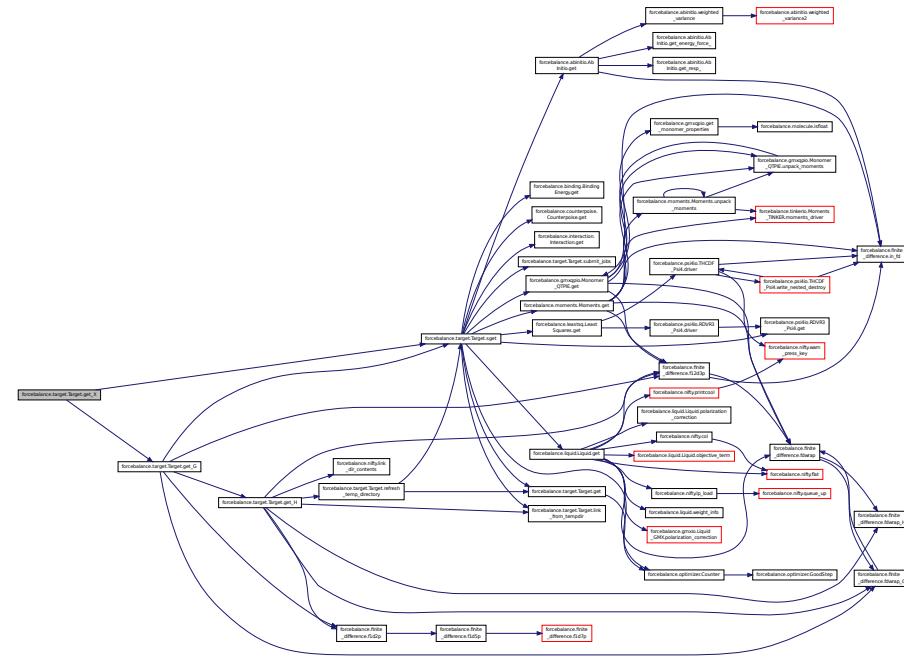
Implements the RESP objective function. (In Python so obviously not optimized.) This function takes the mathematical parameter values and returns the charges on the ATOMS (fancy mapping going on)

Definition at line 1001 of file abinitio.py.

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

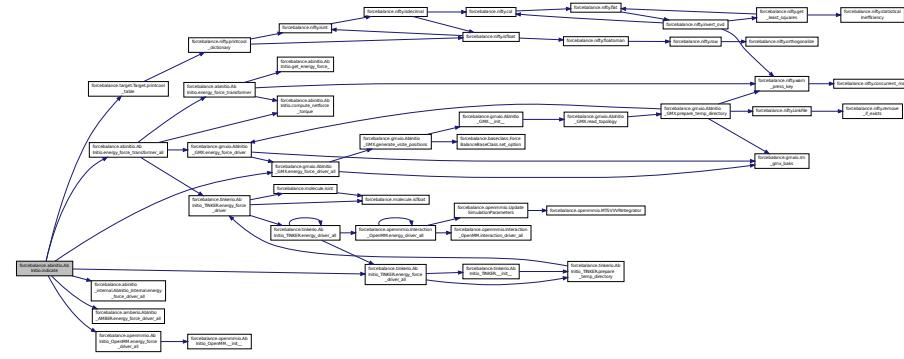
Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.abinitio.AblInitio.indicate ( self ) [inherited]** Definition at line 422 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 213 of file target.py.

**def forcebalance.openmmio.ABInitio\_OpenMM.prepare\_temp\_directory ( self, options, tgt\_opts )** Definition at line 489 of file openmmio.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.

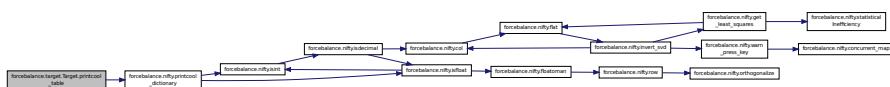
The input should be something like:

## 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 367 of file target.py.

Here is the call graph for this function:



```
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.

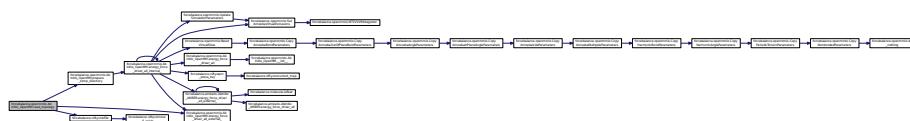
Here, we have a measure for the information content of our snapshots, which comes easily from the definition of information entropy:

```
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 317 of file abinitio.py.

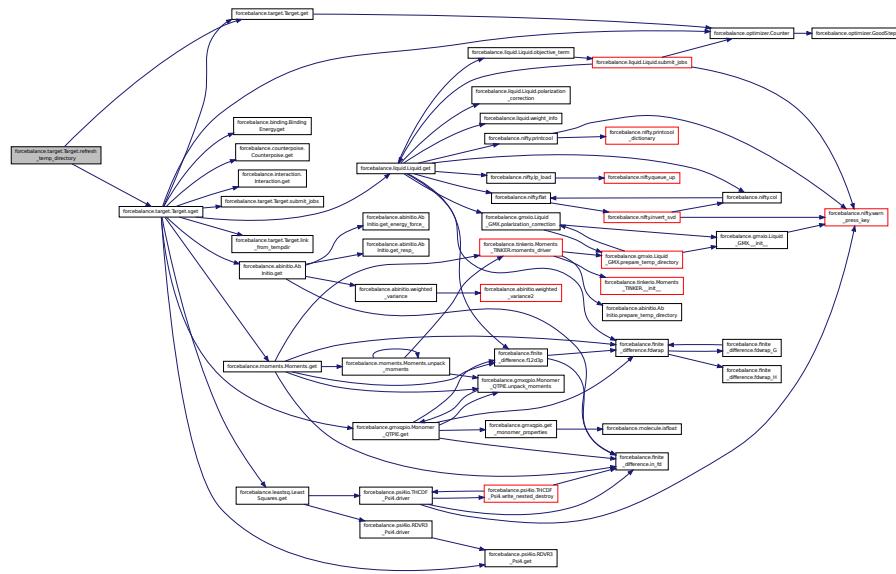
**def forcebalance.openmmio.AbInitio\_OpenMM.read\_topology( self )** Definition at line 479 of file openmmio.py.  
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



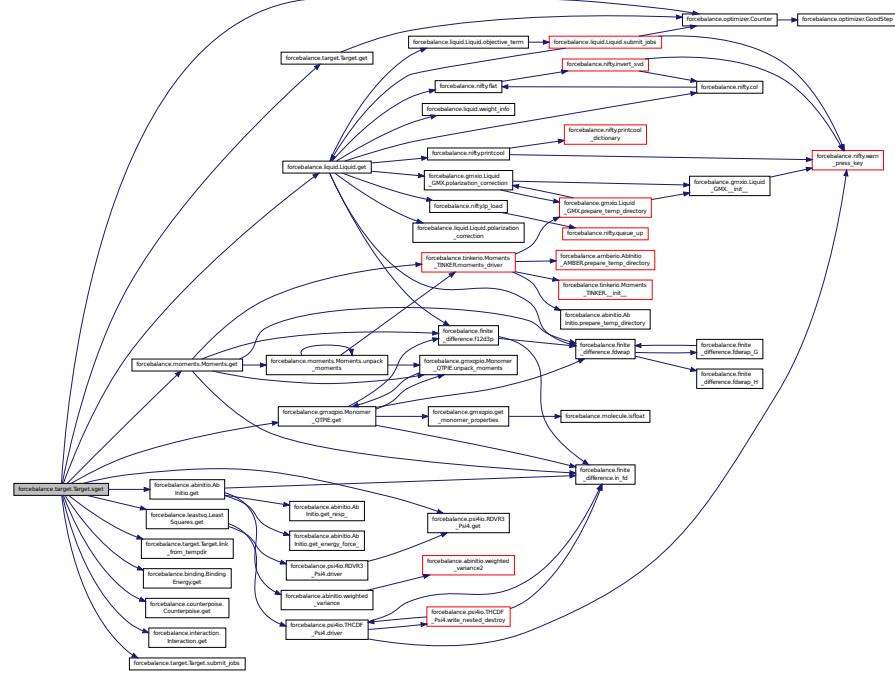
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

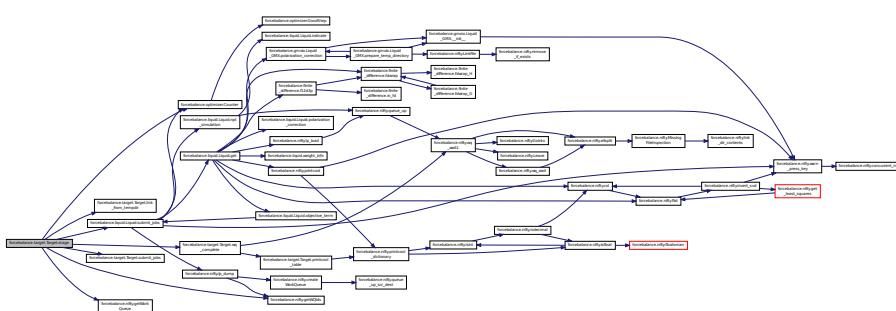


**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

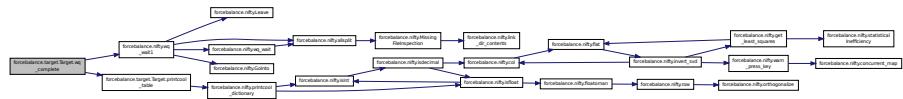


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### **8.5.4 Member Data Documentation**

**forcebalance.abinitio.AblInitio.AtomLists** [inherited] This is a default-dict containing a number of atom-wise lists, such as the residue number of each atom, the mass of each atom, and so on.

Definition at line 150 of file abinitio.py.

**forcebalance.abinitio.AbInitio.e\_err** [inherited] Qualitative Indicator: average energy error (in kJ/mol)  
Definition at line 128 of file abinitio.py.

**forcebalance.abinitio.AblInitio.e\_err\_pct** [inherited] Definition at line 129 of file abinitio.py.

**forcebalance.abinitio.ABInitio.e\_ref** [inherited] Definition at line 978 of file abinitio.py.

**forcebalance.abinitio.ABInitio.emd0** [inherited] Energies of the sampling simulation.

Definition at line 116 of file abinitio.py.

**forcebalance.abinitio.ABInitio.eqm** [inherited] Reference (QM) energies.

Definition at line 114 of file abinitio.py.

**forcebalance.abinitio.AblInitio.esp\_err** [inherited] Qualitative Indicator: "relative RMS" for electrostatic potential.

Definition at line 134 of file abinitio.py.

**forcebalance.abinitio.AbInitio.espval** [inherited] ESP values.

Definition at line 122 of file abinitio.py.

**forcebalance.abinitio.AblInitio.espxyz** [inherited] ESP grid points.

Definition at line 120 of file abinitio.py.

**forcebalance.abinitio.ABInitio.f\_err** [inherited] Qualitative Indicator: average force error (fractional)

Definition at line 131 of file abinitio.py.

**forcebalance.abinitio.AbInitio.f\_err\_pct** [inherited] Definition at line 132 of file abinitio.py.

**forcebalance.abinitio.AblInitio.f\_ref** [inherited] Definition at line 982 of file abinitio.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.abinitio.AbInitio.fitatoms** [inherited] Definition at line 354 of file abinitio.py.

**forcebalance.abinitio.ABInitio.force** [inherited] Definition at line 349 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force\_map** [inherited] Definition at line 212 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fqm** [inherited] Reference (QM) forces.  
Definition at line 118 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fref** [inherited] Definition at line 413 of file abinitio.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.  
Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.  
Definition at line 145 of file target.py.

**forcebalance.abinitio.AbInitio.invdists** [inherited] Definition at line 1009 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nesp** [inherited] Definition at line 351 of file abinitio.py.

**forcebalance.abinitio.AbInitio.new\_vsites** [inherited] Read in the topology.

Read in the reference data Prepare the temporary directory 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 159 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_err** [inherited] Definition at line 135 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_err\_pct** [inherited] Definition at line 136 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_ref** [inherited] Definition at line 986 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nftqm** [inherited] Definition at line 409 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nnf** [inherited] Definition at line 255 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nparticles** [inherited] The number of (atoms + drude particles + virtual sites)  
Definition at line 147 of file abinitio.py.

**forcebalance.abinitio.AbInitio.ns** [inherited] Read in the trajectory file.  
Definition at line 141 of file abinitio.py.

**forcebalance.abinitio.AbInitio.ntq** [inherited] Definition at line 256 of file abinitio.py.

**forcebalance.abinitio.AbInitio.objective** [inherited] Definition at line 1120 of file abinitio.py.

**forcebalance.openmmio.AbInitio\_OpenMM.platform** Initialize the SuperClass!  
Set the device to the environment variable or zero otherwise.  
Set the simulation platform  
Definition at line 434 of file openmmio.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.qfnm** [inherited] The qdata.txt file that contains the QM energies and forces.  
Definition at line 124 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 126 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmboltz\_wts** [inherited] QM Boltzmann weights.  
Definition at line 112 of file abinitio.py.

**forcebalance.abinitio.ABInitio.respterm** [inherited] Definition at line 1088 of file abinitio.py.

**forcebalance.target.Target.rundir** [inherited] 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.  
Definition at line 137 of file target.py.

**forcebalance.abinitio.ABInitio.save\_vvals** [inherited] Save the vvals from the last time we updated the vsites.  
Definition at line 161 of file abinitio.py.

**forcebalance.openmmio.ABInitio\_OpenMM.simulation** Create the simulation object within this class itself.  
Definition at line 467 of file openmmio.py.

**forcebalance.target.Target.tempdir** [inherited] Root directory of the whole project.  
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 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 135 of file target.py.

**forcebalance.openmmio.ABInitio\_OpenMM.topology\_flag** Definition at line 486 of file openmmio.py.

**forcebalance.abinitio.ABInitio.tq\_err** [inherited] Definition at line 990 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_err\_pct** [inherited] Definition at line 137 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_ref** [inherited] Definition at line 989 of file abinitio.py.

**forcebalance.abinitio.ABInitio.traj** [inherited] Definition at line 142 of file abinitio.py.

**forcebalance.openmmio.ABInitio\_OpenMM.trajfnm** Name of the trajectory, we need this BEFORE initializing the SuperClass.  
Definition at line 427 of file openmmio.py.

**forcebalance.abinitio.ABInitio.use\_nft** [inherited] Whether to compute net forces and torques, or not.  
Definition at line 139 of file abinitio.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.w\_energy** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_force** [inherited] Definition at line 350 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_netforce** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_resp** [inherited] Definition at line 1002 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_torque** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz** [inherited] Definition at line 370 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz\_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) WHAM Boltzmann weights

Definition at line 110 of file abinitio.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

Definition at line 141 of file target.py.

**forcebalance.openmmio.ABInitio\_OpenMM.xyz\_omms** Definition at line 469 of file openmmio.py.

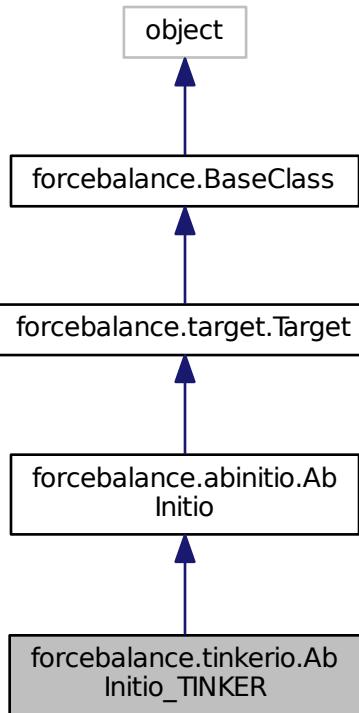
The documentation for this class was generated from the following file:

- [openmmio.py](#)

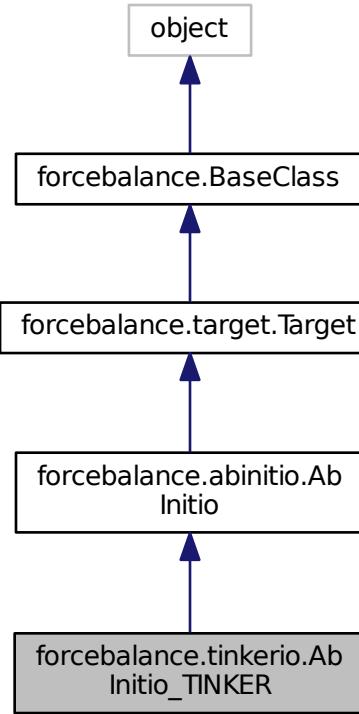
## 8.6 forcebalance.tinkerio.ABInitio\_TINKER Class Reference

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 `prepare_temp_directory`
- def `energy_force_driver`
- def `energy_driver_all`
- def `energy_force_driver_all`
- def `read_topology`
- 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_force_transformer_all`
- def `energy_force_transformer`
- 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\\_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 [sget](#)  
*Stages the directory for the target, and then calls 'get'.*
- 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 [set\\_option](#)

## Public Attributes

- [trajfnm](#)  
*Name of the trajectory.*
- [all\\_at\\_once](#)  
*all\_at\_once is not implemented.*
- [whamboltz\\_wts](#)  
*Initialize the base class.*
- [qm boltz\\_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.*
- **traj**
- **nparticles**  
*The number of (atoms + drude particles + virtual sites)*
- **AtomLists**  
*This is a default-dict containing a number of atom-wise lists, such as the residue number of each atom, the mass of each atom, and so on.*
- **new\_vsites**  
*Read in the topology.*
- **save\_vmvabs**  
*Save the mvabs from the last time we updated the vsites.*
- **topology\_flag**
- **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**
- **tempdir**  
*Root directory of the whole project.*
- **rundir**  
*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.
- **PrintOptionDict**
- **verbose\_options**

### 8.6.1 Detailed Description

Subclass of Target for force and energy matching using TINKER.

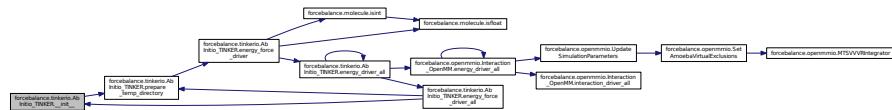
Implements the prepare and energy\_force.driver methods.

Definition at line 295 of file tinkerio.py.

### 8.6.2 Constructor & Destructor Documentation

**def forcebalance.tinkerio.AbInitio\_TINKER.\_\_init\_\_ ( self, options, tgt\_opts, forcefield )** Definition at line 298 of file tinkerio.py.

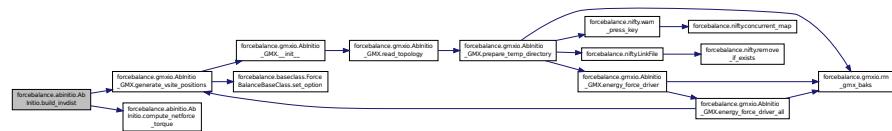
Here is the call graph for this function:



### 8.6.3 Member Function Documentation

**def forcebalance.abinitio.AbInitio.build\_invdist ( self, mvals ) [inherited]** Definition at line 168 of file abinitio.py.

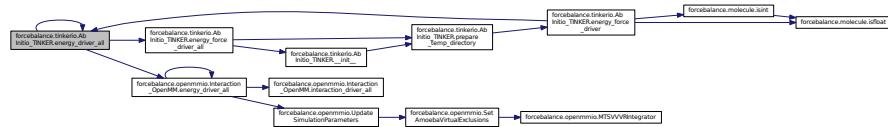
Here is the call graph for this function:



**def forcebalance.abinitio.AbInitio.compute\_netforce\_torque ( self, xyz, force, QM = False ) [inherited]** Definition at line 204 of file abinitio.py.

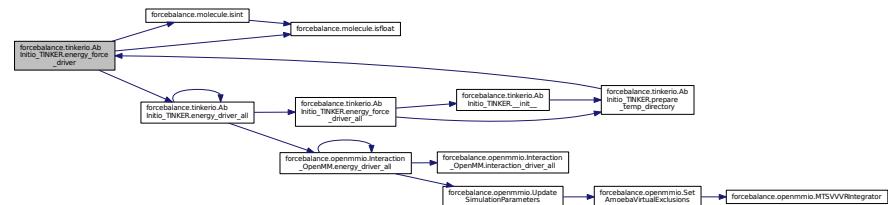
**def forcebalance.tinkerio.AbInitio\_TINKER.energy\_driver.all ( self )** Definition at line 331 of file tinkerio.py.

Here is the call graph for this function:



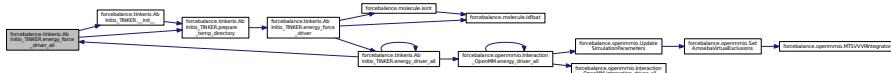
**def forcebalance.tinkerio.AbInitio\_TINKER.energy\_force\_driver ( self, shot )** Definition at line 315 of file tinkerio.py.

Here is the call graph for this function:



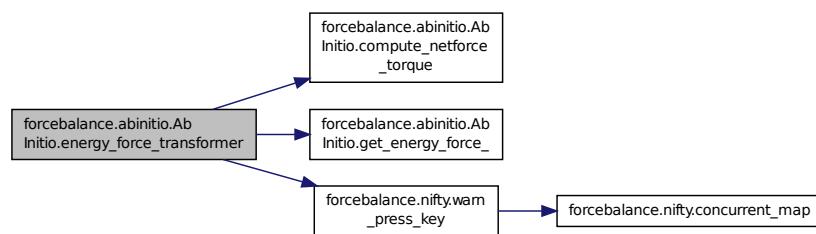
**def forcebalance.tinkerio.AbInitio\_TINKER.energy\_force\_driver\_all ( self )** Definition at line 344 of file tinkerio.py.

Here is the call graph for this function:



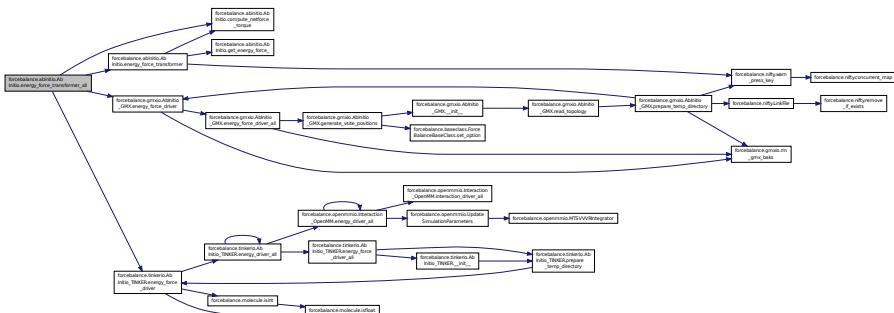
**def forcebalance.abinitio.AbInitio.energy\_force\_transformer ( self, i ) [inherited]** Definition at line 459 of file abinitio.py.

Here is the call graph for this function:



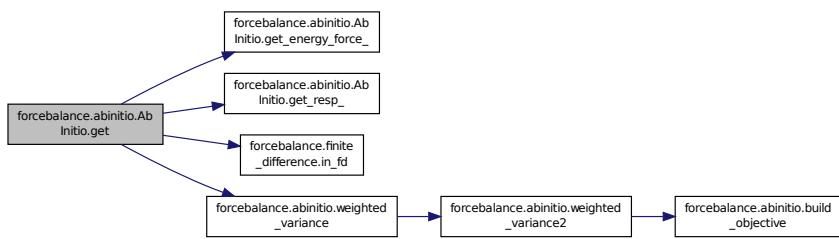
**def forcebalance.abinitio.AbInitio.energy\_force\_transformer\_all ( self ) [inherited]** Definition at line 442 of file abinitio.py.

Here is the call graph for this function:



```
def forcebalance.abinitio.ABInitio.get( self, mvals, AGrad = False, AHess = False ) [inherited]
Definition at line 1100 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
```

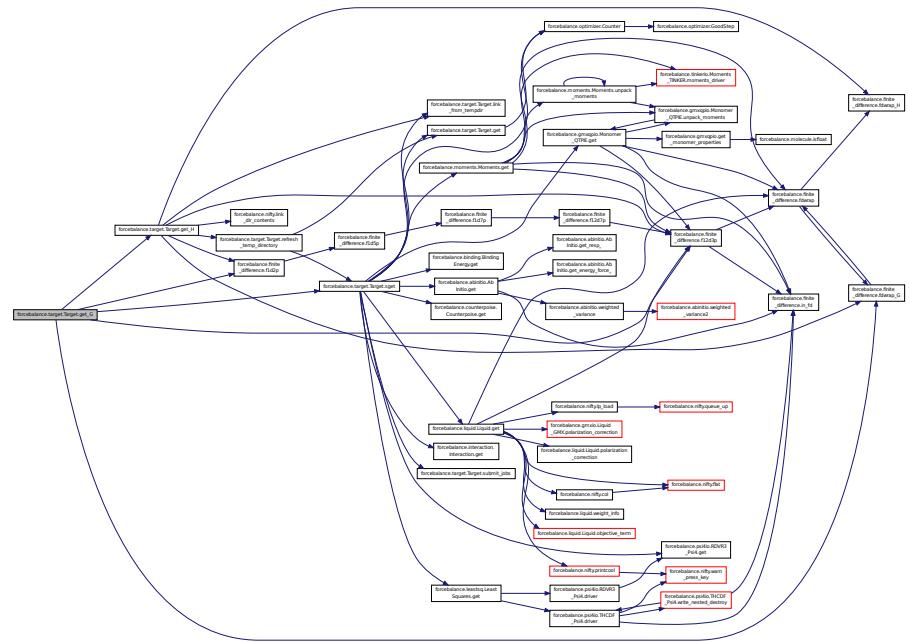
Definition at line 534 of file abinitio.py.

**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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



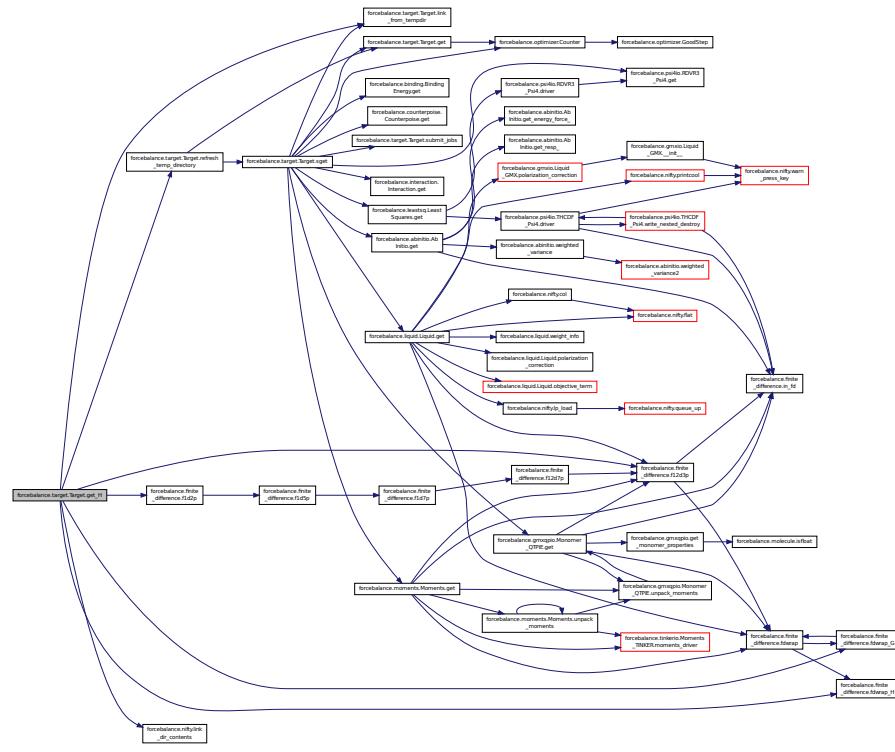
**def forcebalance.target.Target.get.H ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 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.
```

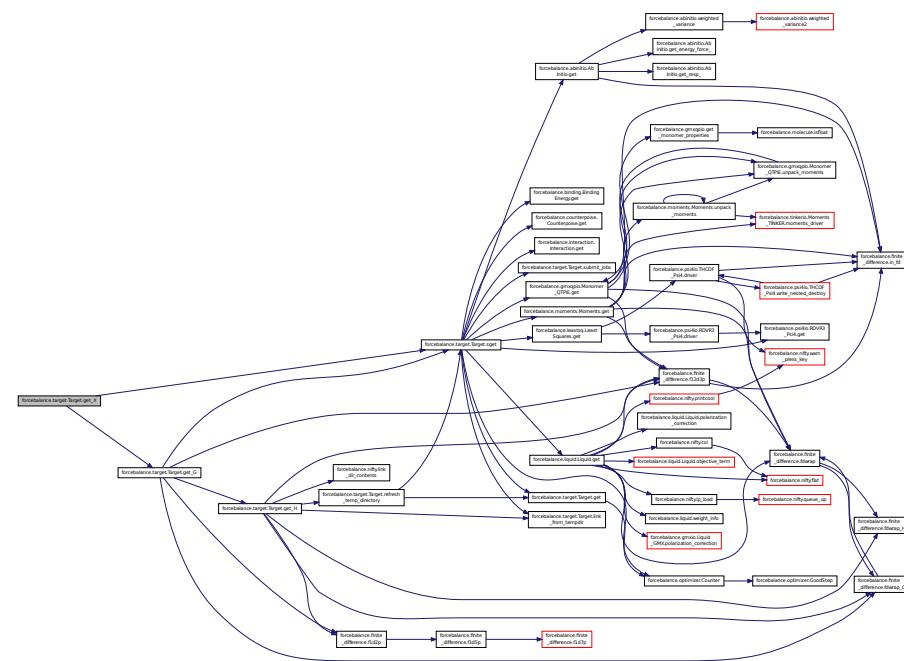
Implements the RESP objective function. (In Python so obviously not optimized.) This function takes the mathematical parameter values and returns the charges on the ATOMS (fancy mapping going on)

Definition at line 1001 of file abinitio.py.

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

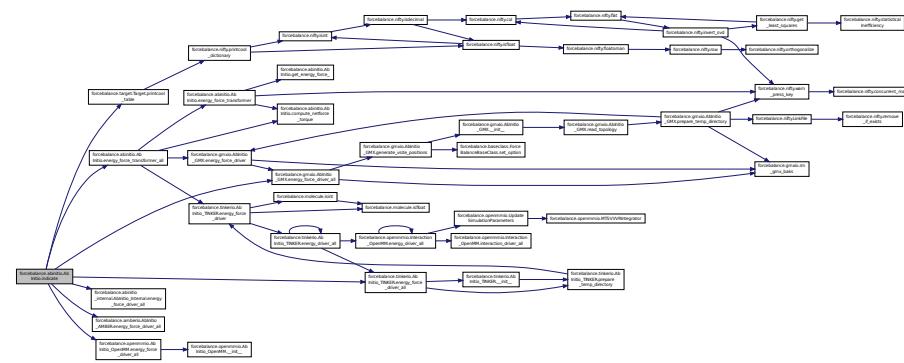
Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.abinitio.ABInitio.indicate ( self ) [inherited]** Definition at line 422 of file abinitio.py.

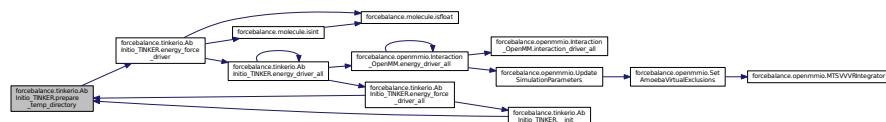
Here is the call graph for this function:



```
def forcebalance.target.Target.link_from_tempdir( self, absdestdir ) [inherited] Definition at line 213 of file target.py.
```

**def forcebalance.tinkerio.AblInitio\_TINKER.prepare\_temp\_directory ( self, options, tgt\_opts )** Definition at line 307 of file tinkerio.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.

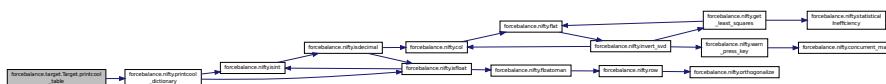
The input should be something like:

## 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 367 of file target.py.

Here is the call graph for this function:



**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.

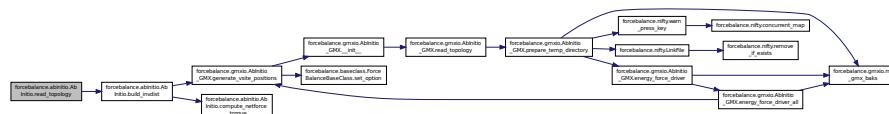
Here, we have a measure for the information content of our snapshots, which comes easily from the definition of information entropy:

```
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 317 of file abinitio.py.

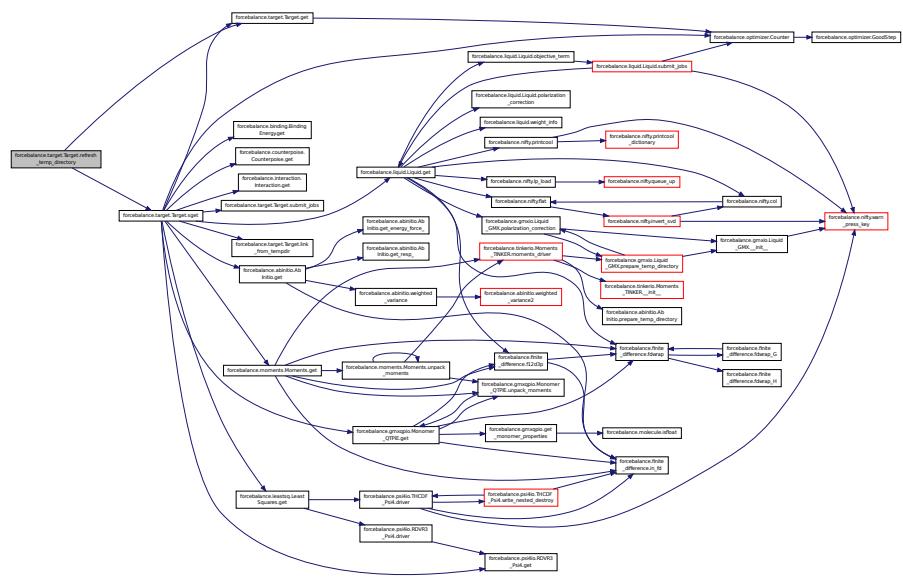
**def forcebalance.abinitio.AbInitio.read\_topology( self ) [inherited]** Definition at line 164 of file abinitio.py.  
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



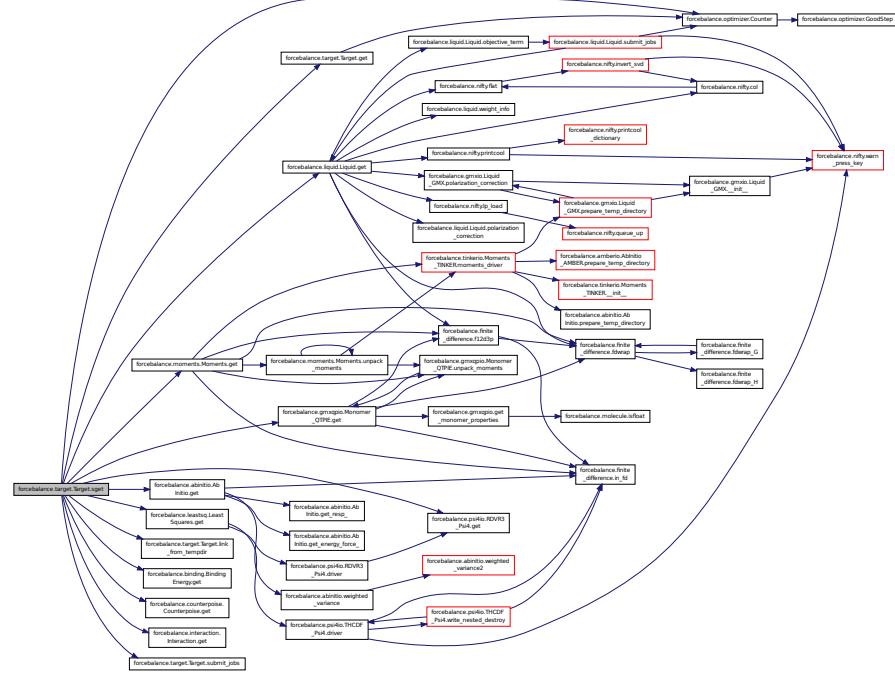
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

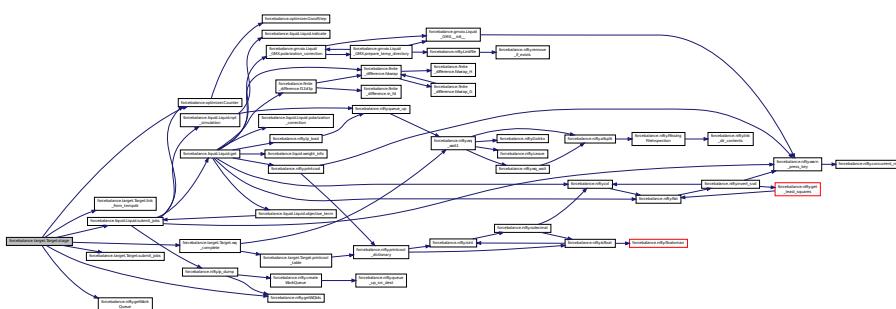


**[inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

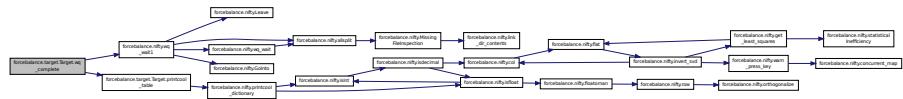


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### **8.6.4 Member Data Documentation**

**forcebalance.tinkerio.ABInitio\_TINKER.all\_at\_once** all\_at\_once is not implemented.

Definition at line 305 of file tinkerio.py.

**forcebalance.abinitio.AblInitio.AtomLists** [inherited] This is a default-dict containing a number of atom-wise lists, such as the residue number of each atom, the mass of each atom, and so on.

Definition at line 150 of file abinitio.py.

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

Definition at line 128 of file abinitio.py.

**forcebalance.abinitio.ABInitio.e\_err\_pct** [inherited] Definition at line 129 of file abinitio.py.

**forcebalance.abinitio.ABInitio.e\_ref** [inherited] Definition at line 978 of file abinitio.py.

**forcebalance.abinitio.ABInitio.emd0** [inherited] Energies of the sampling simulation.

Definition at line 116 of file abinitio.py.

**forcebalance.abinitio.AblInitio.egm** [inherited] Reference (QM) energies.

Definition at line 114 of file abinitio.py.

**forcebalance.abinitio.AblInitio.esp\_err** [inherited] Qualitative Indicator: "relative RMS" for electrostatic potential.

Definition at line 134 of file abinitio.py.

**forcebalance.abinitio.ABInitio.espval** [inherited] ESP values.

Definition at line 122 of file abinitio.py.

**forcebalance.abinitio.AblInitio.espxyz** [inherited] ESP grid points.

Definition at line 120 of file abinitio.py.

**forcebalance.abinitio.AblnInitio.f\_err** [inherited] Qualitative Indicator: average force error (fractional)

Definition at line 131 of file abinitio.py.

**forcebalance.abinitio.AblInitio.f\_err\_pct** [inherited] Definition at line 132 of file abinitio.py.

**forcebalance.abinitio.AblInitio.f\_ref** [inherited] Definition at line 982 of file abinitio.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.abinitio.AbInitio.fitatoms** [inherited] Definition at line 354 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force** [inherited] Definition at line 349 of file abinitio.py.

**forcebalance.abinitio.AbInitio.force\_map** [inherited] Definition at line 212 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fqm** [inherited] Reference (QM) forces.  
Definition at line 118 of file abinitio.py.

**forcebalance.abinitio.AbInitio.fref** [inherited] Definition at line 413 of file abinitio.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.  
Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.  
Definition at line 145 of file target.py.

**forcebalance.abinitio.AbInitio.invdists** [inherited] Definition at line 1009 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nesp** [inherited] Definition at line 351 of file abinitio.py.

**forcebalance.abinitio.AbInitio.new\_vsites** [inherited] Read in the topology.

Read in the reference data Prepare the temporary directory 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 159 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_err** [inherited] Definition at line 135 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_err\_pct** [inherited] Definition at line 136 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nf\_ref** [inherited] Definition at line 986 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nftqm** [inherited] Definition at line 409 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nnf** [inherited] Definition at line 255 of file abinitio.py.

**forcebalance.abinitio.AbInitio.nparticles** [inherited] The number of (atoms + drude particles + virtual sites)  
Definition at line 147 of file abinitio.py.

**forcebalance.abinitio.AbInitio.ns** [inherited] Read in the trajectory file.  
Definition at line 141 of file abinitio.py.

**forcebalance.abinitio.AbInitio.ntq** [inherited] Definition at line 256 of file abinitio.py.

**forcebalance.abinitio.AbInitio.objective** [inherited] Definition at line 1120 of file abinitio.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.qfnm** [inherited] The qdata.txt file that contains the QM energies and forces.  
Definition at line 124 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 126 of file abinitio.py.

**forcebalance.abinitio.ABInitio.qmboltz\_wts** [inherited] QM Boltzmann weights.  
Definition at line 112 of file abinitio.py.

**forcebalance.abinitio.ABInitio.respterm** [inherited] Definition at line 1088 of file abinitio.py.

**forcebalance.target.Target.rundir** [inherited] 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.  
Definition at line 137 of file target.py.

**forcebalance.abinitio.ABInitio.save\_vvals** [inherited] Save the vvals from the last time we updated the vsites.  
Definition at line 161 of file abinitio.py.

**forcebalance.target.Target.tempdir** [inherited] Root directory of the whole project.  
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 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 135 of file target.py.

**forcebalance.abinitio.ABInitio.topology\_flag** [inherited] Definition at line 166 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_err** [inherited] Definition at line 990 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_err\_pct** [inherited] Definition at line 137 of file abinitio.py.

**forcebalance.abinitio.ABInitio.tq\_ref** [inherited] Definition at line 989 of file abinitio.py.

**forcebalance.abinitio.ABInitio.traj** [inherited] Definition at line 142 of file abinitio.py.

**forcebalance.tinkerio.ABInitio\_TINKER.trajfnm** Name of the trajectory.  
Definition at line 300 of file tinkerio.py.

**forcebalance.abinitio.ABInitio.use\_nft** [inherited] Whether to compute net forces and torques, or not.  
Definition at line 139 of file abinitio.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.abinitio.ABInitio.w\_energy** [inherited] Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_force [inherited]** Definition at line 350 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_netforce [inherited]** Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_resp [inherited]** Definition at line 1002 of file abinitio.py.

**forcebalance.abinitio.ABInitio.w\_torque [inherited]** Definition at line 573 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz [inherited]** Definition at line 370 of file abinitio.py.

**forcebalance.abinitio.ABInitio.whamboltz\_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) WHAM Boltzmann weights

Definition at line 110 of file abinitio.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

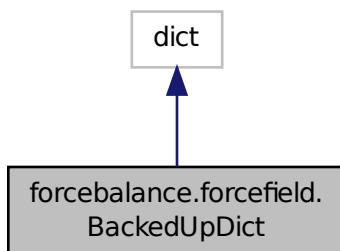
Definition at line 141 of file target.py.

The documentation for this class was generated from the following file:

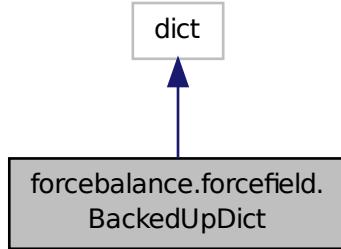
- [tinkerio.py](#)

## 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`

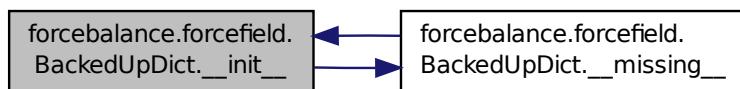
#### 8.7.1 Detailed Description

Definition at line 175 of file forcefield.py.

#### 8.7.2 Constructor & Destructor Documentation

**def forcebalance.forcefield.BackedUpDict.\_\_init\_\_ ( *self*, *backup\_dict* )** Definition at line 176 of file forcefield.py.

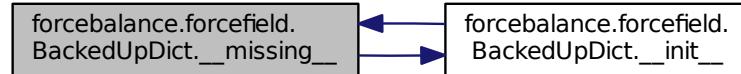
Here is the call graph for this function:



#### 8.7.3 Member Function Documentation

**def forcebalance.forcefield.BackedUpDict.\_\_missing\_\_ ( *self*, *key* )** Definition at line 179 of file forcefield.py.

Here is the call graph for this function:



#### 8.7.4 Member Data Documentation

**forcebalance.forcefield.BackedUpDict.backup\_dict** Definition at line 178 of file `forcefield.py`.

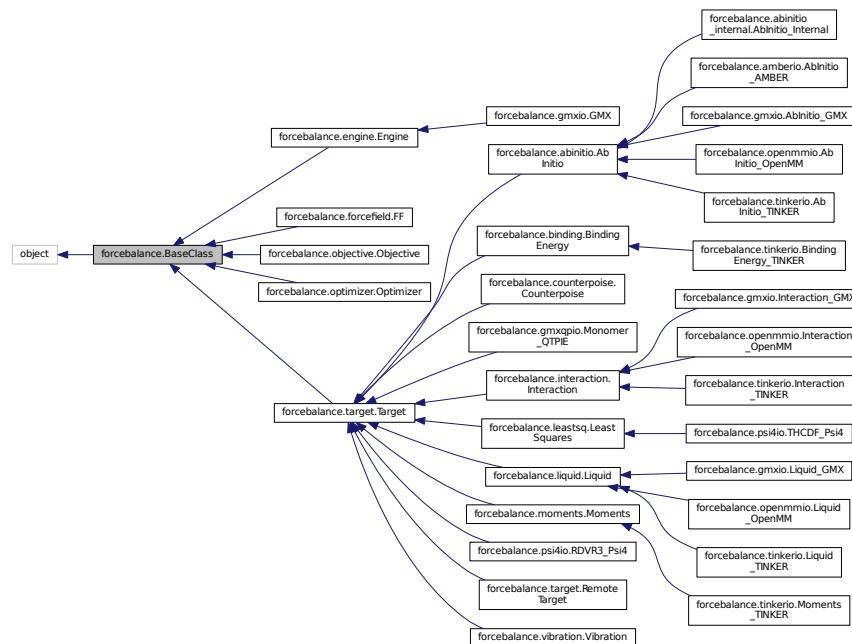
The documentation for this class was generated from the following file:

- [forcefield.py](#)

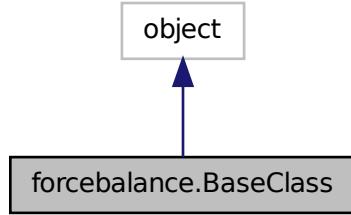
## 8.8 forcebalance.BaseClass Class Reference

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 `__init__`
- def `set_option`

### Public Attributes

- `PrintOptionDict`
- `verbose_options`

#### 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 28 of file `__init__.py`.

#### 8.8.3 Member Function Documentation

`def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False )` Definition at line 32 of file `__init__.py`.

#### 8.8.4 Member Data Documentation

`forcebalance.BaseClass.PrintOptionDict` Definition at line 29 of file `__init__.py`.

`forcebalance.BaseClass.verbose_options` Definition at line 30 of file `__init__.py`.

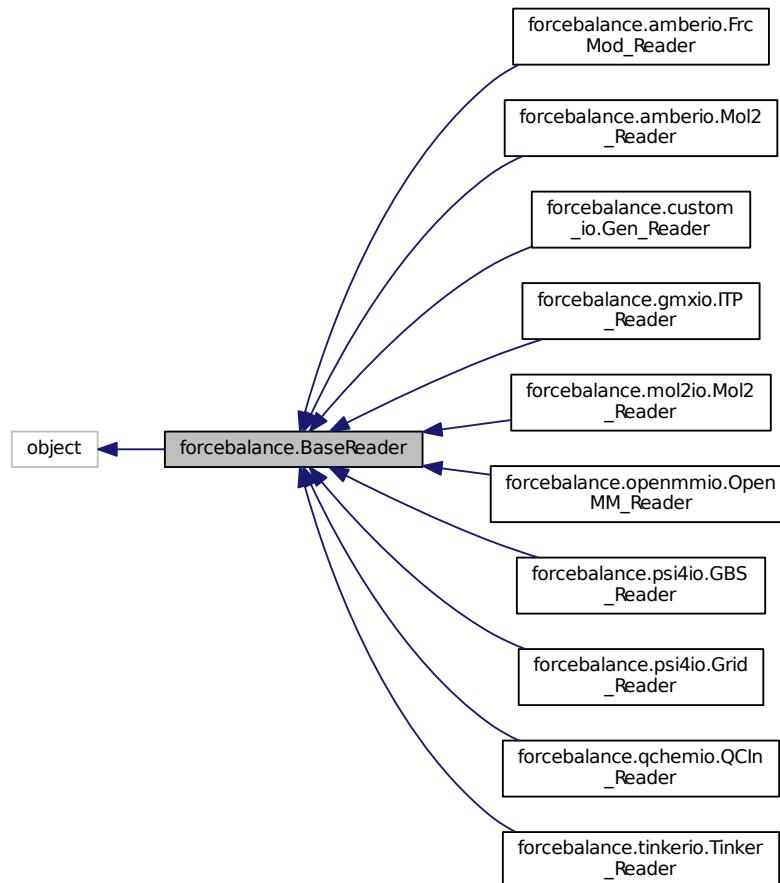
The documentation for this class was generated from the following file:

- `__init__.py`

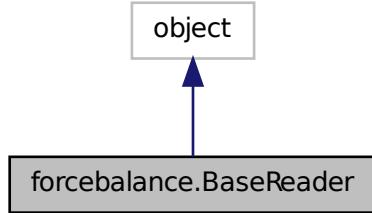
## 8.9 forcebalance.BaseReader Class Reference

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 64 of file `__init__.py`.

#### 8.9.2 Constructor & Destructor Documentation

`def forcebalance.BaseReader.__init__( self, fnm )` Definition at line 66 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 [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 107 of file [\\_\\_init\\_\\_.py](#).

**def forcebalance.BaseReader.feed ( self, line )** Definition at line 88 of file [\\_\\_init\\_\\_.py](#).

Here is the call graph for this function:



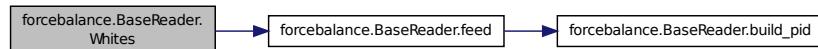
**def forcebalance.BaseReader.Split ( self, line )** Definition at line 82 of file [\\_\\_init\\_\\_.py](#).

Here is the call graph for this function:



**def forcebalance.BaseReader.Whites ( self, line )** Definition at line 85 of file [\\_\\_init\\_\\_.py](#).

Here is the call graph for this function:



### 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 72 of file [\\_\\_init\\_\\_.py](#).

**forcebalance.BaseReader.AtomTypes** Definition at line 80 of file [\\_\\_init\\_\\_.py](#).

**forcebalance.BaseReader.itype** Definition at line 68 of file [\\_\\_init\\_\\_.py](#).

**forcebalance.BaseReader.In** Definition at line 67 of file \_\_init\_\_.py.

**forcebalance.BaseReader.molatom** 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 77 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules** Definition at line 79 of file \_\_init\_\_.py.

**forcebalance.BaseReader.pdict** Definition at line 70 of file \_\_init\_\_.py.

**forcebalance.BaseReader.suffix** Definition at line 69 of file \_\_init\_\_.py.

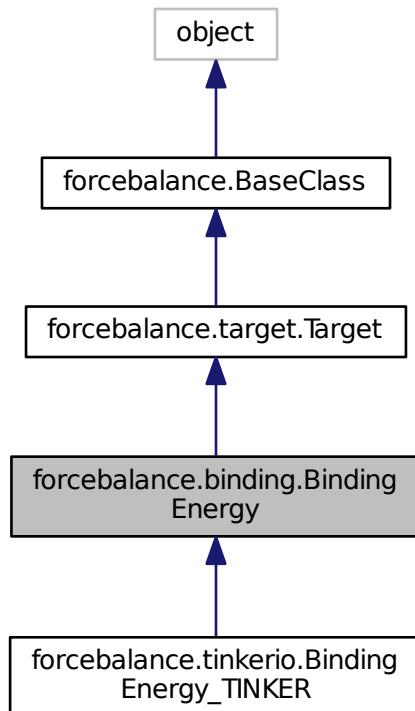
The documentation for this class was generated from the following file:

- [\\_\\_init\\_\\_.py](#)

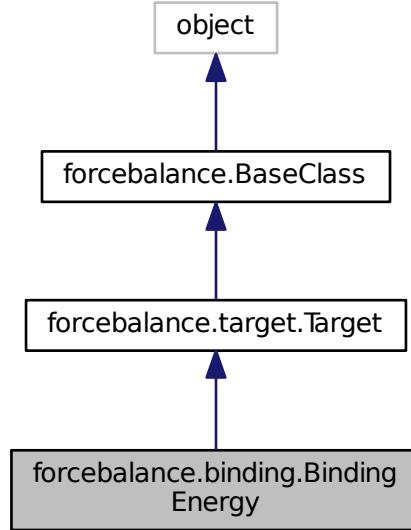
## 8.10 forcebalance.binding.BindingEnergy Class Reference

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 `indicate`
- def `get`
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- 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 `sget`

*Stages the directory for the target, and then calls 'get'.*
- 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 `set_option`

## Public Attributes

- [inter\\_opts](#)
- [PrintDict](#)
- [RMSDDict](#)
- [rmsd\\_part](#)
- [energy\\_part](#)
- [objective](#)
- [tempdir](#)

*Root directory of the whole project.*
- [rundir](#)

*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.*
- [PrintOptionDict](#)
- [verbose\\_options](#)

### 8.10.1 Detailed Description

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

Definition at line 123 of file binding.py.

### 8.10.2 Constructor & Destructor Documentation

**def forcebalance.binding.BindingEnergy.\_\_init\_\_ ( self, options, tgt\_opts, forcefield )** Definition at line 126 of file binding.py.

Here is the call graph for this function:



### 8.10.3 Member Function Documentation

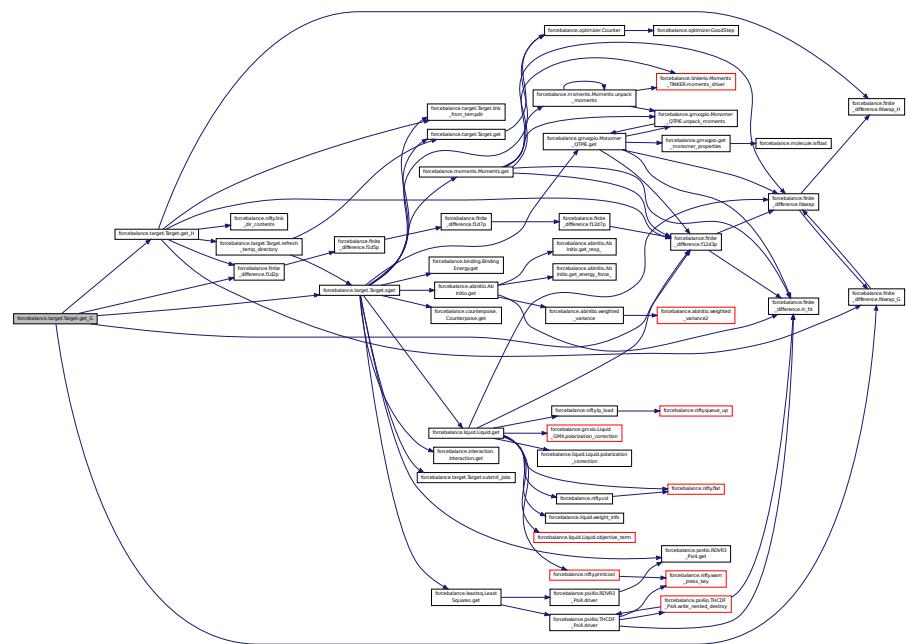
**def forcebalance.binding.BindingEnergy.get ( self, mvals, AGrad = False, AHess = False )** Definition at line 162 of file binding.py.

**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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



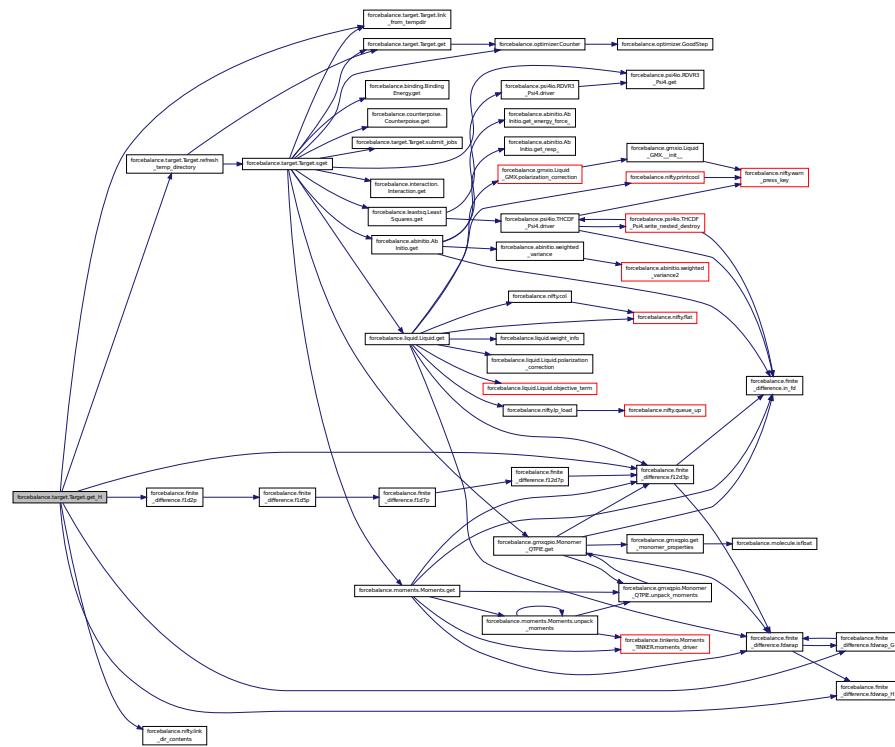
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

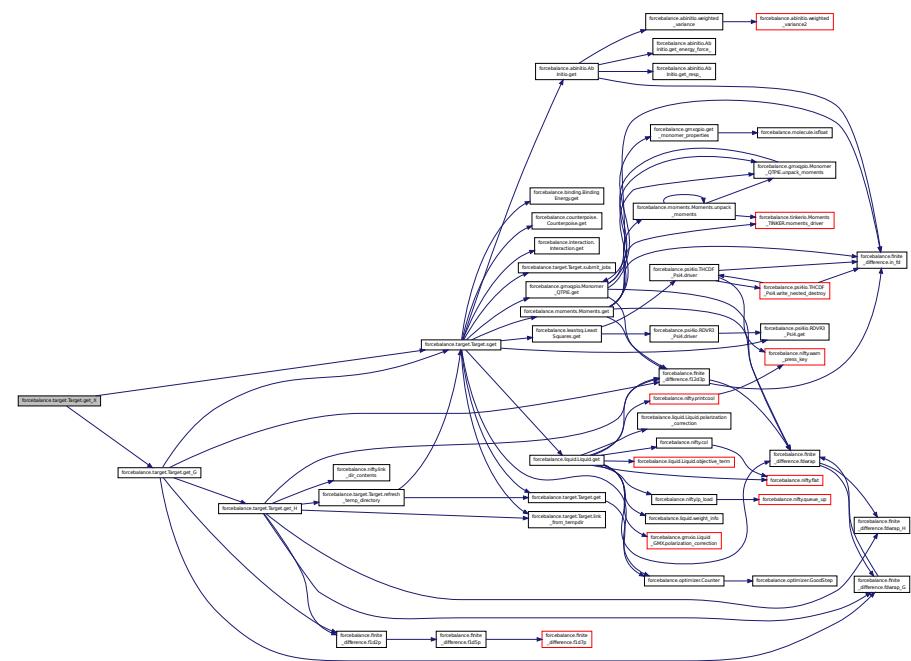
Here is the call graph for this function:



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

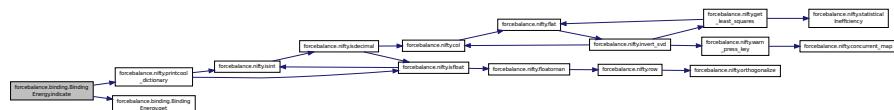
Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.binding.BindingEnergy.indicate ( self )** Definition at line 156 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 213 of file target.py.

```
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:

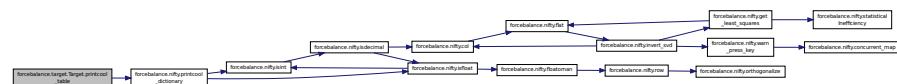
## Parameters

<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 367 of file target.py.

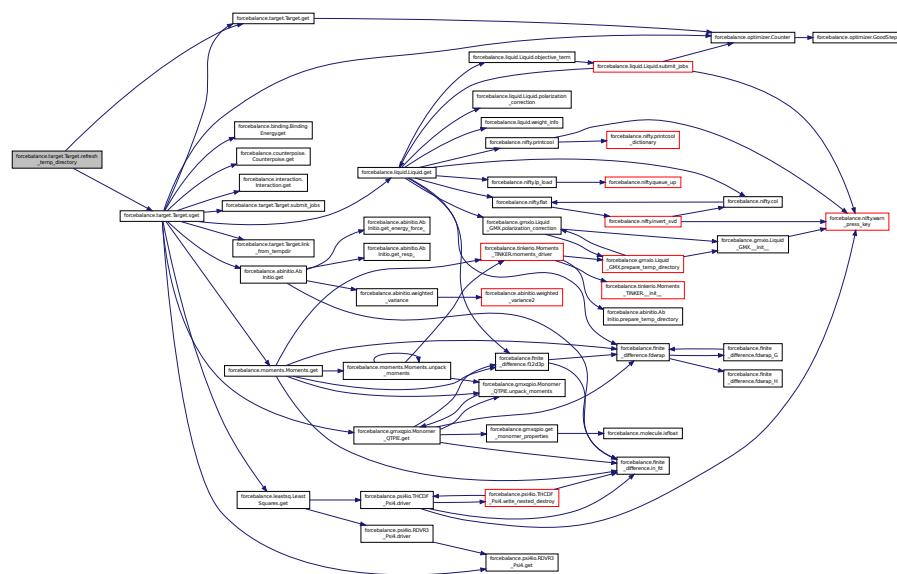
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



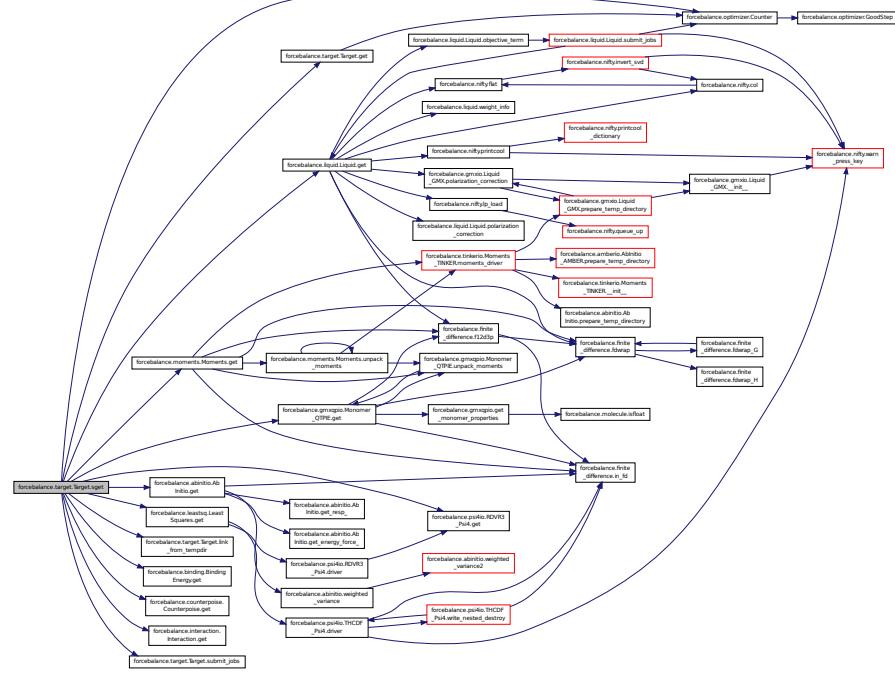
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

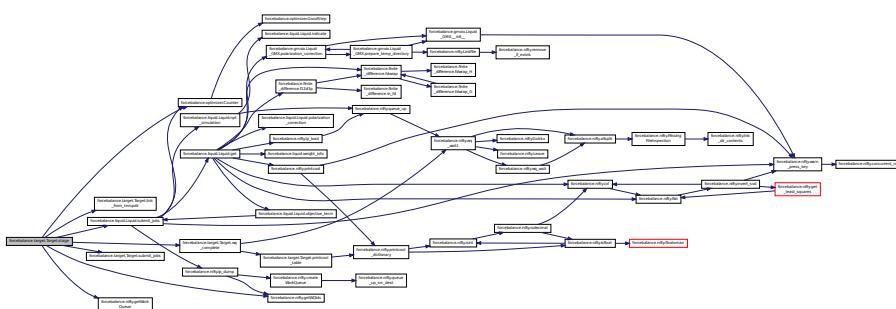


```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

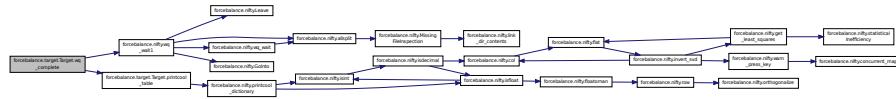


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### 8.10.4 Member Data Documentation

**forcebalance.binding.BindingEnergy.energy\_part** Definition at line 202 of file binding.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.

Definition at line 145 of file target.py.

**forcebalance.binding.BindingEnergy.inter\_opts** Definition at line 129 of file binding.py.

**forcebalance.binding.BindingEnergy.objective** Definition at line 226 of file binding.py.

**forcebalance.binding.BindingEnergy.PrintDict** Definition at line 164 of file binding.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.binding.BindingEnergy.rmsd\_part** Definition at line 200 of file binding.py.

**forcebalance.binding.BindingEnergy.RMSDDict** Definition at line 165 of file binding.py.

**forcebalance.target.Target.rundir** [inherited] 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.

Definition at line 137 of file target.py.

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

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 141 of file target.py.

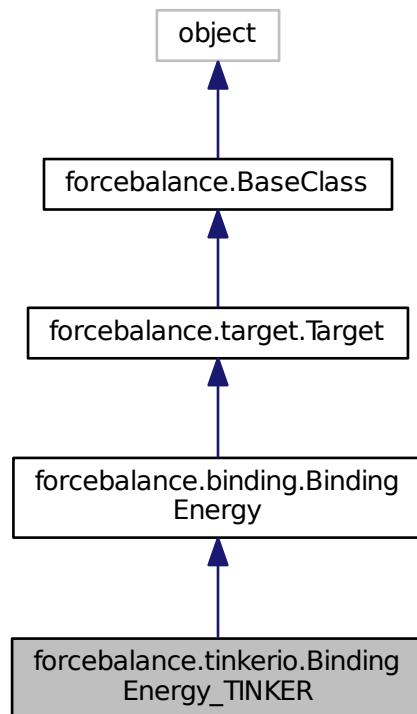
The documentation for this class was generated from the following file:

- [binding.py](#)

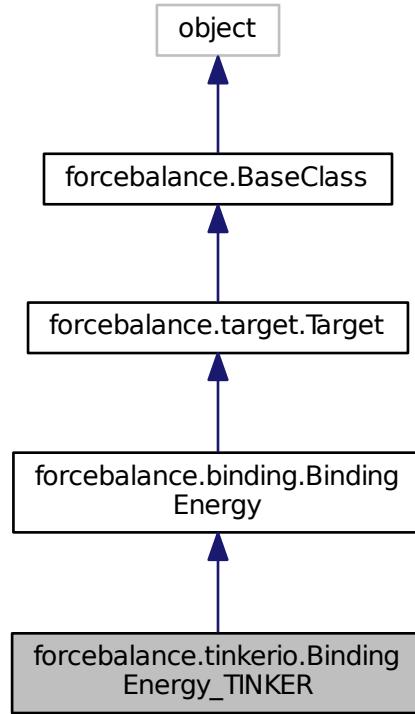
## 8.11 forcebalance.tinkerio.BindingEnergy\_TINKER Class Reference

Subclass of BindingEnergy for 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 `prepare_temp_directory`
- def `system_driver`
- def `indicate`
- def `get`
  - Computes the objective function contribution without any parametric derivatives.
- 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 `sget`
  - Stages the directory for the target, and then calls 'get'.
- 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 `set_option`

## Public Attributes

- `optprog`
- `inter_opts`
- `PrintDict`
- `RMSDDict`
- `rmsd_part`
- `energy_part`
- `objective`
- `tempdir`  
*Root directory of the whole project.*
- `rundir`  
*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.*
- `PrintOptionDict`
- `verbose_options`

### 8.11.1 Detailed Description

Subclass of `BindingEnergy` for binding energy matching using TINKER.

Definition at line 484 of file `tinkerio.py`.

### 8.11.2 Constructor & Destructor Documentation

`def forcebalance.tinkerio.BindingEnergy_TINKER.__init__( self, options, tgt_opts, forcefield )` Definition at line 487 of file `tinkerio.py`.

Here is the call graph for this function:



### 8.11.3 Member Function Documentation

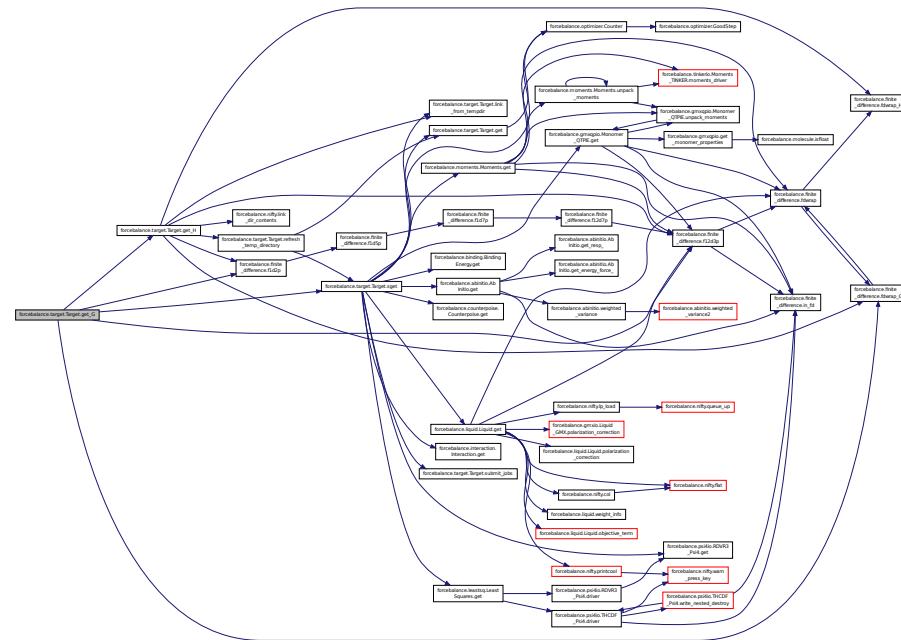
```
def forcebalance.binding.BindingEnergy.get ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 162 of file binding.py.
```

**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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



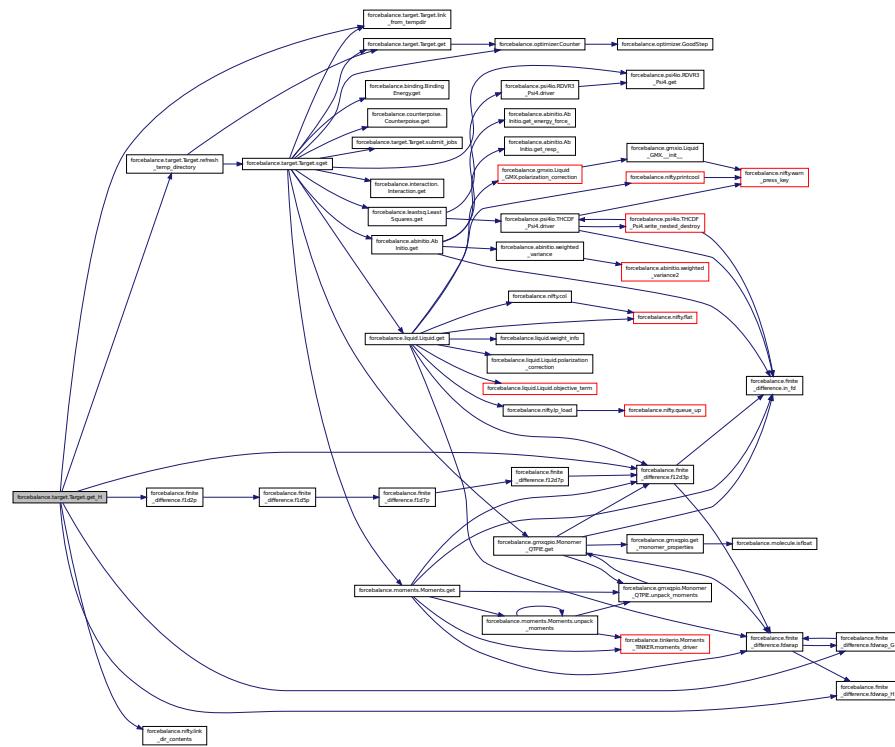
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

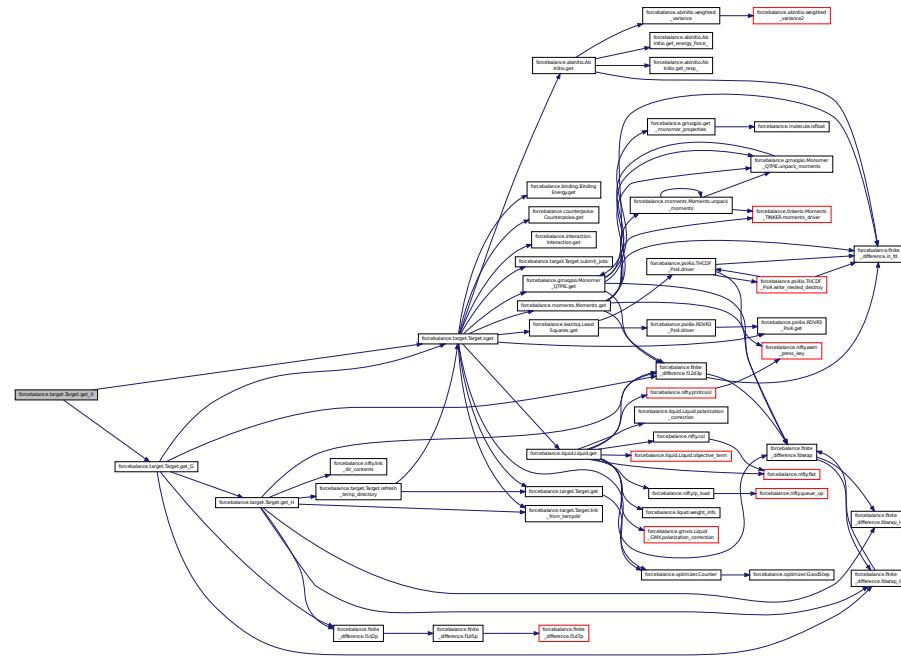
Here is the call graph for this function:



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

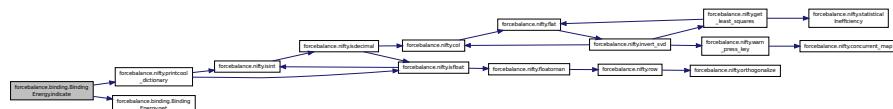
Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.function.function( self ) [inherited]** Definition at line 156 of file function.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

**def forcebalance.tinkerio.BindingEnergy\_TINKER.prepare\_temp\_directory( self, options, tgt\_opts ) [inherited]** Definition at line 491 of file tinkerio.py.

**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.

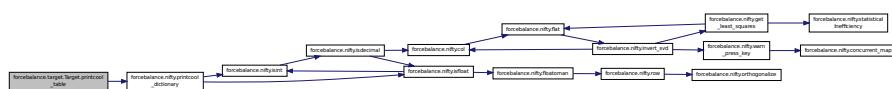
The input should be something like:

## 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 367 of file target.py.

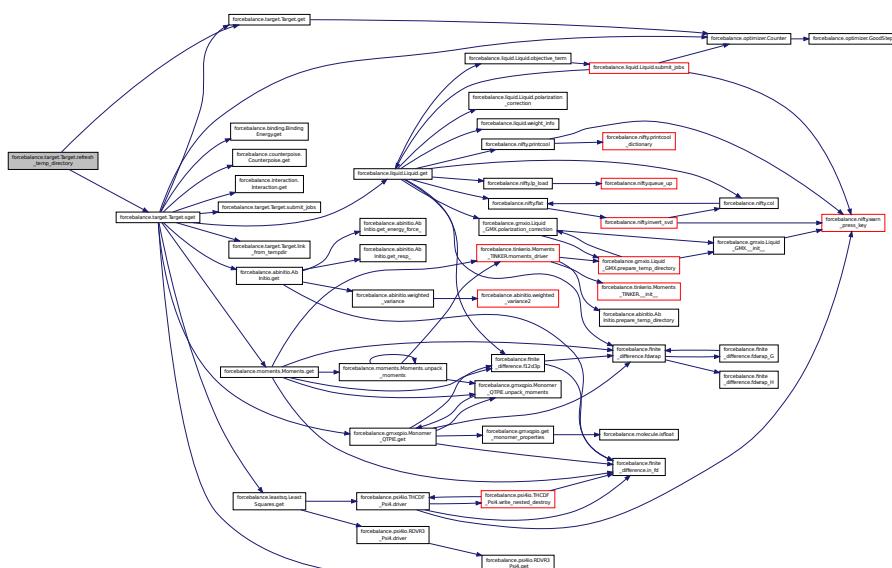
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



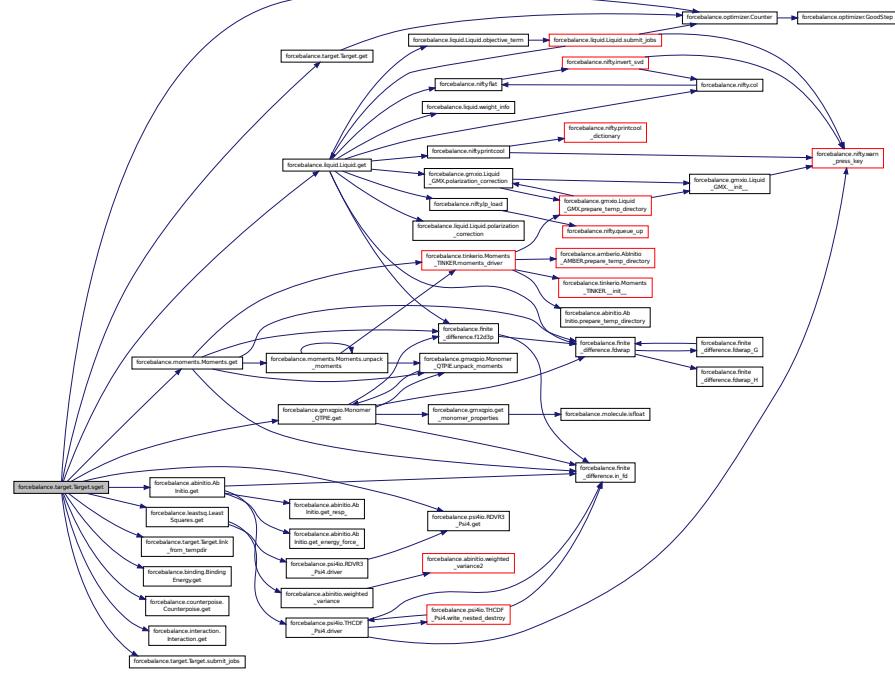
`def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited]` Definition at line 32 of file `_init_.py`.

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
    [inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py

Here is the call graph for this function:

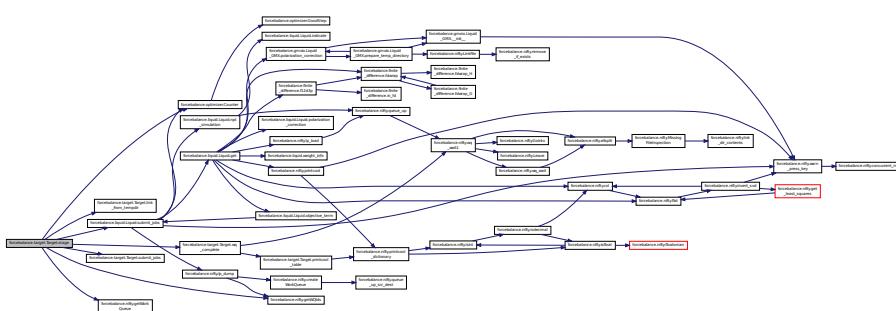


**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.

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

Definition at line 301 of file target.py.

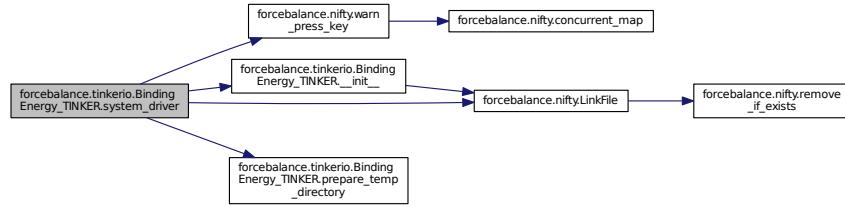
Here is the call graph for this function:



```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**def forcebalance.tinkerio.BindingEnergy\_TINKER.system\_driver ( self, sysname )** Definition at line 545 of file tinkerio.py.

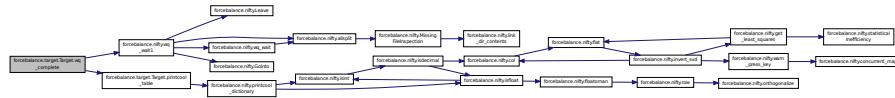
Here is the call graph for this function:



**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 331 of file target.py.

Here is the call graph for this function:



#### **8.11.4 Member Data Documentation**

**forcebalance.binding.BindingEnergy.energy\_part** [inherited] Definition at line 202 of file binding.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.

Definition at line 145 of file target.py.

**forcebalance.binding.BindingEnergy.inter\_opts** [inherited] Definition at line 129 of file binding.py.

**forcebalance.binding.BindingEnergy.objective** [inherited] Definition at line 226 of file binding.py.

**forcebalance.tinkerio.BindingEnergy\_TINKER.optprog** Definition at line 494 of file tinkerio.py.

**forcebalance.binding.BindingEnergy.PrintDict** [inherited] Definition at line 164 of file binding.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 29 of file `__init__.py`.

**forcebalance.binding.BindingEnergy.rmsd\_part** [inherited] Definition at line 200 of file binding.py.

**forcebalance.binding.BindingEnergy.RMSDDict** [inherited] Definition at line 165 of file binding.py.

**forcebalance.target.Target.rundir [inherited]** 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.

Definition at line 137 of file target.py.

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

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 141 of file target.py.

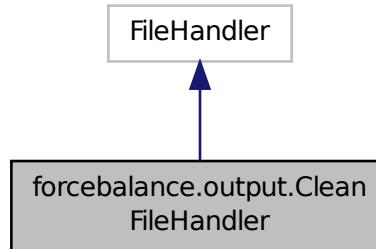
The documentation for this class was generated from the following file:

- [tinkerio.py](#)

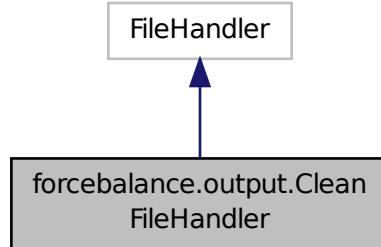
## 8.12 forcebalance.output.CleanFileHandler Class Reference

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](#)

#### 8.12.1 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.

#### 8.12.2 Member Function Documentation

**def forcebalance.output.CleanFileHandler.emit ( *self*, *record* )** Definition at line 70 of file output.py.

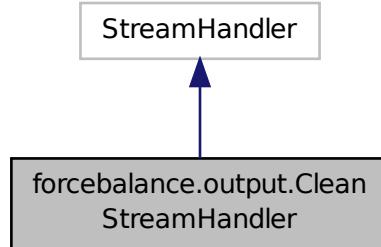
The documentation for this class was generated from the following file:

- [output.py](#)

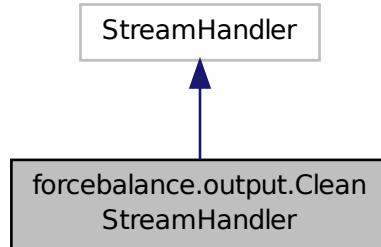
### 8.13 forcebalance.output.CleanStreamHandler Class Reference

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`

### 8.13.1 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`.

### 8.13.2 Constructor & Destructor Documentation

**def forcebalance.output.CleanStreamHandler.\_\_init\_\_( self, stream = sys.stdout )** Definition at line 57 of file `output.py`.

### 8.13.3 Member Function Documentation

```
def forcebalance.output.CleanStreamHandler.emit ( self, record )
```

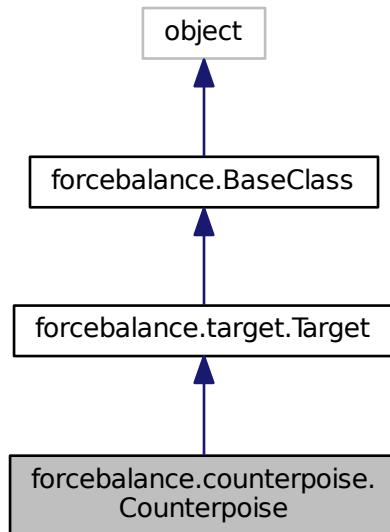
Definition at line 60 of file output.py.  
The documentation for this class was generated from the following file:

- [output.py](#)

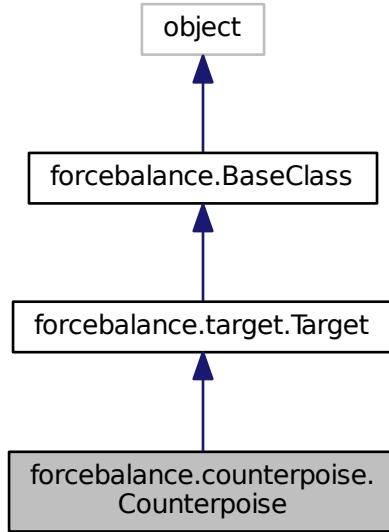
## 8.14 forcebalance.counterpoise.Counterpoise Class Reference

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 `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 `sget`  
    Stages the directory for the target, and then calls 'get'.
- 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 `set_option`

## Public Attributes

- `xyzs`  
*Number of snapshots.*
- `cpqm`  
*Counterpoise correction data.*
- `na`  
*Number of atoms.*
- `ns`
- `tempdir`  
*Root directory of the whole project.*
- `rundir`  
*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.*
- `PrintOptionDict`
- `verbose_options`

### 8.14.1 Detailed Description

Target subclass for matching the counterpoise correction.

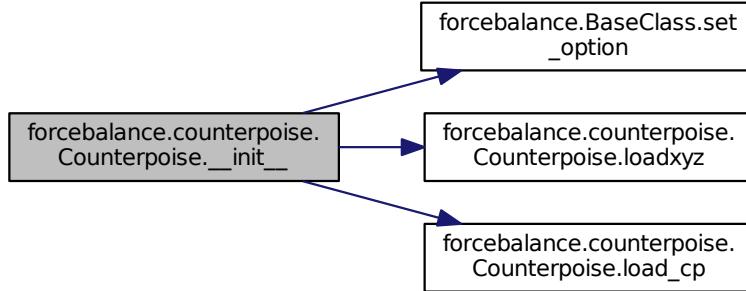
Definition at line 33 of file counterpoise.py.

### 8.14.2 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 37 of file counterpoise.py.

Here is the call graph for this function:



### 8.14.3 Member Function Documentation

**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

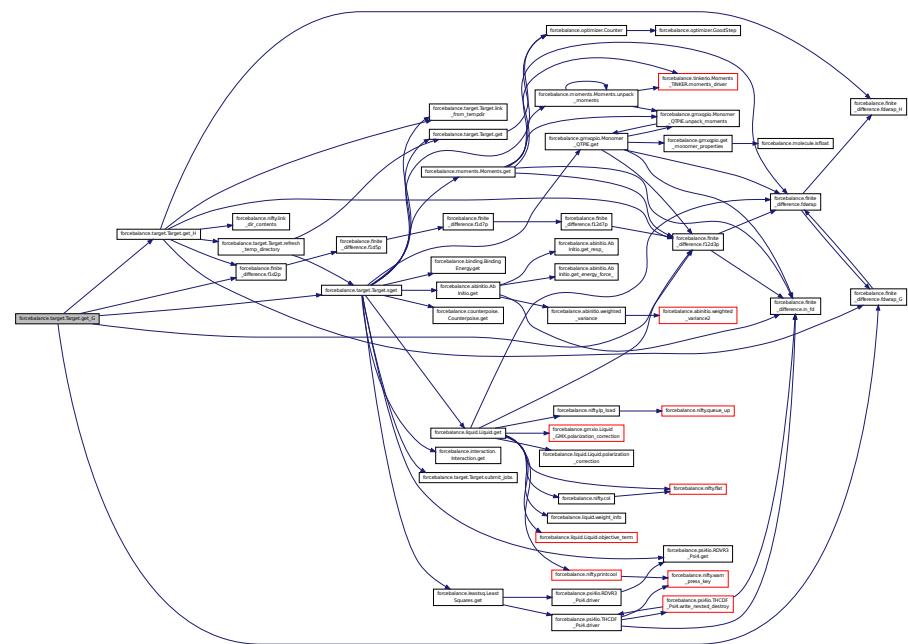
Definition at line 124 of file `counterpoise.py`.

**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.

Definition at line 172 of file `target.py`.

Here is the call graph for this function:



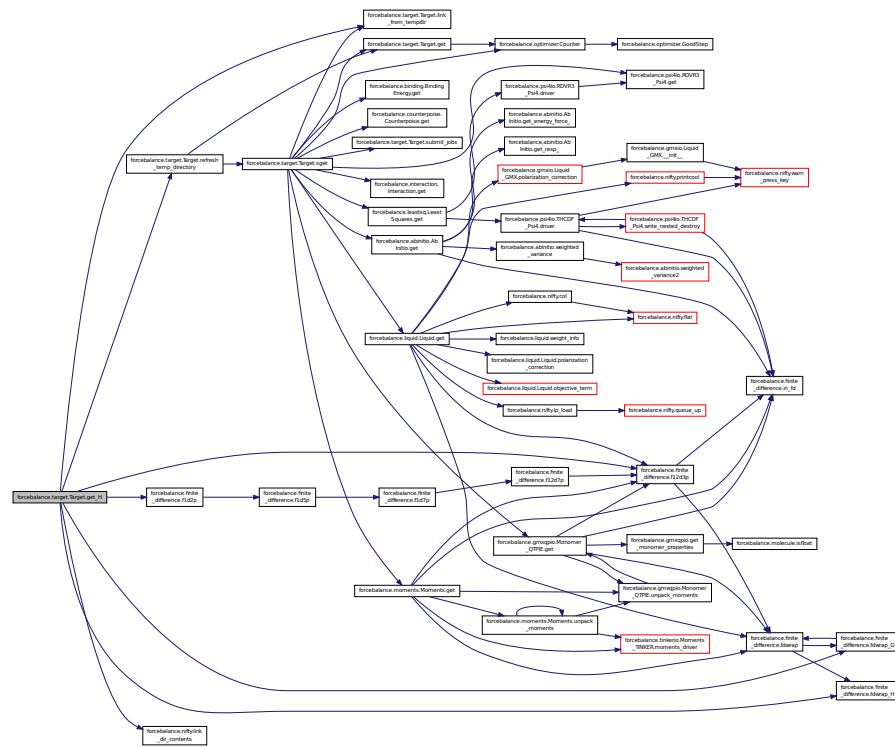
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

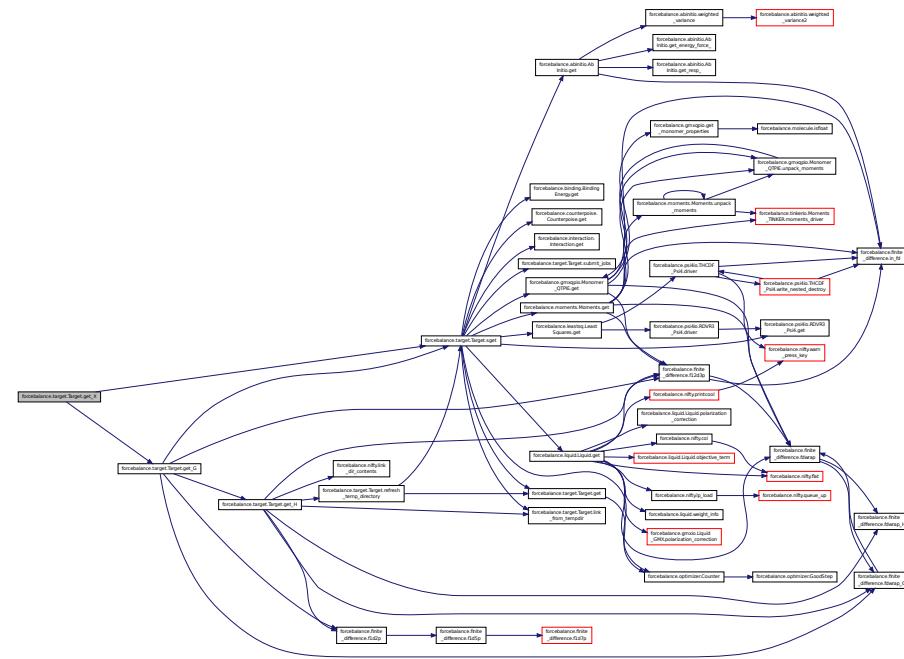
Here is the call graph for this function:



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

Definition at line 155 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 213 of file target.py.

**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 95 of file counterpoise.py.

**def forcebalance.counterpoise.Counterpoise.loadxyz ( self, fnm )** Parse an XYZ file which contains several xyz coordinates, and return their elements.

```

@param[in] fnm The input XYZ file name
@return elem A list of chemical elements in the XYZ file
@return 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 63 of file counterpoise.py.

```
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.

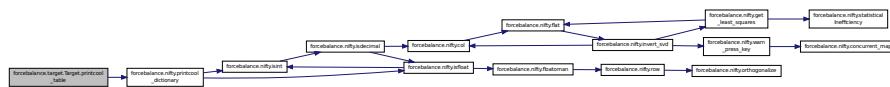
The input should be something like:

## Parameters

<code>data</code>	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).
<code>headings</code>	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.
<code>banner</code>	Optional heading line, which will be printed at the top in the title.
<code>footnote</code>	Optional footnote line, which will be printed at the bottom.

Definition at line 367 of file target.py.

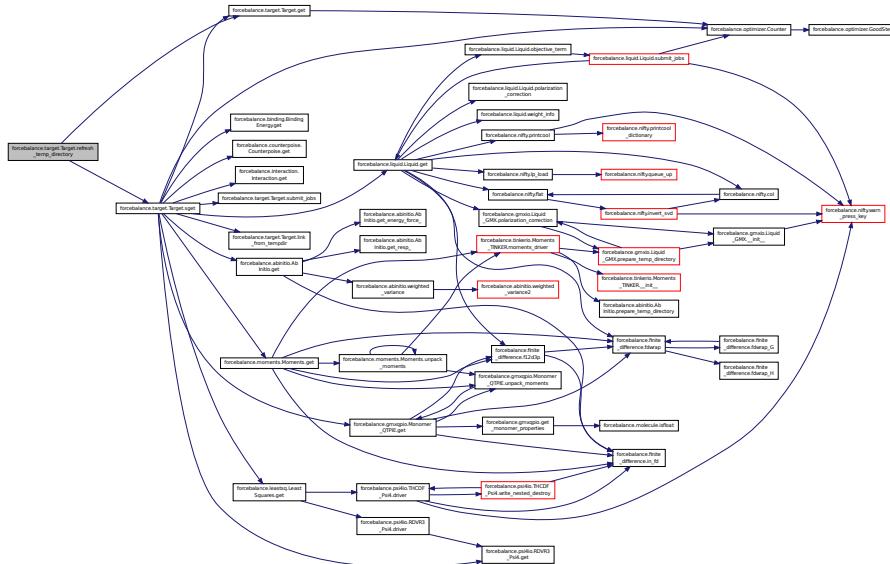
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



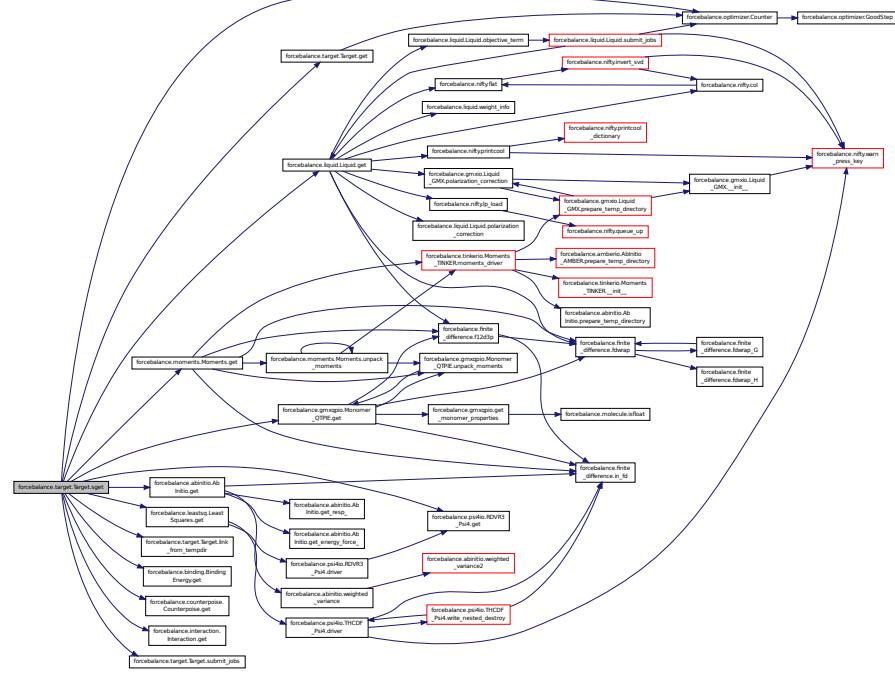
**def forcebalance.BaseClass.set\_option( self, in\_dict, src.key, dest.key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 32 of file \_\_init\_\_.py.

**def forcebalance.target.Target.sget( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited]** Stages the directory for the target, and then calls 'get'.

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

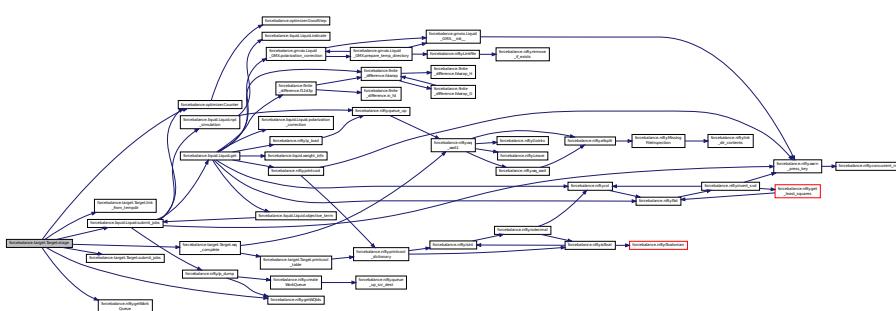


**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

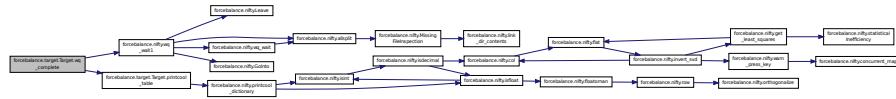


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### 8.14.4 Member Data Documentation

**forcebalance.counterpoise.Counterpoise.cpqm** [Counterpoise](#) correction data.

Definition at line 53 of file counterpoise.py.

**forcebalance.target.Target.FF** [[inherited](#)] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.target.Target.gct** [[inherited](#)] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [[inherited](#)] Counts how often the Hessian was computed.

Definition at line 145 of file target.py.

**forcebalance.counterpoise.Counterpoise.na** Number of atoms.

Definition at line 76 of file counterpoise.py.

**forcebalance.counterpoise.Counterpoise.ns** Definition at line 89 of file counterpoise.py.

**forcebalance.BaseClass.PrintOptionDict** [[inherited](#)] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.target.Target.rundir** [[inherited](#)] 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.

Definition at line 137 of file target.py.

**forcebalance.target.Target.tempdir** [[inherited](#)] Root directory of the whole project.

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options** [[inherited](#)] Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.target.Target.xct** [[inherited](#)] Counts how often the objective function was computed.

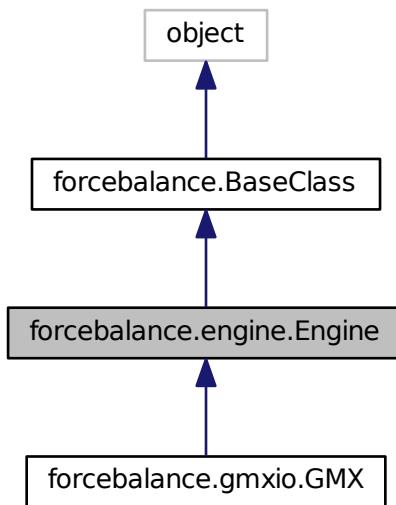
Definition at line 141 of file target.py.

**forcebalance.counterpoise.Counterpoise.xyzs** Number of snapshots.  
XYZ elements and coordinates  
Definition at line 51 of file counterpoise.py.  
The documentation for this class was generated from the following file:  
• [counterpoise.py](#)

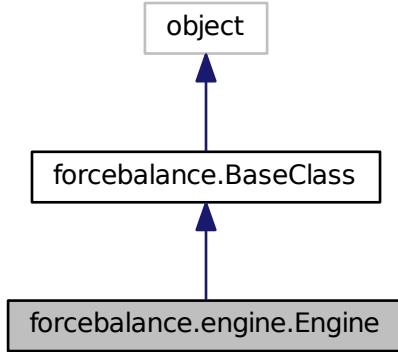
## 8.15 forcebalance.engine.Engine Class Reference

Base class for all engines.

Inheritance diagram for forcebalance.engine.Engine:



Collaboration diagram for forcebalance.engine.Engine:



## Public Member Functions

- def `__init__`
- def `prepare`
- def `evaluate_snapshots`  
*Evaluate properties over a collection of snapshots.*
- def `evaluate_optimized`  
*Evaluate properties on the optimized geometry.*
- def `set_option`

## Public Attributes

- `name`
- `target`
- `root`
- `PrintOptionDict`
- `verbose_options`

### 8.15.1 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.

The `Engine` implements methods that execute operations such as:

- Input: trajectory object
- Return: energy over trajectory
- Return: energy and force over trajectory

- Return: electrostatic potential over trajectory
- How about:
- `evaluate_snapshots(Molecule, Energy=True, Force=True, ESP=True)` where all information passed in belongs in the Molecule object. :)
- Return a dictionary
- Input: molecular geometry,
- Return: optimized geometry
- Return: vibrational modes at optimized geometry
- Return: multipole moments at optimized geometry
- `evaluate_optimized(Molecule, Energy=True, Frequencies=True, Moments=True)`
- Engine objects may be initialized using a molecule object and a force field object.

## 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 60 of file engine.py.

## 8.15.2 Constructor & Destructor Documentation

```
def forcebalance.engine.Engine.__init__( self, name = "engine", kwargs ) Definition at line 63 of file engine.py.
```

## 8.15.3 Member Function Documentation

```
def forcebalance.engine.Engine.evaluate_optimized( self, M ) Evaluate properties on the optimized geometry.  
Definition at line 86 of file engine.py.
```

```
def forcebalance.engine.Engine.evaluate_snapshots( self, M ) Evaluate properties over a collection of snapshots.
```

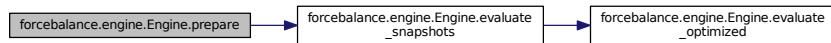
Definition at line 80 of file engine.py.

Here is the call graph for this function:



```
def forcebalance.engine.Engine.prepare( ) Definition at line 74 of file engine.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 32 of file __init__.py.
```

#### 8.15.4 Member Data Documentation

**forcebalance.engine.Engine.name** Definition at line 65 of file engine.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.engine.Engine.root** Definition at line 68 of file engine.py.

**forcebalance.engine.Engine.target** Definition at line 67 of file engine.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 30 of file \_\_init\_\_.py.

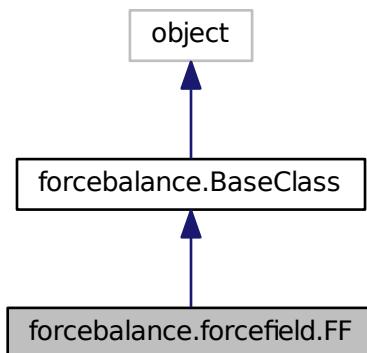
The documentation for this class was generated from the following file:

- [engine.py](#)

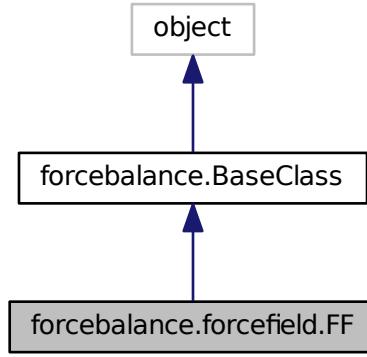
## 8.16 forcebalance.forcefield.FF Class Reference

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`  
*Parse a text force field and create several important instance variables.*
- def `addff_xml`  
*Parse an XML force field file and create 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 `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 `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).*
- `parmdestroy_save`
- `linedestroy_this`
- `parmdestroy_this`
- `tinkerprm`
- `openmmxml`
- `qmap`
- `qid`

- qid2
- PrintOptionDict
- verbose\_options

### 8.16.1 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 196 of file forcefield.py.

### 8.16.2 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 204 of file forcefield.py.

Here is the call graph for this function:



### 8.16.3 Member Function Documentation

**def forcebalance.forcefield.FF.\_\_eq\_\_( self, other )** Definition at line 1145 of file forcefield.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, <tt> fname force\_field.xml:openmm </tt> 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 <tt> parameter identifier -> index in parameter vector </tt>
- 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 <tt> unique parameter identifier </tt>. 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 # PARM 3 (where # denotes a comment, the word PARM 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.

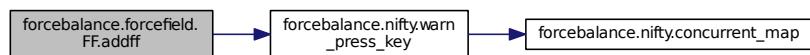
I don't think the multiplier actually works for analytic derivatives unless the interaction calculator knows the multiplier as well. I'm sure I can make this work in the future if necessary.

## Parameters

in	ffname	Name of the force field file
----	--------	------------------------------

Definition at line 395 of file forcefield.py.

Here is the call graph for this function:



**def forcebalance.forcefield.FF.addff\_txt ( self, fname, fftype )** 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 'PARM' 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 'PARM' is encountered:

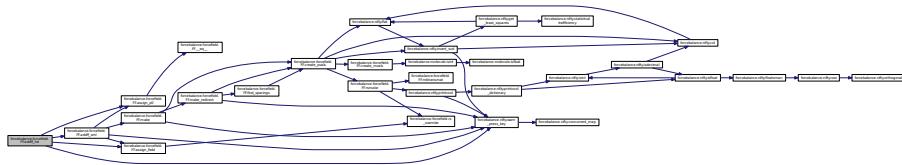
- Store the physical parameter value (this is permanent; it's the original value)

- 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 465 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.

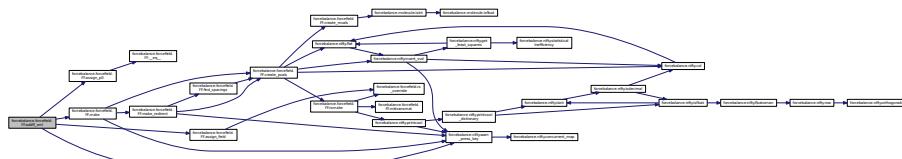
It's hard to specify precisely the location in an XML file to change a force field parameter. I can create a list of tree elements (essentially pointers to elements within a tree), but this method breaks down when I copy the tree because I have no way to refer to the copied tree elements. Fortunately, 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 573 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.

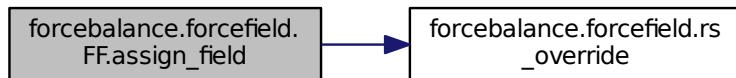
#### Parameters

in	idx	The index of the parameter.
----	-----	-----------------------------

in	<i>fnm</i>	The file name of the parameter field.
in	<i>ln</i>	The line number within the file (or the node index in the flattened xml)
in	<i>pfld</i>	The field within the line (or the name of the attribute in the xml)
in	<i>mult</i>	The multiplier (this is usually 1.0)

Definition at line 1139 of file forcefield.py.

Here is the call graph for this function:



**def forcebalance.forcefield.FF.assign\_p0 ( self, idx, val )** Assign physical parameter values to the 'pvals0' array.

Parameters

in	<i>idx</i>	The index to which we assign the parameter value.
in	<i>val</i>	The parameter value to be inserted.

Definition at line 1121 of file forcefield.py.

Here is the call graph for this function:



**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 854 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

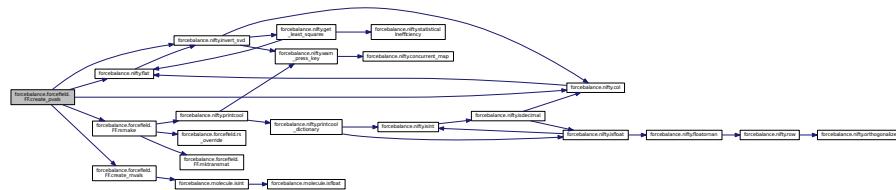
in	mvals	The mathematical parameters
----	-------	-----------------------------

Returns

pvals The physical parameters

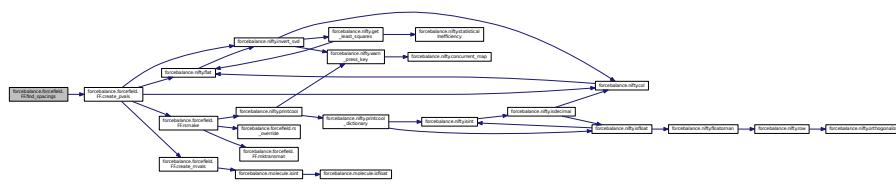
Definition at line 819 of file forcefield.py.

Here is the call graph for this function:



**def forcebalance.forcefield.FF.find\_spacings ( self )** Definition at line 774 of file forcefield.py.

Here is the call graph for this function:

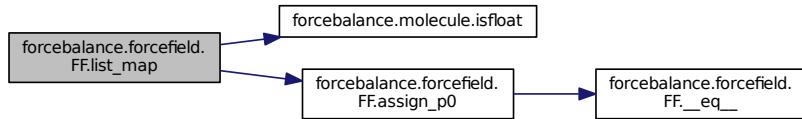


**def forcebalance.forcefield.FF.list\_map ( self )** Create the plist, which is like a reversed version of the parameter map.

More convenient for printing.

Definition at line 1097 of file forcefield.py.

Here is the call graph for this function:



```
def forcebalance.forcefield.FF.make( self, vals, 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.

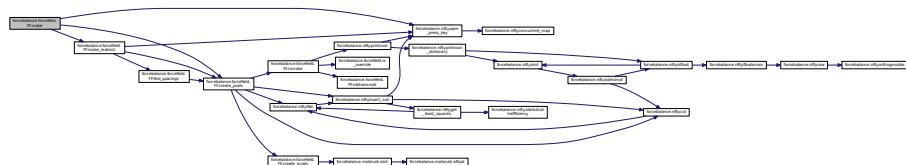
It does NOT store the mathematical parameters in the class state (since we can only hold one set of parameters).

## 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.

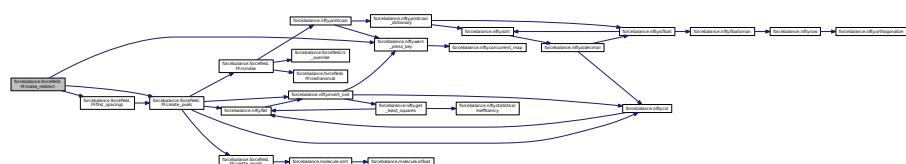
Definition at line 621 of file forcefield.py.

Here is the call graph for this function:



**def forcebalance.forcefield.FF.make\_redirect ( self, mvals )** Definition at line 733 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.

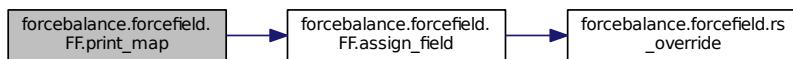
The AMOEBA selection of charge depends not only on the atom type, but what that atom is bonded to.

Definition at line 947 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 1109 of file forcefield.py.

Here is the call graph for this function:



```
def forcebalance.forcefield.FF.rsmake ( self, printfac = True ) Create the rescaling factors for the coordinate transformation in parameter space.
```

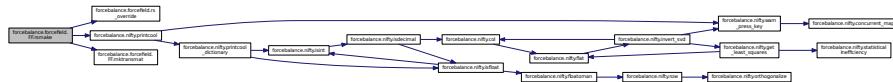
The proper choice of rescaling factors (read: prior widths in maximum likelihood analysis) is still a black art. This is a topic of current research.

**Todo** Pass in rsfactors through the input file

```
@param[in] printfac List for printing out the rescaling factors
```

Definition at line 871 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 32 of file __init__.py.
```

#### 8.16.4 Member Data Documentation

**forcebalance.forcefield.FF.atomnames** A list of atom names (this is new, for ESP fitting)  
Definition at line 265 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 257 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 275 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. 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 237 of file forcefield.py.

**forcebalance.forcefield.FF.ffdata\_isxml** Definition at line 238 of file forcefield.py.

**forcebalance.forcefield.FF.FFMolecules** Definition at line 288 of file forcefield.py.

**forcebalance.forcefield.FF.linedestroy\_save** Destruction dictionary (experimental).  
Definition at line 316 of file forcefield.py.

**forcebalance.forcefield.FF.linedestroy\_this** Definition at line 318 of file forcefield.py.

**forcebalance.forcefield.FF.map** The mapping of interaction type -> parameter number.  
Definition at line 240 of file forcefield.py.

**forcebalance.forcefield.FF.np** The total number of parameters.  
Definition at line 259 of file forcefield.py.

**forcebalance.forcefield.FF.openmmxml** Definition at line 411 of file forcefield.py.

**forcebalance.forcefield.FF.parmdestroy\_save** Definition at line 317 of file forcefield.py.

**forcebalance.forcefield.FF.parmdestroy\_this** Definition at line 319 of file forcefield.py.

**forcebalance.forcefield.FF.patoms** A listing of parameter number -> atoms involved.  
Definition at line 244 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 249 of file forcefield.py.

**forcebalance.forcefield.FF.plist** The listing of parameter number -> interaction types.  
Definition at line 242 of file forcefield.py.

**forcebalance.BaseClass.PrintOptionDict** [**inherited**] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.forcefield.FF.pvals0** Initial value of physical parameters.  
Definition at line 261 of file forcefield.py.

**forcebalance.forcefield.FF.qid** Definition at line 949 of file forcefield.py.

**forcebalance.forcefield.FF.qid2** Definition at line 950 of file forcefield.py.

**forcebalance.forcefield.FF.qmap** Definition at line 948 of file forcefield.py.

**forcebalance.forcefield.FF.Readers** A dictionary of force field reader classes.  
Definition at line 263 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 314 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 251 of file forcefield.py.

**forcebalance.forcefield.FF.tinkerprm** Definition at line 404 of file forcefield.py.

**forcebalance.forcefield.FF.tm** The transformation matrix for mathematical -> physical parameters.  
Definition at line 253 of file forcefield.py.

**forcebalance.forcefield.FF.tml** The transpose of the transformation matrix.  
Definition at line 255 of file forcefield.py.

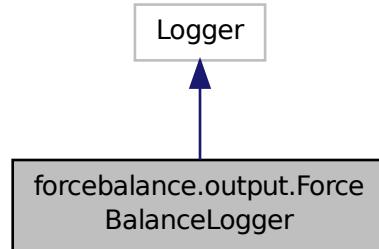
**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 30 of file \_\_init\_\_.py.  
The documentation for this class was generated from the following file:

- [forcefield.py](#)

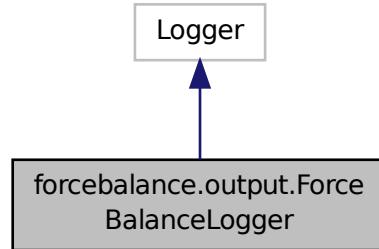
## 8.17 forcebalance.output.ForceBalanceLogger Class Reference

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`

#### 8.17.1 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.

### 8.17.2 Constructor & Destructor Documentation

**def forcebalance.output.ForceBalanceLogger.\_\_init\_\_ ( self, name )** Definition at line 11 of file output.py.

### 8.17.3 Member Function Documentation

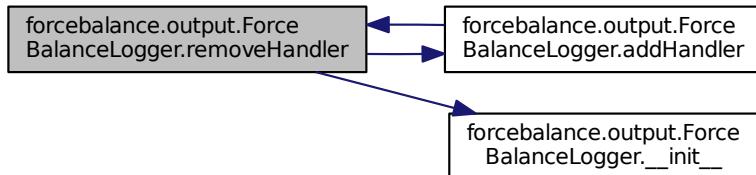
**def forcebalance.output.ForceBalanceLogger.addHandler ( self, hdlr )** Definition at line 17 of file output.py.

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:



### 8.17.4 Member Data Documentation

**forcebalance.output.ForceBalanceLogger.defaultHandler** Definition at line 13 of file output.py.

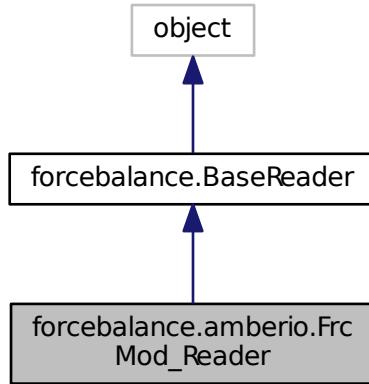
The documentation for this class was generated from the following file:

- [output.py](#)

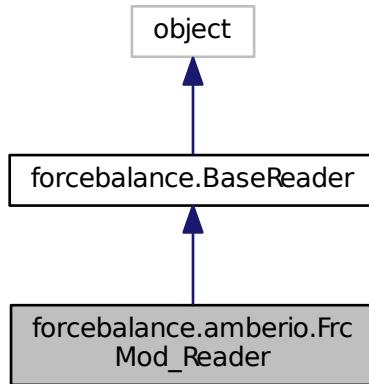
## 8.18 forcebalance.amberio.FrcMod\_Reader Class Reference

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**

### 8.18.1 Detailed Description

Finite state machine for parsing FrcMod force field file.  
Definition at line 99 of file amberio.py.

### 8.18.2 Constructor & Destructor Documentation

**def forcebalance.amberio.FrcMod\_Reader.\_\_init\_\_ ( self, fnm )** Definition at line 101 of file amberio.py.

### 8.18.3 Member Function Documentation

**def forcebalance.BaseReader.build\_pid ( self, pfld ) [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 107 of file \_\_init\_\_.py.

**def forcebalance.amberio.FrcMod\_Reader.feed ( self, line )** Definition at line 119 of file amberio.py.

**def forcebalance.amberio.FrcMod\_Reader.Split ( self, line )** Definition at line 113 of file amberio.py.  
Here is the call graph for this function:



**def forcebalance.amberio.FrcMod.Reader.Whites ( self, line )** Definition at line 116 of file amberio.py.

Here is the call graph for this function:



#### 8.18.4 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 80 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 67 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 77 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules [inherited]** Definition at line 79 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.

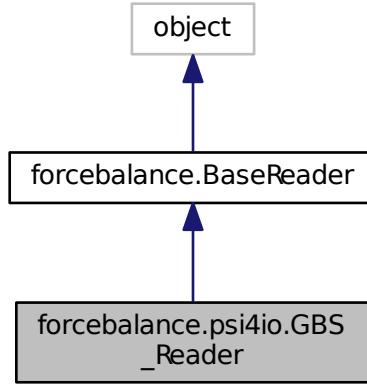
The documentation for this class was generated from the following file:

- [amberio.py](#)

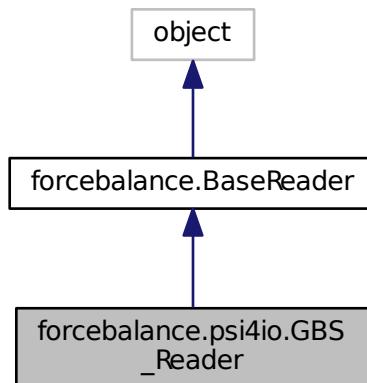
### 8.19 forcebalance.psi4io.GBS\_Reader Class Reference

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](#)

### 8.19.1 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.

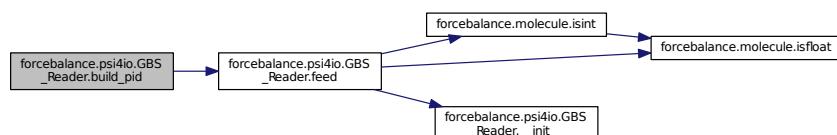
### 8.19.2 Constructor & Destructor Documentation

`def forcebalance.psi4io.GBS_Reader.__init__ ( self, fnm = None )` Definition at line 37 of file psi4io.py.

### 8.19.3 Member Function Documentation

`def forcebalance.psi4io.GBS_Reader.build_pid ( self, pfld )` Definition at line 48 of file psi4io.py.

Here is the call graph for this function:



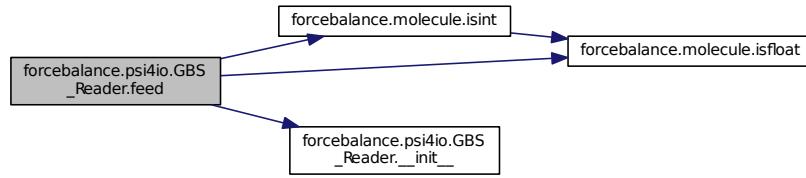
`def forcebalance.psi4io.GBS_Reader.feed ( self, line, linindep = False )` Feed in a line.

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 88 of file \_\_init\_\_.py.

Here is the call graph for this function:



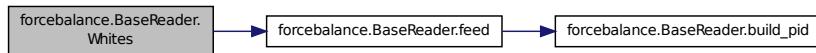
**def forcebalance.BaseReader.Split ( self, line ) [inherited]** Definition at line 82 of file \_\_init\_\_.py.

Here is the call graph for this function:



**def forcebalance.BaseReader.Whites ( self, line ) [inherited]** Definition at line 85 of file \_\_init\_\_.py.

Here is the call graph for this function:



#### 8.19.4 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 80 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 68 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 67 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 77 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules** [inherited] Definition at line 79 of file \_\_init\_\_.py.

**forcebalance.BaseReader.pdict** [inherited] Definition at line 70 of file \_\_init\_\_.py.

**forcebalance.BaseReader.suffix** [inherited] Definition at line 69 of file \_\_init\_\_.py.

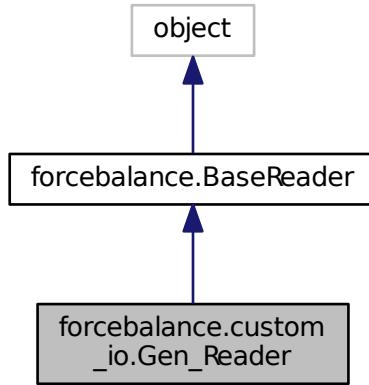
The documentation for this class was generated from the following file:

- [psi4io.py](#)

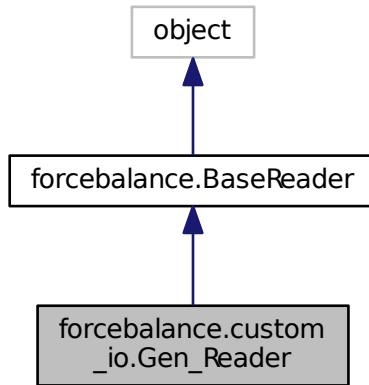
## 8.20 forcebalance.custom\_io.Gen\_Reader Class Reference

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**

### 8.20.1 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.

### 8.20.2 Constructor & Destructor Documentation

**def forcebalance.custom.io.Gen\_Reader.\_\_init\_\_ ( self, fnm )** Definition at line 43 of file custom.io.py.

### 8.20.3 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 107 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 82 of file \_\_init\_\_.py.

Here is the call graph for this function:



```
def forcebalance.BaseReader.Whites( self, line ) [inherited] Definition at line 85 of file __init__.py.
```

Here is the call graph for this function:



#### 8.20.4 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 72 of file \_\_init\_\_.py.

**forcebalance.BaseReader.AtomTypes [inherited]** Definition at line 80 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 67 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 77 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules [inherited]** Definition at line 79 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.

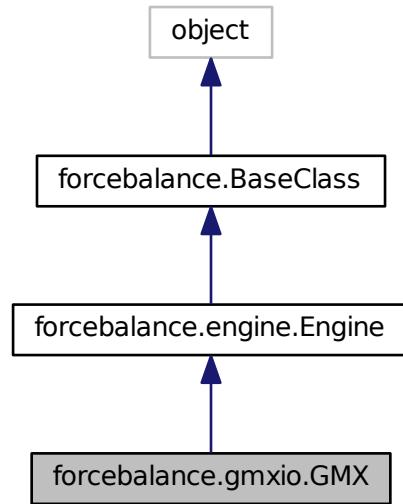
The documentation for this class was generated from the following file:

- [custom\\_io.py](#)

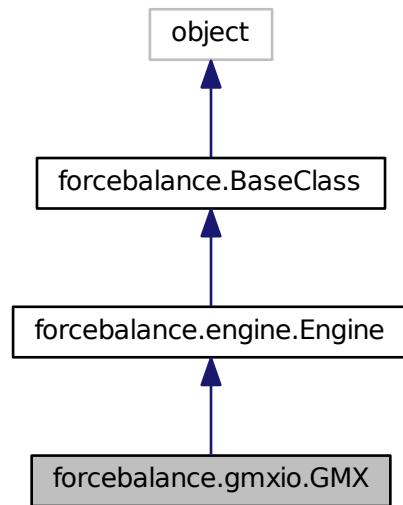
### 8.21 forcebalance.gmxio.GMX Class Reference

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_`
- def `callgmx`

*Call GROMACS; prepend the gmxpath to the call to the GROMACS program.*
- def `prepare`

*Prepare the calculation.*
- def `energy_force_driver`

*Computes the energy and force using GROMACS for a single snapshot.*
- def `energy_force_driver_all`

*Computes the energy and force using GROMACS over a trajectory.*
- def `generate_vsites_positions`
- def `prepare`
- def `evaluate_snapshots`

*Evaluate properties over a collection of snapshots.*
- def `evaluate_optimized`

*Evaluate properties on the optimized geometry.*
- def `set_option`

## Public Attributes

- `gmxsuffix`

*Disable some optimizations.*
- `gmxpath`

*The directory containing GROMACS executables (e.g.*
- `srcdir`

*Attempt to determine file names of .gro, .top, and .mdp files.*
- `top`
- `mdp`
- `mol`
- `AtomMask`

*First move into the temp directory if specified by the input arguments.*
- `AtomLists`
- `name`
- `target`
- `root`
- `PrintOptionDict`
- `verbose_options`

### 8.21.1 Detailed Description

Derived from Engine object for carrying out general purpose GROMACS calculations.

Definition at line 461 of file gmxio.py.

### **8.21.2 Constructor & Destructor Documentation**

**def forcebalance.gmxio.GMX.\_\_init\_\_( self, name = "gmx", kwargs )** Definition at line 462 of file gmxio.py.  
Here is the call graph for this function:



### 8.21.3 Member Function Documentation

`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 511 of file gmxio.py.

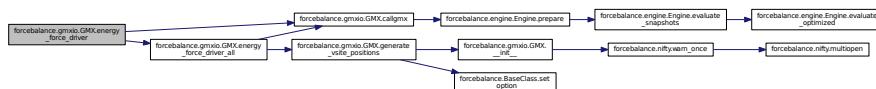
Here is the call graph for this function:



**def forcebalance.gmxio.GMX.energy\_force\_driver ( self, shot )** Computes the energy and force using GROMACS for a single snapshot.

Definition at line 583 of file gmxio.py.

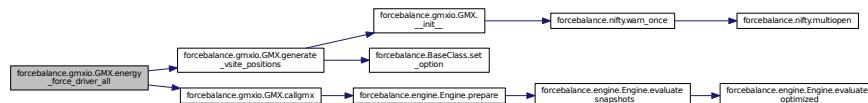
Here is the call graph for this function:



**def forcebalance.gmxio.GMX.energy\_force\_driver\_all ( self )** Computes the energy and force using GROMACS over a trajectory.

Definition at line 605 of file qmxic.py.

Here is the call graph for this function:



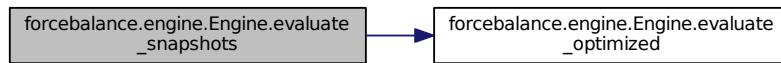
```
def forcebalance.engine.Engine.evaluate_optimized ( self, M ) [inherited] Evaluate properties on the optimized geometry.
```

Definition at line 86 of file engine.py.

```
def forcebalance.engine.Engine.evaluate_snapshots ( self, M ) [inherited] Evaluate properties over a collection of snapshots.
```

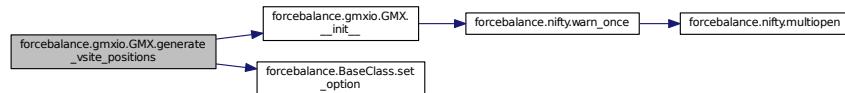
Definition at line 80 of file engine.py.

Here is the call graph for this function:



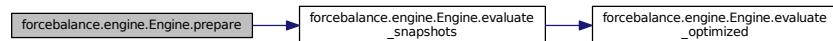
```
def forcebalance.gmxio.GMX.generate_vsite_positions ( self ) Definition at line 627 of file gmxio.py.
```

Here is the call graph for this function:



```
def forcebalance.engine.Engine.prepare ( ) [inherited] Definition at line 74 of file engine.py.
```

Here is the call graph for this function:

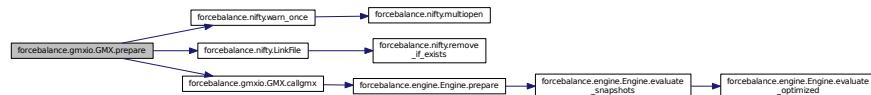


```
def forcebalance.gmxio.GMX.prepare ( self ) Prepare the calculation.
```

Write conformation to the temporary directory. Read the topology.

Definition at line 523 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 32 of file __init__.py.
```

#### 8.21.4 Member Data Documentation

**forcebalance.gmxio.GMX.AtomLists** Definition at line 554 of file gmxio.py.

**forcebalance.gmxio.GMX.AtomMask** First move into the temp directory if specified by the input arguments.

Link files into the temp directory because it's good for reproducibility. Write the appropriate coordinate files. Call grompp followed by gmxdump to read the trajectory

Definition at line 553 of file gmxio.py.

**forcebalance.gmxio.GMX.gmxpath** The directory containing GROMACS executables (e.g. mdrun)

Definition at line 480 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 473 of file gmxio.py.

**forcebalance.gmxio.GMX.mdp** Definition at line 498 of file gmxio.py.

**forcebalance.gmxio.GMX.mol** Definition at line 500 of file gmxio.py.

**forcebalance.engine.Engine.name** [*inherited*] Definition at line 65 of file engine.py.

**forcebalance.BaseClass.PrintOptionDict** [*inherited*] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.engine.Engine.root** [*inherited*] Definition at line 68 of file engine.py.

**forcebalance.gmxio.GMX.srkdir** Attempt to determine file names of .gro, .top, and .mdp files.

Definition at line 492 of file gmxio.py.

**forcebalance.engine.Engine.target** [*inherited*] Definition at line 67 of file engine.py.

**forcebalance.gmxio.GMX.top** Definition at line 497 of file gmxio.py.

**forcebalance.BaseClass.verbose\_options** [*inherited*] Definition at line 30 of file \_\_init\_\_.py.

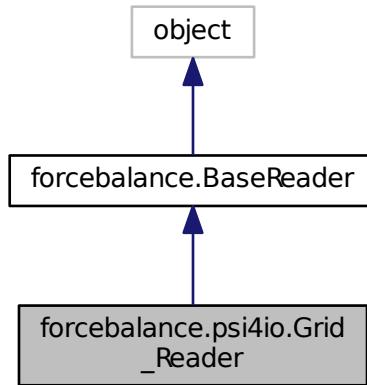
The documentation for this class was generated from the following file:

- [gmxio.py](#)

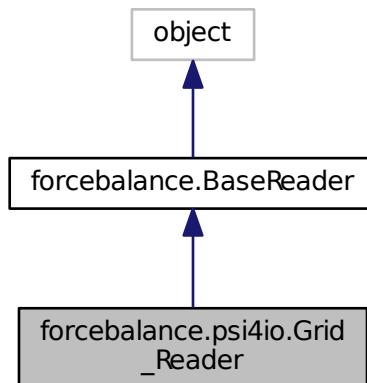
## 8.22 forcebalance.psi4io.Grid\_Reader Class Reference

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`

### 8.22.1 Detailed Description

Finite state machine for parsing DVR grid files.

Definition at line 247 of file psi4io.py.

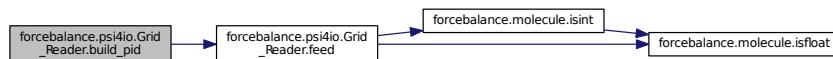
### 8.22.2 Constructor & Destructor Documentation

`def forcebalance.psi4io.Grid_Reader.__init__ ( self, fnm = None )` Definition at line 249 of file psi4io.py.

### 8.22.3 Member Function Documentation

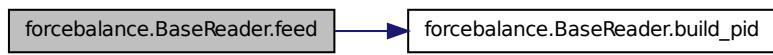
`def forcebalance.psi4io.Grid_Reader.build_pid ( self, pfld )` Definition at line 255 of file psi4io.py.

Here is the call graph for this function:



`def forcebalance.BaseReader.feed ( self, line ) [inherited]` Definition at line 88 of file \_\_init\_\_.py.

Here is the call graph for this function:



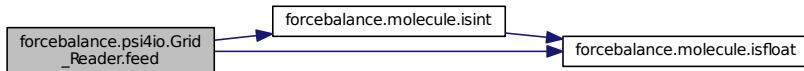
`def forcebalance.psi4io.Grid_Reader.feed ( self, line, linindep = False )` Feed in a line.

#### Parameters

in	line	The line of data
----	------	------------------

Definition at line 270 of file psi4io.py.

Here is the call graph for this function:



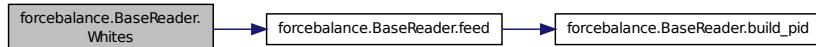
**def forcebalance.BaseReader.Split ( self, line ) [inherited]** Definition at line 82 of file \_\_init\_\_.py.

Here is the call graph for this function:



**def forcebalance.BaseReader.Whites ( self, line ) [inherited]** Definition at line 85 of file \_\_init\_\_.py.

Here is the call graph for this function:



#### 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 72 of file \_\_init\_\_.py.

**forcebalance.BaseReader.AtomTypes [inherited]** Definition at line 80 of file \_\_init\_\_.py.

**forcebalance.psi4io.Grid\_Reader.element** Definition at line 251 of file psi4io.py.

**forcebalance.psi4io.Grid\_Reader.isdata** Definition at line 281 of file psi4io.py.

**forcebalance.BaseReader.itype [inherited]** Definition at line 68 of file \_\_init\_\_.py.

**forcebalance.BaseReader.in [inherited]** Definition at line 67 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 77 of file `__init__.py`.

**forcebalance.BaseReader.Molecules** [inherited] Definition at line 79 of file `__init__.py`.

**forcebalance.BaseReader.pdict** [inherited] Definition at line 70 of file `__init__.py`.

**forcebalance.psi4io.Grid\_Reader.point** Definition at line 252 of file `psi4io.py`.

**forcebalance.psi4io.Grid\_Reader.radius** Definition at line 253 of file `psi4io.py`.

**forcebalance.BaseReader.suffix** [inherited] Definition at line 69 of file `__init__.py`.

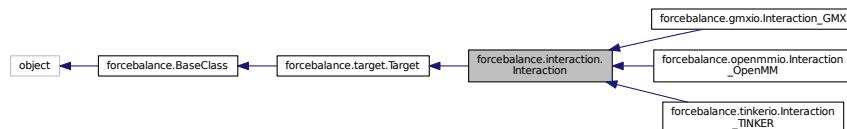
The documentation for this class was generated from the following file:

- [psi4io.py](#)

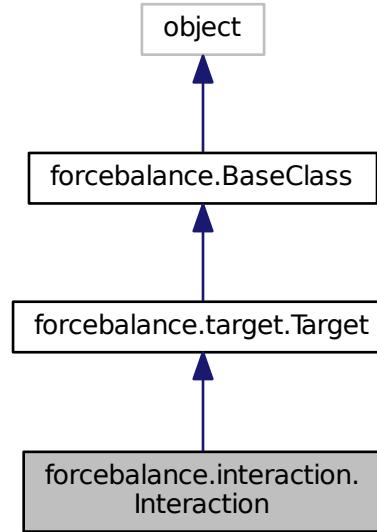
## 8.23 forcebalance.interaction.Interaction Class Reference

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 `prepare_temp_directory`

*Prepare the temporary directory, by default does nothing.*
- def `indicate`
- def `get`

*Evaluate objective function.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- 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 `sget`

*Stages the directory for the target, and then calls 'get'.*
- 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 `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`  
*Qualitative Indicator: average energy error (in kJ/mol)*
- `e_err_pct`
- `ns`  
*Read in the trajectory file.*
- `traj`
- `divisor`  
*Read in the reference data.*
- `prefactor`
- `weight`
- `emm`
- `objective`
- `tempdir`  
*Root directory of the whole project.*
- `rundir`  
*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.*
- `PrintOptionDict`
- `verbose_options`

### **8.23.1 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)

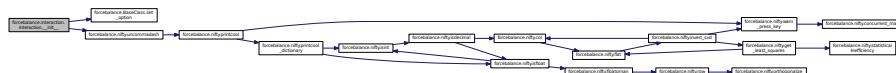
This subclass contains the 'get' method for building the objective function from any simulation software (a driver to run the program and read output is still required).

Definition at line 35 of file interaction.py.

### 8.23.2 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:



### 8.23.3 Member Function Documentation

```
def forcebalance.interaction.Interaction.get ( self, mvals, AGrad = False, AHess = False ) Evaluate  
objective function.
```

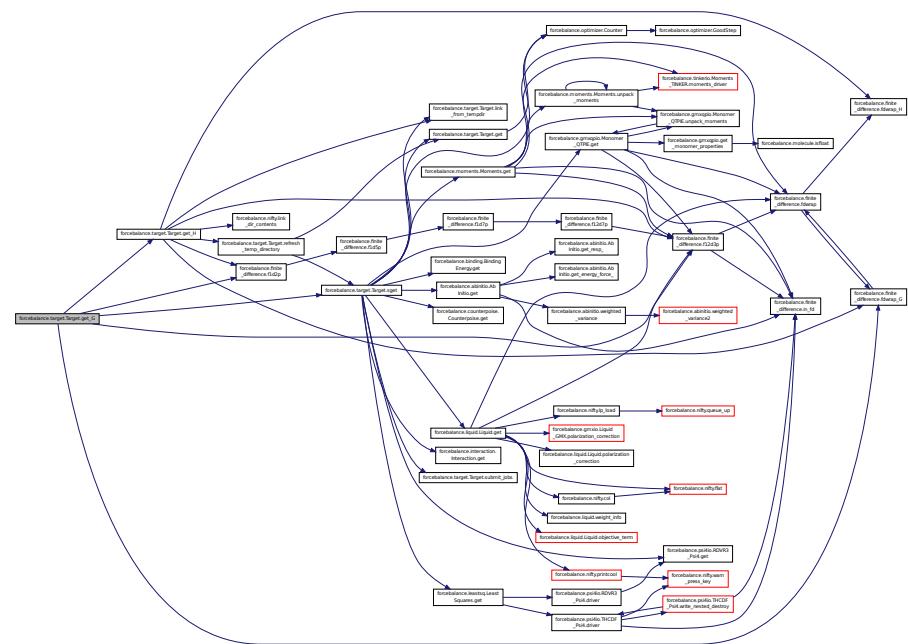
Definition at line 163 of file interaction.py.

**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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



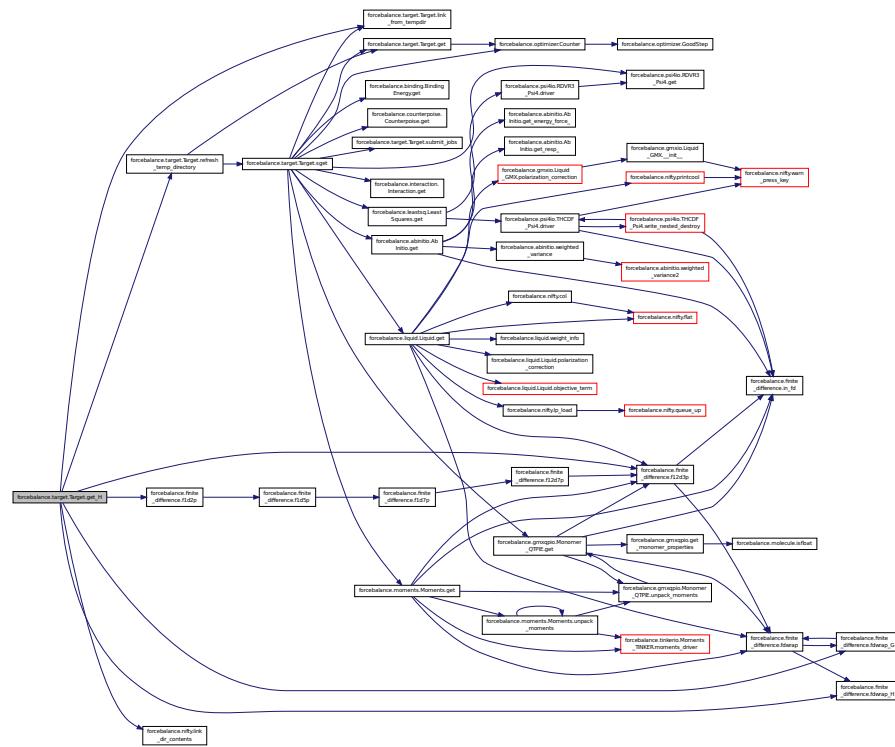
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

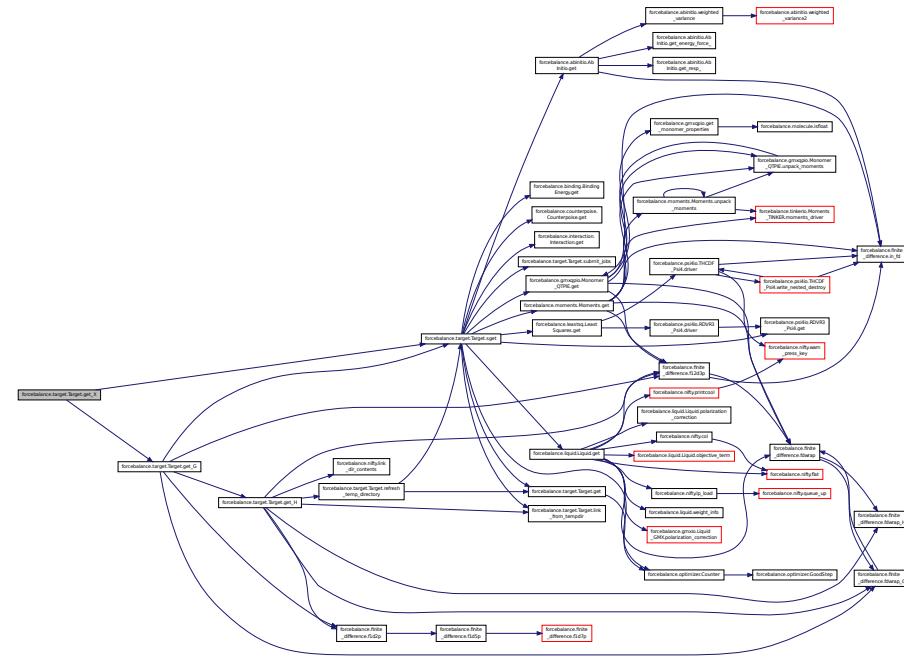
Here is the call graph for this function:



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

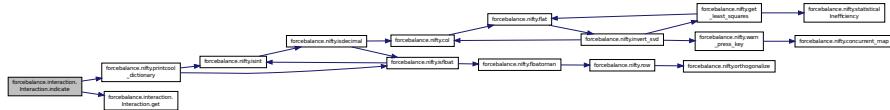
Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.interaction.Interaction.indicate( self )** Definition at line 147 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 213 of file target.py.

```
def forcebalance.interaction.Interaction.prepare_temp_directory ( self, options, tgt_opts ) Prepare the temporary directory, by default does nothing.
```

Definition at line 144 of file interaction.py.

```
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.

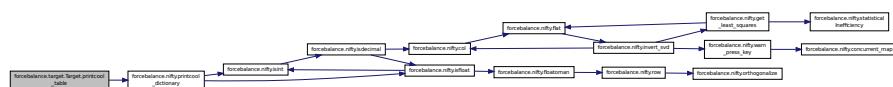
The input should be something like:

## 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 367 of file target.py.

Here is the call graph for this function:

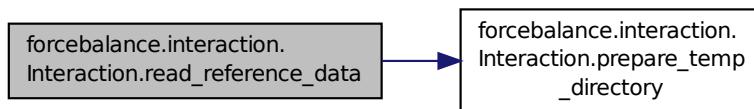


```
def forcebalance.interaction.Interaction.read_reference_data ( self ) Read the reference ab initio data from a file such as qdata.txt.
```

After reading in the information from qdata.txt, it is converted into the GROMACS/OpenMM units (kJ/mol for energy, kJ/mol/nm force).

Definition at line 125 of file interaction.py.

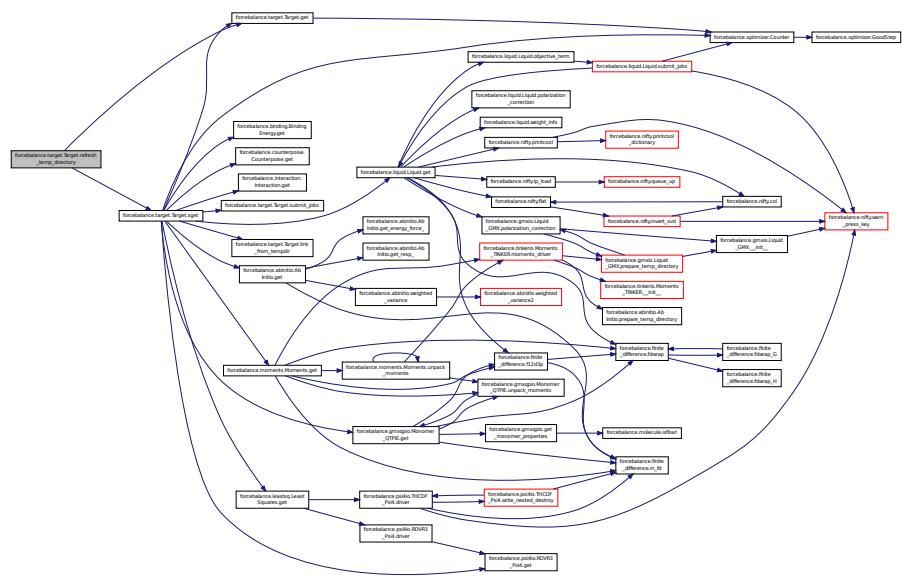
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



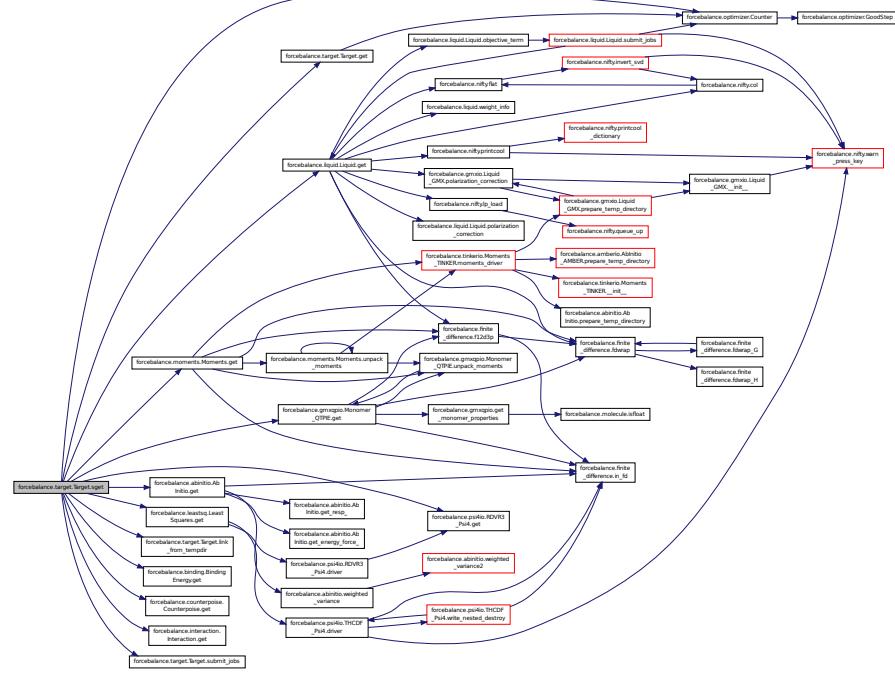
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

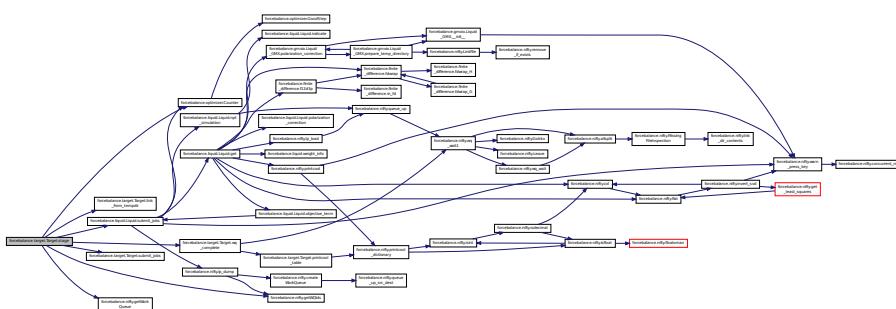


```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

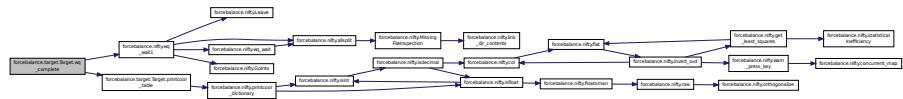


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### **8.23.4 Member Data Documentation**

**forcebalance.interaction.Interaction.divisor** Read in the reference data.

## Prepare the temporary directory

Definition at line 96 of file interaction.py.

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

Definition at line 78 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err\_pct** Definition at line 79 of file interaction.py.

**forcebalance.interaction.Interaction.emm** Definition at line 200 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 139 of file target.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 145 of file target.py

**forcebalance.interaction.InteractionLabel** Snapshot label, useful for graphing

Definition at line 74 of file interaction.py.

**forcebalance.interaction.Interaction.ns** Read in the trajectory file.

Definition at line 81 of file interaction.py.

**forcebalance.interaction.Interaction.objective** Definition at line 201 of file interaction.py.

**forcebalance\_interaction Interaction prefactor** Definition at line 114 of file interaction.py.

**forcebalance** BaseClass PrintOptionDict [inherited] Definition at line 29 of file `init.py`

`cebalance.BaseClass.FaintoptionDict`

**forcebalance.target.Target.rundir** [**inherited**] 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.

Definition at line 137 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.tempdir** [**inherited**] Root directory of the whole project.

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 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 135 of file target.py.

**forcebalance.interaction.Interaction.traj** Definition at line 82 of file interaction.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.interaction.Interaction.weight** Definition at line 167 of file interaction.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 141 of file target.py.

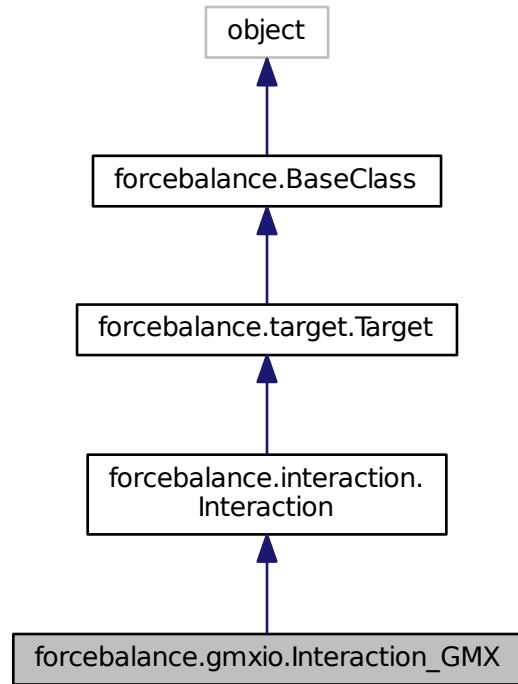
The documentation for this class was generated from the following file:

- [interaction.py](#)

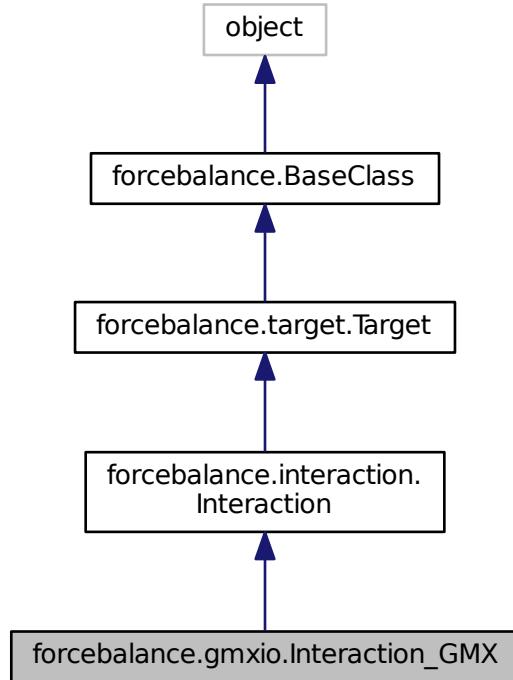
## 8.24 forcebalance.gmxio.Interaction\_GMX Class Reference

Subclass of Interaction for 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 `prepare_temp_directory`
- def `interaction_driver`  
    *Computes the energy and force using GROMACS for a single snapshot.*
- def `interaction_driver_all`  
    *Computes the energy and force using GROMACS for a trajectory.*
- 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 `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 `sget`  
*Stages the directory for the target, and then calls 'get'.*
- 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 `set_option`

## Public Attributes

- `trajfnm`  
*Name of the trajectory.*
- `topfnm`
- `Dielectric`
- `Diel_B`
- `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`  
*Qualitative Indicator: average energy error (in kJ/mol)*
- `e_err_pct`
- `ns`  
*Read in the trajectory file.*
- `traj`
- `divisor`  
*Read in the reference data.*
- `prefactor`
- `weight`
- `emm`
- `objective`
- `tempdir`  
*Root directory of the whole project.*
- `rundir`  
*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.
- **PrintOptionDict**
- **verbose\_options**

### 8.24.1 Detailed Description

Subclass of Interaction for interaction energy matching using GROMACS.

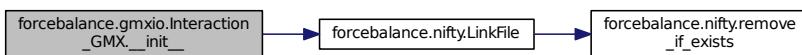
Definition at line 732 of file gmxio.py.

### 8.24.2 Constructor & Destructor Documentation

```
def forcebalance.gmxio.Interaction_GMX.__init__ ( self, options, tgt_opts, forcefield )
```

Definition at line 734 of file gmxio.py.

Here is the call graph for this function:



### 8.24.3 Member Function Documentation

```
def forcebalance.interaction.Interaction.get ( self, mvals, AGrad = False, AHess = False )
```

[inherited] Evaluate objective function.

Definition at line 163 of file interaction.py.

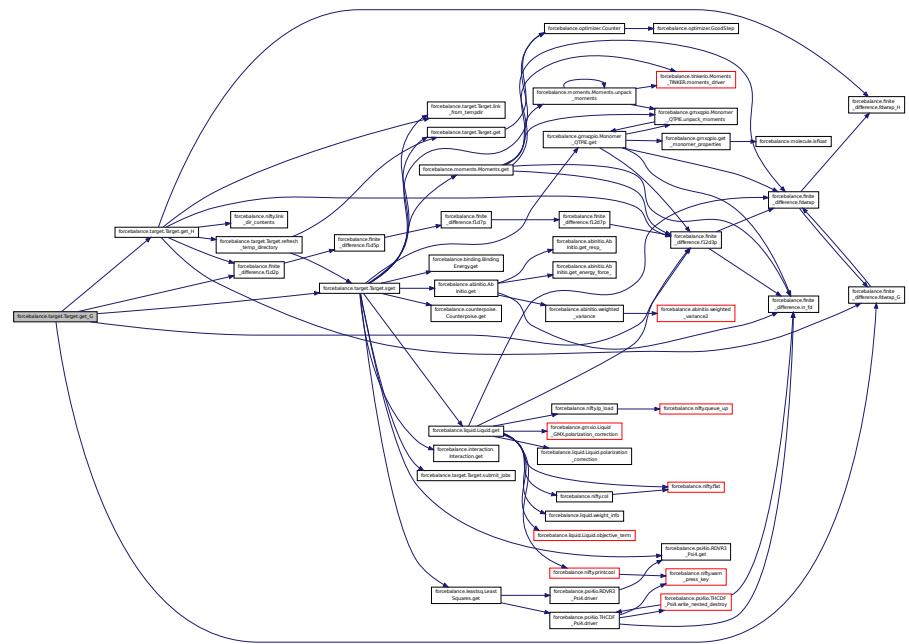
```
def forcebalance.target.Target.get_G ( self, mvals = None )
```

[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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



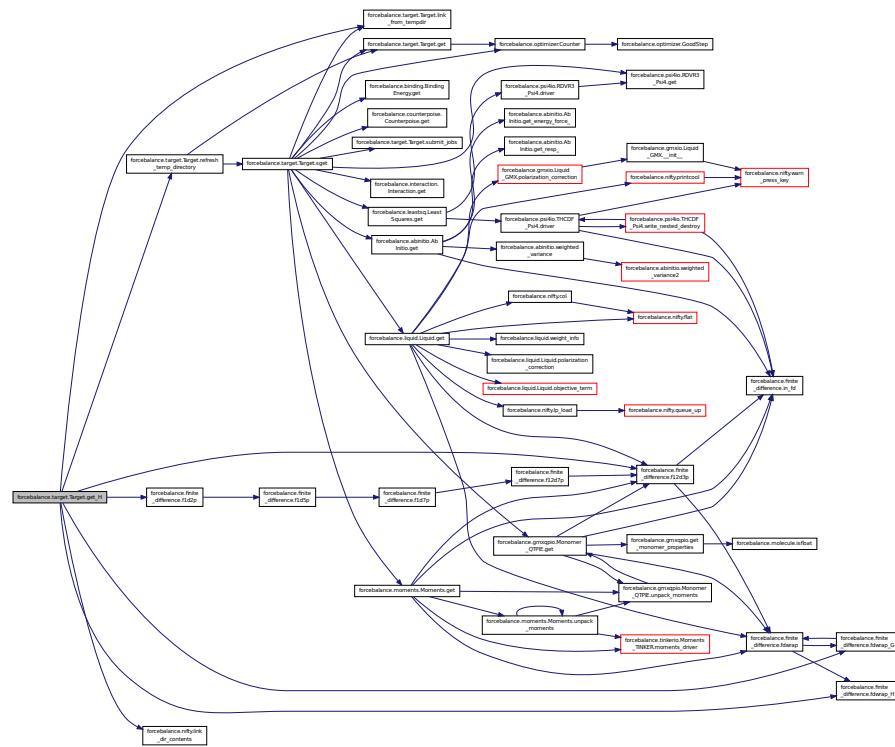
**def forcebalance.target.Target.get.H ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

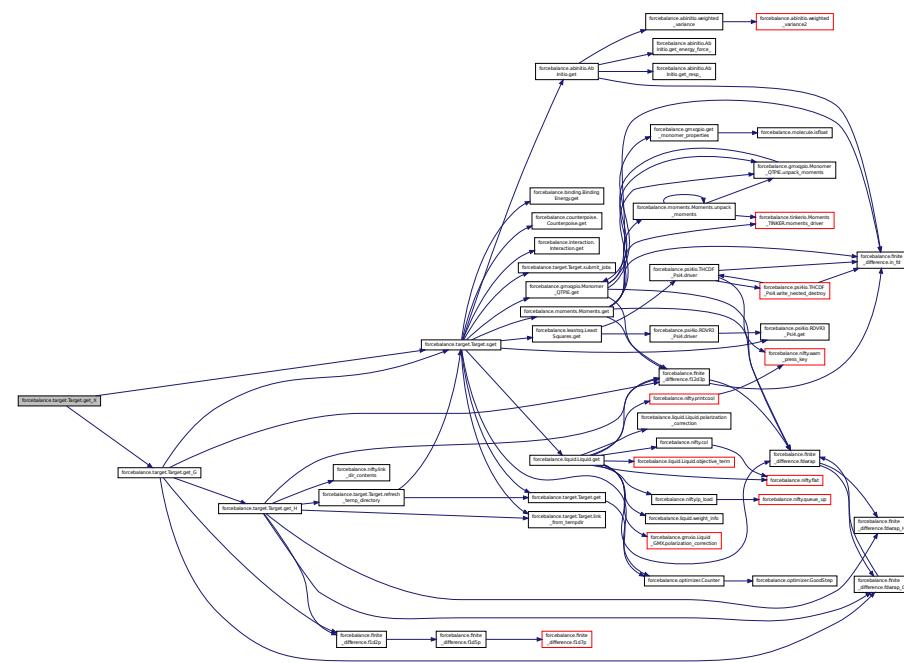
Here is the call graph for this function:



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

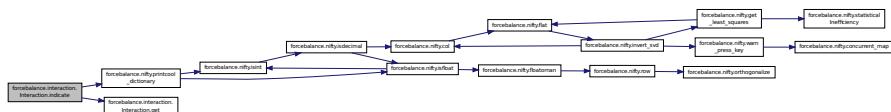
Definition at line 155 of file target.py.

Here is the call graph for this function:



```
def forcebalance.interaction.Interaction.indicate ( self ) [inherited] Definition at line 147 of file interaction.py.
```

Here is the call graph for this function:



**def forcebalance.gmxio.Interaction\_GMX.interaction\_driver( self, shot )** Computes the energy and force using GROMACS for a single snapshot.

This does not require GROMACS-X2.

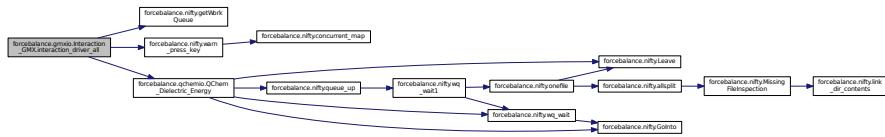
Definition at line 765 of file gmxio.py.

**def forcebalance.gmxio.Interaction\_GMX.interaction\_driver\_all ( self, dielectric = False )** Computes the energy and force using GROMACS for a trajectory.

This does not require GROMACS-X2.

Definition at line 770 of file gmxio.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

**def forcebalance.gmxio.Interaction\_GMX.prepare\_temp\_directory ( self, options, tgt\_opts )** Definition at line 742 of file gmxio.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.

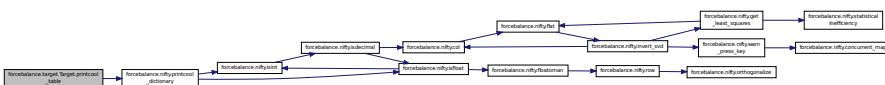
The input should be something like:

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 367 of file target.py.

Here is the call graph for this function:

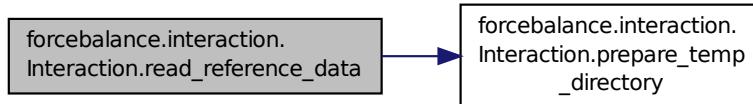


**def forcebalance.interaction.Interaction.read\_reference\_data ( self ) [inherited]** Read the reference ab initio data from a file such as qdata.txt.

After reading in the information from qdata.txt, it is converted into the GROMACS/OpenMM units (kJ/mol for energy, kJ/mol/nm force).

Definition at line 125 of file interaction.py.

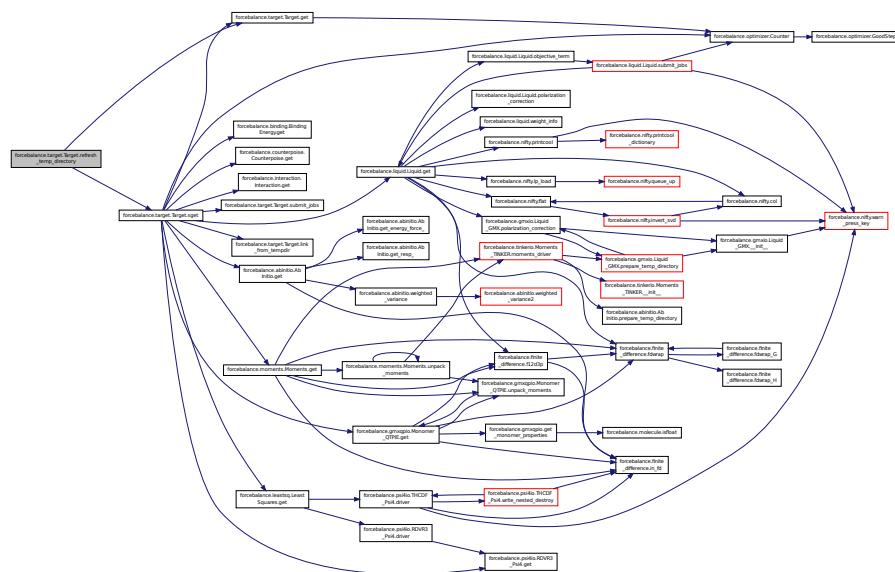
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



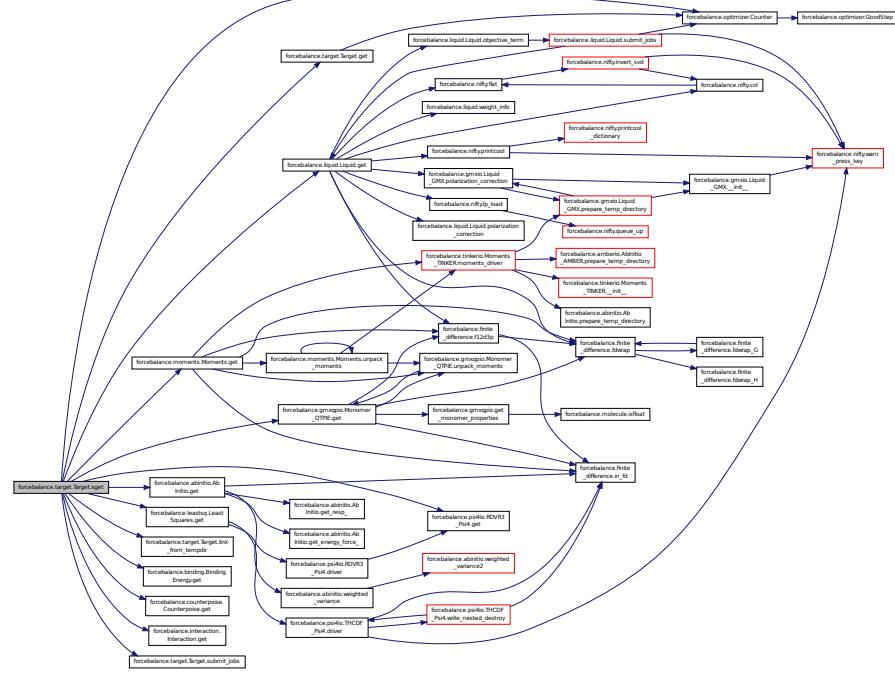
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

**def forcebalance.target.Target.sget( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then calls 'get'.

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

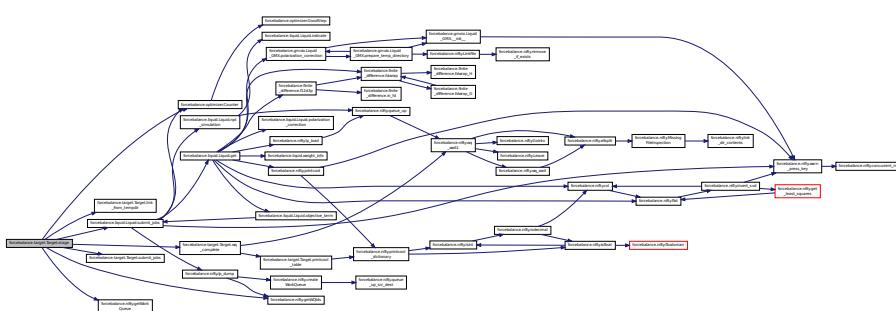


**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

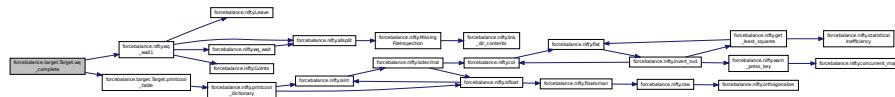


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### 8.24.4 Member Data Documentation

**forcebalance.gmxio.Interaction\_GMX.Diel\_B** Definition at line 810 of file gmxio.py.

**forcebalance.gmxio.Interaction\_GMX.Dielectric** Definition at line 739 of file gmxio.py.

**forcebalance.interaction.Interaction.divisor [inherited]** Read in the reference data.

Prepare the temporary directory

Definition at line 96 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err [inherited]** Qualitative Indicator: average energy error (in kJ/mol)

Definition at line 78 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err\_pct [inherited]** Definition at line 79 of file interaction.py.

**forcebalance.interaction.Interaction.emm [inherited]** Definition at line 200 of file interaction.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 139 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 145 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.ns [inherited]** Read in the trajectory file.

Definition at line 81 of file interaction.py.

**forcebalance.interaction.Interaction.objective [inherited]** Definition at line 201 of file interaction.py.

**forcebalance.interaction.Interaction.prefactor [inherited]** Definition at line 114 of file interaction.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 29 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.rundir** [**inherited**] 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.

Definition at line 137 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.tempdir** [**inherited**] Root directory of the whole project.

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 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 135 of file target.py.

**forcebalance.gmxio.Interaction\_GMX.topfnm** Definition at line 737 of file gmxio.py.

**forcebalance.interaction.Interaction.traj** [**inherited**] Definition at line 82 of file interaction.py.

**forcebalance.gmxio.Interaction\_GMX.trajfnm** Name of the trajectory.

Definition at line 736 of file gmxio.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.interaction.Interaction.weight** [**inherited**] Definition at line 167 of file interaction.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 141 of file target.py.

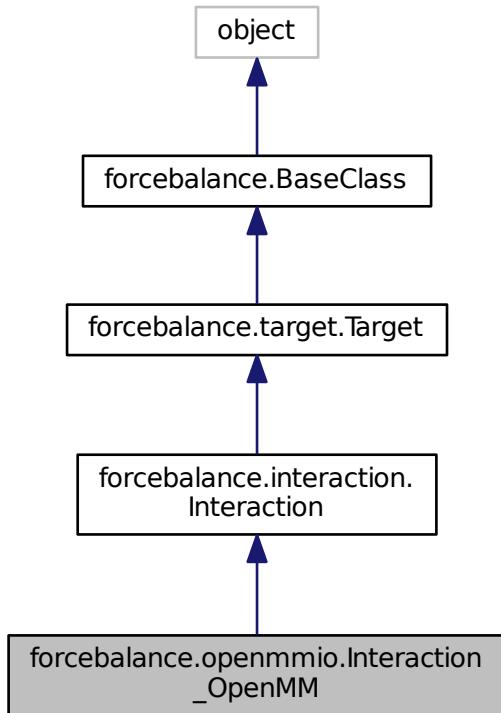
The documentation for this class was generated from the following file:

- [gmxio.py](#)

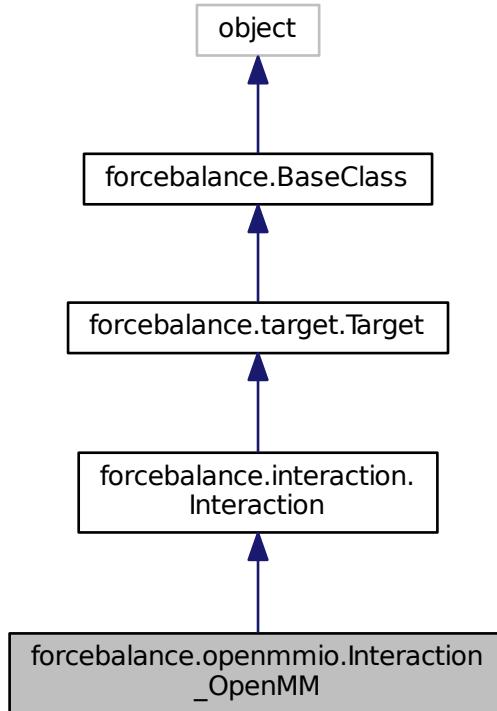
## 8.25 forcebalance.openmmio.Interaction\_OpenMM Class Reference

Subclass of Target for interaction matching using OpenMM.

Inheritance diagram for forcebalance.openmmio.Interaction\_OpenMM:



Collaboration diagram for forcebalance.openmmio.Interaction\_OpenMM:



### Public Member Functions

- def `__init__`
- def `prepare_temp_directory`
- def `energy_driver_all`
- def `interaction_driver_all`
- 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 `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 **sget**  
*Stages the directory for the target, and then calls 'get'.*
- 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 **set\_option**

## Public Attributes

- **trajfnm**  
*Name of the trajectory file containing snapshots.*
- **simulations**  
*Dictionary of simulation objects (dimer, fraga, fragb)*
- **platform**  
*Set up three OpenMM System objects.*
- **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**  
*Qualitative Indicator: average energy error (in kJ/mol)*
- **e\_err\_pct**
- **ns**  
*Read in the trajectory file.*
- **traj**
- **divisor**  
*Read in the reference data.*
- **prefactor**
- **weight**
- **emm**
- **objective**
- **tempdir**  
*Root directory of the whole project.*
- **rundir**  
*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.
- **PrintOptionDict**
- **verbose\_options**

### 8.25.1 Detailed Description

Subclass of Target for interaction matching using OpenMM.

Definition at line 560 of file openmmio.py.

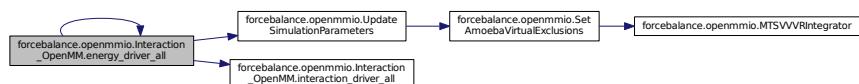
### 8.25.2 Constructor & Destructor Documentation

**def forcebalance.openmmio.Interaction\_OpenMM.\_\_init\_\_ ( self, options, tgt\_opts, forcefield )** Definition at line 563 of file openmmio.py.

### 8.25.3 Member Function Documentation

**def forcebalance.openmmio.Interaction\_OpenMM.energy\_driver\_all ( self, mode )** Definition at line 625 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.interaction.Interaction.get ( self, mvals, AGrad = False, AHess = False ) [inherited]** Evaluate objective function.

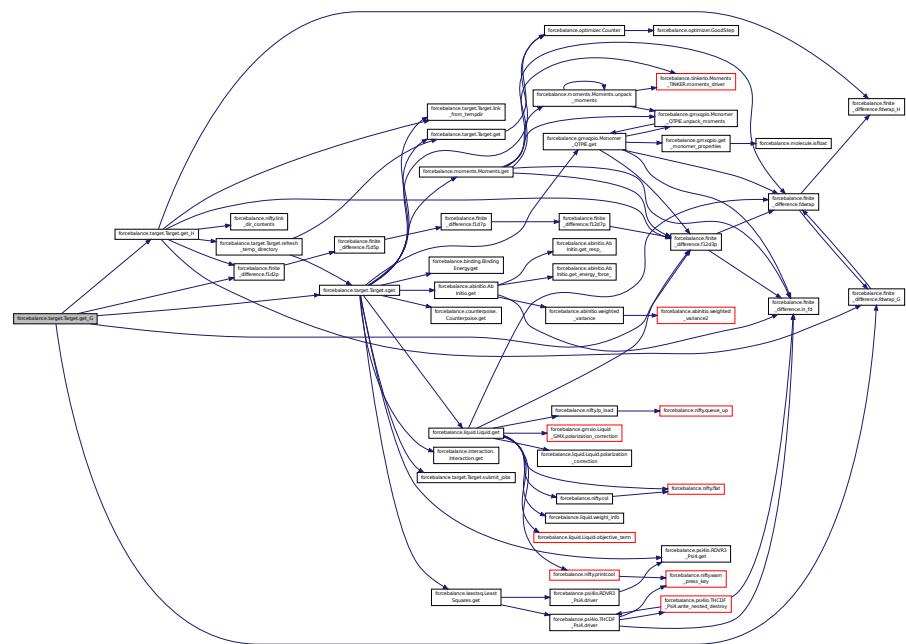
Definition at line 163 of file interaction.py.

**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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



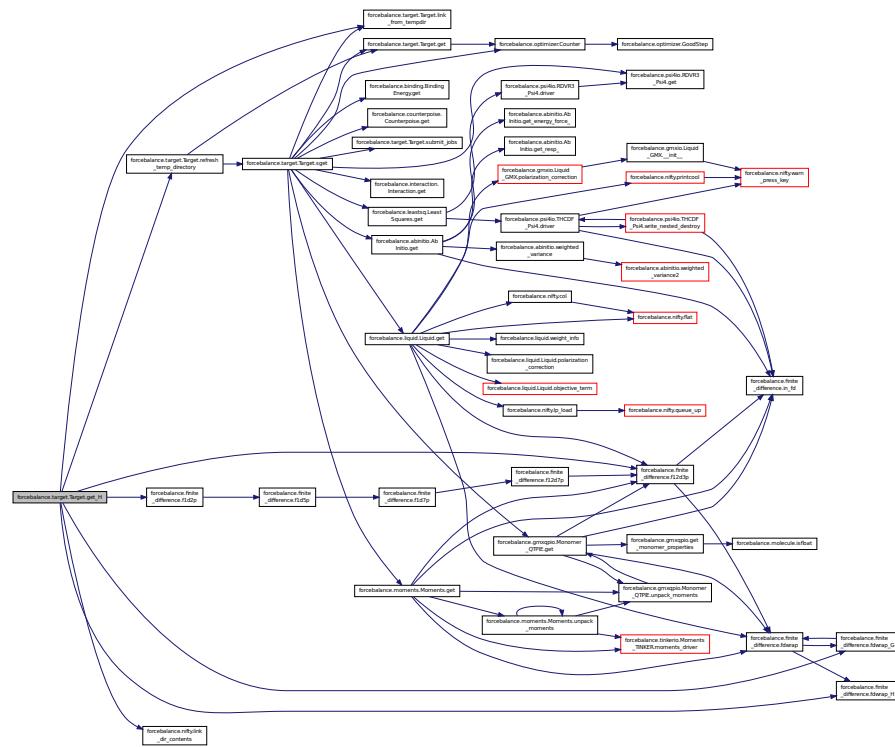
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

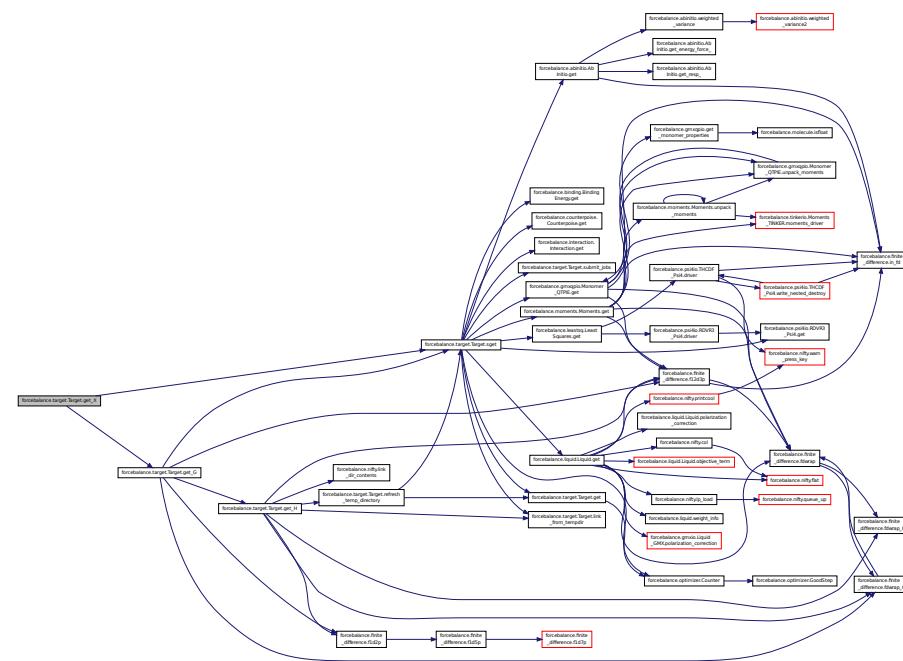
Here is the call graph for this function:



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

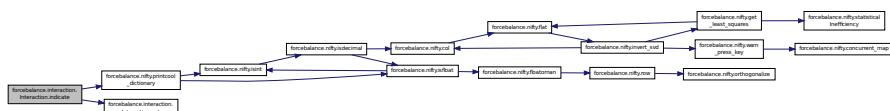
Definition at line 155 of file target.py.

Here is the call graph for this function:



```
def forcebalance.interaction.Interaction.indicate ( self ) [inherited] Definition at line 147 of file interaction.py.
```

Here is the call graph for this function:



**def forcebalance.openmmio.Interaction\_OpenMM.interaction\_driver\_all( self, dielectric = False )** Definition at line 679 of file openmmio.py.

```
def forcebalance.target.Target.link_from_tempdir( self, absdestdir ) [inherited] Definition at line 213 of file target.py.
```

**def forcebalance.openmmio.Interaction\_OpenMM.prepare\_temp\_directory ( self, options, tgt\_opts )** Definition at line 571 of file openmmio.py.

```
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.

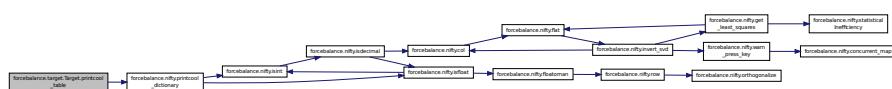
The input should be something like:

## 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 367 of file target.py.

Here is the call graph for this function:

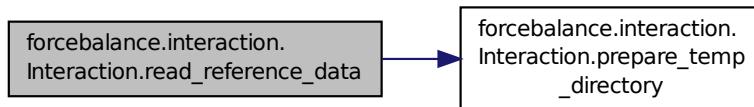


```
def forcebalance.interaction.Interaction.read_reference_data ( self ) [inherited] Read the reference ab initio data from a file such as qdata.txt.
```

After reading in the information from qdata.txt, it is converted into the GROMACS/OpenMM units (kJ/mol for energy, kJ/mol/nm force).

Definition at line 125 of file interaction.py.

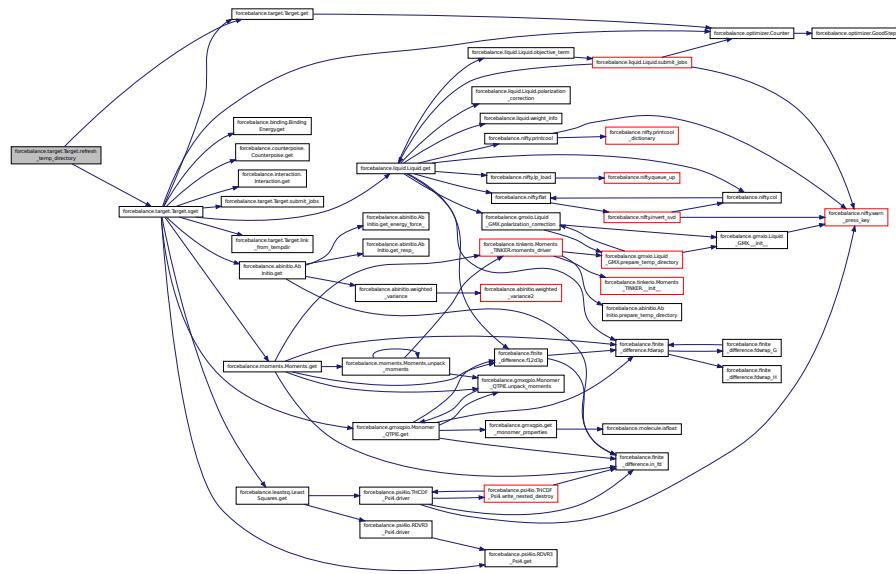
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



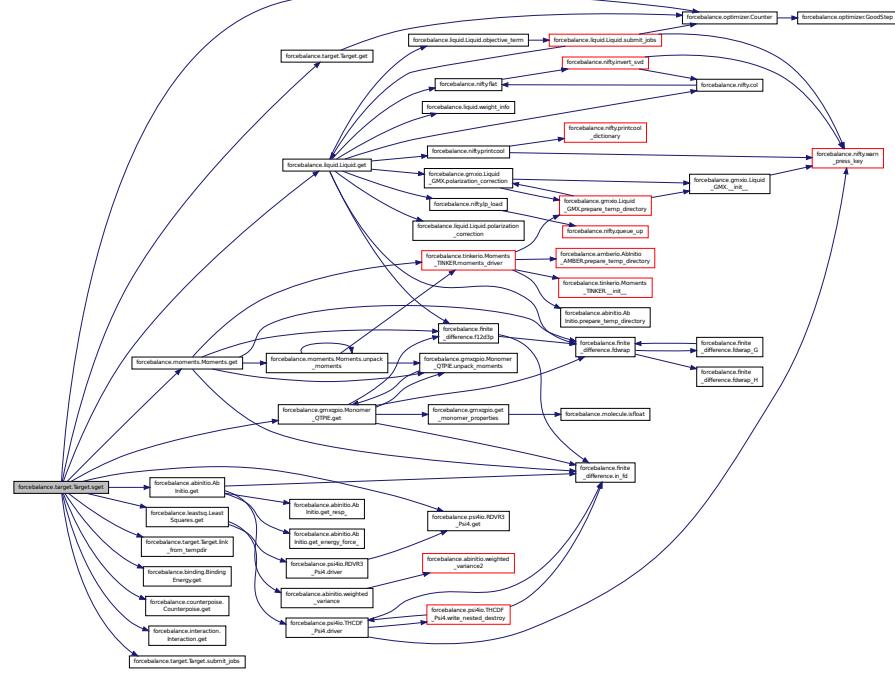
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

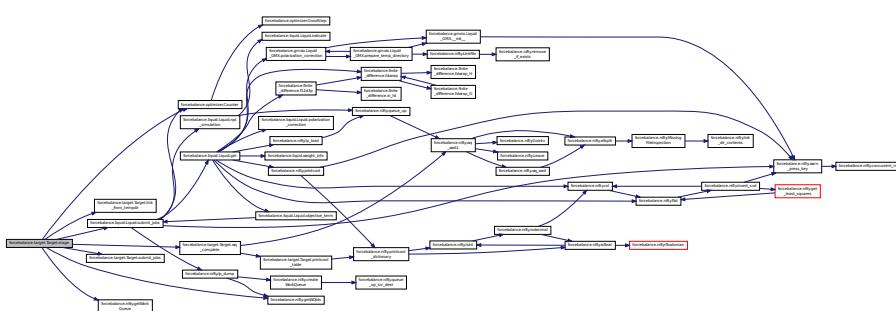


**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

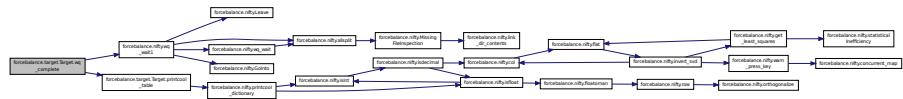


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### **8.25.4 Member Data Documentation**

**forcebalance.interaction.Interaction.divisor** [inherited] Read in the reference data.

Prepare the temporary directory

Definition at line 96 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err** [inherited] Qualitative Indicator: average energy error (in kJ/mol)  
Definition at line 78 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err\_pct** [inherited] Definition at line 79 of file interaction.py.

**forcebalance.interaction.Interaction.emm** [inherited] Definition at line 200 of file interaction.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 139 of file target.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 145 of file target.py.

**forcebalance.interaction.Interaction.label** [inherited] Snapshot label, useful for graphing.

Definition at line 74 of file interaction.m.

**forcebalance.interaction.Interaction.ns** [inherited] Read in the trajectory file.

Definition at line 81 of file interaction.py.

**forcebalance.interaction.Interaction.objective** [inherited] Definition at line 201 of file interaction.py.

**forcebalance openmmio Interaction OpenMM platform** Set up three OpenMM System objects

Set the device to the environment variable or zero otherwise

**TODO:** The following code should not be repeated everywhere. Set the simulation platform

TODO: The following code should not be

**forcebalance interaction Interaction prefactor** [inherited] Definition at line 114 of file interaction.py.

**forcebalance** `BaseClass PrintOptionDict` [inherited] Definition at line 29 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.rundir** [**inherited**] 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.

Definition at line 137 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.openmmio.Interaction\_OpenMM.simulations** Dictionary of simulation objects (dimer, fraga, fragb)

Definition at line 567 of file openmmio.py.

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

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 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 135 of file target.py.

**forcebalance.interaction.Interaction.traj** [**inherited**] Definition at line 82 of file interaction.py.

**forcebalance.openmmio.Interaction\_OpenMM.traifnm** Name of the trajectory file containing snapshots.

Definition at line 565 of file openmmio.py.

**forcebalance.BaseClass.verbose\_options** [**inherited**] Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.interaction.Interaction.weight** [**inherited**] Definition at line 167 of file interaction.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 141 of file target.py.

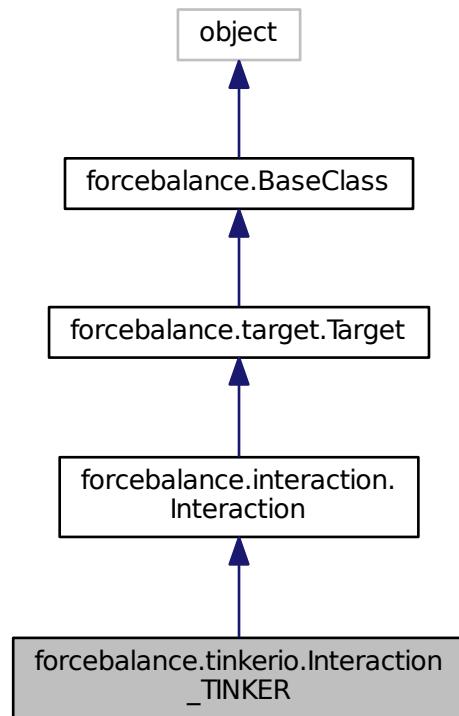
The documentation for this class was generated from the following file:

- [openmmio.py](#)

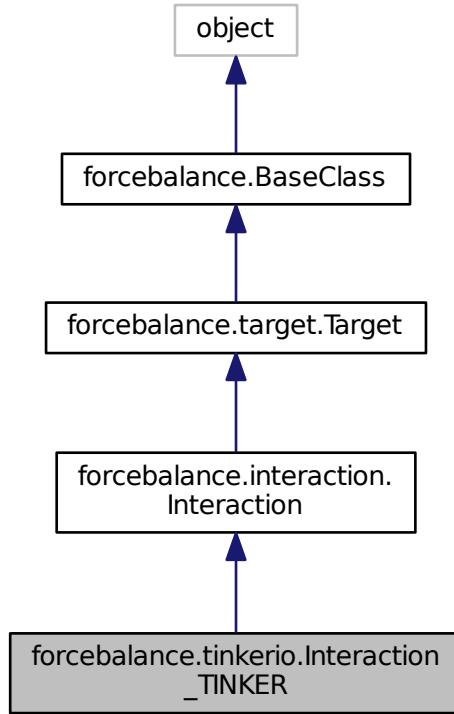
## 8.26 forcebalance.tinkerio.Interaction\_TINKER Class Reference

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 `prepare_temp_directory`
- def `energy_driver_all`
- def `interaction_driver_all`
- 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 `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 **sget**  
*Stages the directory for the target, and then calls 'get'.*
- 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 **set\_option**

## Public Attributes

- **trajfnm**  
*Name of the trajectory.*
- **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**  
*Qualitative Indicator: average energy error (in kJ/mol)*
- **e\_err\_pct**
- **ns**  
*Read in the trajectory file.*
- **traj**
- **divisor**  
*Read in the reference data.*
- **prefactor**
- **weight**
- **emm**
- **objective**
- **tempdir**  
*Root directory of the whole project.*
- **rundir**  
*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.*
- `PrintOptionDict`
- `verbose_options`

### 8.26.1 Detailed Description

Subclass of Target for interaction matching using TINKER.

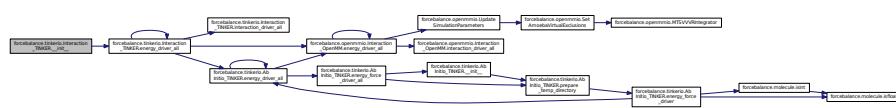
Definition at line 583 of file tinkerio.py.

### 8.26.2 Constructor & Destructor Documentation

```
def forcebalance.tinkerio.Interaction_TINKER.__init__ ( self, options, tgt_opts, forcefield )
```

Definition at line 586 of file tinkerio.py.

Here is the call graph for this function:

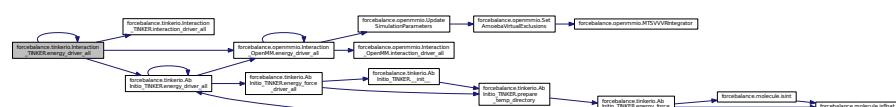


### 8.26.3 Member Function Documentation

```
def forcebalance.tinkerio.Interaction_TINKER.energy_driver_all ( self, select = None )
```

Definition at line 599 of file tinkerio.py.

Here is the call graph for this function:



```
def forcebalance.interaction.Interaction.get ( self, mvals, AGrad = False, AHess = False )
```

**[inherited]** Evaluate objective function.

Definition at line 163 of file interaction.py.

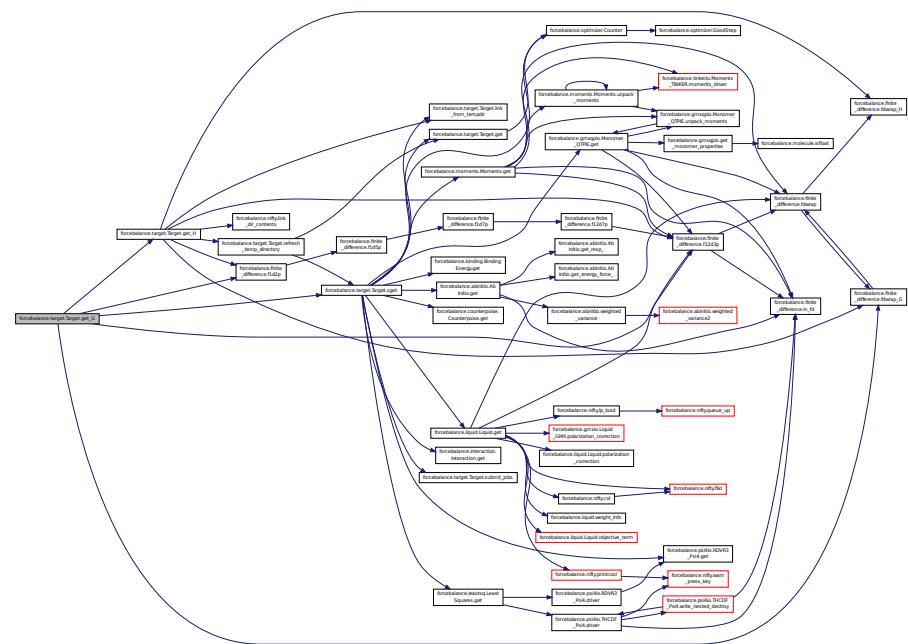
```
def forcebalance.target.Target.get_G ( self, mvals = None )
```

**[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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



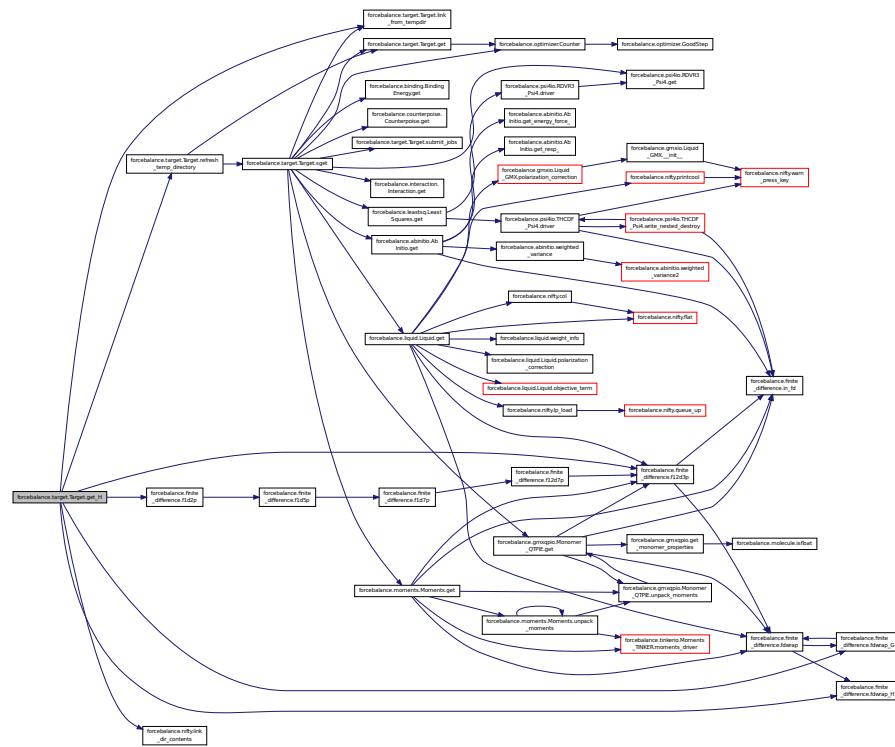
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

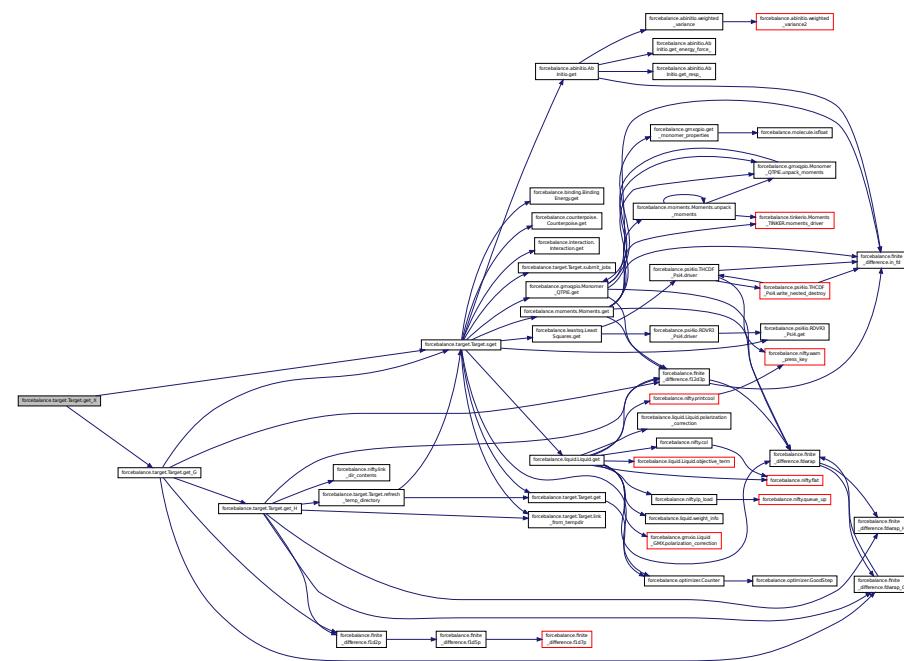
Here is the call graph for this function:



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

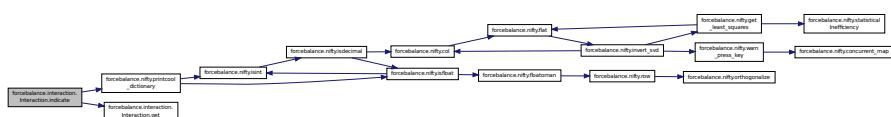
Definition at line 155 of file target.py.

Here is the call graph for this function:



```
def forcebalance.interaction.Interaction.indicate ( self ) [inherited] Definition at line 147 of file interaction.py.
```

Here is the call graph for this function:



**def forcebalance.tinkerio.Interaction.TINKER.interaction\_driver\_all ( self, dielectric = False )** Definition at line 616 of file tinkerio.py.

**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

**def forcebalance.tinkerio.Interaction\_TINKER.prepare\_temp\_directory ( self, options, tgt\_opts )** Definition at line 591 of file tinkerio.py.

**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.

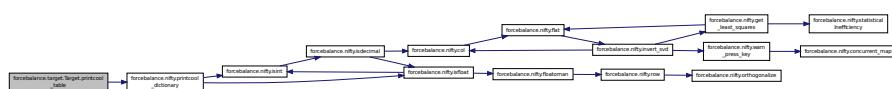
The input should be something like:

## 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 367 of file target.py.

Here is the call graph for this function:

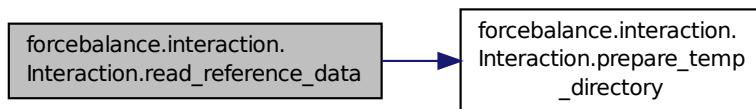


```
def forcebalance.interaction.Interaction.read_reference_data ( self ) [inherited] Read the reference ab initio data from a file such as qdata.txt.
```

After reading in the information from qdata.txt, it is converted into the GROMACS/OpenMM units (kJ/mol for energy, kJ/mol/nm force).

Definition at line 125 of file interaction.py.

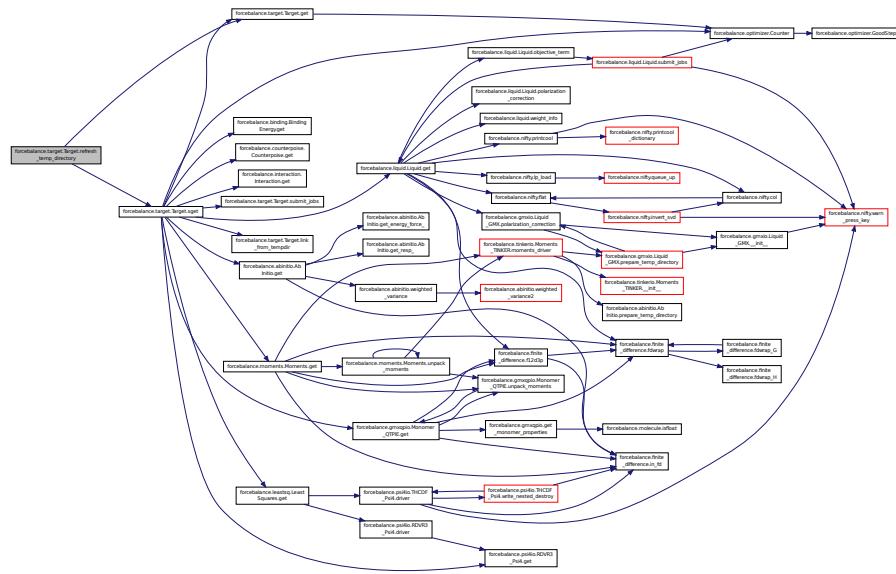
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



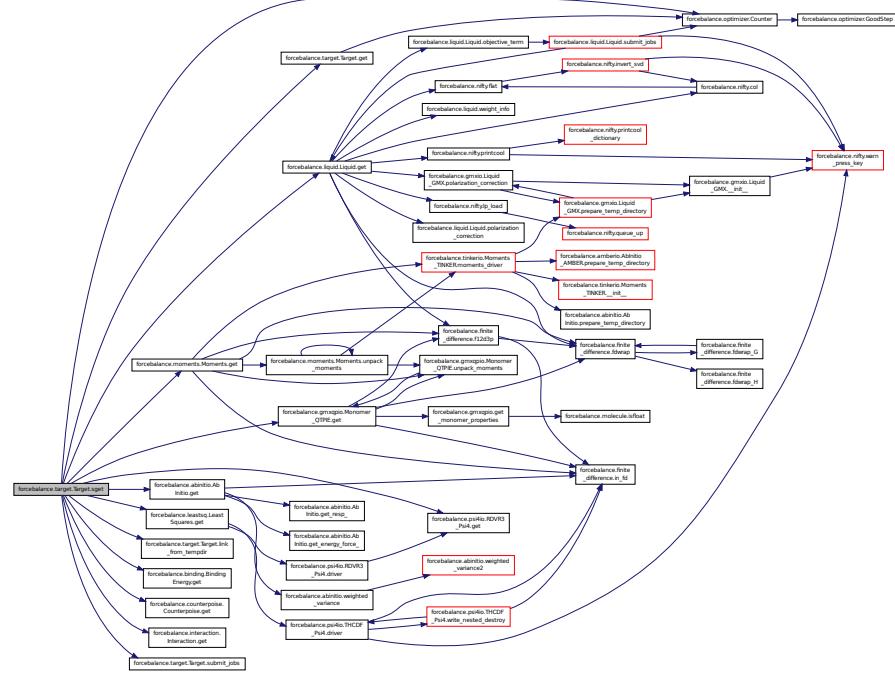
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

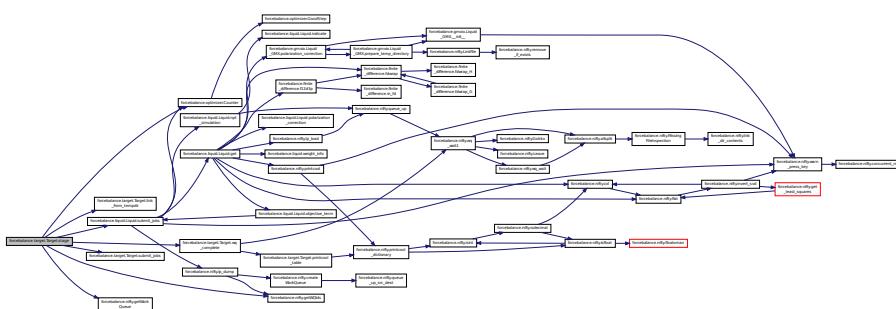


**[inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

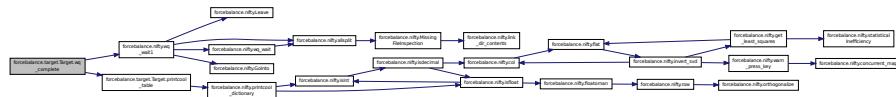


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### **8.26.4 Member Data Documentation**

**forcebalance.interaction.Interaction.divisor** [inherited] Read in the reference data.

Prepare the temporary directory

Definition at line 96 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err** [inherited] Qualitative Indicator: average energy error (in kJ/mol)  
Definition at line 78 of file interaction.py.

**forcebalance.interaction.Interaction.e\_err\_pct** [inherited] Definition at line 79 of file interaction.py.

**forcebalance.interaction.Interaction.emm** [inherited] Definition at line 200 of file interaction.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 139 of file target.py.

**forcebalance.target.Target.act** [inherited] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.

Definition at line 145 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.ns** [inherited] Read in the trajectory file.

Definition at line 81 of file interaction.py.

**forcebalance.interaction.Interaction.objective** [inherited] Definition at line 201 of file interaction.py.

**forcebalance\_interaction** **Interaction\_prefactor** [inherited] Definition at line 114 of file interaction.py

**forcebalance**.BaseClass PrintOptionDict [inherited] Definition at line 29 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.rundir [inherited]** 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.

Definition at line 137 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.tempdir [inherited]** Root directory of the whole project.

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 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 135 of file target.py.

**forcebalance.interaction.Interaction.traj [inherited]** Definition at line 82 of file interaction.py.

**forcebalance.tinkerio.Interaction.TINKER.trajfnm** Name of the trajectory.

Definition at line 588 of file tinkerio.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.interaction.Interaction.weight [inherited]** Definition at line 167 of file interaction.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 141 of file target.py.

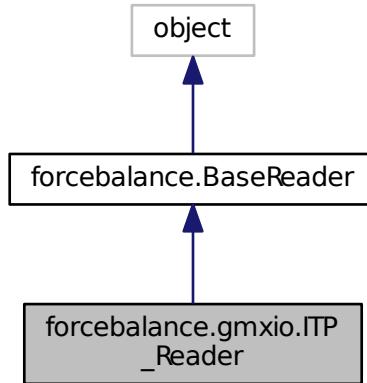
The documentation for this class was generated from the following file:

- [tinkerio.py](#)

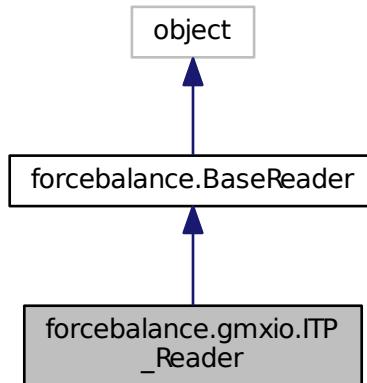
## 8.27 forcebalance.gmxio.ITP\_Reader Class Reference

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.*
- **nbtype**  
*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**

### 8.27.1 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 'PARM' or 'RPT' keywords.

As we go through the file, we figure out the atoms involved in the interaction described on each line.

When a 'PARM' 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 ; PARM 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 277 of file gmxio.py.

## 8.27.2 Constructor & Destructor Documentation

**def forcebalance.gmxio.ITP\_Reader.\_\_init\_\_ ( self, fnm )** Definition at line 280 of file gmxio.py.

## 8.27.3 Member Function Documentation

**def forcebalance.BaseReader.build\_pid ( self, pid ) [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 107 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
; PARM 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

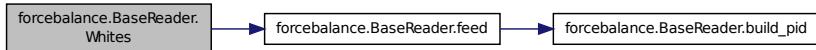
Definition at line 318 of file gmxio.py.

**def forcebalance.BaseReader.Split ( self, line ) [inherited]** Definition at line 82 of file \_\_init\_\_.py.

Here is the call graph for this function:



```
def forcebalance.BaseReader.Whites( self, line ) [inherited] Definition at line 85 of file __init__.py.  
Here is the call graph for this function:
```



#### 8.27.4 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 72 of file \_\_init\_\_.py.

**forcebalance.gmxio.ITP\_Reader.atomnames** Listing of all atom names in the file, (probably unnecessary)  
Definition at line 292 of file gmxio.py.

**forcebalance.gmxio.ITP\_Reader.atomtype\_to\_mass** A dictionary of atomic masses.  
Definition at line 296 of file gmxio.py.

**forcebalance.BaseReader.AtomTypes** [inherited] Definition at line 80 of file \_\_init\_\_.py.

**forcebalance.gmxio.ITP\_Reader.atomtypes** Listing of all atom types in the file, (probably unnecessary)  
Definition at line 294 of file gmxio.py.

**forcebalance.gmxio.ITP\_Reader.itype** Definition at line 321 of file gmxio.py.

**forcebalance.BaseReader.In** [inherited] Definition at line 67 of file \_\_init\_\_.py.

**forcebalance.gmxio.ITP\_Reader.mol** The current molecule (set by the moleculetype keyword)  
Definition at line 288 of file gmxio.py.

**forcebalance.gmxio.ITP\_Reader.molatom** Definition at line 428 of file gmxio.py.

**forcebalance.BaseReader.Molecules** [inherited] Definition at line 79 of file \_\_init\_\_.py.

**forcebalance.gmxio.ITP\_Reader.nbtype** Nonbonded type.  
Definition at line 286 of file gmxio.py.

**forcebalance.gmxio.ITP\_Reader.pdict** The parameter dictionary (defined in this file)  
Definition at line 290 of file gmxio.py.

**forcebalance.gmxio.ITP\_Reader.sec** The current section that we're in.  
Definition at line 284 of file gmxio.py.

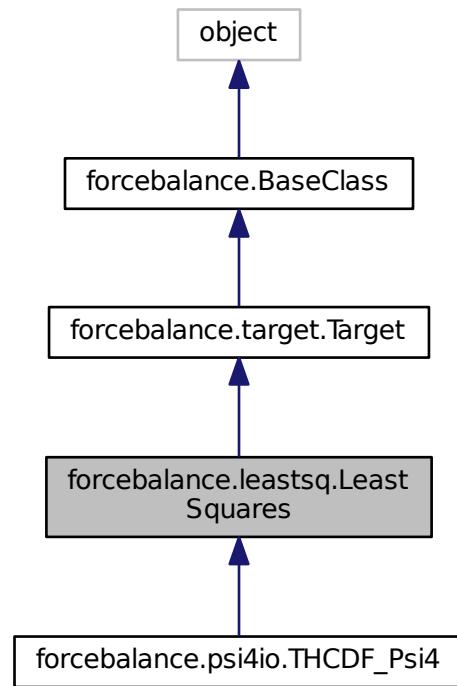
**forcebalance.gmxio.ITP\_Reader.suffix** Definition at line 423 of file gmxio.py.  
The documentation for this class was generated from the following file:

- [gmxio.py](#)

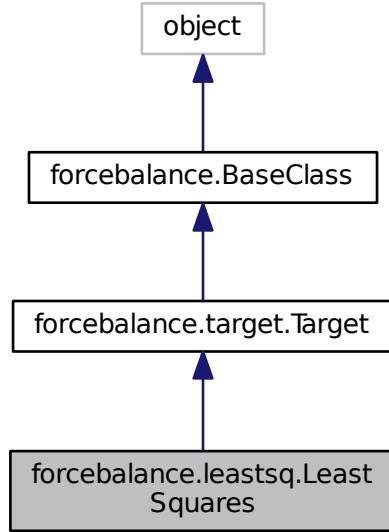
## 8.28 forcebalance.leastsq.LeastSquares Class Reference

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 `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 `sget`
  - Stages the directory for the target, and then calls 'get'.
- 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 `set_option`

## Public Attributes

- **call\_derivatives**  
*Number of snapshots.*
- **MAQ**  
*Dictionary for derivative terms.*
- **D**
- **objective**
- **tempdir**  
*Root directory of the whole project.*
- **rundir**  
*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.*
- **PrintOptionDict**
- **verbose\_options**

### 8.28.1 Detailed Description

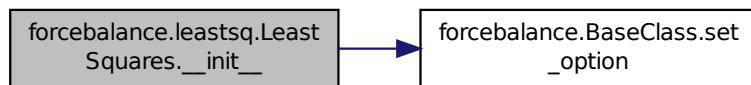
Subclass of Target for general least squares fitting.

Definition at line 35 of file leastsq.py.

### 8.28.2 Constructor & Destructor Documentation

**def forcebalance.leastsq.LeastSquares.\_\_init\_\_( self, options, tgt\_opts, forcefield )** Definition at line 38 of file leastsq.py.

Here is the call graph for this function:



### 8.28.3 Member Function Documentation

**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

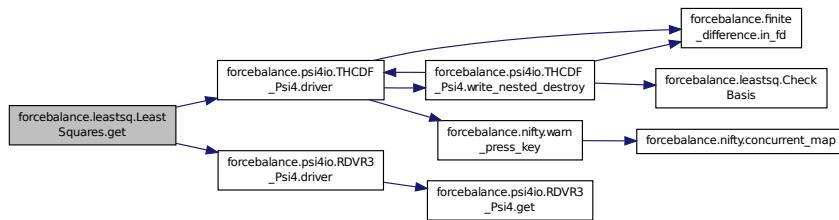
in	<i>mvals</i>	Mathematical parameter values
in	<i>AGrad</i>	Switch to turn on analytic gradient
in	<i>AHess</i>	Switch to turn on analytic Hessian

## Returns

Answer Contribution to the objective function

Definition at line 74 of file leastsq.py.

Here is the call graph for this function:

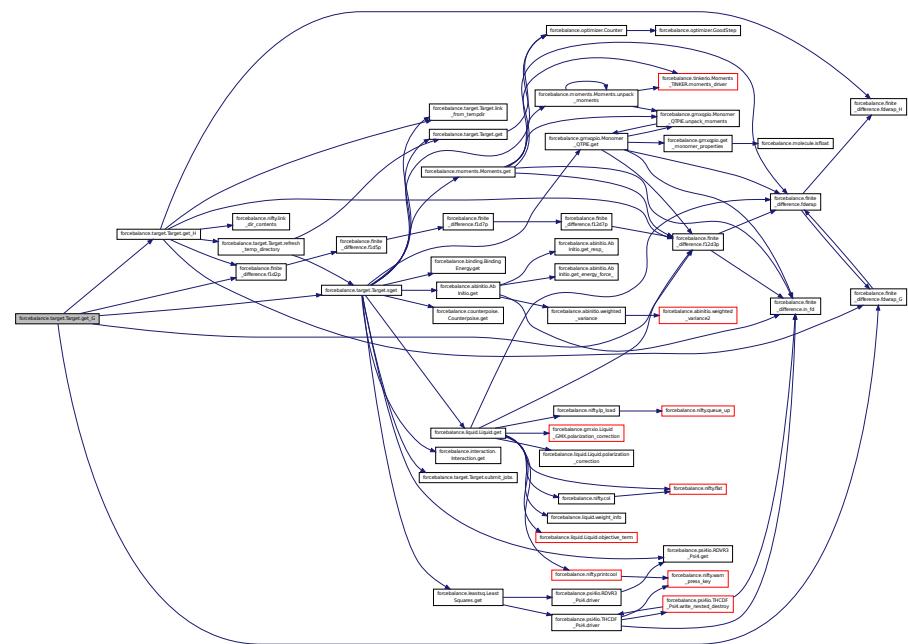


**def forcebalance.target.Target.get\_G ( *self*, *mvals = None* ) [inherited]** Computes the objective function contribution and its gradient.

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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



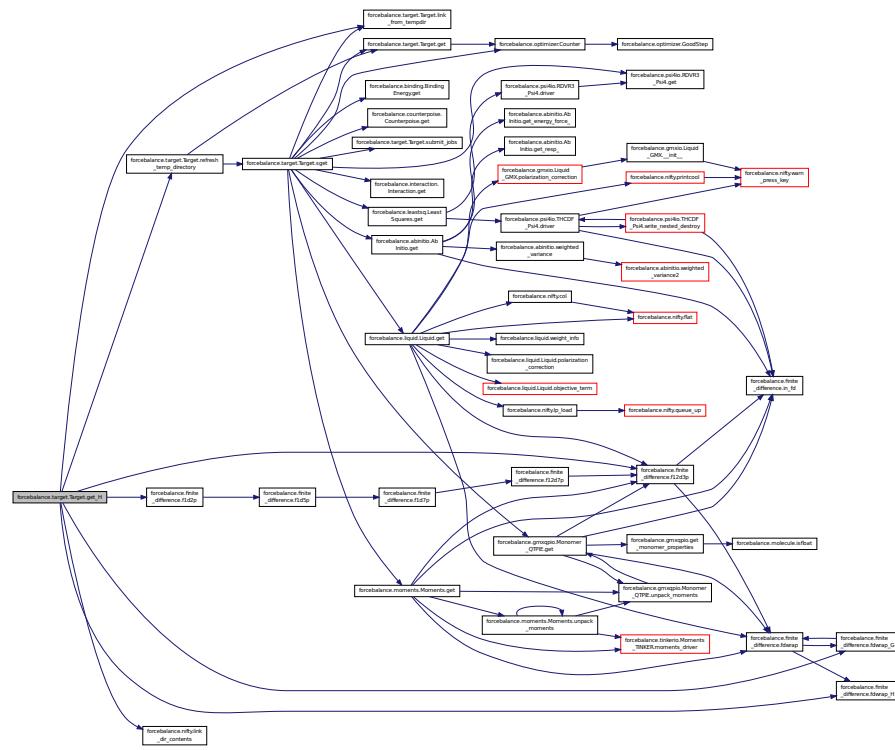
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

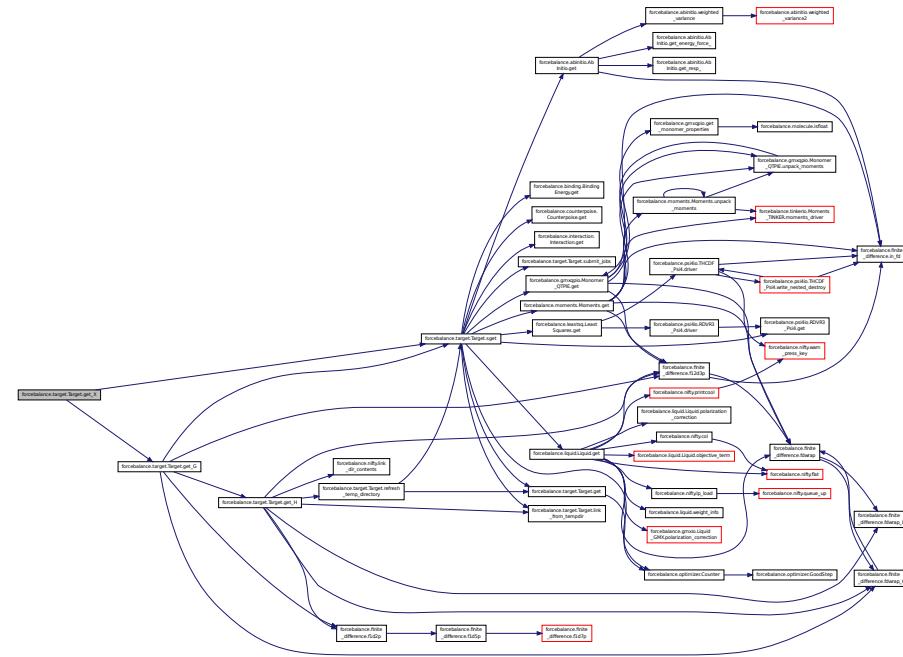
Here is the call graph for this function:



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

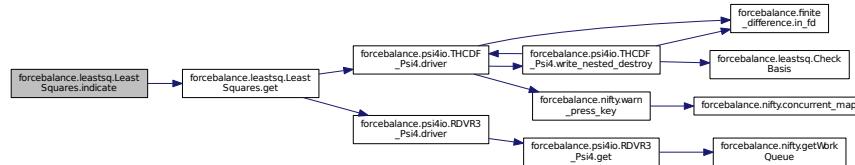
Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.leastsq.LeastSquares.indicate ( self )** Definition at line 53 of file leastsq.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

```
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.

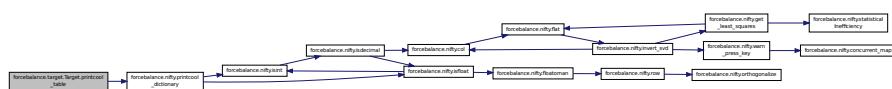
The input should be something like:

## 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 367 of file target.py.

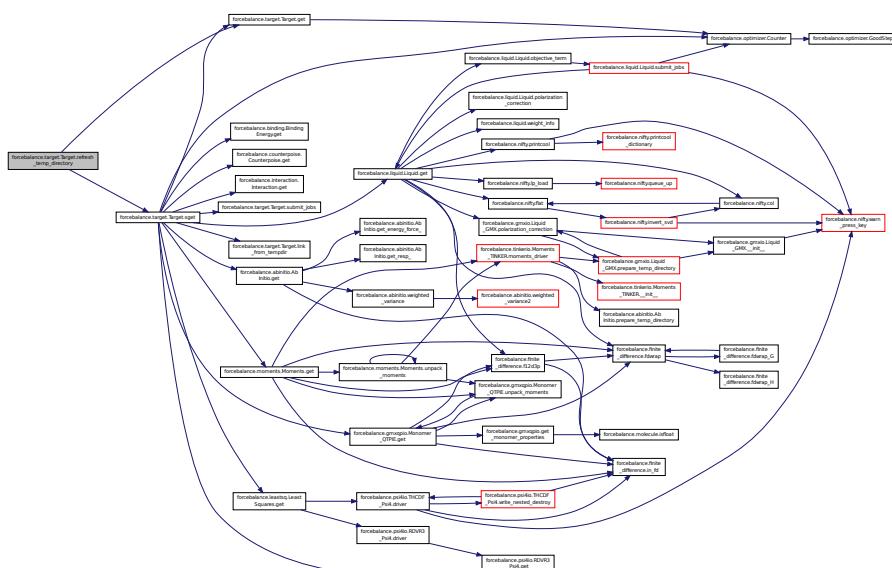
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



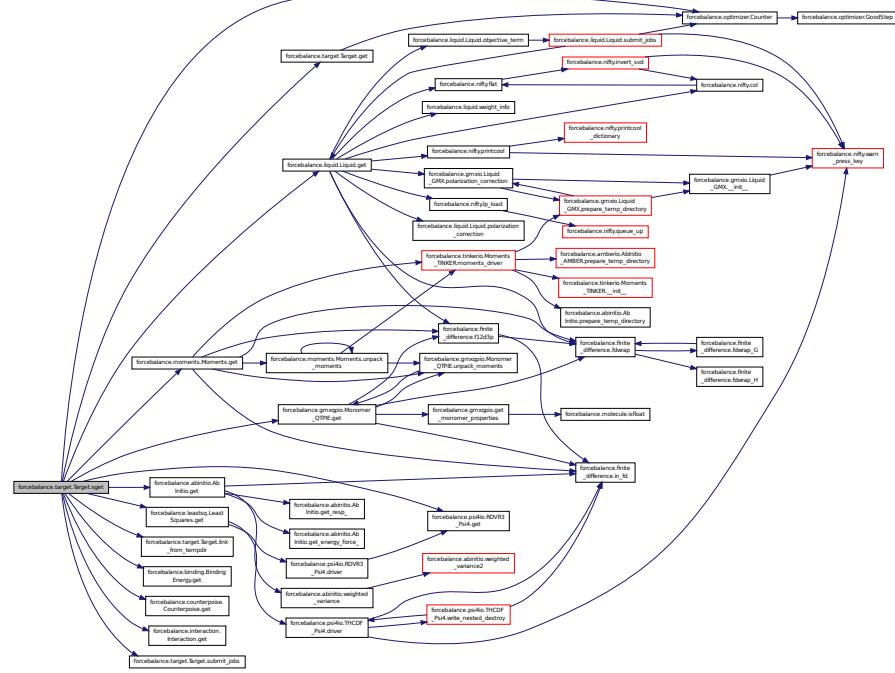
`def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited]` Definition at line 32 of file `_init_.py`.

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
    [inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py

Here is the call graph for this function:

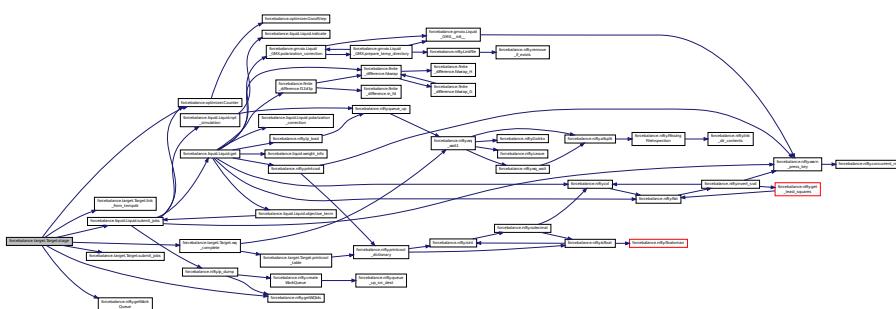


```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

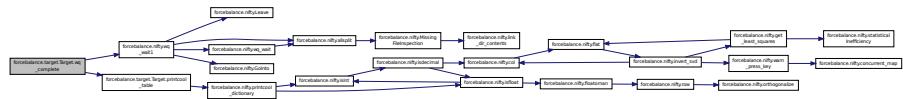


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### **8.28.4 Member Data Documentation**

**forcebalance.leastsq.LeastSquares.call\_derivatives** Number of snapshots.

Which parameters are differentiated?

Definition at line 51 of file leastsq.py.

**forcebalance.leastsq.LeastSquares.D** Definition at line 137 of file leastsq.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.

Definition at line 145 of file target.py.

**forcebalance.leastsq.LeastSquares.MAQ** Dictionary for derivative terms.

Definition at line 100 of file leastsq.py.

**forcebalance.leastsq.LeastSquares.objective** Definition at line 138 of file leastsq.py.

**forcebalance**.**BaseClass**.**PrintOptionDict** [inherited] Definition at line 29 of file **init**.py.

**forcebalance.target.Target.rundir** [inherited] 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.

Definition at line 137 of file target.py.

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

**forcebalance,target,targettempdir** [**target\_name**] Root directory of the whole project.  
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 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 135 of file target.py.

**forcebalance.BaseClass.verbose** options [inherited] Definition at line 30 of file init.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

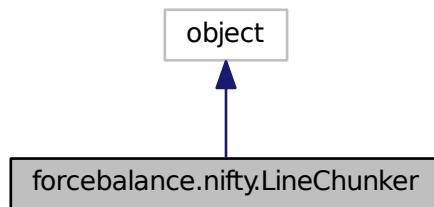
Definition at line 141 of file target.py.

The documentation for this class was generated from the following file:

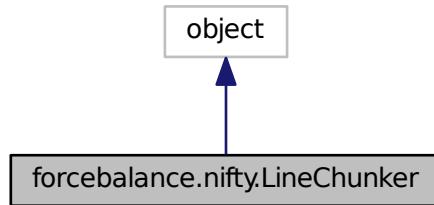
- [leastsq.py](#)

## 8.29 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](#)

### 8.29.1 Detailed Description

Definition at line 850 of file nifty.py.

### 8.29.2 Constructor & Destructor Documentation

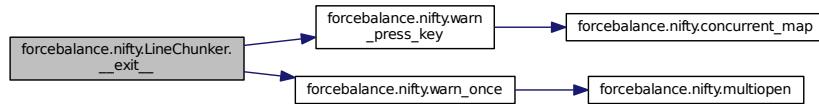
**def forcebalance.nifty.LineChunker.\_\_init\_\_ ( self, callback )** Definition at line 851 of file nifty.py.

### 8.29.3 Member Function Documentation

**def forcebalance.nifty.LineChunker.\_\_enter\_\_ ( self )** Definition at line 870 of file nifty.py.

**def forcebalance.nifty.LineChunker.\_\_exit\_\_ ( self, args, kwargs )** Definition at line 873 of file nifty.py.

Here is the call graph for this function:



**def forcebalance.nifty.LineChunker.close ( self )** Definition at line 859 of file nifty.py.

**def forcebalance.nifty.LineChunker.nomnom ( self )** Definition at line 863 of file nifty.py.

**def forcebalance.nifty.LineChunker.push ( self, data )** Definition at line 855 of file nifty.py.

### 8.29.4 Member Data Documentation

**forcebalance.nifty.LineChunker.buf** Definition at line 853 of file nifty.py.

**forcebalance.nifty.LineChunker.callback** Definition at line 852 of file nifty.py.

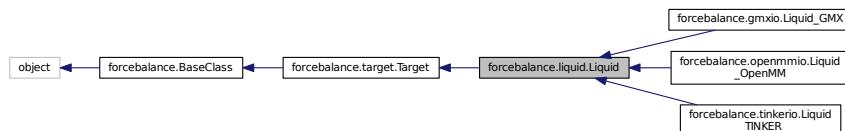
The documentation for this class was generated from the following file:

- [nifty.py](#)

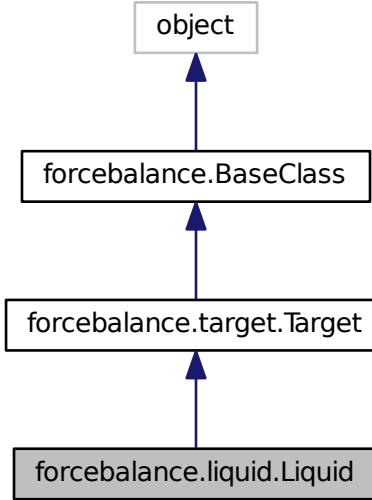
## 8.30 forcebalance.liquid.Liquid Class Reference

Subclass of Target for liquid property matching.

Inheritance diagram for `forcebalance.liquid.Liquid`:



Collaboration diagram for forcebalance.liquid.Liquid:



## Public Member Functions

- def `_init_`  
*Create an instance of the class.*
- def `read_data`
- def `npt_simulation`  
*Submit a NPT simulation to the Work Queue.*
- def `indicate`
- def `objective_term`
- def `submit_jobs`
- def `get`  
*Fitting of liquid bulk properties.*
- def `polarization_correction`  
*Return the self-polarization correction as described in Berendsen et al., JPC 1987.*
- def `get_X`  
*Computes the objective function contribution without any parametric derivatives.*
- 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 `sget`  
*Stages the directory for the target, and then calls 'get'.*

- 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 `set_option`

## Public Attributes

- `do_self_pol`
- `extra_output`  
*Read the reference data.*
- `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.*
- `nptpx`
- `nptfiles`
- `nptsfx`
- `last_traj`
- `RefData`
- `PhasePoints`
- `Labels`
- `w_rho`  
*Density.*
- `w_hvap`
- `w_alpha`
- `w_kappa`
- `w_cp`
- `w_eps0`
- `tempdir`  
*Root directory of the whole project.*
- `rundir`  
*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.*
- `PrintOptionDict`
- `verbose_options`

### 8.30.1 Detailed Description

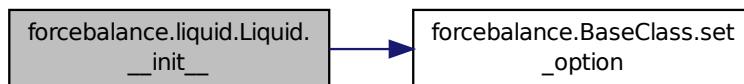
Subclass of Target for liquid property matching.  
Definition at line 51 of file liquid.py.

### 8.30.2 Constructor & Destructor Documentation

**def forcebalance.liquid.Liquid.\_\_init\_\_( self, options, tgt.opts, forcefield )** Create an instance of the class.

Definition at line 56 of file liquid.py.

Here is the call graph for this function:



### 8.30.3 Member Function Documentation

**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 (Hvap) of liquid 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

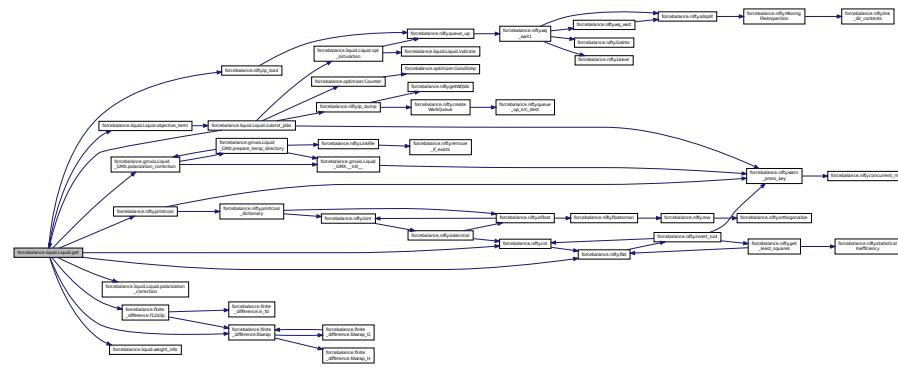
in	<i>mvals</i>	Mathematical parameter values
in	<i>AGrad</i>	Switch to turn on analytic gradient
in	<i>AHess</i>	Switch to turn on analytic Hessian

## 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 401 of file liquid.py.

Here is the call graph for this function:

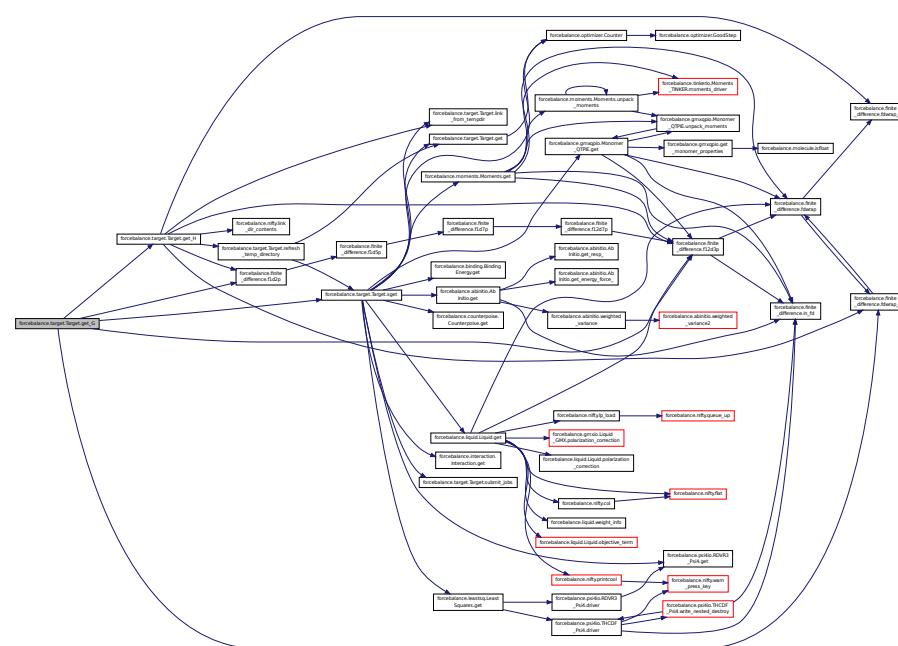


**def forcebalance.target.Target.get\_G( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



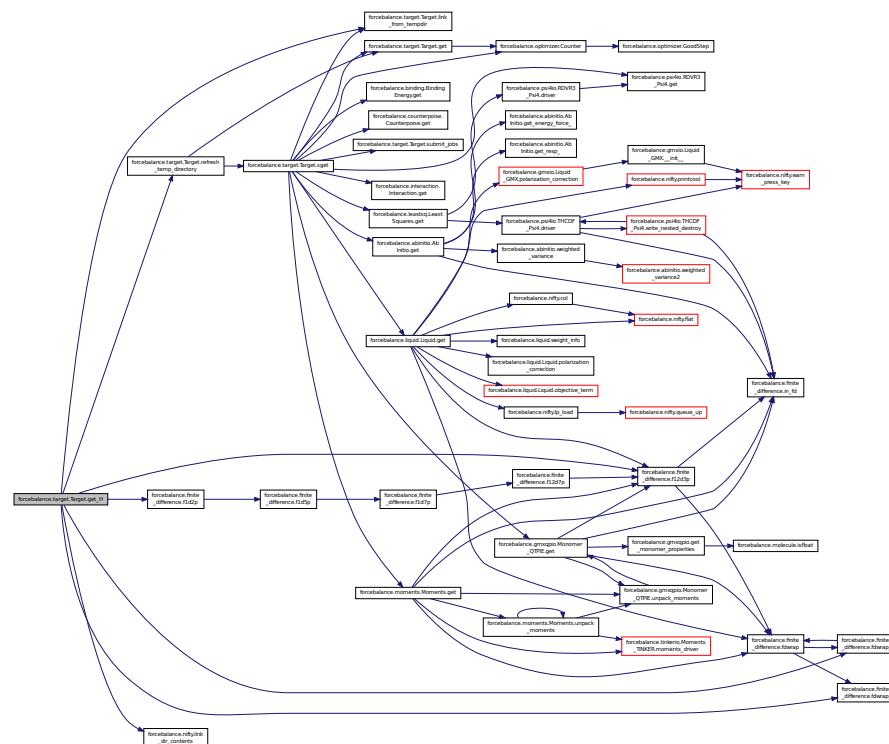
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

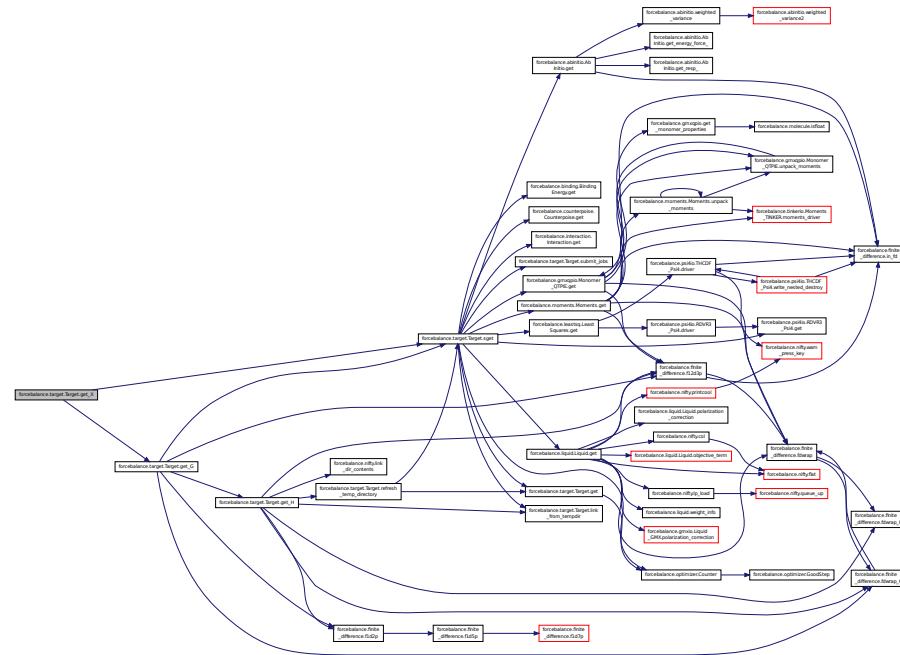
Here is the call graph for this function:



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

Definition at line 155 of file target.py.

Here is the call graph for this function:



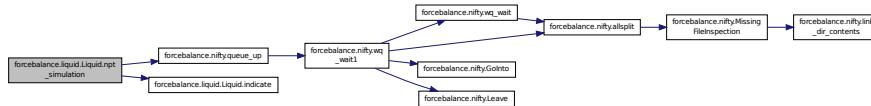
**def forcebalance.liquid.Liquid.indicate ( self )** Definition at line 254 of file liquid.py.

**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

**def forcebalance.liquid.Liquid.npt\_simulation ( self, temperature, pressure, simnum )** Submit a NPT simulation to the Work Queue.

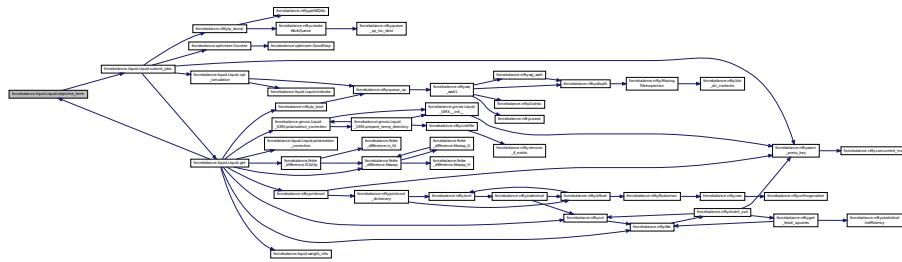
Definition at line 233 of file liquid.py.

Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.objective.term ( self, points, exname, calc, err, grad, name = "Quantity", SubAverage = False ) Definition at line 258 of file liquid.py.
```

Here is the call graph for this function:



**def forcebalance.liquid.Liquid.polarization\_correction ( self, mvals )** Return the self-polarization correction as described in Berendsen et al., JPC 1987.

Definition at line 759 of file liquid.py.

**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.

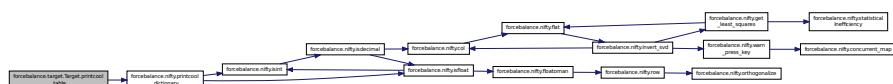
The input should be something like:

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 367 of file target.py.

Here is the call graph for this function:

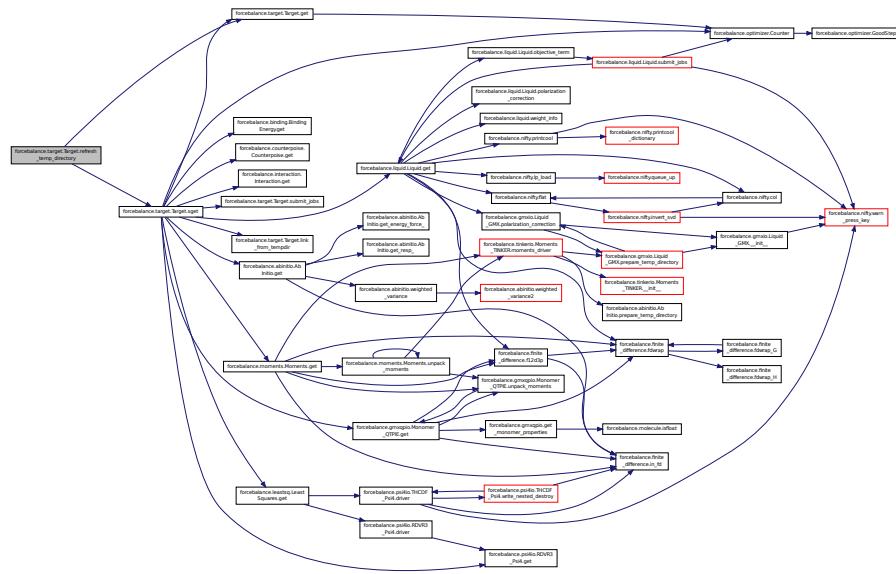


**def forcebalance.liquid.Liquid.read\_data ( self )** Definition at line 141 of file liquid.py.

**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 219 of file target.py.

Here is the call graph for this function:



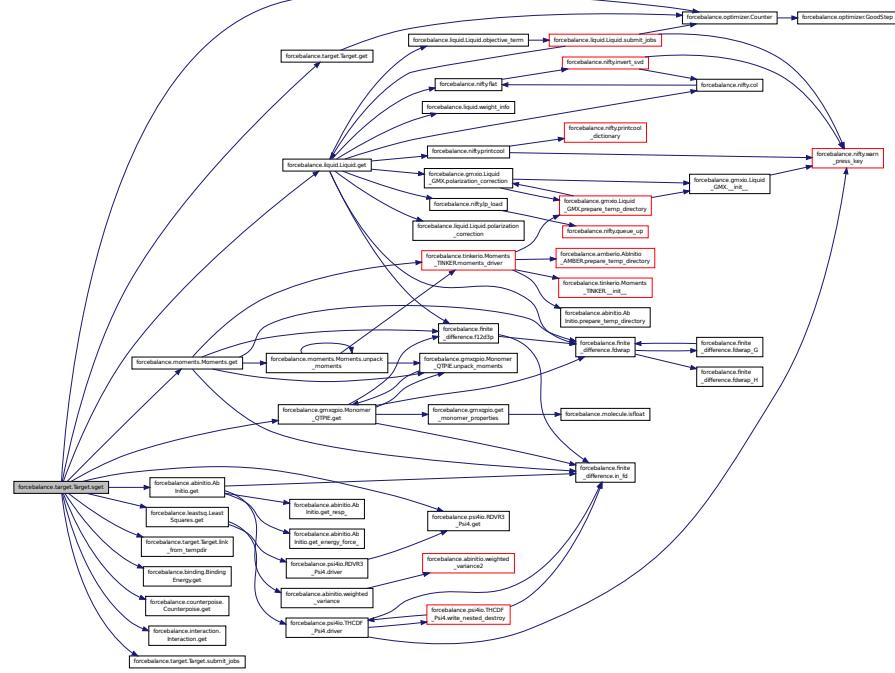
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

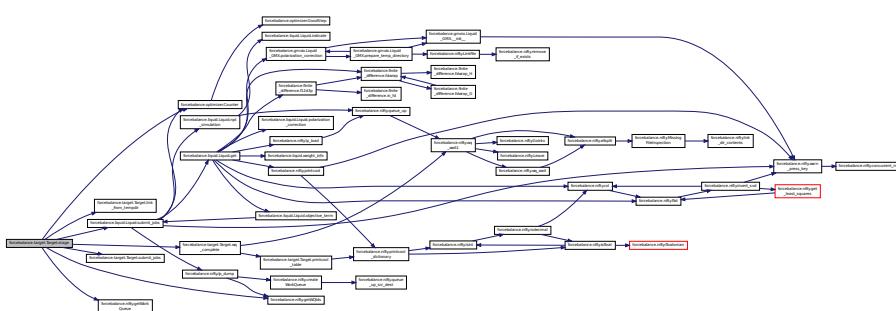


**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.

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

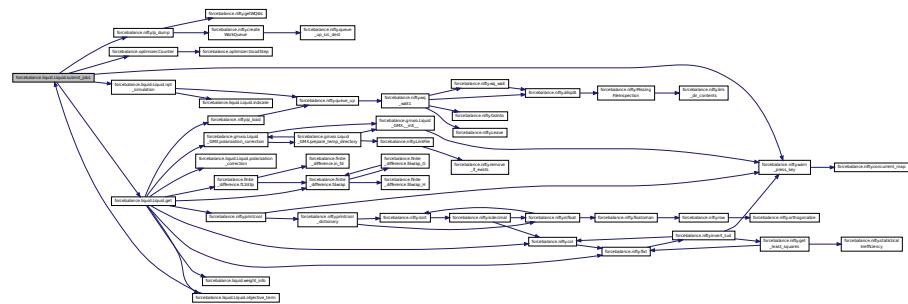
Definition at line 301 of file target.py.

Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.submit_jobs ( self, mvals, AGrad = True, AHess = True ) Definition at line  
336 of file liquid.py.
```

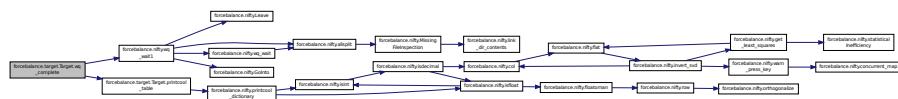
Here is the call graph for this function:



**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 331 of file target.py.

Here is the call graph for this function:



#### **8.30.4 Member Data Documentation**

**forcebalance.liquid.Liquid.do\_self\_pol** Definition at line 99 of file liquid.py.

**forcebalance.liquid.Liquid.extra\_output** Read the reference data.

Prepare the temporary directory Extra platform-dependent data to send back

Definition at line 114 of file liquid.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance\_target.Target.act** [inherited] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance target Target hct [inherited]** Counts how often the Hessian was computed.

`cebalance.target.target.net` [1:111]  
Definition at line 145 of file target.py.

**forcebalance\_liquid** | **Liquid Labels** | Definition at line 221 of file liquid.py

**forcebalance\_liquid** | liquid last trial | Definition at line 139 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 126 of file liquid.py.

**forcebalance\_liquid.Liquid.pptfilec** - Definition at line 120 of file liquid.py

**forcebalance.liquid.Liquid.nptpx** Definition at line 128 of file liquid.py.

**forcebalance.liquid.Liquid.nptsfx** Definition at line 132 of file liquid.py.

**forcebalance.liquid.Liquid.PhasePoints** Definition at line 217 of file liquid.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.liquid.Liquid.RefData** Definition at line 151 of file liquid.py.

**forcebalance.target.Target.rundir [inherited]** 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.

Definition at line 137 of file target.py.

**forcebalance.liquid.Liquid.SavedMVal** Saved force field mvals for all iterations.

Definition at line 122 of file liquid.py.

**forcebalance.liquid.Liquid.SavedTraj** Saved trajectories for all iterations and all temperatures :)

Definition at line 124 of file liquid.py.

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

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.liquid.Liquid.w\_alpha** Definition at line 668 of file liquid.py.

**forcebalance.liquid.Liquid.w\_cp** Definition at line 670 of file liquid.py.

**forcebalance.liquid.Liquid.w\_eps0** Definition at line 671 of file liquid.py.

**forcebalance.liquid.Liquid.w\_hvap** Definition at line 667 of file liquid.py.

**forcebalance.liquid.Liquid.w\_kappa** Definition at line 669 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 666 of file liquid.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

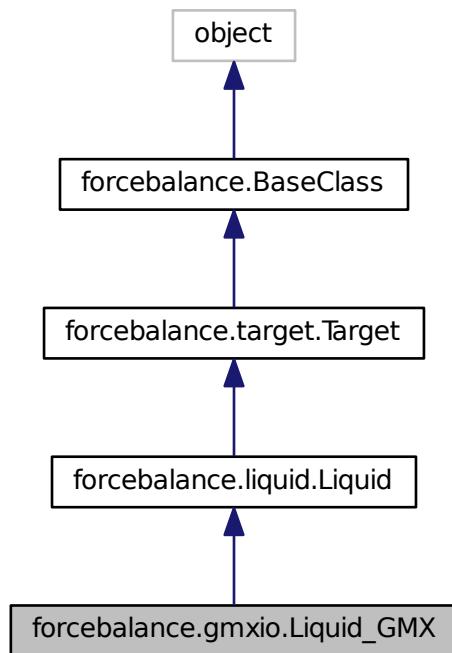
Definition at line 141 of file target.py.

The documentation for this class was generated from the following file:

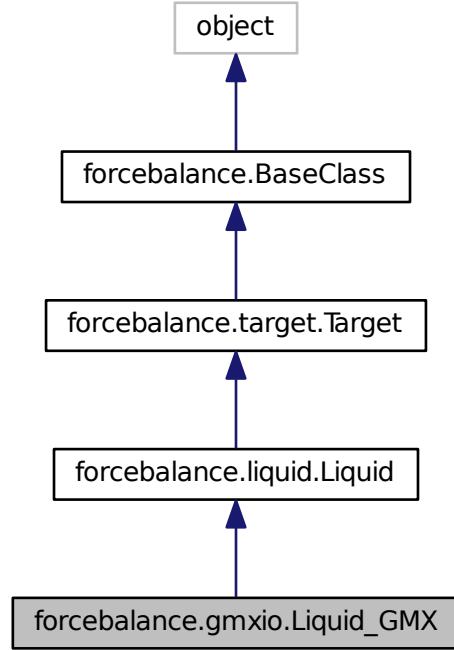
- [liquid.py](#)

## 8.31 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 [prepare\\_temp\\_directory](#)

*Prepare the temporary directory by copying in important files.*

- def [polarization\\_correction](#)
- def [read\\_data](#)
- def [npt\\_simulation](#)

*Submit a NPT simulation to the Work Queue.*

- def [indicate](#)
- def [objective\\_term](#)
- def [submit\\_jobs](#)
- def [get](#)

*Fitting of liquid bulk properties.*

- def [get\\_X](#)

*Computes the objective function contribution without any parametric derivatives.*

- 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 `sget`  
*Stages the directory for the target, and then calls 'get'.*
- 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 `set_option`

## Public Attributes

- `liquid_fnm`
- `liquid_conf`
- `liquid_traj`
- `gas_fnm`
- `nptpx`
- `engine`
- `extra_output`
- `do_self_pol`
- `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.*
- `nptfiles`
- `nptsfy`
- `last_traj`
- `RefData`
- `PhasePoints`
- `Labels`
- `w_rho`  
*Density.*
- `w_hvap`
- `w_alpha`
- `w_kappa`
- `w_cp`
- `w_eps0`
- `tempdir`  
*Root directory of the whole project.*
- `rundir`  
*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.*
- [PrintOptionDict](#)
- [verbose\\_options](#)

### 8.31.1 Detailed Description

Definition at line 681 of file gmxio.py.

### 8.31.2 Constructor & Destructor Documentation

**def forcebalance.gmxio.Liquid\_GMX.\_\_init\_\_ ( *self*, *options*, *tgt\_opts*, *forcefield* )** Definition at line 682 of file gmxio.py.

### 8.31.3 Member Function Documentation

**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 (Hvap) of liquid 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

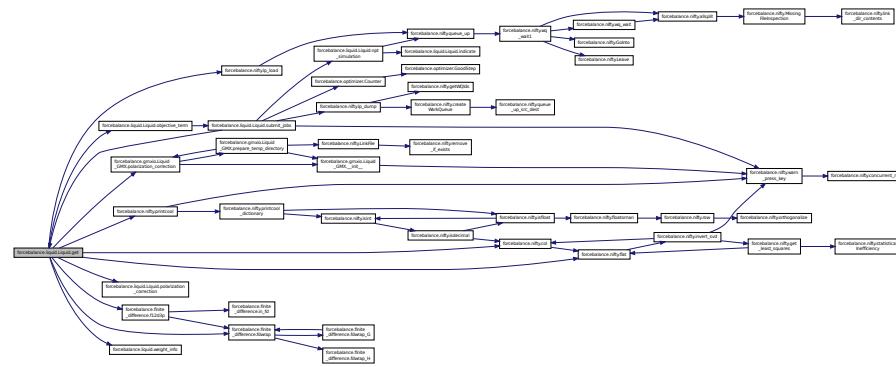
in	<i>mvals</i>	Mathematical parameter values
in	<i>AGrad</i>	Switch to turn on analytic gradient
in	<i>AHess</i>	Switch to turn on analytic Hessian

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 401 of file liquid.py.

Here is the call graph for this function:

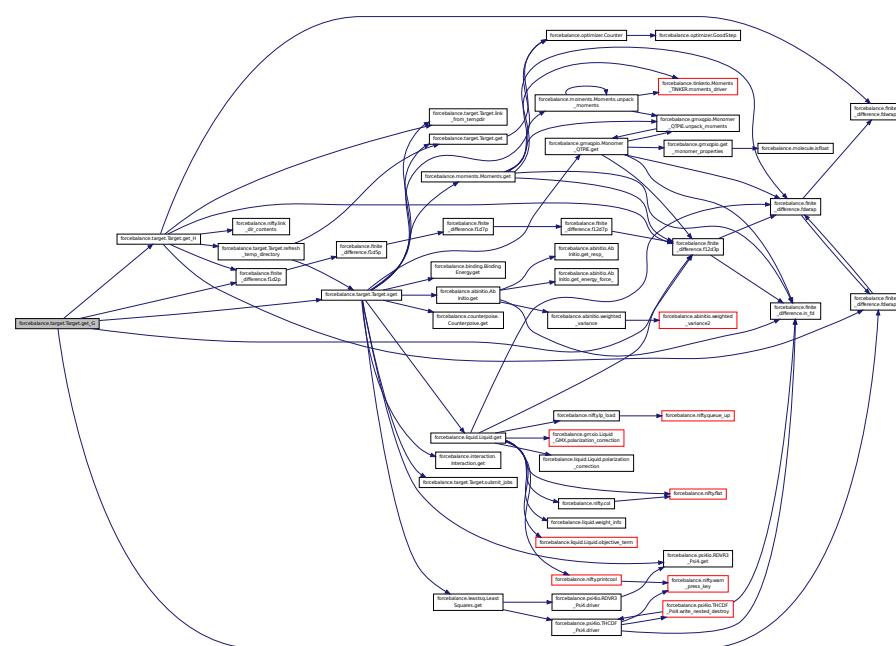


**def forcebalance.Target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



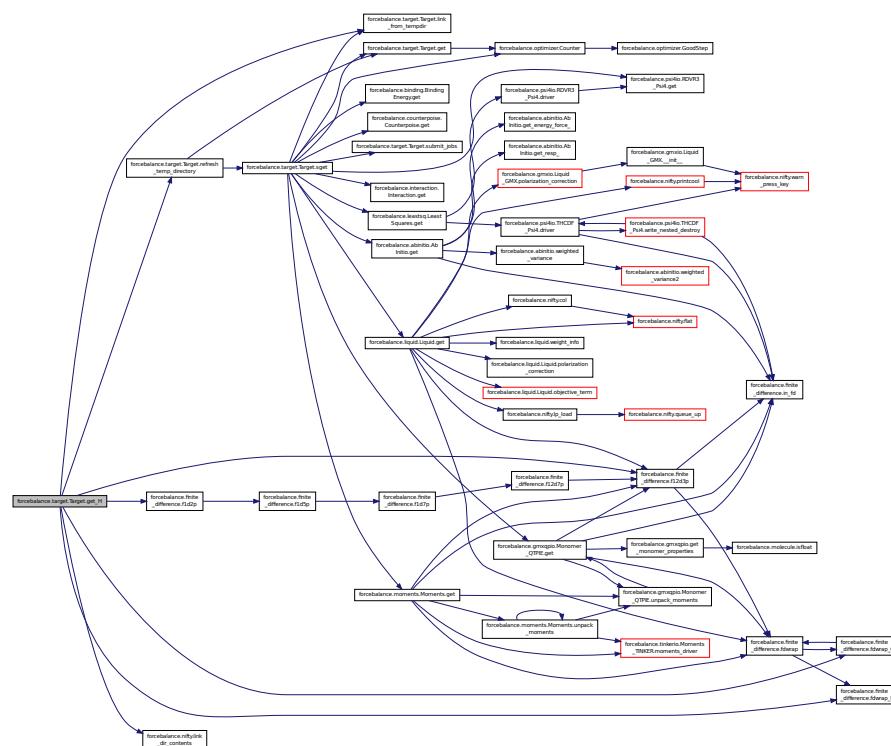
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

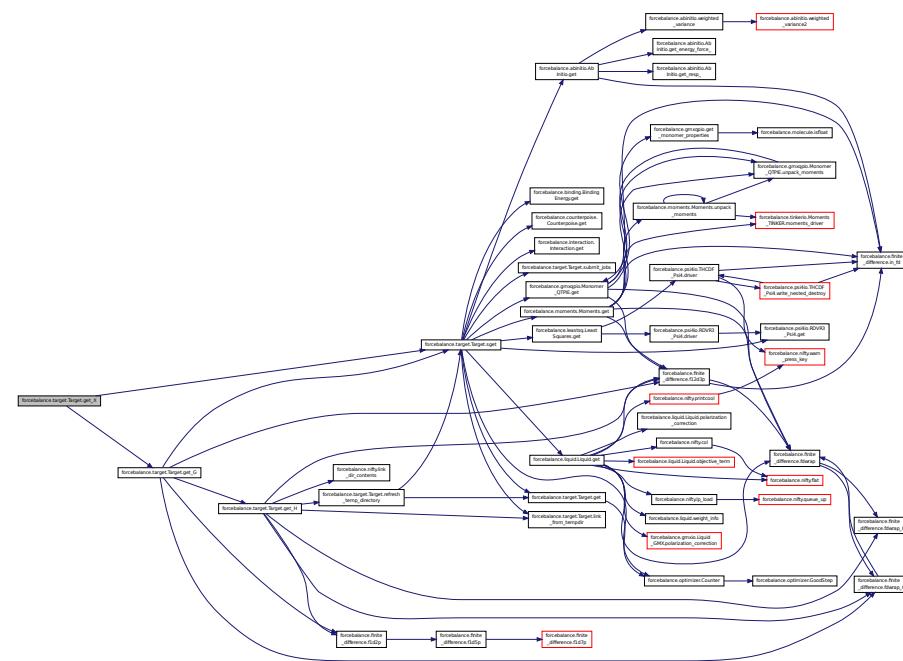
Here is the call graph for this function:



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

Definition at line 155 of file target.py.

Here is the call graph for this function:



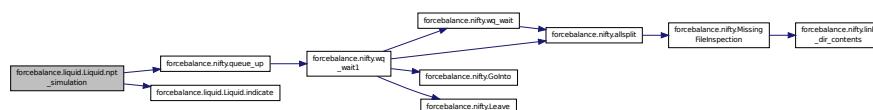
**def** forcebalance.liquid.Liquid.indicate ( self ) [inherited] Definition at line 254 of file liquid.py.

**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

```
def forcebalance.liquid.Liquid.npt_simulation ( self, temperature, pressure, simnum ) [inherited]
Submit a NPT simulation to the Work Queue.
```

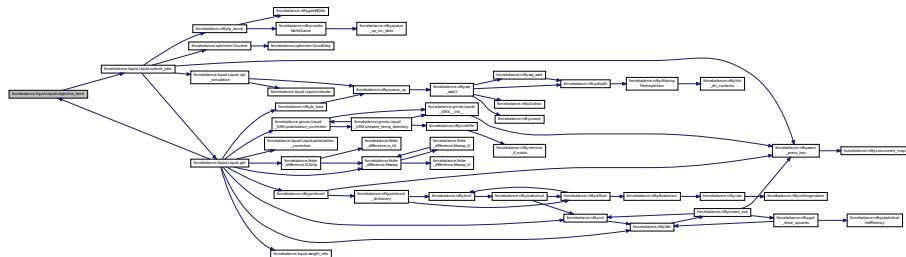
Definition at line 233 of file liquid.py.

Here is the call graph for this function:



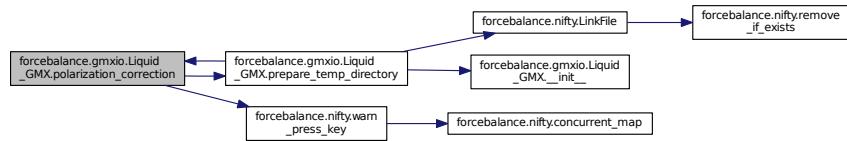
```
def forcebalance.liquid.Liquid.objective_term ( self, points, exname, calc, err, grad, name = "Quantity", SubAverage = False ) [inherited] Definition at line 258 of file liquid.py.
```

Here is the call graph for this function:



**def forcebalance.gmxio.Liquid\_GMX.polarization\_correction ( self, mvals )** Definition at line 726 of file gmxio.py.

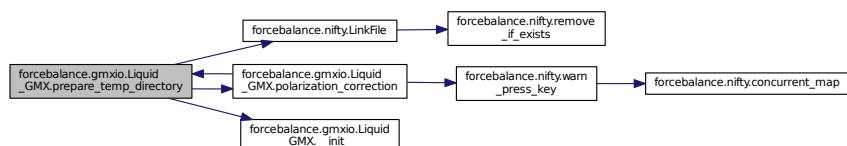
Here is the call graph for this function:



**def forcebalance.gmxio.Liquid\_GMX.prepare\_temp\_directory ( self, options, tgt\_opts )** Prepare the temporary directory by copying in important files.

Definition at line 709 of file gmxio.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.

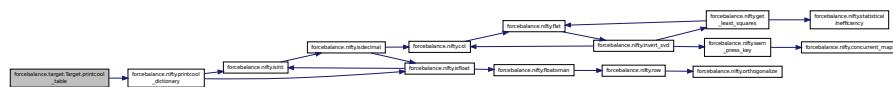
The input should be something like:

## 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 367 of file target.py.

Here is the call graph for this function:

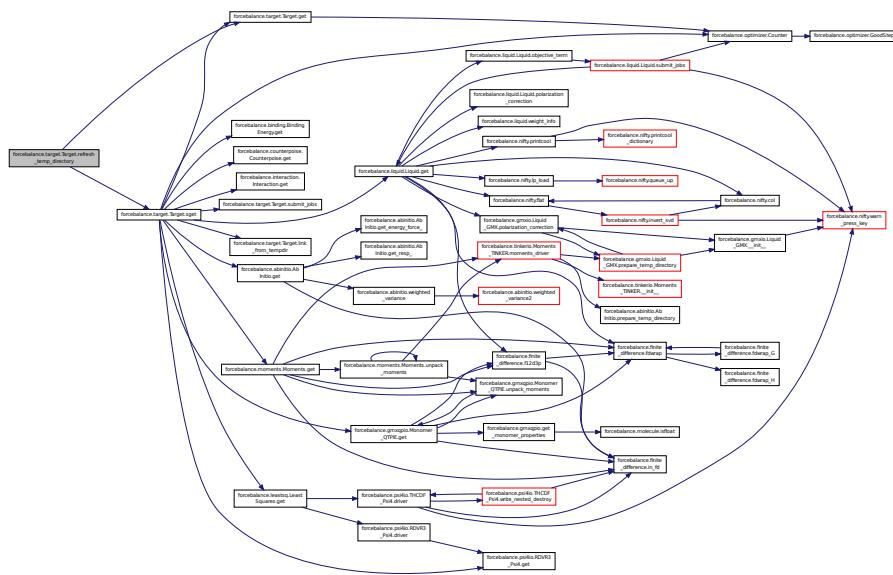


`def forcebalance.liquid.Liquid.read_data( self ) [inherited]` Definition at line 141 of file liquid.py.

**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 219 of file target.py.

Here is the call graph for this function:



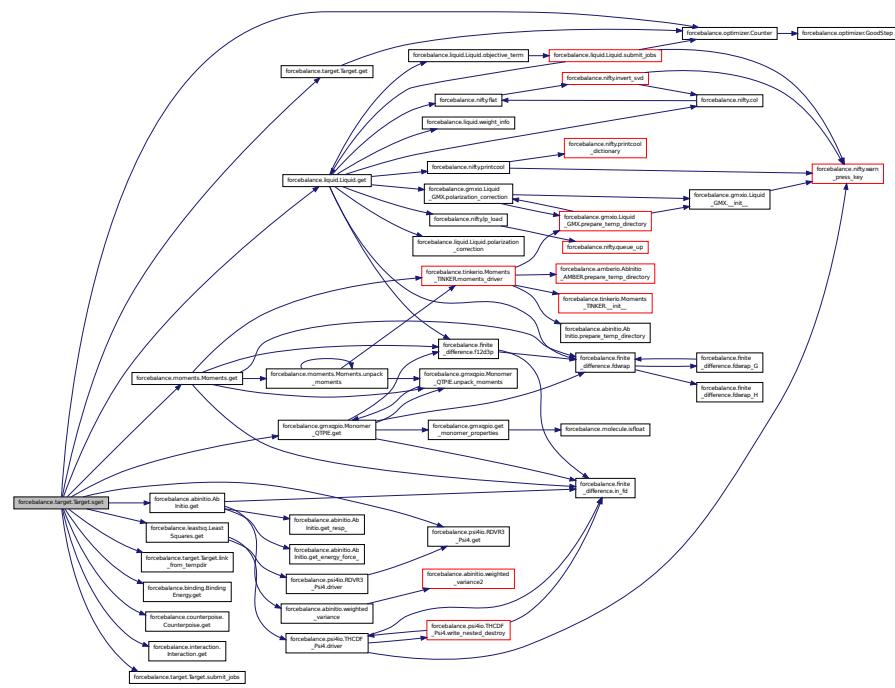
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

**[inherited]** Stages the directory for the target, and then calls 'get'.

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

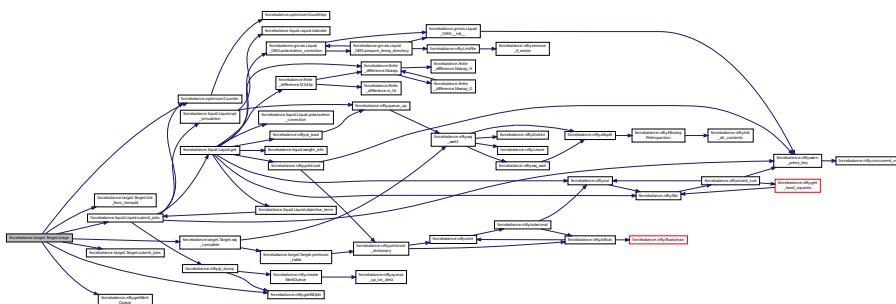


**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.

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

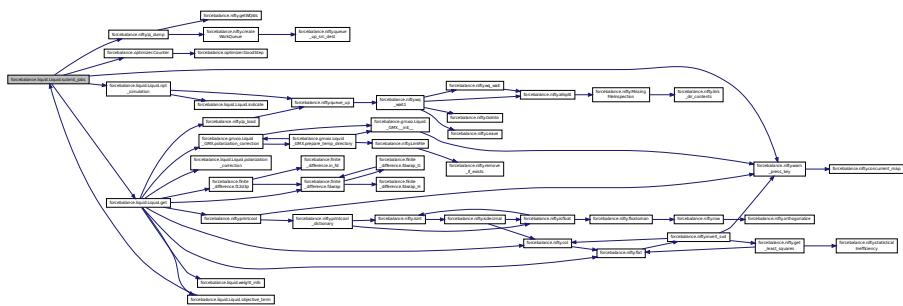
Definition at line 301 of file target.py.

Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.submit.jobs ( self, mvals, AGrad = True, AHess = True ) [inherited]
Definition at line 336 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 331 of file target.py.

Here is the call graph for this function:



#### **8.31.4 Member Data Documentation**

**forcebalance.liquid.Liquid.do\_self\_pol** [inherited] Definition at line 99 of file liquid.py.

**forcebalance.gmxio.Liquid\_GMX.engine** Definition at line 702 of file gmxio.py.

**forcebalance.qmxio.Liquid\_GMX.extra\_output** Definition at line 705 of file qmxio.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

**forcebalance gmxio Liquid GMX gas fnm** Definition at line 689 of file gmxio.py

Journal of Management Education 37(10) 1333–1356 © 2013 Sage Publications

Definition at line 143 of file target.py.

Definition at line 145 of file target.py.

**forcebalance.gmxio.Liquid\_GMX.liquid\_fnm** Definition at line 686 of file gmxio.py.

**forcebalance.gmxio.Liquid\_GMX.liquid\_traj** Definition at line 688 of file gmxio.py.

**forcebalance.liquid.Liquid.MBarEnergy [inherited]** Evaluated energies for all trajectories (i.e. all iterations and all temperatures), using all mvals  
Definition at line 126 of file liquid.py.

**forcebalance.liquid.Liquid.nptfiles [inherited]** Definition at line 130 of file liquid.py.

**forcebalance.gmxio.Liquid\_GMX.nptpx** Definition at line 696 of file gmxio.py.

**forcebalance.liquid.Liquid.nptsfx [inherited]** Definition at line 132 of file liquid.py.

**forcebalance.liquid.Liquid.PhasePoints [inherited]** Definition at line 217 of file liquid.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.liquid.Liquid.RefData [inherited]** Definition at line 151 of file liquid.py.

**forcebalance.target.Target.rundir [inherited]** 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.

Definition at line 137 of file target.py.

**forcebalance.liquid.Liquid.SavedMVal [inherited]** Saved force field mvals for all iterations.  
Definition at line 122 of file liquid.py.

**forcebalance.liquid.Liquid.SavedTraj [inherited]** Saved trajectories for all iterations and all temperatures :)  
Definition at line 124 of file liquid.py.

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

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.liquid.Liquid.w\_alpha [inherited]** Definition at line 668 of file liquid.py.

**forcebalance.liquid.Liquid.w\_cp [inherited]** Definition at line 670 of file liquid.py.

**forcebalance.liquid.Liquid.w\_eps0 [inherited]** Definition at line 671 of file liquid.py.

**forcebalance.liquid.Liquid.w\_hvap [inherited]** Definition at line 667 of file liquid.py.

**forcebalance.liquid.Liquid.w\_kappa** [inherited] Definition at line 669 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 666 of file liquid.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

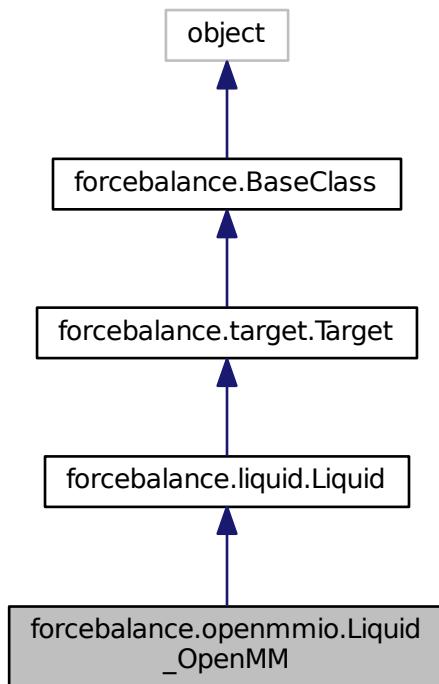
Definition at line 141 of file target.py.

The documentation for this class was generated from the following file:

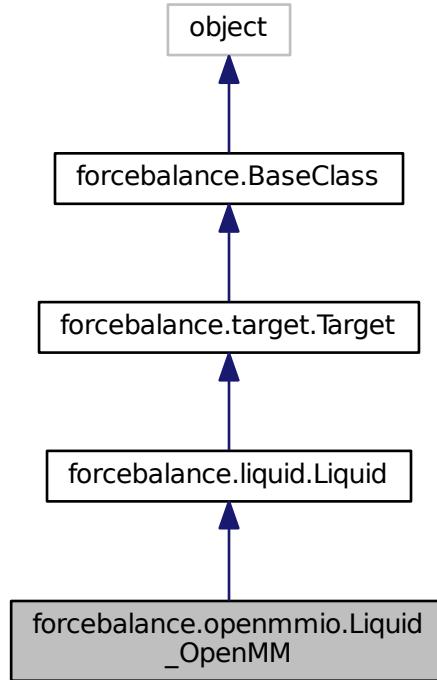
- [gmxio.py](#)

## 8.32 forcebalance.openmmio.Liquid\_OpenMM Class Reference

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 `polarization_correction`
- def `read_data`
- def `npt_simulation`

*Submit a NPT simulation to the Work Queue.*
- def `indicate`
- def `objective_term`
- def `submit_jobs`
- def `get`

*Fitting of liquid bulk properties.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- 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 [sget](#)  
*Stages the directory for the target, and then calls 'get'.*
- 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 [set\\_option](#)

## Public Attributes

- [mpdb](#)
- [platform](#)
- [msim](#)
- [liquid\\_fnm](#)
- [liquid\\_conf](#)
- [liquid\\_traj](#)
- [gas\\_fnm](#)
- [engine](#)
- [do\\_self\\_pol](#)
- [extra\\_output](#)  
*Read the reference data.*
- [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.*
- [nptpx](#)
- [nptfiles](#)
- [nptsfx](#)
- [last\\_traj](#)
- [RefData](#)
- [PhasePoints](#)
- [Labels](#)
- [w\\_rho](#)  
*Density.*
- [w\\_hvap](#)
- [w\\_alpha](#)
- [w\\_kappa](#)
- [w\\_cp](#)
- [w\\_eps0](#)
- [tempdir](#)  
*Root directory of the whole project.*
- [rundir](#)

*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.*
- **PrintOptionDict**
- **verbose\_options**

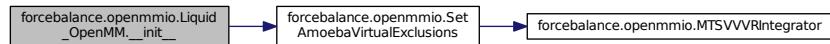
### 8.32.1 Detailed Description

Definition at line 349 of file openmmio.py.

### 8.32.2 Constructor & Destructor Documentation

**def forcebalance.openmmio.Liquid\_OpenMM\_\_init\_\_( self, options, tgt\_opts, forcefield )** Definition at line 350 of file openmmio.py.

Here is the call graph for this function:



### 8.32.3 Member Function Documentation

**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

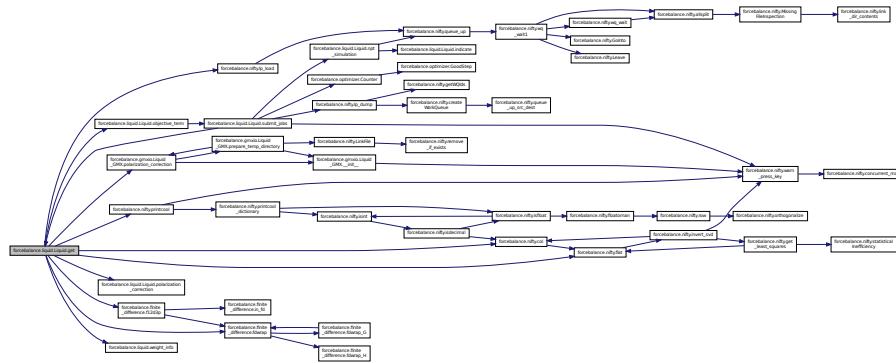
`in`      *AHess*      Switch to turn on analytic Hessian

## 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 401 of file liquid.py.

Here is the call graph for this function:

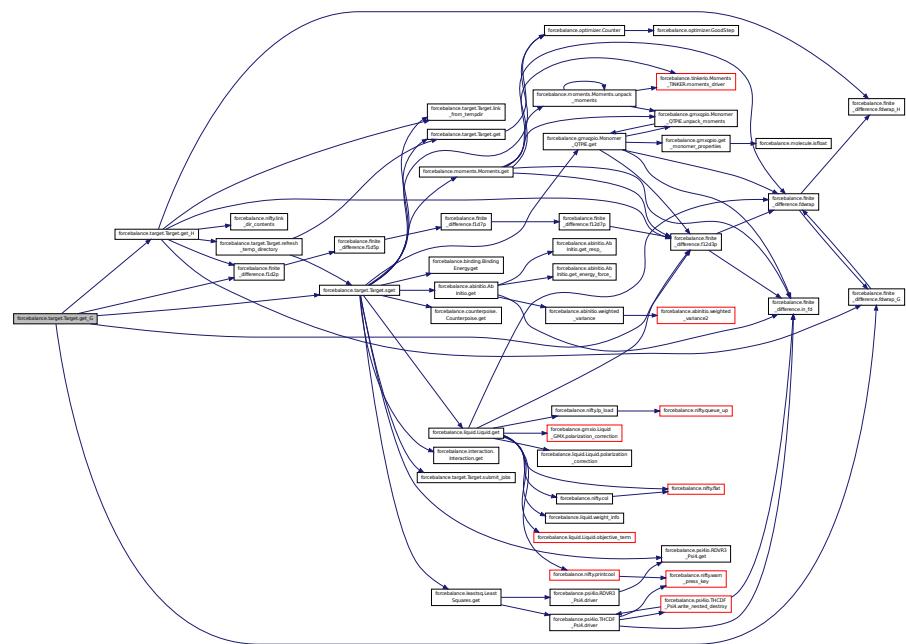


**def forcebalance.target.Target.get\_G( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



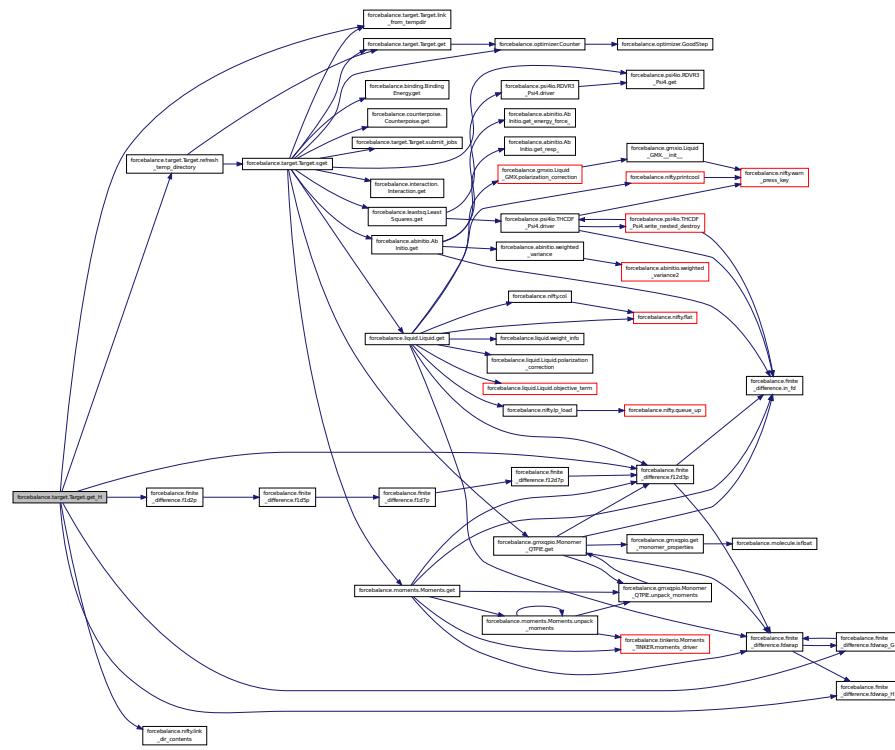
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

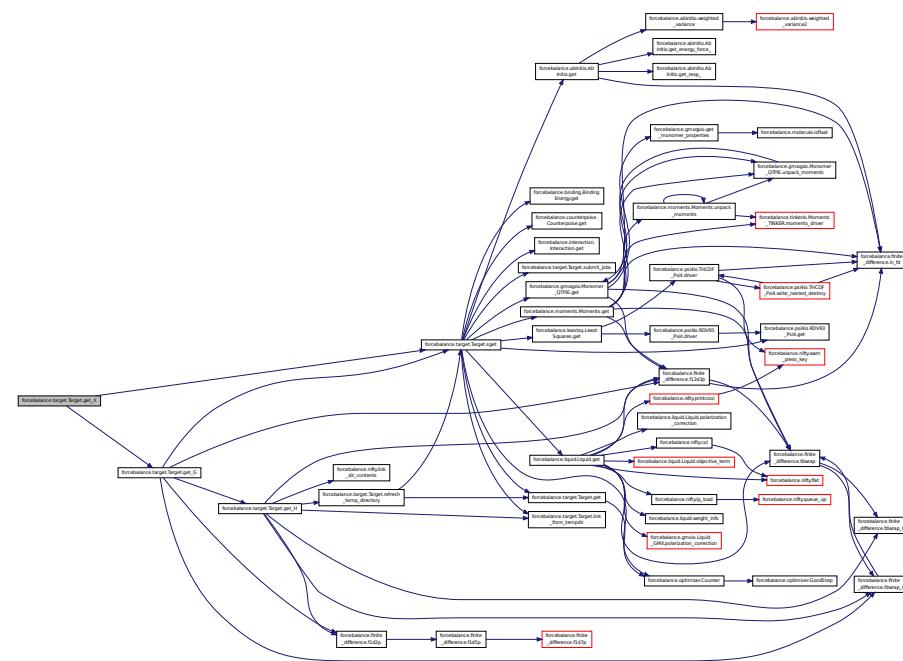
Here is the call graph for this function:



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

Definition at line 155 of file target.py.

Here is the call graph for this function:



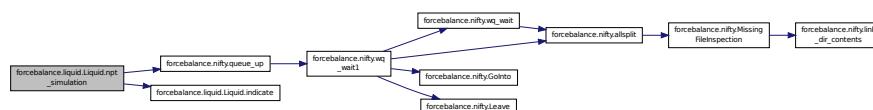
**def** forcebalance.liquid.Liquid.indicate ( self ) [inherited] Definition at line 254 of file liquid.py.

**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

```
def forcebalance.liquid.Liquid.npt_simulation ( self, temperature, pressure, simnum ) [inherited]
Submit a NPT simulation to the Work Queue.
```

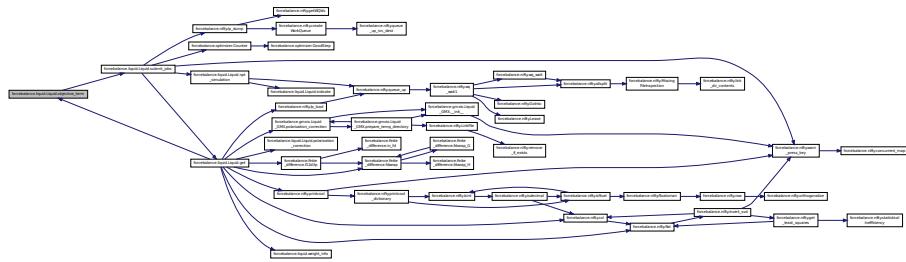
Definition at line 233 of file liquid.py.

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 258 of file liquid.py.
```

Here is the call graph for this function:



**def forcebalance.openmmio.Liquid\_OpenMM.polarization\_correction ( self, mvals )** Definition at line 398 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.openmmio.Liquid\_OpenMM.prepare\_temp\_directory ( self, options, tgt\_opts )** Prepare the temporary directory by copying in important files.

Definition at line 391 of file openmmio.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:

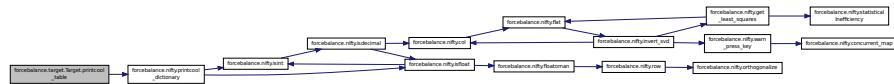
Parameters

<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.

<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 367 of file target.py.

Here is the call graph for this function:

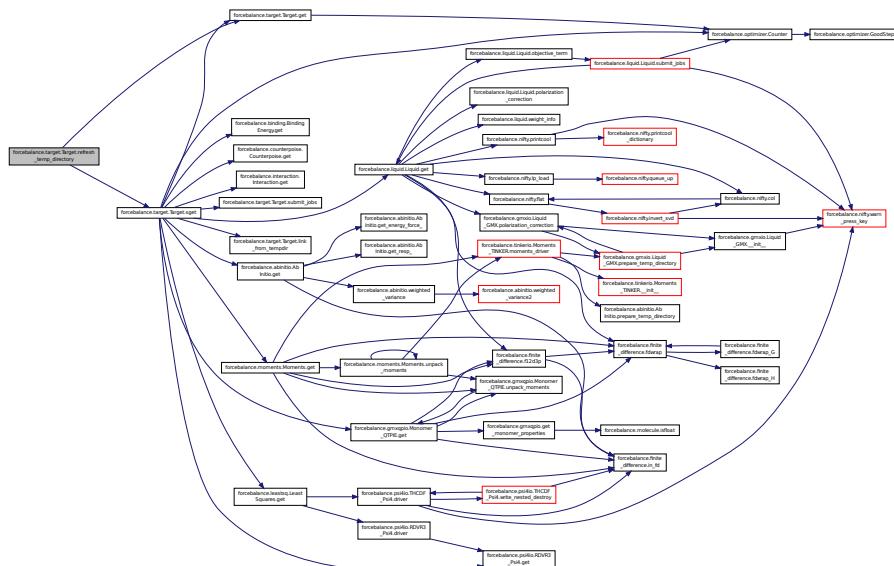


**def forcebalance.liquid.Liquid.read\_data ( self ) [inherited]** Definition at line 141 of file liquid.py.

**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 219 of file target.py.

Here is the call graph for this function:



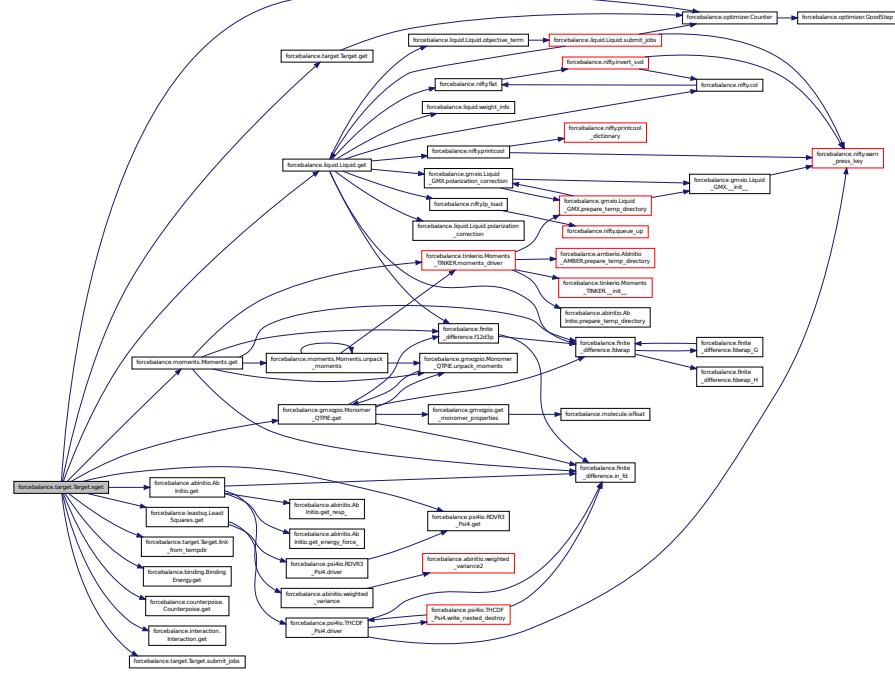
**def forcebalance.BaseClass.set\_option( self, in\_dict, src\_key, dest\_key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 32 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

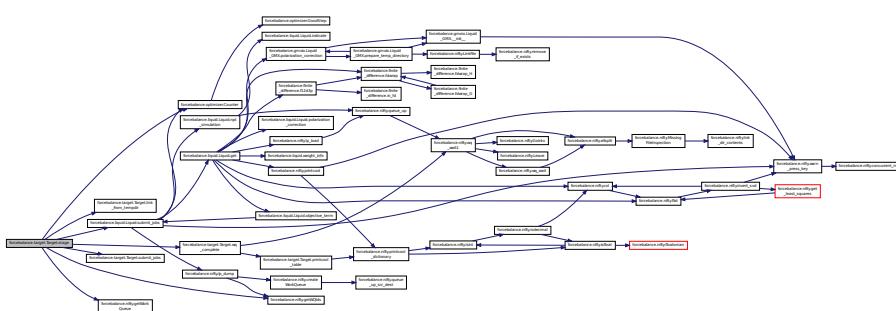


```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

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

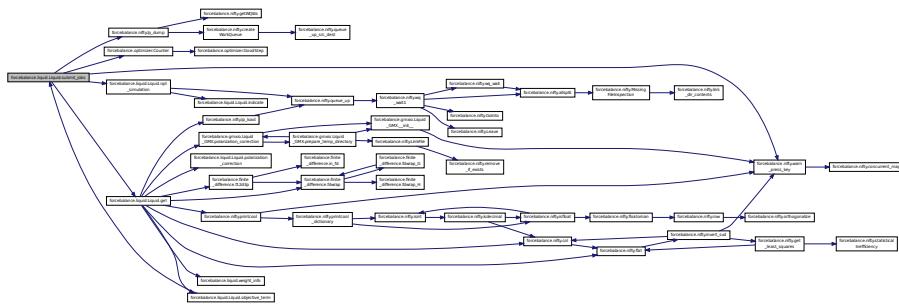
Definition at line 301 of file target.py.

Here is the call graph for this function:



**def** forcebalance.liquid.Liquid.submit\_jobs ( self, mvals, AGrad = True, AHess = True ) [inherited]  
Definition at line 336 of file liquid.py.

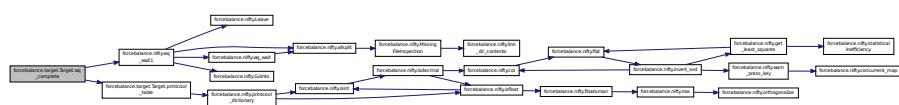
Here is the call graph for this function:



**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 331 of file target.py.

Here is the call graph for this function:



#### **8.32.4 Member Data Documentation**

**forcebalance.liquid.Liquid.do\_self\_pol** [inherited] Definition at line 99 of file liquid.py.

**forcebalance.openmmio.Liquid\_OpenMM.engine** Definition at line 387 of file openmmio.py.

**forcebalance.liquid.Liquid.extra\_output** [inherited] Read the reference data.

Prepare the temporary directory Extra platform-dependent data to send back

Definition at line 114 of file liquid.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.openmmio.Liquid\_OpenMM.gas\_fnm** Definition at line 374 of file openmmio.py.

10. The following table summarizes the results of the study. The first column lists the variables, the second column lists the sample size, and the third column lists the estimated effect sizes.

Definition at line 143 of file target.py.

#### SECTION 5: THE USE OF THE LANGUAGE

Definition at line 145 of file target.py.

Definition at line 145 of file target.py.

`forcebalance.liquid.Liquid.Labels` [Inherited] Definition at line 221 of file liquid.py.

**forcebalance.liquid.Liquid.last\_traj** [inherited] Definition at line 139 of file liquid.py.

**forcebalance.openmmio.Liquid\_OpenMM.liquid\_conf** Definition at line 372 of file openmmio.py.

**forcebalance.openmmio.Liquid\_OpenMM.liquid\_fnm** Definition at line 371 of file openmmio.py.

**forcebalance.openmmio.Liquid\_OpenMM.liquid\_traj** Definition at line 373 of file openmmio.py.

**forcebalance.liquid.Liquid.MBarEnergy [inherited]** Evaluated energies for all trajectories (i.e.

all iterations and all temperatures), using all mvals

Definition at line 126 of file liquid.py.

**forcebalance.openmmio.Liquid\_OpenMM.mpdb** Definition at line 358 of file openmmio.py.

**forcebalance.openmmio.Liquid\_OpenMM.msim** Definition at line 369 of file openmmio.py.

**forcebalance.liquid.Liquid.nptfiles [inherited]** Definition at line 130 of file liquid.py.

**forcebalance.liquid.Liquid.nptpx [inherited]** Definition at line 128 of file liquid.py.

**forcebalance.liquid.Liquid.nptsfx [inherited]** Definition at line 132 of file liquid.py.

**forcebalance.liquid.Liquid.PhasePoints [inherited]** Definition at line 217 of file liquid.py.

**forcebalance.openmmio.Liquid\_OpenMM.platform** Definition at line 367 of file openmmio.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.liquid.Liquid.RefData [inherited]** Definition at line 151 of file liquid.py.

**forcebalance.target.Target.rundir [inherited]** 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.

Definition at line 137 of file target.py.

**forcebalance.liquid.Liquid.SavedMVal [inherited]** Saved force field mvals for all iterations.

Definition at line 122 of file liquid.py.

**forcebalance.liquid.Liquid.SavedTraj [inherited]** Saved trajectories for all iterations and all temperatures :)

Definition at line 124 of file liquid.py.

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

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.liquid.Liquid.w\_alpha** [inherited] Definition at line 668 of file liquid.py.

**forcebalance.liquid.Liquid.w\_cp** [inherited] Definition at line 670 of file liquid.py.

**forcebalance.liquid.Liquid.w\_eps0** [inherited] Definition at line 671 of file liquid.py.

**forcebalance.liquid.Liquid.w\_hvap** [inherited] Definition at line 667 of file liquid.py.

**forcebalance.liquid.Liquid.w\_kappa** [inherited] Definition at line 669 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 666 of file liquid.py.

**forcebalance.target.Target.xct** [inherited] Counts how often the objective function was computed.

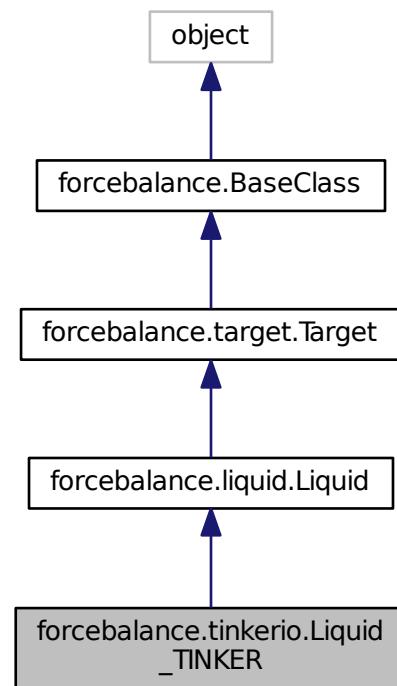
Definition at line 141 of file target.py.

The documentation for this class was generated from the following file:

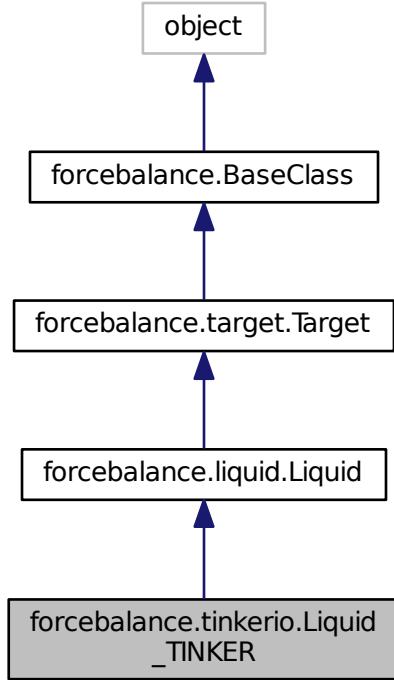
- [openmmio.py](#)

### 8.33 forcebalance.tinkerio.Liquid\_TINKER Class Reference

Inheritance diagram for forcebalance.tinkerio.Liquid\_TINKER:



Collaboration diagram for forcebalance.tinkerio.Liquid\_TINKER:



### Public Member Functions

- def `_init_`
- def `prepare_temp_directory`

*Prepare the temporary directory by copying in important files.*
- def `npt_simulation`

*Submit a NPT simulation to the Work Queue.*
- def `read_data`
- def `indicate`
- def `objective_term`
- def `submit_jobs`
- def `get`

*Fitting of liquid bulk properties.*
- def `polarization_correction`

*Return the self-polarization correction as described in Berendsen et al., JPC 1987.*
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- 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 [sget](#)  
*Stages the directory for the target, and then calls 'get'.*
- 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 [set\\_option](#)

## Public Attributes

- [DynDict](#)  
*Read the reference data.*
- [DynDict\\_New](#)  
*Saved force field mvals for all iterations.*
- [do\\_self\\_pol](#)  
*Saved trajectories for all iterations and all temperatures :)*
- [extra\\_output](#)  
*Evaluuated energies for all trajectories (i.e.*
- [nptpx](#)  
• [nptfiles](#)  
• [nptsfx](#)  
• [last\\_traj](#)  
• [RefData](#)  
• [PhasePoints](#)  
• [Labels](#)  
• [w\\_rho](#)  
*Density.*
- [w\\_hvap](#)  
• [w\\_alpha](#)  
• [w\\_kappa](#)  
• [w\\_cp](#)  
• [w\\_eps0](#)  
• [tempdir](#)  
*Root directory of the whole project.*
- [rundir](#)  
*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.*
- `PrintOptionDict`
- `verbose_options`

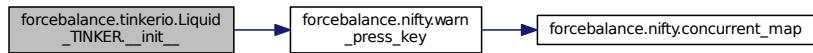
### 8.33.1 Detailed Description

Definition at line 233 of file `tinkerio.py`.

### 8.33.2 Constructor & Destructor Documentation

`def forcebalance.tinkerio.Liquid_TINKER.__init__ ( self, options, tgt_opts, forcefield )` Definition at line 234 of file `tinkerio.py`.

Here is the call graph for this function:



### 8.33.3 Member Function Documentation

`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 (Hvap) of liquid 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

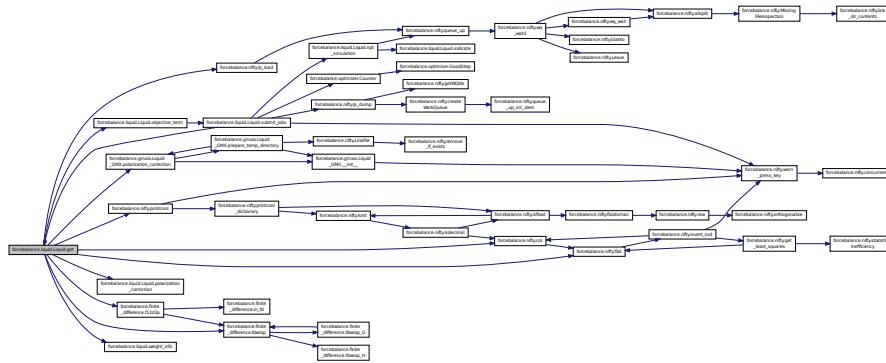
in	<code>mvals</code>	Mathematical parameter values
in	<code>AGrad</code>	Switch to turn on analytic gradient
in	<code>AHess</code>	Switch to turn on analytic Hessian

#### 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 401 of file `liquid.py`.

Here is the call graph for this function:

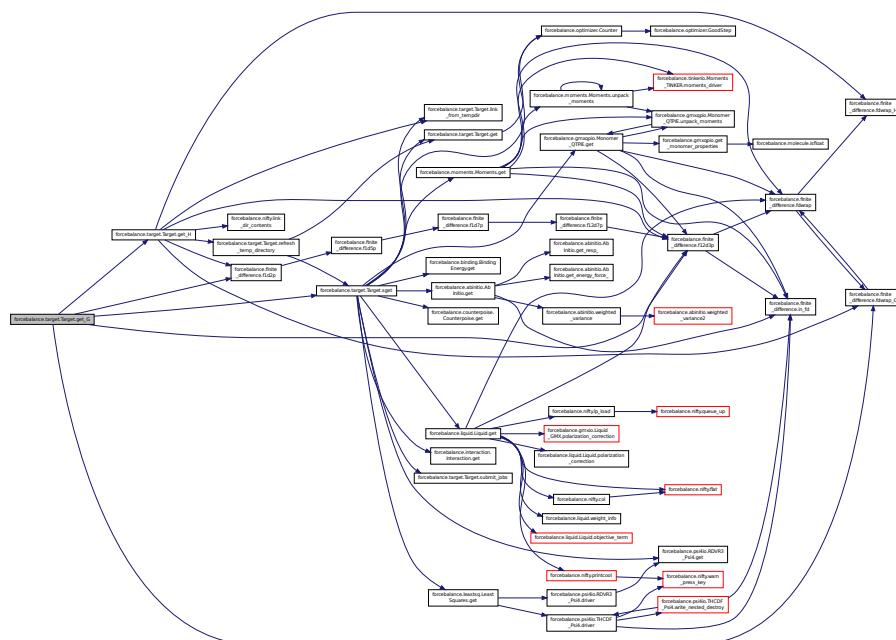


**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



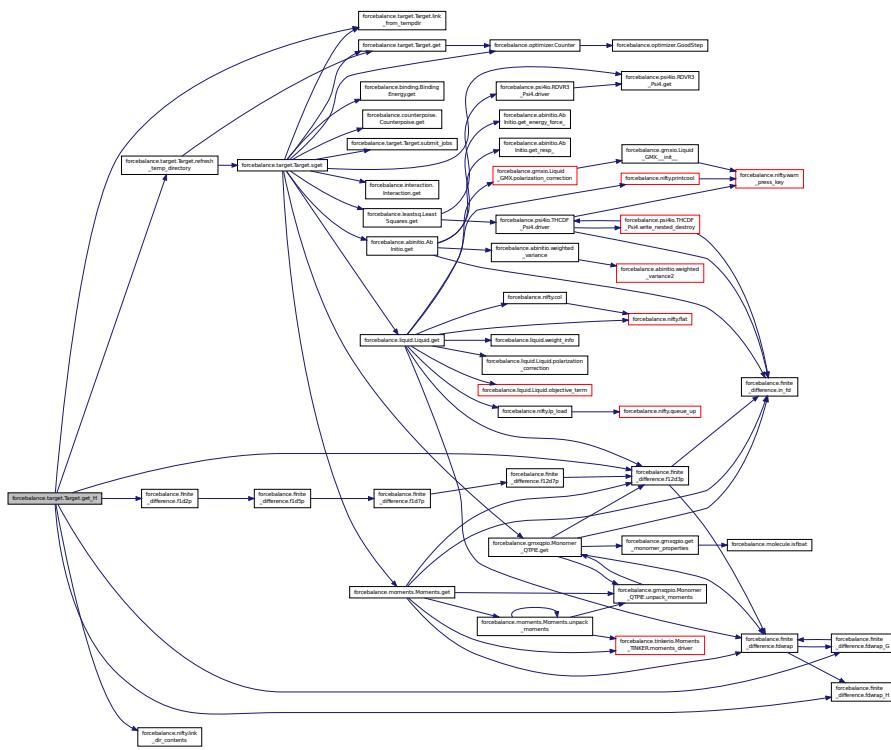
**def forcebalance.target.Target.get\_H ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

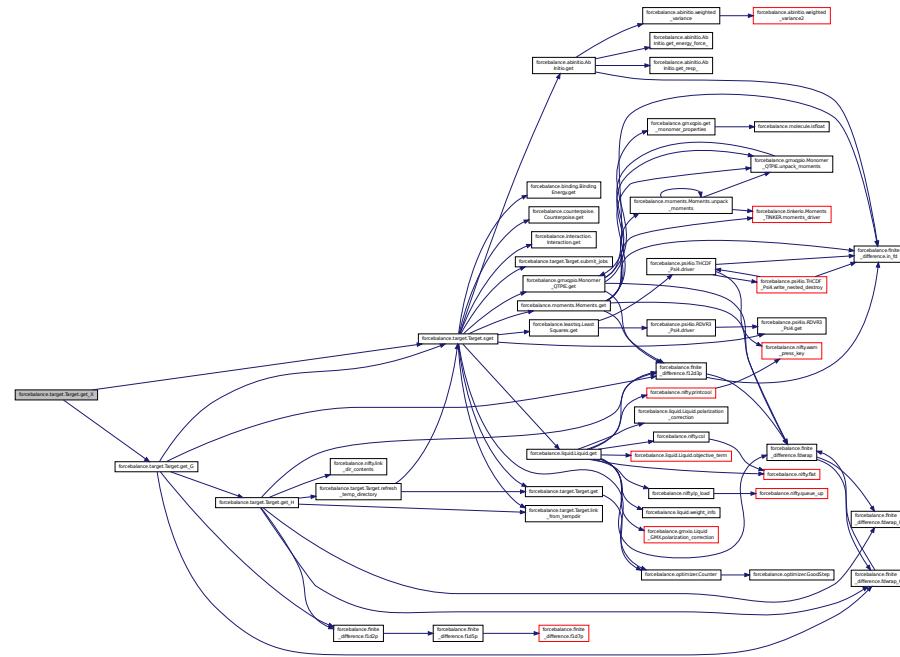
Here is the call graph for this function:



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

Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.liquid.Liquid.indicate ( self ) [inherited]** Definition at line 254 of file liquid.py.

**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

**def forcebalance.tinkerio.Liquid\_TINKER.npt\_simulation ( self, temperature, pressure, simnum )** Submit a NPT simulation to the Work Queue.

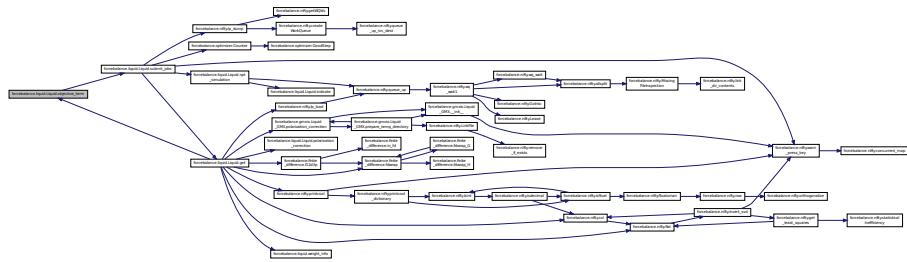
Definition at line 261 of file tinkerio.py.

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 258 of file liquid.py.
```

Here is the call graph for this function:



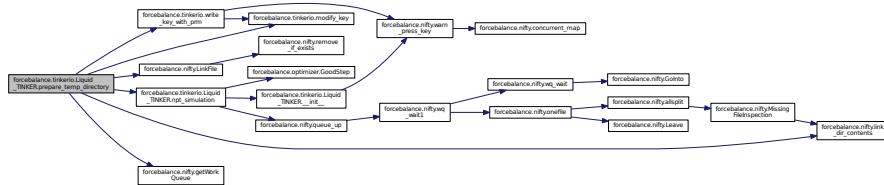
**def forcebalance.liquid.Liquid.polarization\_correction ( self, mvals ) [inherited]** Return the self-polarization correction as described in Berendsen et al., JPC 1987.

Definition at line 759 of file liquid.py.

```
def forcebalance.tinkerio.Liquid_TINKER.prepare_temp_directory ( self, options, tgt_opts ) Prepare the temporary directory by copying in important files.
```

Definition at line 243 of file tinkerio.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:

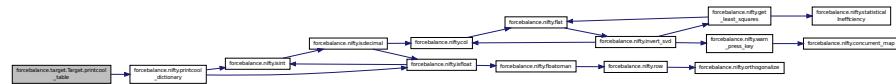
## Parameters

<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 367 of file target.py.

Here is the call graph for this function:

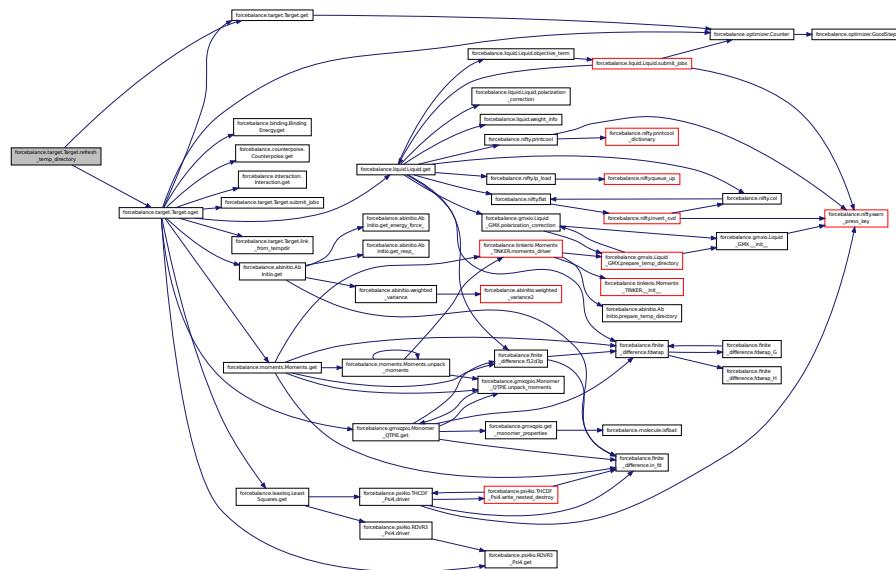


**def forcebalance.liquid.Liquid.read\_data ( self ) [inherited]** Definition at line 141 of file liquid.py.

**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 219 of file target.py.

Here is the call graph for this function:



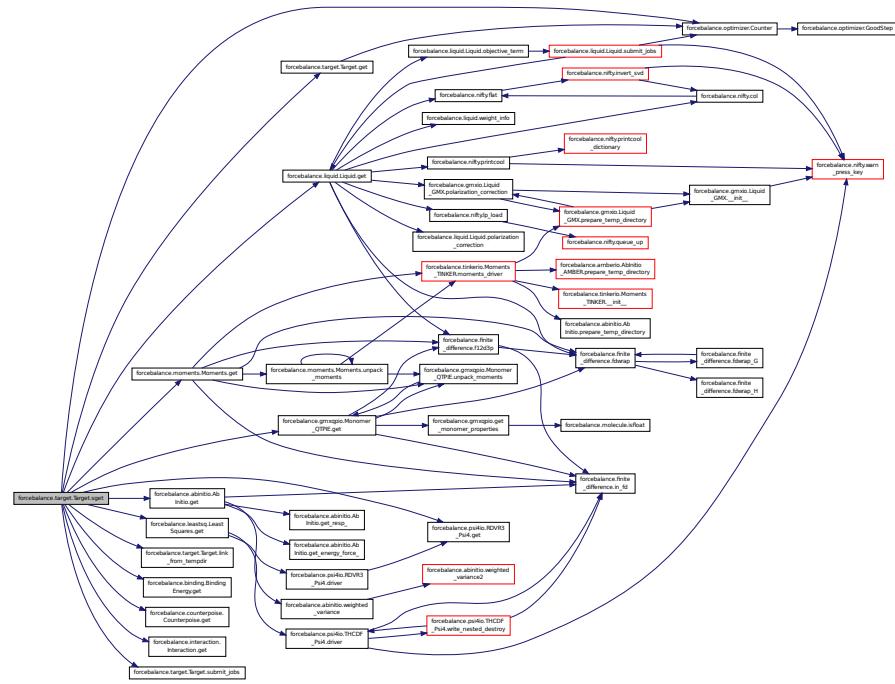
**def forcebalance.BaseClass.set\_option( self, in\_dict, src\_key, dest\_key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 32 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

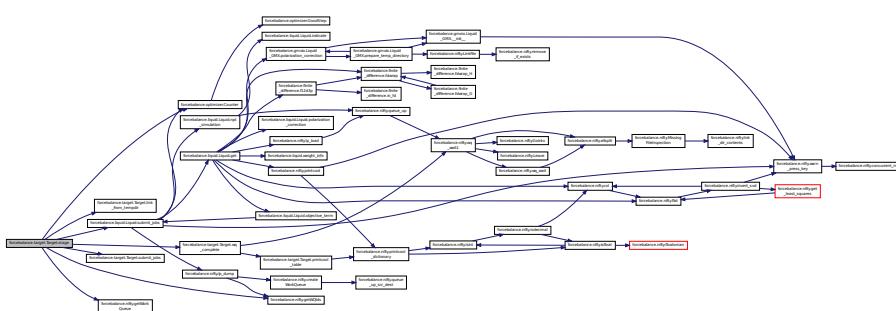


**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.

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

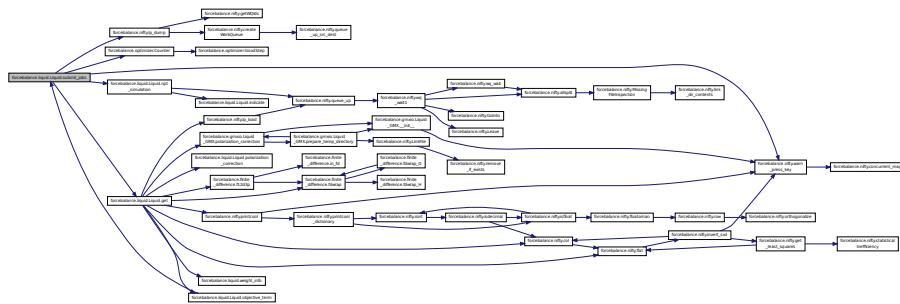
Definition at line 301 of file target.py.

Here is the call graph for this function:



```
def forcebalance.liquid.Liquid.submit.jobs( self, mvals, AGrad = True, AHess = True ) [inherited]  
Definition at line 336 of file liquid.py.
```

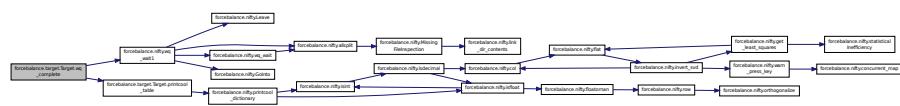
Here is the call graph for this function:



**def forcebalance.liquid.Liquid.\_compute( self ) [inherited]** This method determines whether the Work Queue tasks for the current target have completed.

Definition at line 331 of file liquid.py.

Here is the call graph for this function:



#### 8.33.4 Member Data Documentation

**forcebalance.liquid.Liquid.do\_self\_pol [inherited]** Definition at line 99 of file liquid.py.

**forcebalance.tinkerio.Liquid\_TINKER.DynDict** Definition at line 236 of file tinkerio.py.

**forcebalance.tinkerio.Liquid\_TINKER.DynDict\_New** Definition at line 237 of file tinkerio.py.

**forcebalance.liquid.Liquid.extra\_output [inherited]** Read the reference data.

Prepare the temporary directory Extra platform-dependent data to send back

Definition at line 114 of file liquid.py.

**forcebalance.target.Target.FF [inherited]** Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.target.Target.gct [inherited]** Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.

Definition at line 145 of file target.py.

**forcebalance.liquid.Liquid.Labels [inherited]** Definition at line 221 of file liquid.py.

**forcebalance.liquid.Liquid.last\_traj [inherited]** Definition at line 139 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 126 of file liquid.py.

**forcebalance.liquid.Liquid.nptfiles [inherited]** Definition at line 130 of file liquid.py.

**forcebalance.liquid.Liquid.nptpx [inherited]** Definition at line 128 of file liquid.py.

**forcebalance.liquid.Liquid.nptsfx [inherited]** Definition at line 132 of file liquid.py.

**forcebalance.liquid.Liquid.PhasePoints [inherited]** Definition at line 217 of file liquid.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.liquid.Liquid.RefData [inherited]** Definition at line 151 of file liquid.py.

**forcebalance.target.Target.rundir [inherited]** 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.

  Definition at line 137 of file target.py.

**forcebalance.liquid.Liquid.SavedMVal [inherited]** Saved force field mvals for all iterations.

  Definition at line 122 of file liquid.py.

**forcebalance.liquid.Liquid.SavedTraj [inherited]** Saved trajectories for all iterations and all temperatures :)

  Definition at line 124 of file liquid.py.

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

  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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.liquid.Liquid.w\_alpha [inherited]** Definition at line 668 of file liquid.py.

**forcebalance.liquid.Liquid.w\_cp [inherited]** Definition at line 670 of file liquid.py.

**forcebalance.liquid.Liquid.w\_eps0 [inherited]** Definition at line 671 of file liquid.py.

**forcebalance.liquid.Liquid.w\_hvap [inherited]** Definition at line 667 of file liquid.py.

**forcebalance.liquid.Liquid.w\_kappa [inherited]** Definition at line 669 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 666 of file liquid.py.

**forcebalance.target.Target.xct** [**inherited**] Counts how often the objective function was computed.

Definition at line 141 of file target.py.

The documentation for this class was generated from the following file:

- [tinkerio.py](#)

## 8.34 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](#)  
*number of sets (integer)*
- 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 dictionnary 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`

### 8.34.1 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.

### 8.34.2 Constructor & Destructor Documentation

```
def forcebalance.Mol2.mol2.__init__ ( self, data ) Definition at line 289 of file Mol2.py.
```

### 8.34.3 Member Function Documentation

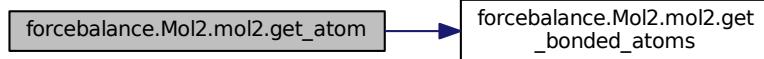
```
def forcebalance.Mol2.mol2.__repr__ ( self ) Definition at line 305 of file Mol2.py.
```

Here is the call graph for this function:



```
def forcebalance.Mol2.mol2.get_atom ( self, id ) return the atom instance given its atom identifier
Definition at line 461 of file Mol2.py.
```

Here is the call graph for this function:

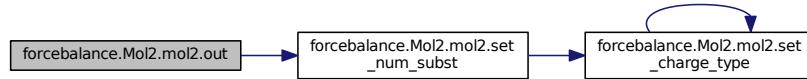


**def forcebalance.Mol2.mol2.get\_bonded\_atoms ( self, id )** return a dictionnary of atom instances bonded to the atom, and their types

Definition at line 475 of file Mol2.py.

**def forcebalance.Mol2.mol2.out ( self, f = sys.stdout )** Definition at line 325 of file Mol2.py.

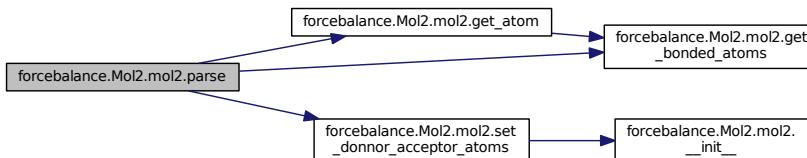
Here is the call graph for this function:



**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.

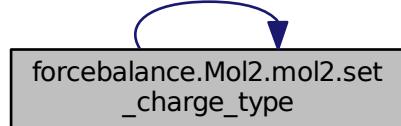
Here is the call graph for this function:



**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.

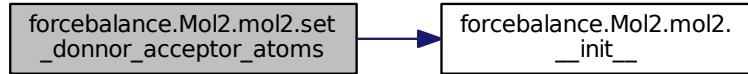
Here is the call graph for this function:



**def forcebalance.Mol2.mol2.set\_donnor\_acceptor\_atoms ( self, verbose = 0 )** modify atom types to specify  
donnor, acceptor, or both

Definition at line 490 of file Mol2.py.

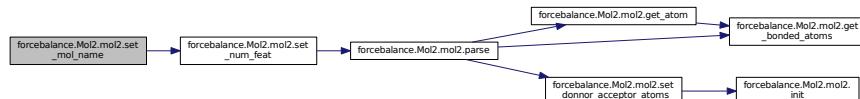
Here is the call graph for this function:



**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.

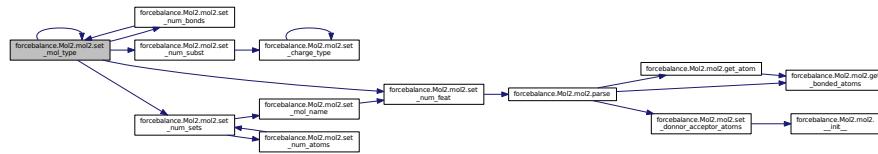
Here is the call graph for this function:



**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.

Here is the call graph for this function:



**def forcebalance.Mol2.mol2.set.num\_atoms ( self, num\_atoms = None )** number of atoms (integer)

Definition at line 341 of file Mol2.py.

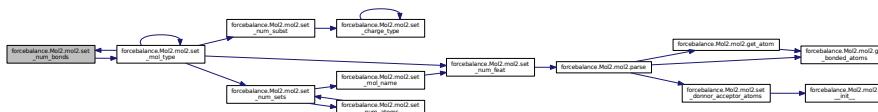
Here is the call graph for this function:



**def forcebalance.Mol2.mol2.set.num\_bonds ( self, num\_bonds = None )** number of bonds (integer)

Definition at line 350 of file Mol2.py.

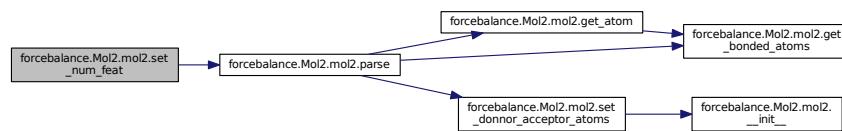
Here is the call graph for this function:



**def forcebalance.Mol2.mol2.set.num\_feat ( self, num\_feat = None )** number of features (integer)

Definition at line 368 of file Mol2.py.

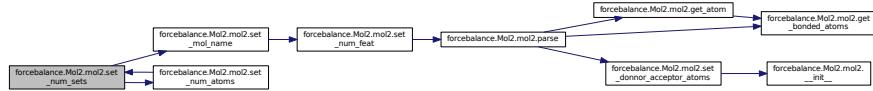
Here is the call graph for this function:



**def forcebalance.Mol2.mol2.set.num\_sets ( self, num\_sets = None )** number of sets (integer)

Definition at line 377 of file Mol2.py.

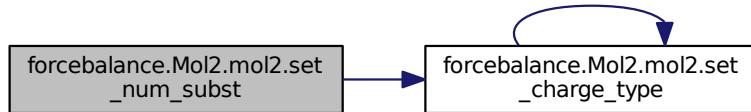
Here is the call graph for this function:



**def forcebalance.Mol2.mol2.set\_num\_subst ( self, num\_subst = None )** number of substructures (integer)

Definition at line 359 of file Mol2.py.

Here is the call graph for this function:



#### 8.34.4 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](#)

## 8.35 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`

#### 8.35.1 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.

#### 8.35.2 Constructor & Destructor Documentation

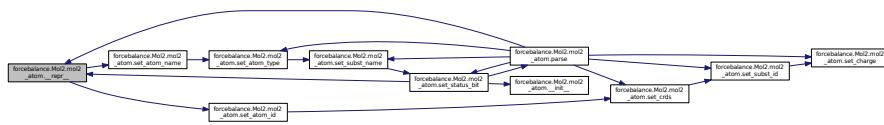
`def forcebalance.Mol2.mol2_atom.__init__ ( self, data = None )` if data is passed, it will be installed  
Definition at line 37 of file Mol2.py.

### 8.35.3 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.

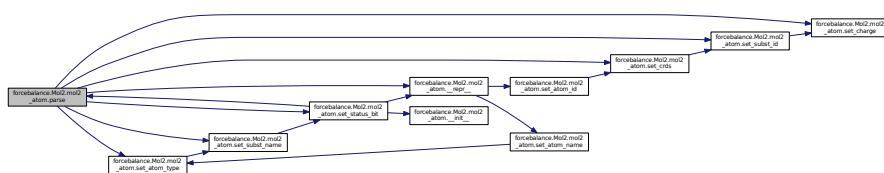
Here is the call graph for this function:



**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.

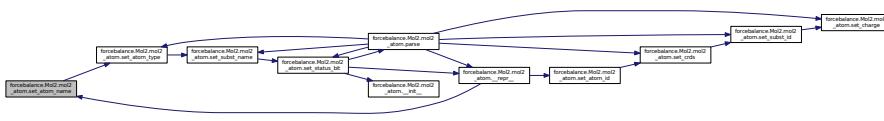
Here is the call graph for this function:



**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.

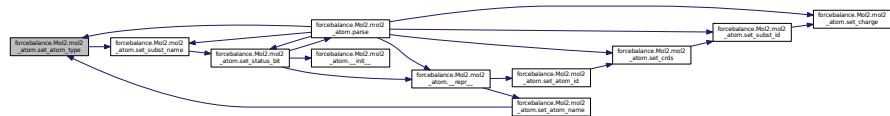
Here is the call graph for this function:



**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.

Here is the call graph for this function:



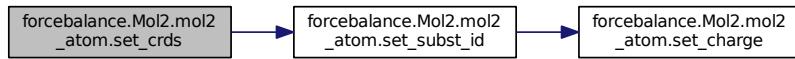
**def forcebalance.Mol2.mol2\_atom.set\_charge ( self, charge = None )** atomic charge

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.

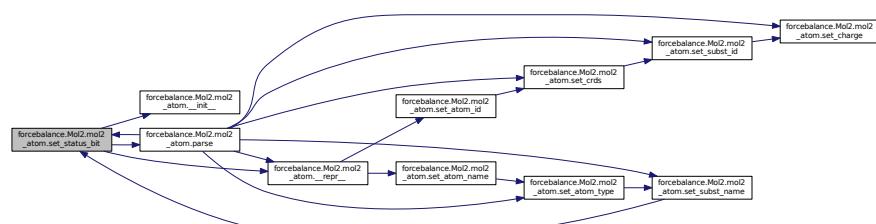
Here is the call graph for this function:



**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.

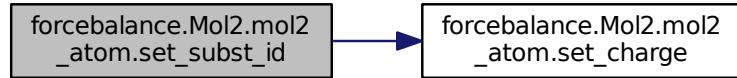
Here is the call graph for this function:



**def forcebalance.Mol2.mol2\_atom.set\_subst\_id ( self, subst\_id = None )** substructure identifier

Definition at line 128 of file Mol2.py.

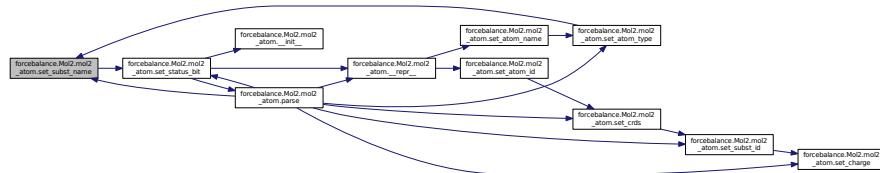
Here is the call graph for this function:



**def forcebalance.Mol2.mol2.atom.set\_subst\_name ( self, subst\_name = None )** substructure name

Definition at line 137 of file Mol2.py.

Here is the call graph for this function:



#### 8.35.4 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](#)

## 8.36 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`

#### 8.36.1 Detailed Description

This is to manage mol2 bond lines on the form: 1 1 2 ar.

Definition at line 172 of file Mol2.py.

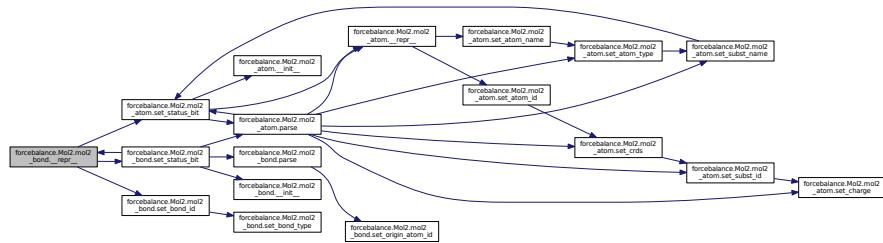
#### 8.36.2 Constructor & Destructor Documentation

```
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.
```

#### 8.36.3 Member Function Documentation

```
def forcebalance.Mol2.mol2_bond.__repr__( self ) Definition at line 186 of file Mol2.py.
```

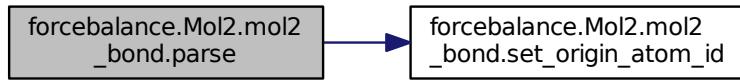
Here is the call graph for this function:



**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.

Here is the call graph for this function:



**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.

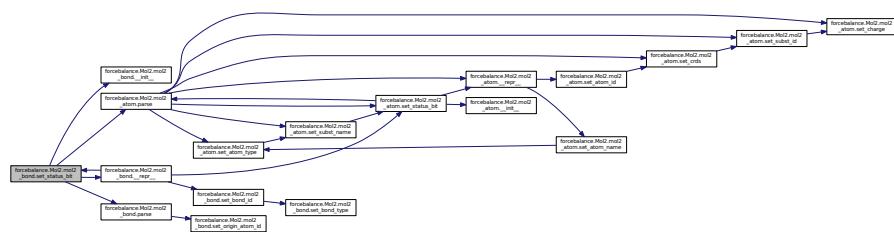
**def forcebalance.Mol2.mol2.bond.set\_origin\_atom\_id ( self, origin\_atom\_id = None )** the origin atom identifier (integer)

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.

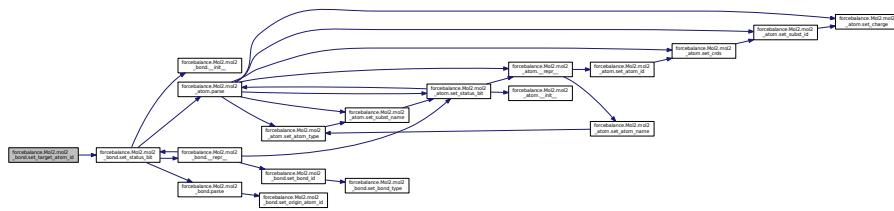
Here is the call graph for this function:



```
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.

Here is the call graph for this function:



#### 8.36.4 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.

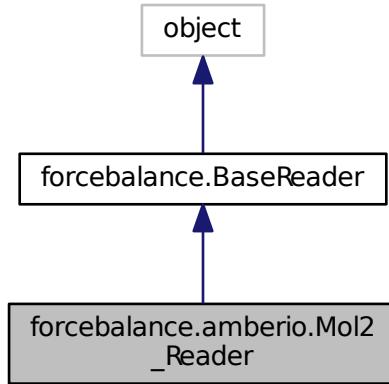
The documentation for this class was generated from the following file:

- [Mol2.py](#)

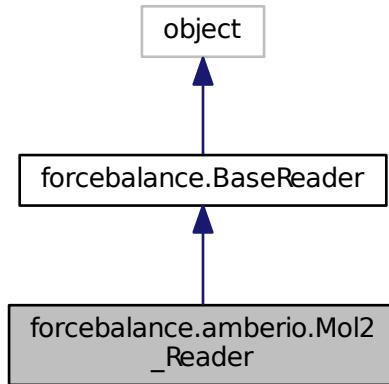
### 8.37 forcebalance.amberio.Mol2.Reader Class Reference

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](#)

### 8.37.1 Detailed Description

Finite state machine for parsing [Mol2](#) force field file.  
(just for parameterizing the charges)  
Definition at line 43 of file amberio.py.

### 8.37.2 Constructor & Destructor Documentation

`def forcebalance.amberio.Mol2_Reader.__init__ ( self, fnm )` Definition at line 45 of file amberio.py.

### 8.37.3 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 [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 107 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 82 of file \_\_init\_\_.py.  
Here is the call graph for this function:



```
def forcebalance.BaseReader.Whites( self, line ) [inherited] Definition at line 85 of file __init__.py.  
Here is the call graph for this function:
```



#### 8.37.4 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 72 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 80 of file \_\_init\_\_.py.

**forcebalance.amberio.Mol2\_Reader.itype** Definition at line 64 of file amberio.py.

**forcebalance.BaseReader.In** [inherited] Definition at line 67 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 79 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.

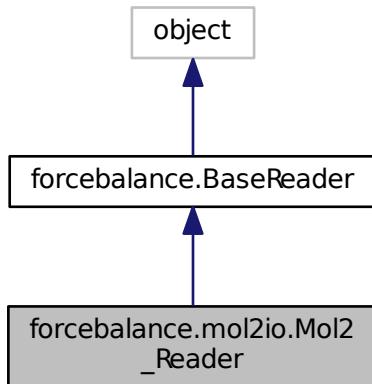
**forcebalance.amberio.Mol2\_Reader.suffix** Definition at line 93 of file amberio.py.  
The documentation for this class was generated from the following file:

- [amberio.py](#)

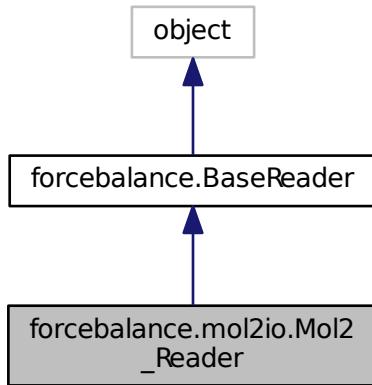
## 8.38 forcebalance.mol2io.Mol2\_Reader Class Reference

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](#)
- [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](#)

### 8.38.1 Detailed Description

Finite state machine for parsing [Mol2](#) force field file.  
(just for parameterizing the charges)  
Definition at line 22 of file mol2io.py.

### 8.38.2 Constructor & Destructor Documentation

**def forcebalance.mol2io.Mol2\_Reader.\_\_init\_\_ ( self, fnm )** Definition at line 24 of file mol2io.py.

### 8.38.3 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 [gmxiom.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 107 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 82 of file \_\_init\_\_.py.  
Here is the call graph for this function:



```
def forcebalance.BaseReader.Whites( self, line ) [inherited] Definition at line 85 of file __init__.py.
```

Here is the call graph for this function:



#### 8.38.4 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 72 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 80 of file \_\_init\_\_.py.

**forcebalance.mol2io.Mol2\_Reader.itype** Definition at line 36 of file mol2io.py.

**forcebalance.BaseReader.In** [inherited] Definition at line 67 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 77 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules** [inherited] Definition at line 79 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](#)

### 8.39 forcebalance.Mol2.mol2\_set Class Reference

#### Public Member Functions

- def [\\_\\_init\\_\\_](#)

A collection is organized as a dictionary of compounds self.num\_compounds : the number of compounds self.compounds : the dictionary 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](#)

### 8.39.1 Detailed Description

Definition at line 568 of file Mol2.py.

### 8.39.2 Constructor & Destructor Documentation

**def forcebalance.Mol2.mol2\_set.\_init\_ ( self, data = None, subset = None )** A collection is organized as a dictionary of compounds self.num\_compounds : the number of compounds self.compounds : the dictionary 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.

### 8.39.3 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.

### 8.39.4 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.

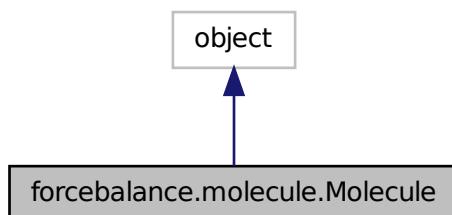
The documentation for this class was generated from the following file:

- [Mol2.py](#)

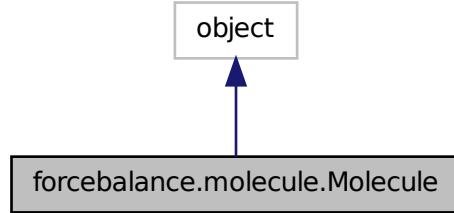
## 8.40 forcebalance.molecule.Molecule Class Reference

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 `append`  
*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 `load_frames`
- def `edit_qcrems`
- 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." Note that we're following the MSMBuild convention of using all ones for the masses.*
- def [align](#)  
*Align molecules.*
- def [build\\_topology](#)  
*A bare-bones implementation of the bond graph capability in the nanoreactor code.*
- 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 MSMBuild.)*
- 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](#)  
*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`

### 8.40.1 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 2 through 10.

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 662 of file molecule.py.

### 8.40.2 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 832 of file molecule.py.

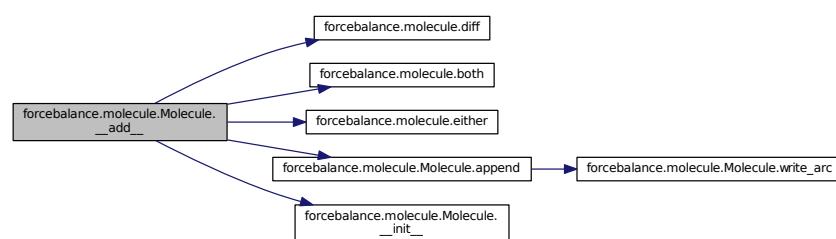
### 8.40.3 Member Function Documentation

```
def forcebalance.molecule.Molecule.__add__ ( self, other )
```

Add method for `Molecule` objects.

Definition at line 771 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 751 of file molecule.py.

Here is the call graph for this function:

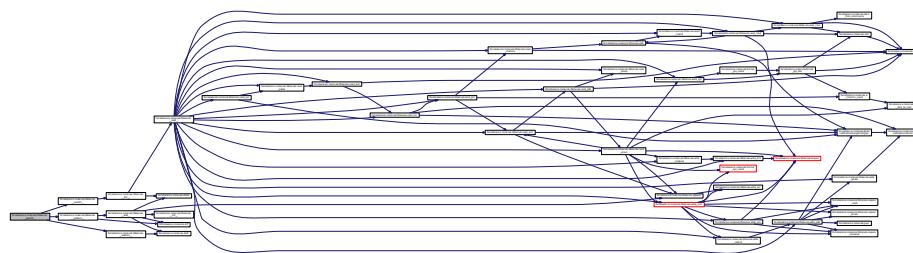


```
def forcebalance.molecule.Molecule.__getattr__( self, key )
```

Whenever we try to get a class attribute, it first tries to get the attribute from the Data dictionary.

Definition at line 682 of file molecule.py.

Here is the call graph for this function:



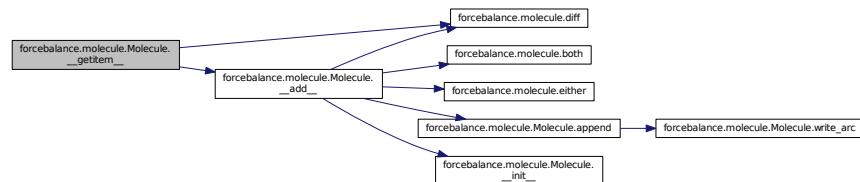
```
def forcebalance.molecule.Molecule.__getitem__( self, key )
```

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 730 of file molecule.py.

Here is the call graph for this function:

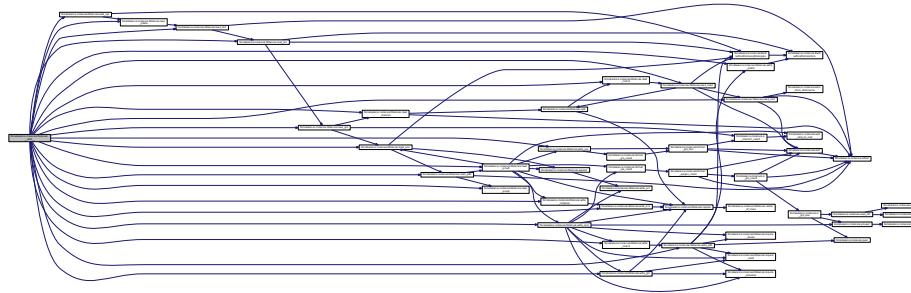


```
def forcebalance.molecule.Molecule.__iadd__( self, other )
```

Add method for `Molecule` objects.

Definition at line 801 of file molecule.py.

Here is the call graph for this function:

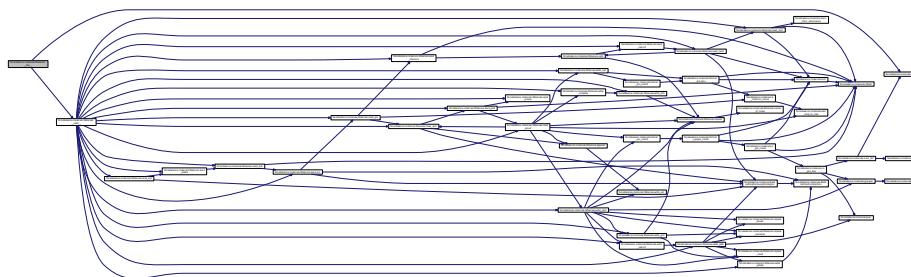


**def forcebalance.molecule.Molecule.\_\_iter\_\_( self )** List-like behavior for looping over trajectories.

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 760 of file molecule.py.

Here is the call graph for this function:



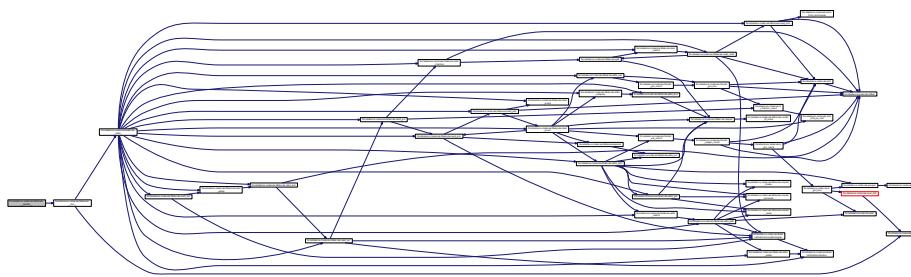
**def forcebalance.molecule.Molecule.\_\_len\_\_( self )** Return the number of frames in the trajectory.

Definition at line 666 of file molecule.py.

**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.

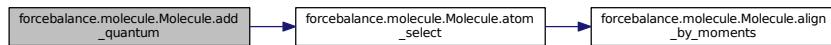
Definition at line 718 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.add\_quantum ( self, other )** Definition at line 1014 of file molecule.py.

Here is the call graph for this function:



**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 1026 of file molecule.py.

**def forcebalance.molecule.Molecule.align ( self, smooth = False, center = True, select = None )** Align molecules.

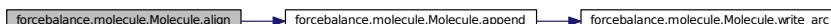
Has the option to create smooth trajectories (align each frame to the previous one) or to align each frame to the first one.

Also has the option to remove the center of mass.

Provide a list of atom indices to align along selected atoms.

Definition at line 1166 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."

Note that we're following the MSMBuilder convention of using all ones for the masses.

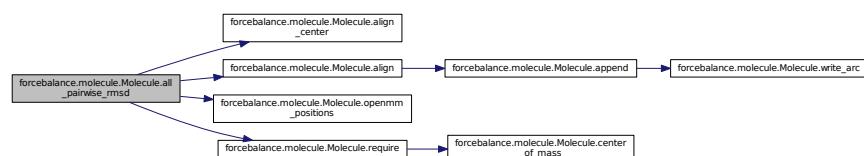
Definition at line 1145 of file molecule.py.

**def forcebalance.molecule.Molecule.align\_center ( self )** Definition at line 1364 of file molecule.py.

**def forcebalance.molecule.Molecule.all\_pairwise\_rmsd ( self )** Find pairwise RMSD (super slow, not like the one in MSMBuilder.)

Definition at line 1315 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.append ( self, other )** Definition at line 825 of file molecule.py.

Here is the call graph for this function:

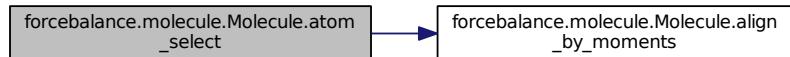


```
def forcebalance.molecule.Molecule.atom_select( self, atomslice ) Return a copy of the object with certain atoms selected.
```

Takes an integer, list or array as argument.

Definition at line 1064 of file molecule.py.

Here is the call graph for this function:

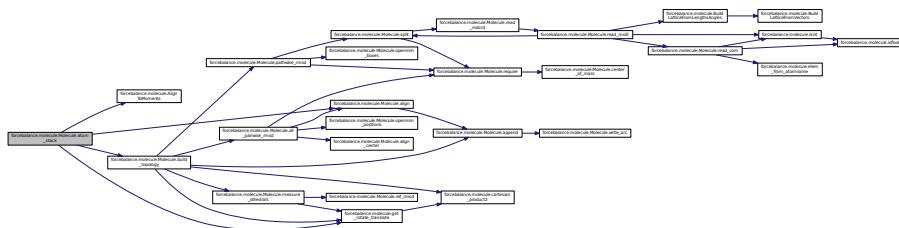


**def forcebalance.molecule.Molecule.atom\_stack( self, other )** Return a copy of the object with another molecule object appended.

WARNING: This function may invalidate stuff like QM energies.

Definition at line 1101 of file molecule.py.

Here is the call graph for this function:

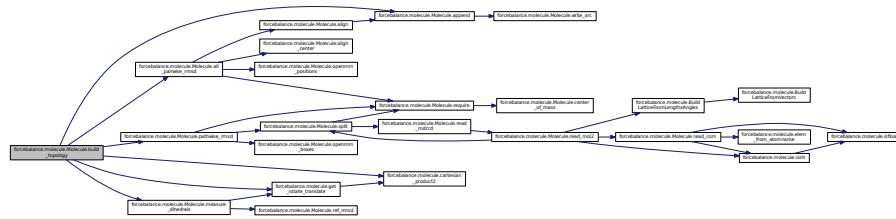


**def forcebalance.molecule.Molecule.build\_topology ( self, sn = None, Fac = 1.2 )** A bare-bones implementation of the bond graph capability in the nanoreactor code.

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 1191 of file molecule.py.

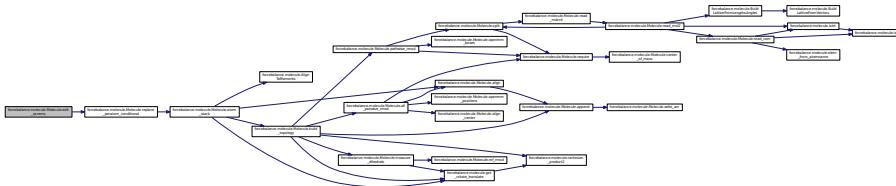
Here is the call graph for this function:



**def forcebalance.molecule.Molecule.center\_of\_mass ( self )** Definition at line 971 of file molecule.py.

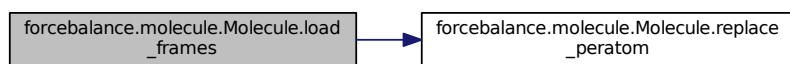
**def forcebalance.molecule.Molecule.edit\_qcrems ( self, in\_dict, subcalc = None )** Definition at line 1005 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.load\_frames ( self, fnm )** Definition at line 998 of file molecule.py.

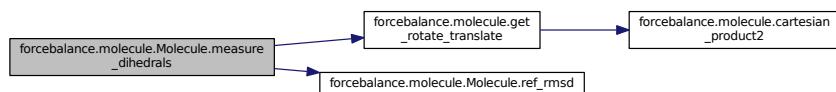
Here is the call graph for this function:



**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 1288 of file molecule.py.

Here is the call graph for this function:



**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(Molecule.openmm_boxes()[0])` or something like that.

Definition at line 1390 of file molecule.py.

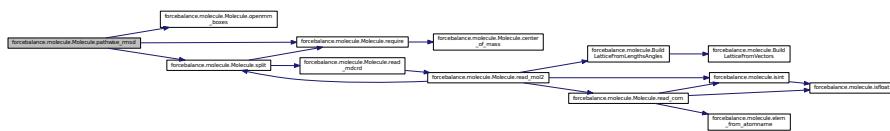
**def forcebalance.molecule.Molecule.openmm\_positions ( self )** Returns the Cartesian coordinates in the Molecule object in a list of OpenMM-compatible positions, so it is possible to type simulation.context.setPositions(Mol.-openmm\_positions()[0]) or something like that.

Definition at line 1373 of file molecule.py.

**def forcebalance.molecule.Molecule.pathwise\_rmsd ( self )** Find RMSD between frames along path.

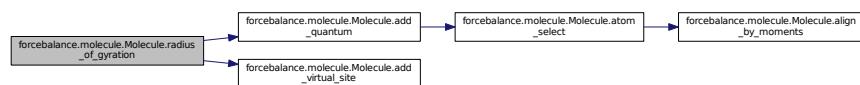
Definition at line 1333 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.radius\_of\_gyration ( self )** Definition at line 975 of file molecule.py.

Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.read_arc ( self, fnm ) Read a TINKER .arc file.
```

## Parameters

in                    *fnm*    The input file name

## Returns

**xyzs** A list for the XYZ coordinates.

**resid** The residue ID numbers. These are not easy to get!

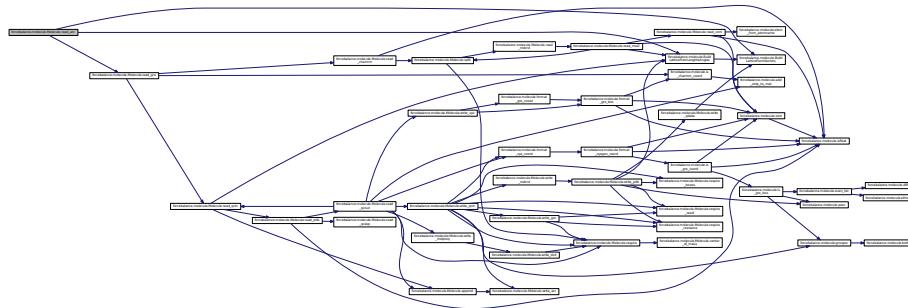
elem A list of chemical elements in the XYZ file

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 1671 of file molecule.py.

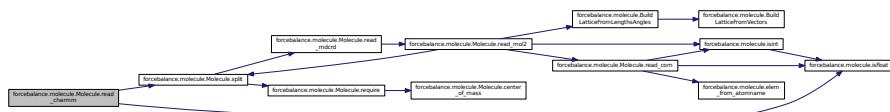
Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_charmm ( self, fnm )** Read a CHARMM .cor (or .crd) file.

Definition at line 1815 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_com ( self, fnm )** Parse a Gaussian .com file and return a SINGLE-ELEMENT list of xyz coordinates (no multiple file support)

Parameters

in	fnm	The input file name
----	-----	---------------------

Returns

elem A list of chemical elements in the XYZ file

comms A single-element list for the comment.

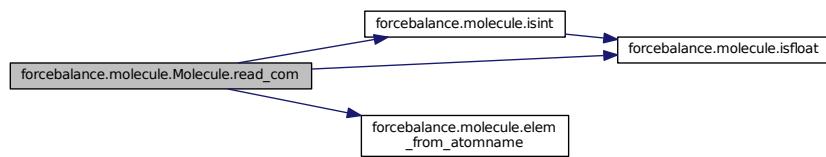
xyzs A single-element list for the XYZ coordinates.

charge The total charge of the system.

mult The spin multiplicity of the system.

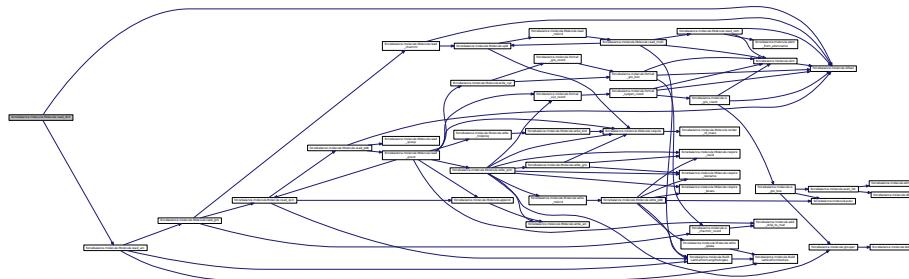
Definition at line 1627 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_dcd ( self, fnm )** Definition at line 1588 of file molecule.py.

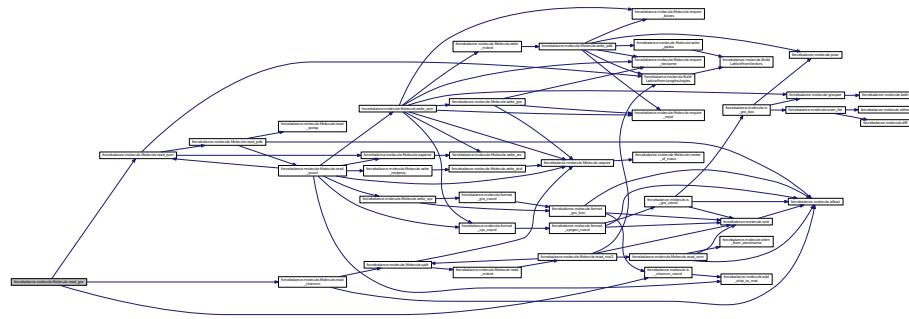
Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_gro ( self, fnm )** Read a GROMACS .gro file.

Definition at line 1735 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_mdcrd ( self, fnm )** 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

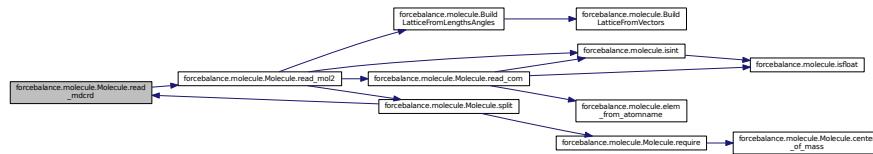
in	fnm	The input file name
----	-----	---------------------

#### Returns

xyzs A list of XYZ coordinates (number of snapshots times number of atoms)  
boxes Boxes (if present.)

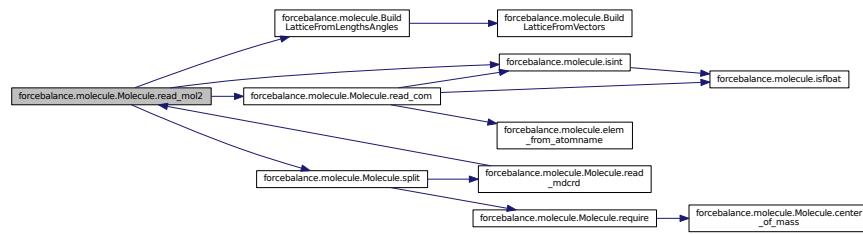
Definition at line 1478 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_mol2 ( self, fnm )** Definition at line 1539 of file molecule.py.

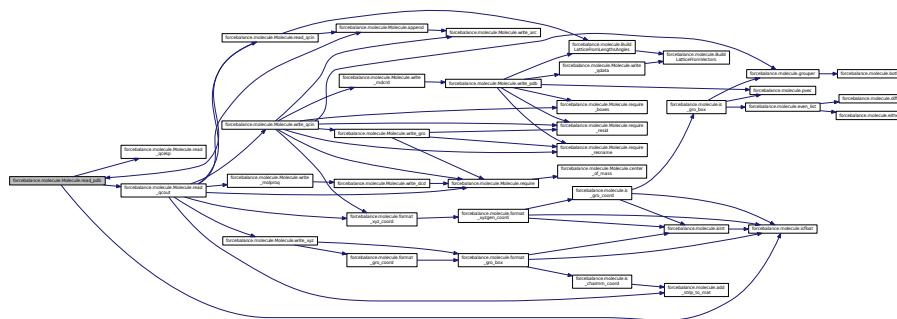
Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_pdb ( self, fnm )** Loads a PDB and returns a dictionary containing its data.

Definition at line 1998 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_qcesp( self, fnm )** Definition at line 2073 of file molecule.py.

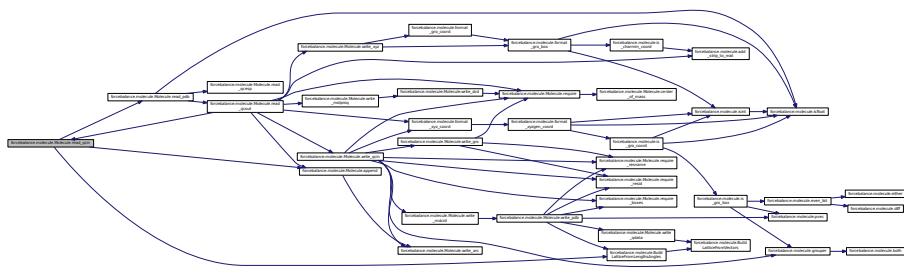
**def forcebalance.molecule.Molecule.read\_qcin ( self, fnm )** Read a Q-Chem input file.

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 1884 of file molecule.py.

Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.read_qcout ( self, fnm, maxopt = 1 ) 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

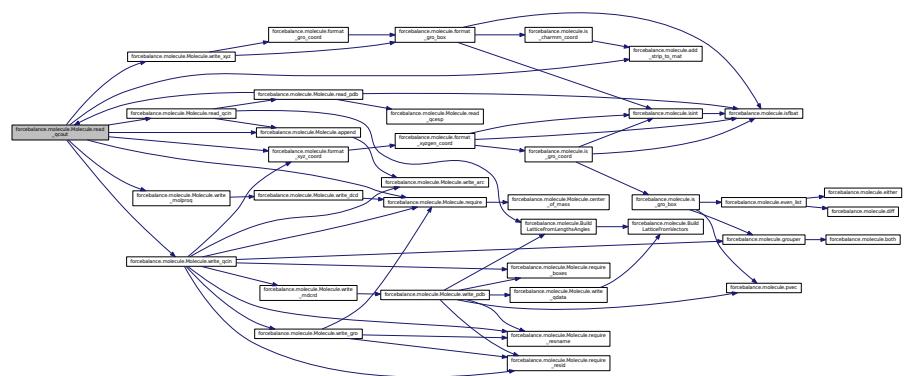
Calling with `maxopt = 1` will result in a successful read (with warning) even when maximum optimization cycles are reached.

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.

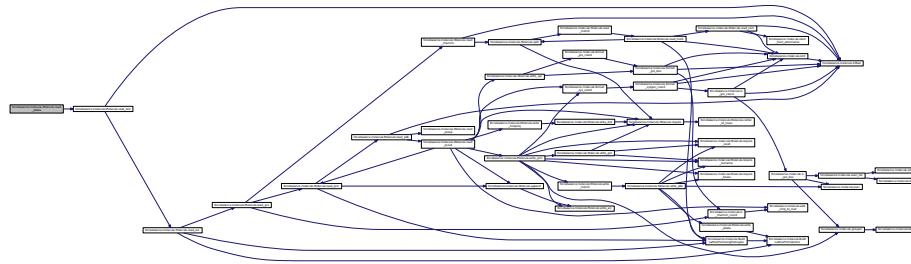
Definition at line 2105 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_qdata ( self, fnm )** Definition at line 1503 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.read\_xyz ( self, fnm )** Parse a .xyz file which contains several xyz coordinates, and return their elements.

Parameters

in	fnm	The input file name
----	-----	---------------------

Returns

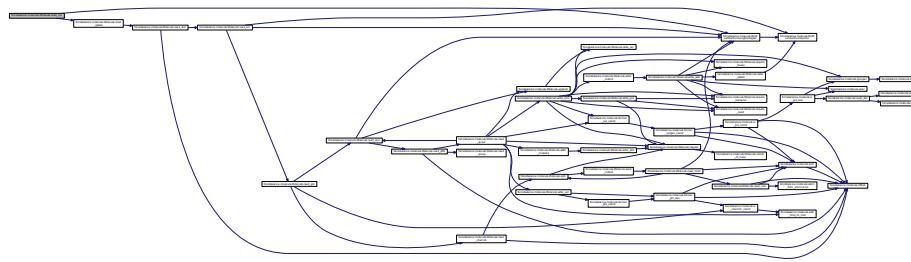
elem A list of chemical elements in the XYZ file

comms A list of comments.

xyzs A list of XYZ coordinates (number of snapshots times number of atoms)

Definition at line 1431 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.ref\_rmsd ( self, i )** Find RMSD to a reference frame.

Definition at line 1350 of file molecule.py.

**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 1043 of file molecule.py.

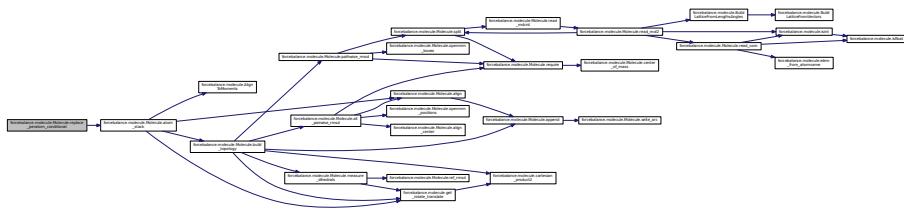
**def forcebalance.molecule.Molecule.replace\_peratom\_conditional ( self, key1, cond, key2, orig, want )**

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.

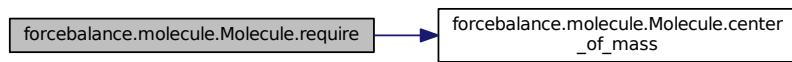
Definition at line 1054 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.require( self, args )** Definition at line 922 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.require\_boxes( self )** Definition at line 2595 of file molecule.py.

**def forcebalance.molecule.Molecule.require\_resid( self )** Definition at line 2582 of file molecule.py.

**def forcebalance.molecule.Molecule.require\_resname( self )** Definition at line 2590 of file molecule.py.

**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.

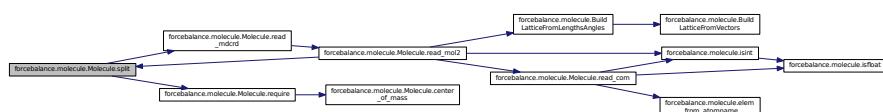
If the number of chunks / frames is not specified, then one file is written for each frame.

## Returns

**fnms** A list of the file names that were written.

Definition at line 1413 of file molecule.py.

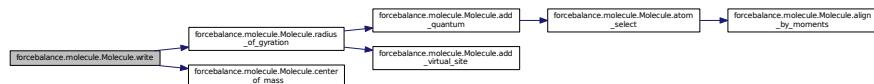
Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.write ( self, fnm = None, ftype = None, append = False, select = None )
```

Definition at line 937 of file molecule.py.

Here is the call graph for this function:



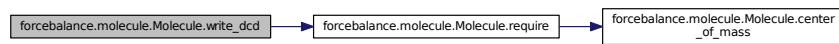
```
def forcebalance.molecule.Molecule.write_arc ( self, select )
```

Definition at line 2373 of file molecule.py.

```
def forcebalance.molecule.Molecule.write_dcd ( self, select )
```

Definition at line 2412 of file molecule.py.

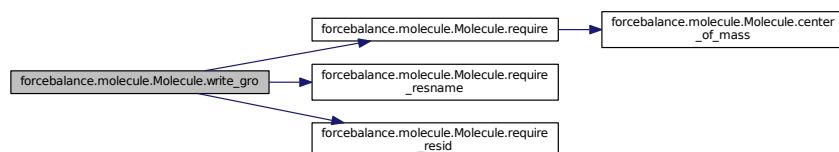
Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.write_gro ( self, select )
```

Definition at line 2385 of file molecule.py.

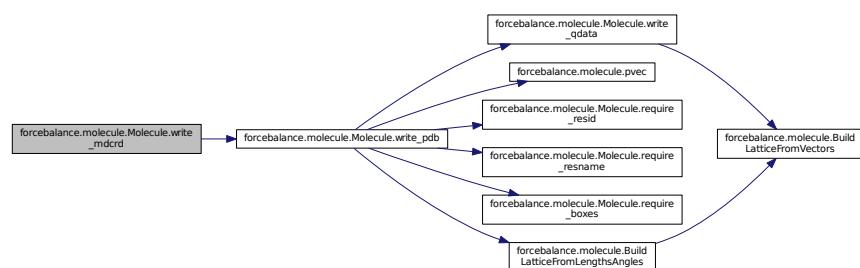
Here is the call graph for this function:



```
def forcebalance.molecule.Molecule.write_mdcrd ( self, select )
```

Definition at line 2362 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.write\_molproq ( self, select )** Definition at line 2350 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.write\_pdb ( self, select )** Save to a PDB.

Copied wholesale from MSMBuild.

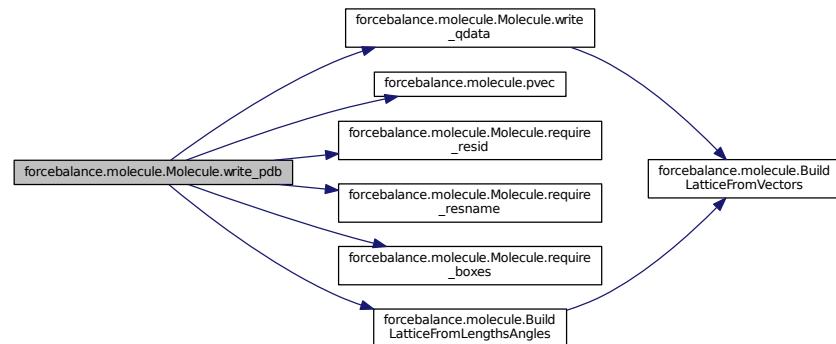
**COLUMNS TYPE FIELD DEFINITION** 7-11 int serial Atom serial number. 13-16 string name Atom name. 17 string altLoc Alternate location indicator. 18-20 (17-21 KAB) string resName Residue name. 22 string chainID Chain identifier. 23-26 int resSeq Residue sequence number. 27 string iCode Code for insertion 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.

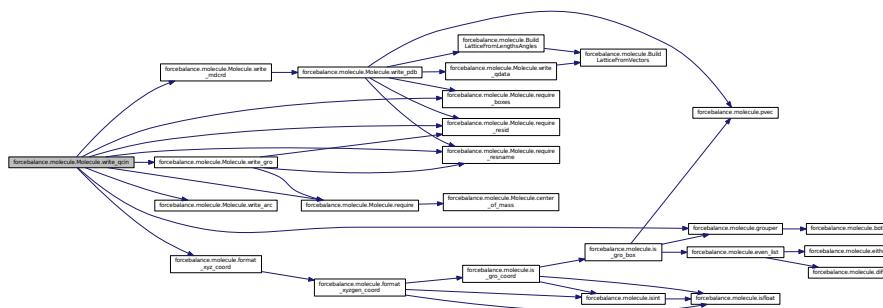
Definition at line 2468 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.write\_qcin ( self, select )** Definition at line 2293 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.write\_qdata( self, select )** Text quantum data format.

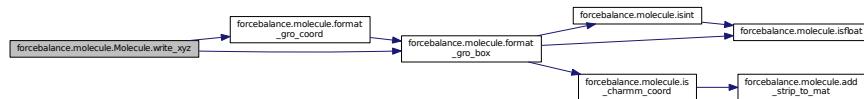
Definition at line 2561 of file molecule.py.

Here is the call graph for this function:



**def forcebalance.molecule.Molecule.write\_xyz( self, select )** Definition at line 2339 of file molecule.py.

Here is the call graph for this function:



#### **8.40.4 Member Data Documentation**

**forcebalance.molecule.Molecule.boxes** Definition at line 2643 of file molecule.py.

**forcebalance.molecule.Molecule.built\_bonds** Definition at line 879 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 900 of file molecule.py.

**forcebalance.molecule.Molecule.Data** Definition at line 883 of file molecule.py.

**forcebalance.molecule.Moleculefout** 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 948 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 864 of file molecule.py.

**forcebalance.molecule.Molecule.molecules** Definition at line 911 of file molecule.py.

**forcebalance.molecule.Molecule.positive\_resid** Creates entries like 'gromacs' : 'gromacs' and 'xyz' : 'xyz' in the Funnel.

Definition at line 878 of file molecule.py.

**forcebalance.molecule.Molecule.Read\_Tab** The table of file readers.

Definition at line 839 of file molecule.py.

**forcebalance.molecule.Molecule.resid** Definition at line 2586 of file molecule.py.

**forcebalance.molecule.Molecule.resname** Definition at line 2593 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 910 of file molecule.py.

**forcebalance.molecule.Molecule.Write\_Tab** The table of file writers.

Definition at line 853 of file molecule.py.

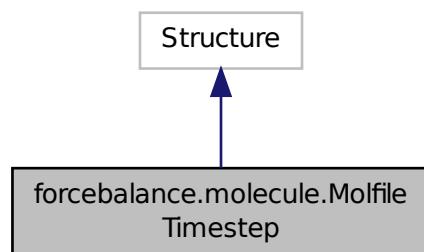
The documentation for this class was generated from the following file:

- [molecule.py](#)

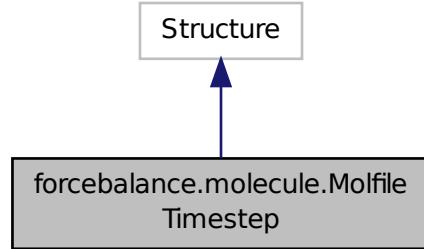
## 8.41 forcebalance.molecule.MolfileTimestep Class Reference

Wrapper for the timestep C structure used in molfile plugins.

Inheritance diagram for forcebalance.molecule.MolfileTimestep:



Collaboration diagram for forcebalance.molecule.MolfileTimestep:



#### 8.41.1 Detailed Description

Wrapper for the timestep C structure used in molfile plugins.

Definition at line 474 of file molecule.py.

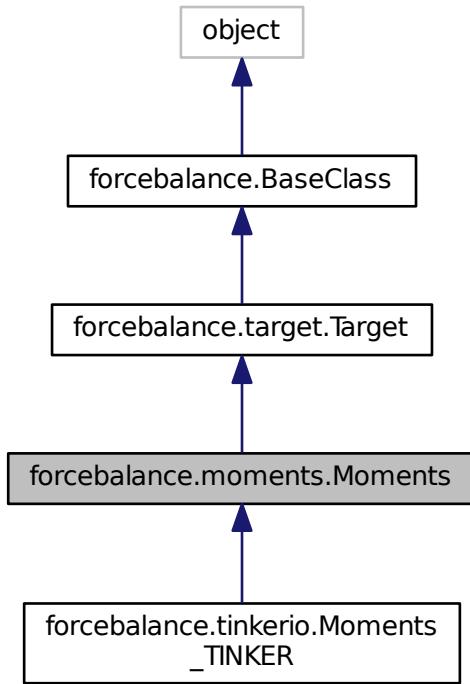
The documentation for this class was generated from the following file:

- [molecule.py](#)

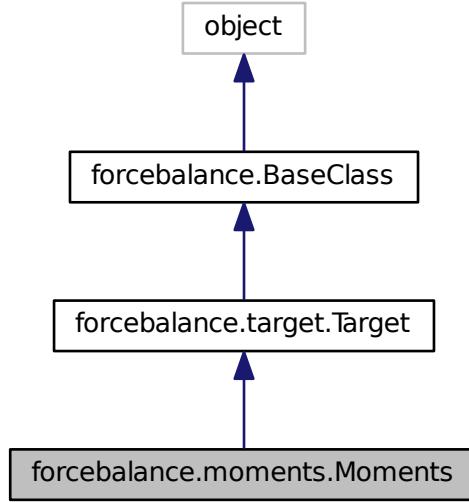
### 8.42 forcebalance.moments.Moments Class Reference

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 `prepare_temp_directory`  
*Prepare the temporary directory.*
- 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 `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 `sget`  
*Stages the directory for the target, and then calls 'get'.*

- 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 `set_option`

## Public Attributes

- `denoms`
- `mfnm`  
*The mdata.txt file that contains the moments.*
- `ref_moments`
- `na`  
*Number of atoms.*
- `ref_eigvals`
- `ref_eigvecs`
- `calc_moments`
- `objective`
- `tempdir`  
*Root directory of the whole project.*
- `rundir`  
*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.*
- `PrintOptionDict`
- `verbose_options`

### 8.42.1 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.

## 8.42.2 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:

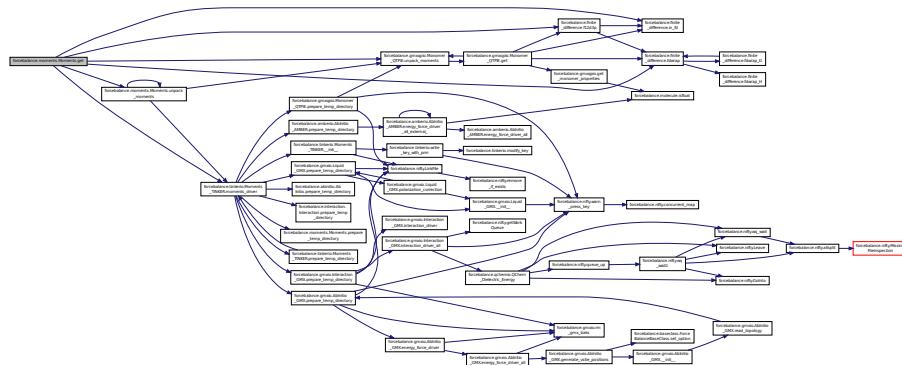


## 8.42.3 Member Function Documentation

**def forcebalance.moments.Moments.get ( self, mvals, AGrad = False, AHess = False )** Evaluate objective function.

Definition at line 173 of file moments.py.

Here is the call graph for this function:

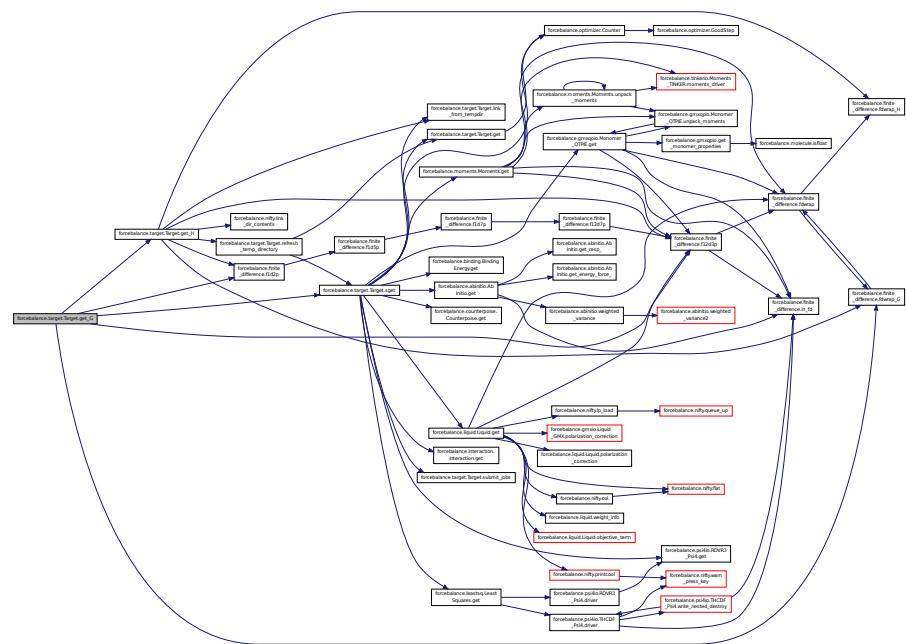


**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



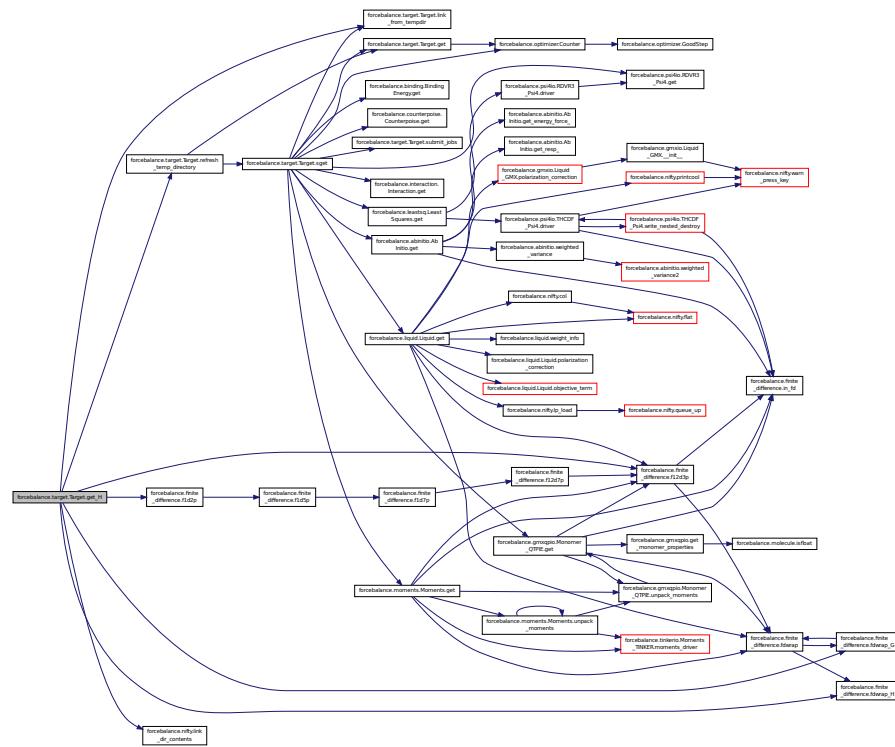
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

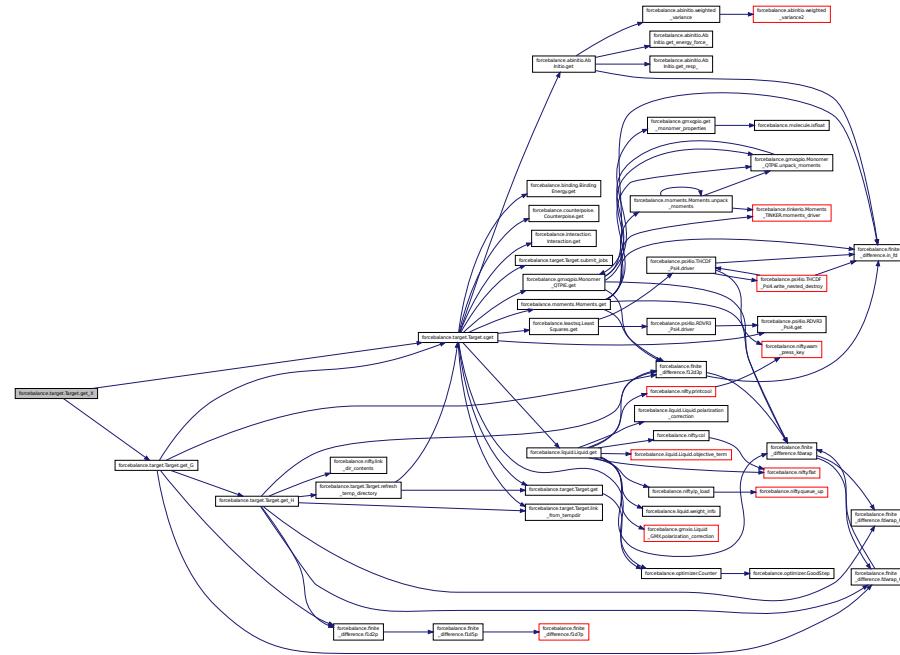
Here is the call graph for this function:



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

Definition at line 155 of file target.py.

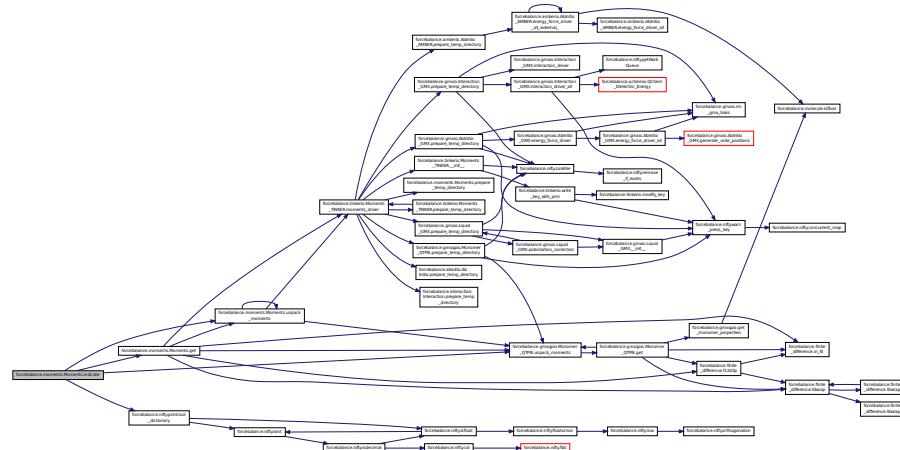
Here is the call graph for this function:



**def forcebalance.moments.Moments.indicate ( self )** Print qualitative indicator.

Definition at line 141 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 213 of file target.py.
```

```
def forcebalance.moments.Moments.prepare_temp_directory ( self, options, tgt_opts ) Prepare the temporary directory.
```

Definition at line 136 of file moments.py.

```
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.

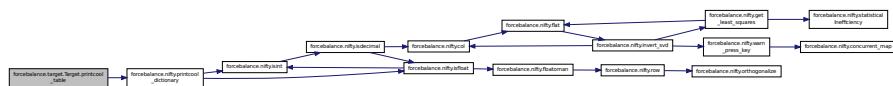
The input should be something like:

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 367 of file target.py.

Here is the call graph for this function:



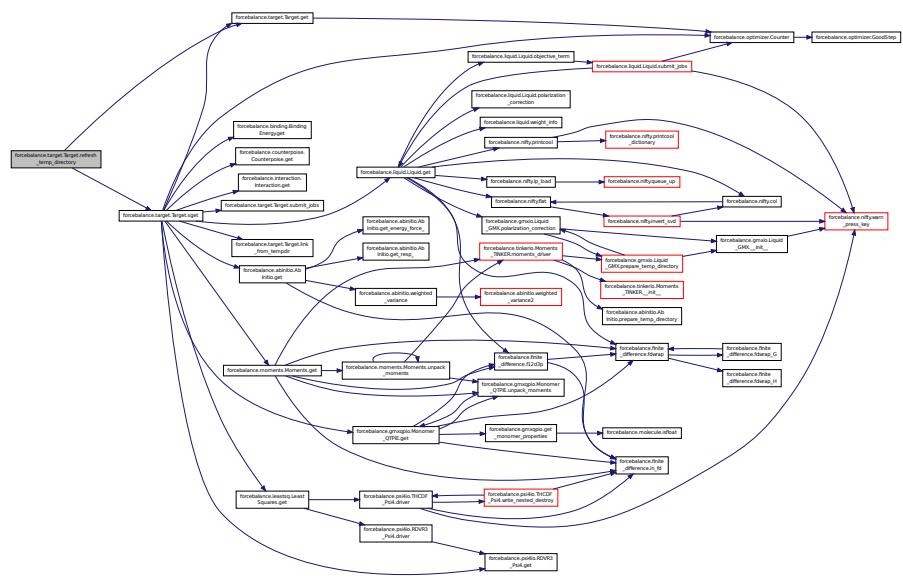
```
def forcebalance.moments.Moments.read_reference_data( self ) Read the reference data from a file.
```

Definition at line 67 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 219 of file target.py.

Here is the call graph for this function:



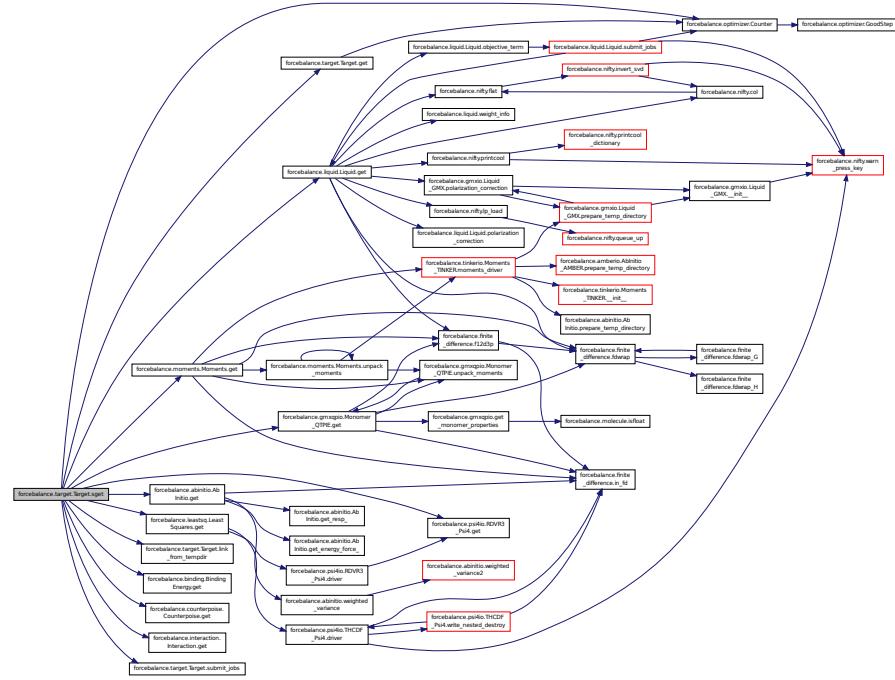
**def forcebalance.BaseClass.set\_option( self, in\_dict, src\_key, dest\_key = None, val = None, default = None, forceprint = False ) [inherited]** Definition at line 32 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

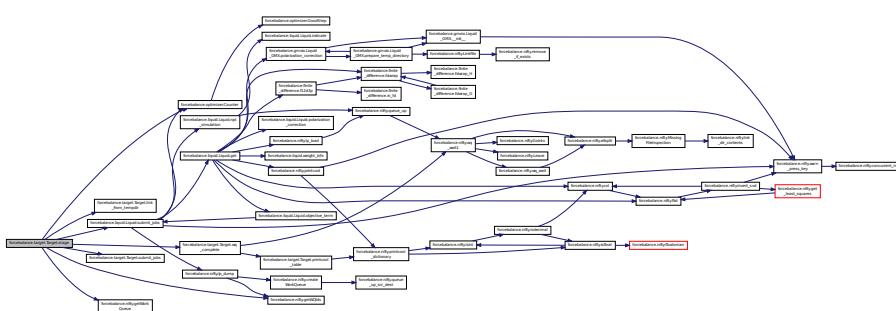


```
def forcebalance.target.Target.stage( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

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

Definition at line 301 of file target.py.

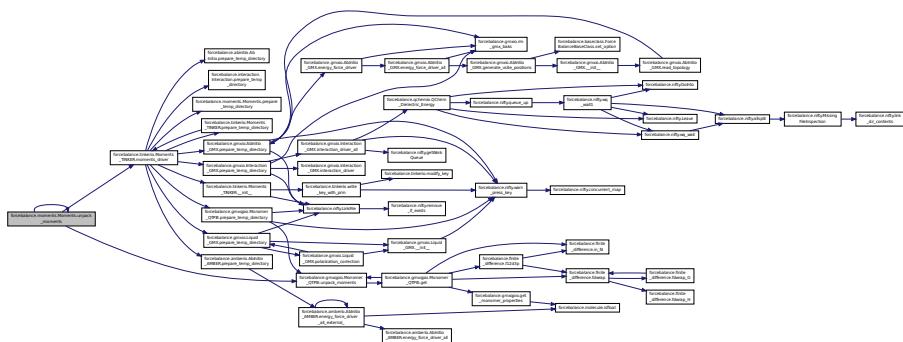
Here is the call graph for this function:



```
def forcebalance.target.Target.submit_jobs( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 291 of file target.py.
```

```
def forcebalance.moments.Moments.unpack_moments( self, moment_dict ) Definition at line 167 of file moments.py.
```

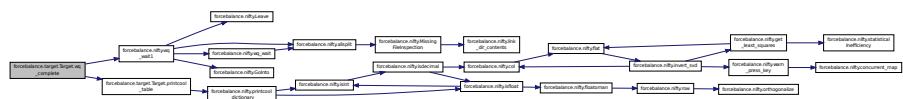
Here is the call graph for this function:



**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 331 of file target.py.

Here is the call graph for this function:



#### **8.42.4 Member Data Documentation**

**forcebalance.moments.Moments.calc\_moments** Definition at line 201 of file moments.py.

**forcebalance.moments.Moments.denoms** Definition at line 48 of file moments.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

cebalance-target-Target-qct [in ph]

**ccbalance-target** Target hot [link]

Definition at line 175 of file target.h.

Journal of Management Education

Definition at line 57 of file moments.py.

Definition at line 69 of file moments.py.

**forcebalance.moments.Moments.ref\_eigvals** Definition at line 70 of file moments.py.

**forcebalance.moments.Moments.ref\_eigvecs** Definition at line 71 of file moments.py.

**forcebalance.moments.Moments.ref\_moments** Definition at line 58 of file moments.py.

**forcebalance.target.Target.rundir [inherited]** 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.

Definition at line 137 of file target.py.

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

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 141 of file target.py.

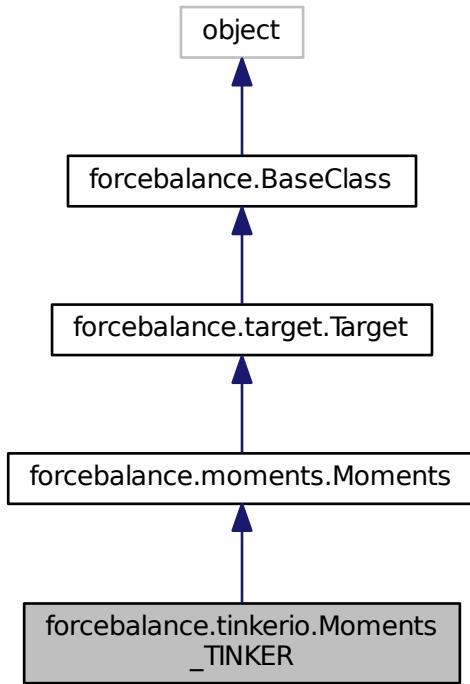
The documentation for this class was generated from the following file:

- [moments.py](#)

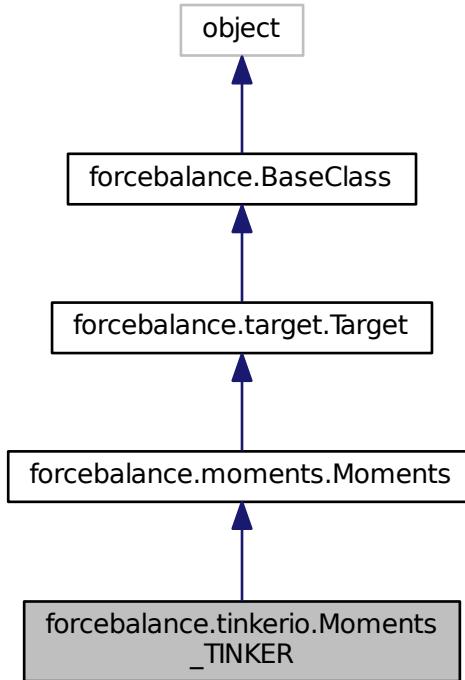
## 8.43 forcebalance.tinkerio.Moments\_TINKER Class Reference

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 `prepare_temp_directory`
- def `moments_driver`
- 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 `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 [sget](#)  
*Stages the directory for the target, and then calls 'get'.*
- 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 [set\\_option](#)

## Public Attributes

- [denoms](#)
- [mfnm](#)  
*The mdata.txt file that contains the moments.*
- [ref\\_moments](#)
- [na](#)  
*Number of atoms.*
- [ref\\_eigvals](#)
- [ref\\_eigvecs](#)
- [calc\\_moments](#)
- [objective](#)
- [tempdir](#)  
*Root directory of the whole project.*
- [rundir](#)  
*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.*
- [PrintOptionDict](#)
- [verbose\\_options](#)

### 8.43.1 Detailed Description

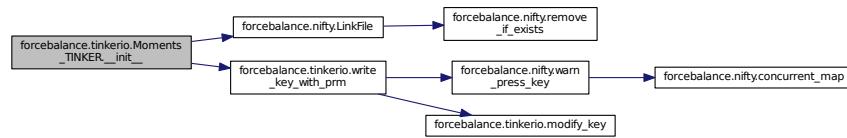
Subclass of Target for multipole moment matching using TINKER.

Definition at line 403 of file tinkerio.py.

### 8.43.2 Constructor & Destructor Documentation

**def forcebalance.tinkerio.Moments.TINKER\_\_init\_\_( self, options, tgt\_opts, forcefield )** Definition at line 406 of file tinkerio.py.

Here is the call graph for this function:



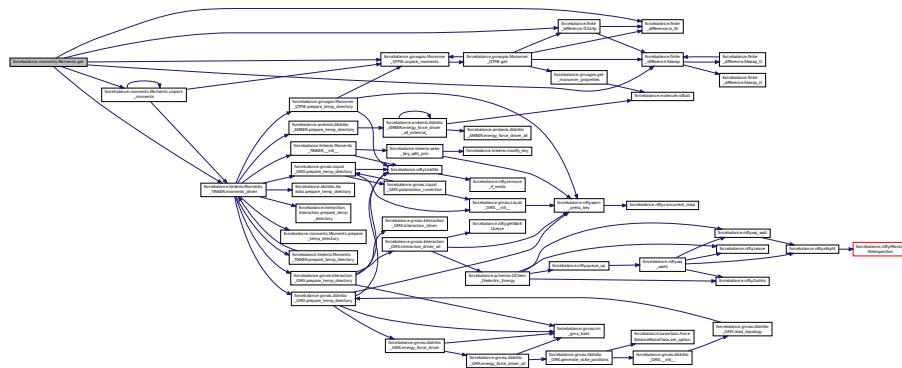
### 8.43.3 Member Function Documentation

**def forcebalance.moments.Moments.get( self, mvals, AGrad=False, AHess=False ) [inherited]**

Evaluate objective function.

Definition at line 173 of file moments.py.

Here is the call graph for this function:

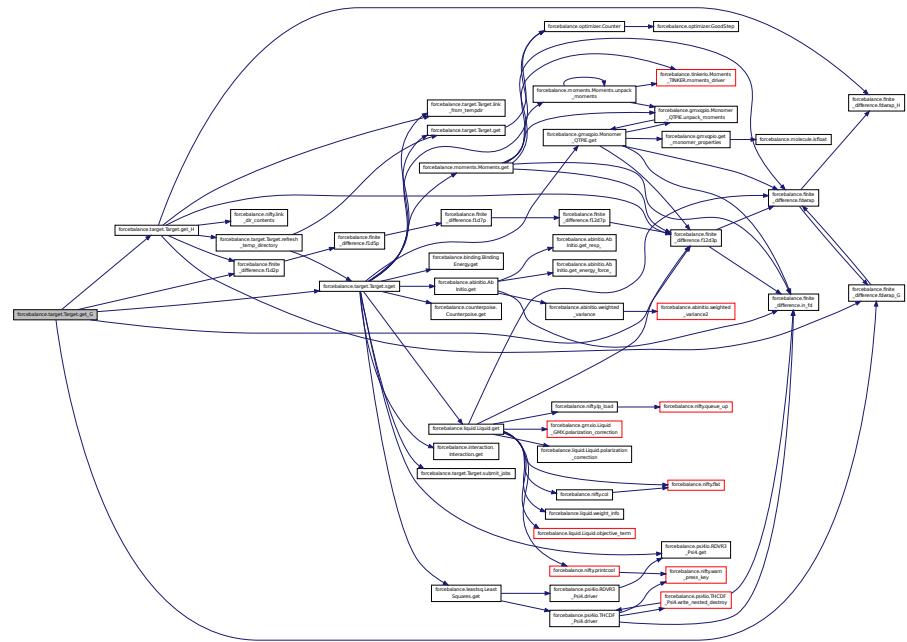


**def forcebalance.target.Target.get\_G( self, mvals=None ) [inherited]** Computes the objective function contribution and its gradient.

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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



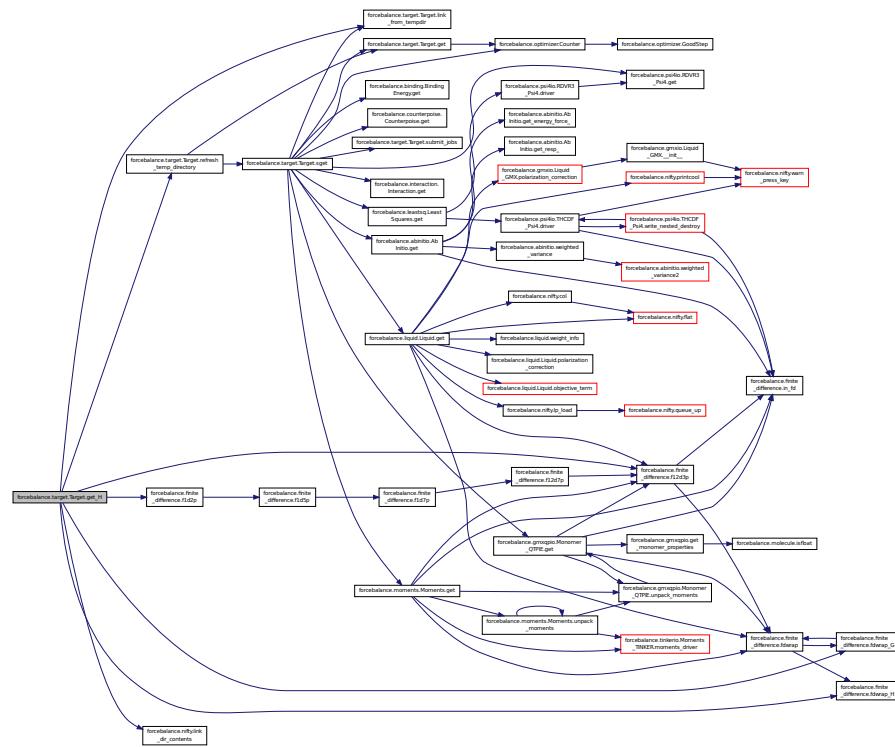
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

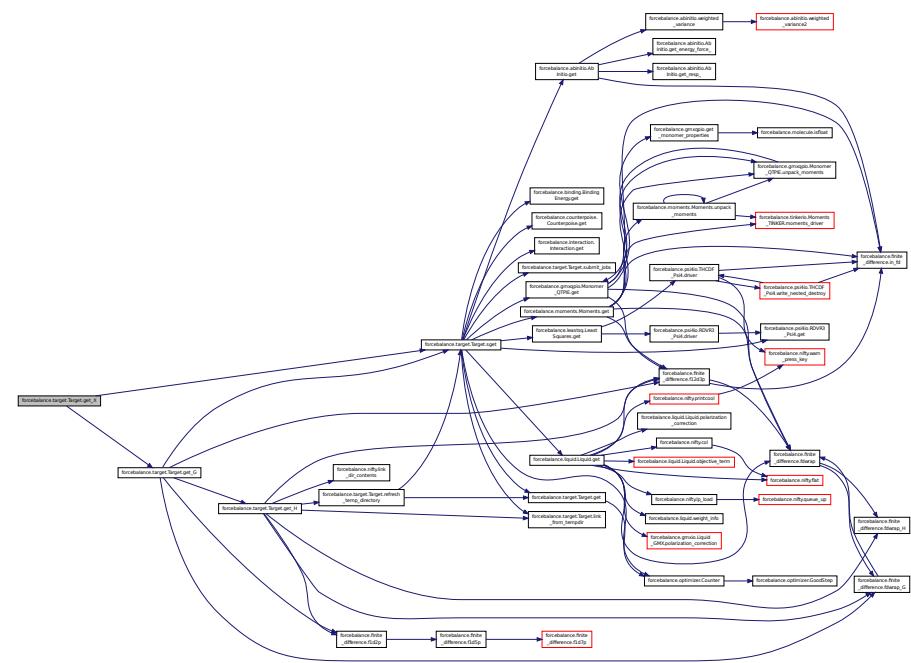
Here is the call graph for this function:



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

Definition at line 155 of file target.py.

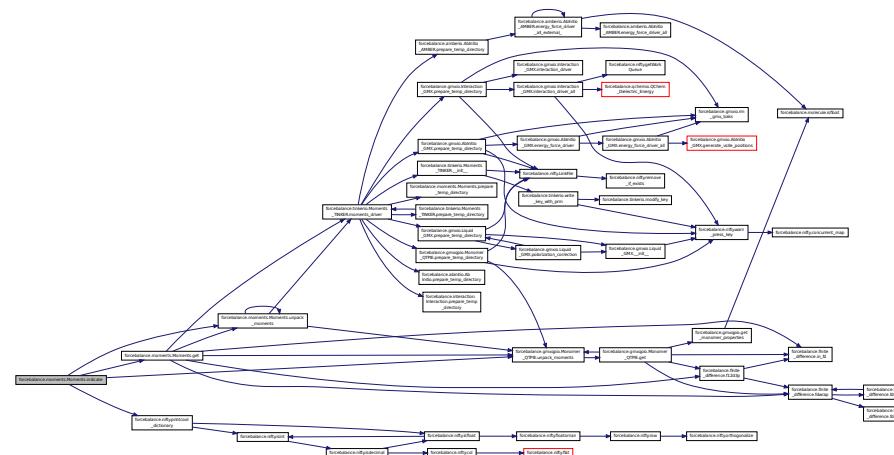
Here is the call graph for this function:



**def forcebalance.moments.Moments.indicate( self ) [inherited]** Print qualitative indicator.

Definition at line 141 of file moments.py.

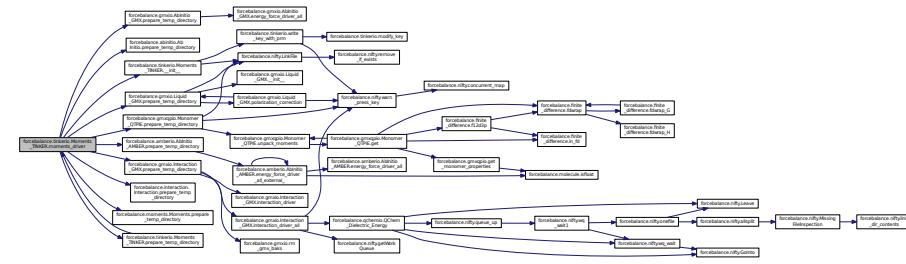
Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

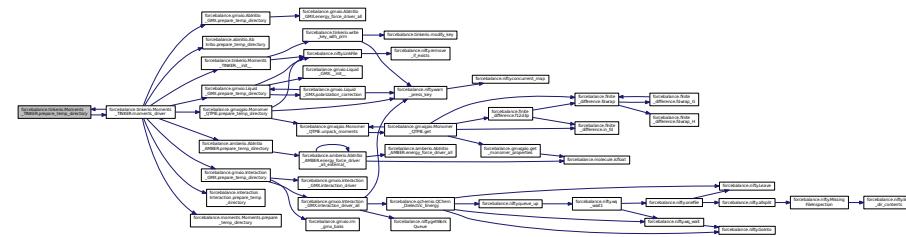
**def forcebalance.tinkerio.Moments.TINKER.moments\_driver ( self )** Definition at line 422 of file tinkerio.py.

Here is the call graph for this function:



**def forcebalance.tinkerio.Moments.TINKER.prepare\_temp\_directory ( self, options, tgt\_opts )** Definition at line 411 of file tinkerio.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 367 of file target.py.

Here is the call graph for this function:



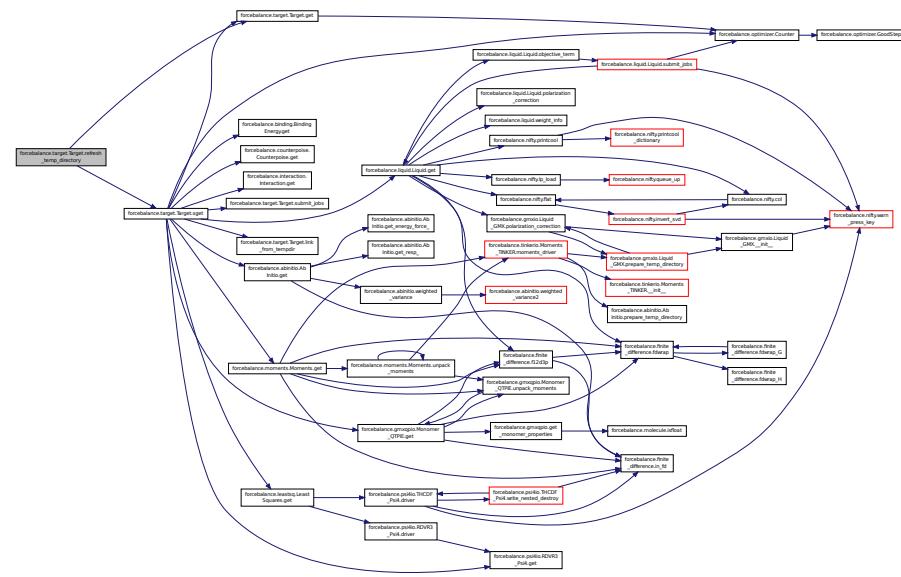
```
def forcebalance.moments.Moments.read_reference_data ( self ) [inherited] Read the reference data from a file.
```

Definition at line 67 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 219 of file target.py.

Here is the call graph for this function:



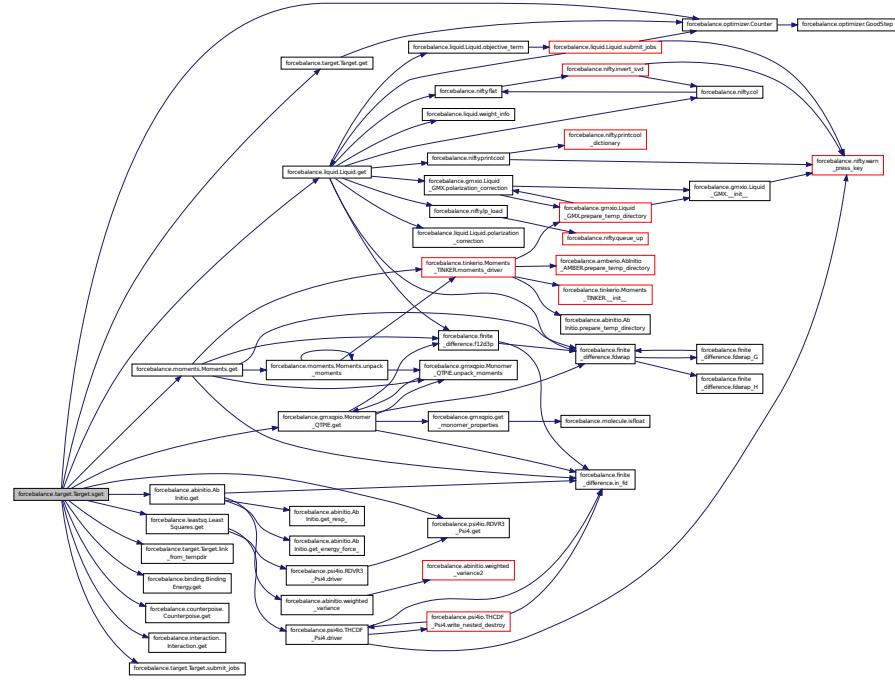
`def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited]` Definition at line 32 of file `__init__.py`.

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

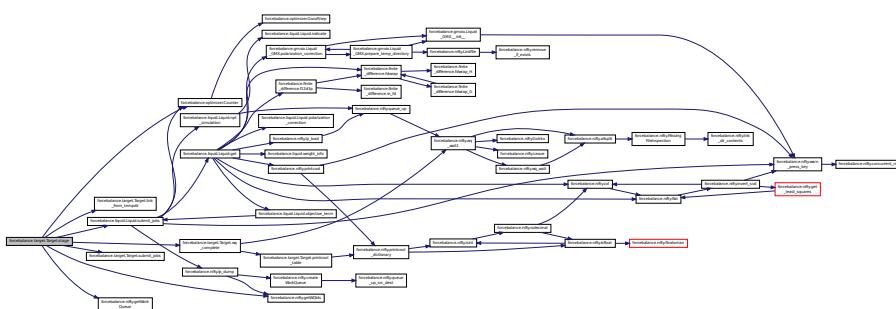


```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

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

Definition at line 301 of file target.py.

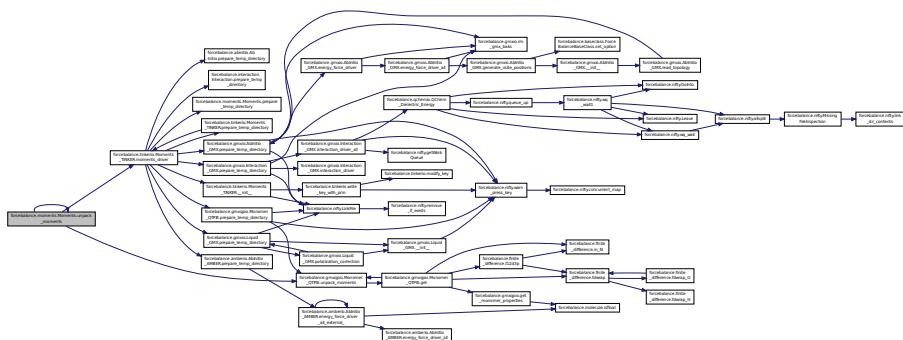
Here is the call graph for this function:



```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**def forcebalance.moments.Moments.unpack\_moments ( self, moment\_dict ) [inherited]** Definition at line 167 of file moments.py.

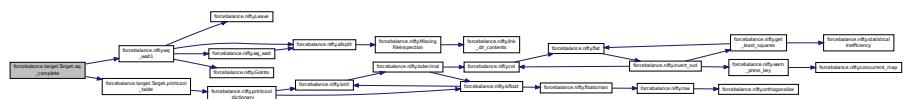
Here is the call graph for this function:



**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 331 of file target.py.

Here is the call graph for this function:



#### **8.43.4 Member Data Documentation**

**forcebalance.moments.Moments.calc\_moments** [inherited] Definition at line 201 of file moments.py.

**forcebalance.moments.Moments.denoms** [inherited] Definition at line 48 of file moments.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.

Definition at line 145 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 69 of file moments.py.

**forcebalance.moments.Momentsobjective** [inherited] Definition at line 202 of file moments.py.

**forcebalance** BaseClass PrintOptionDict [inherited] Definition at line 39 of file `init.py`.

**forcebalance.moments.Moments.ref\_eigvals [inherited]** Definition at line 70 of file moments.py.

**forcebalance.moments.Moments.ref\_eigvecs [inherited]** Definition at line 71 of file moments.py.

**forcebalance.moments.Moments.ref\_moments [inherited]** Definition at line 58 of file moments.py.

**forcebalance.target.Target.rundir [inherited]** 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.

Definition at line 137 of file target.py.

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

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 141 of file target.py.

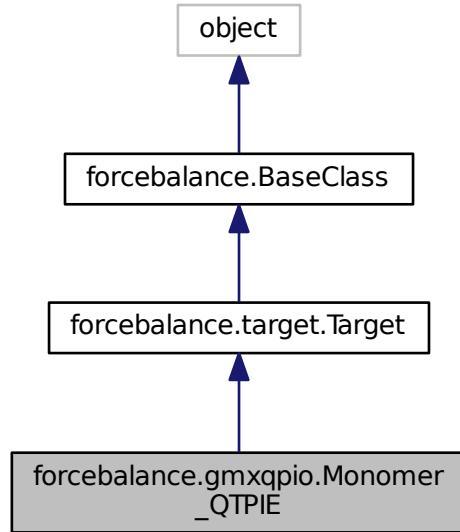
The documentation for this class was generated from the following file:

- [tinkerio.py](#)

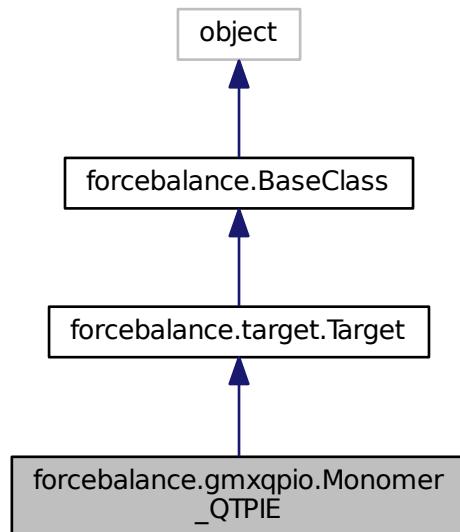
## 8.44 forcebalance.gmxqpio.Monomer\_QTPIE Class Reference

Subclass of Target for monomer properties of QTPIE (implemented within gromacs WCV branch).

Inheritance diagram for forcebalance.gmxqpio.Monomer\_QTPIE:



Collaboration diagram for forcebalance.gmxqpio.Monomer\_QTPIE:



## Public Member Functions

- def `_init_`
- def `indicate`

*Print qualitative indicator.*
- def `prepare_temp_directory`
- def `unpack_moments`
- def `get`
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- 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 `sget`

*Stages the directory for the target, and then calls 'get'.*
- 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 `set_option`

## Public Attributes

- `ref_moments`
- `weights`
- `calc_moments`
- `objective`
- `tempdir`

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

*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.*
- `PrintOptionDict`
- `verbose_options`

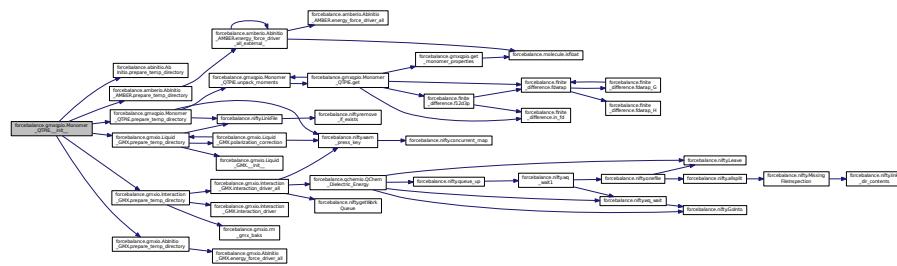
### 8.44.1 Detailed Description

Subclass of Target for monomer properties of QTPIE (implemented within gromacs WCV branch).  
Definition at line 130 of file gmxqpio.py.

### 8.44.2 Constructor & Destructor Documentation

**def forcebalance.gmxqpio.Monomer\_QTPIE.\_\_init\_\_ ( self, options, tgt\_opts, forcefield )** Definition at line 132 of file gmxqpio.py.

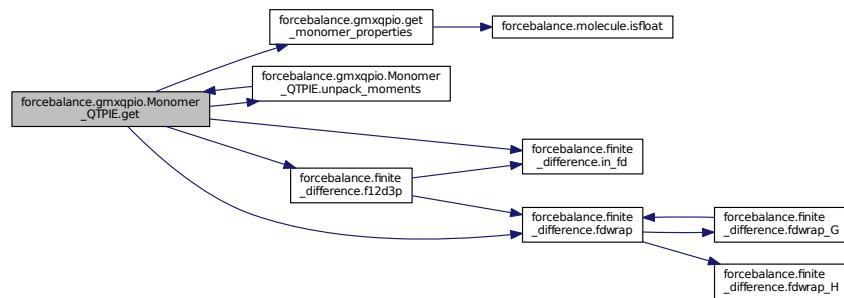
Here is the call graph for this function:



### 8.44.3 Member Function Documentation

**def forcebalance.gmxqpio.Monomer\_QTPIE.get ( self, mvals, AGrad = False, AHess = False )** Definition at line 221 of file gmxqpio.py.

Here is the call graph for this function:

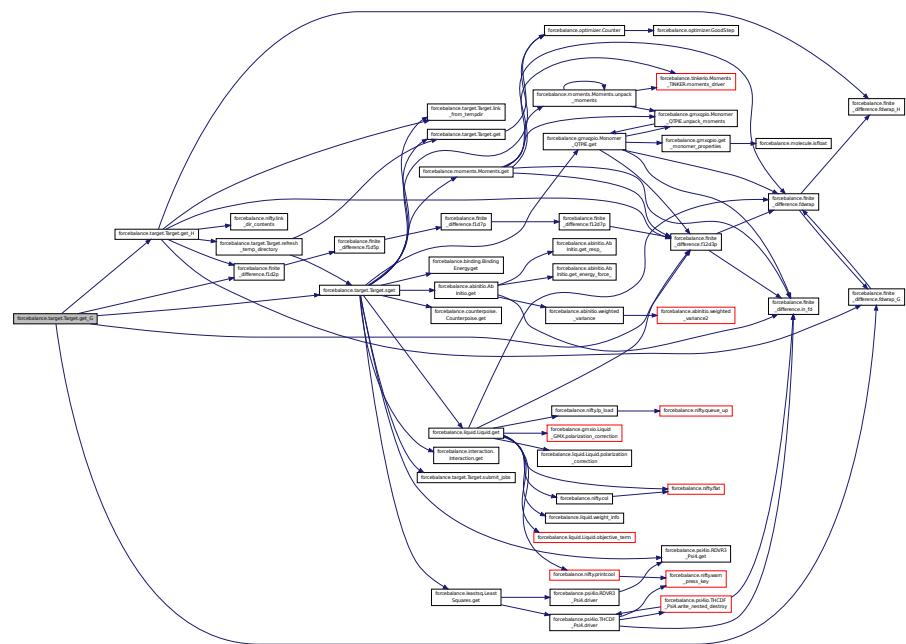


**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



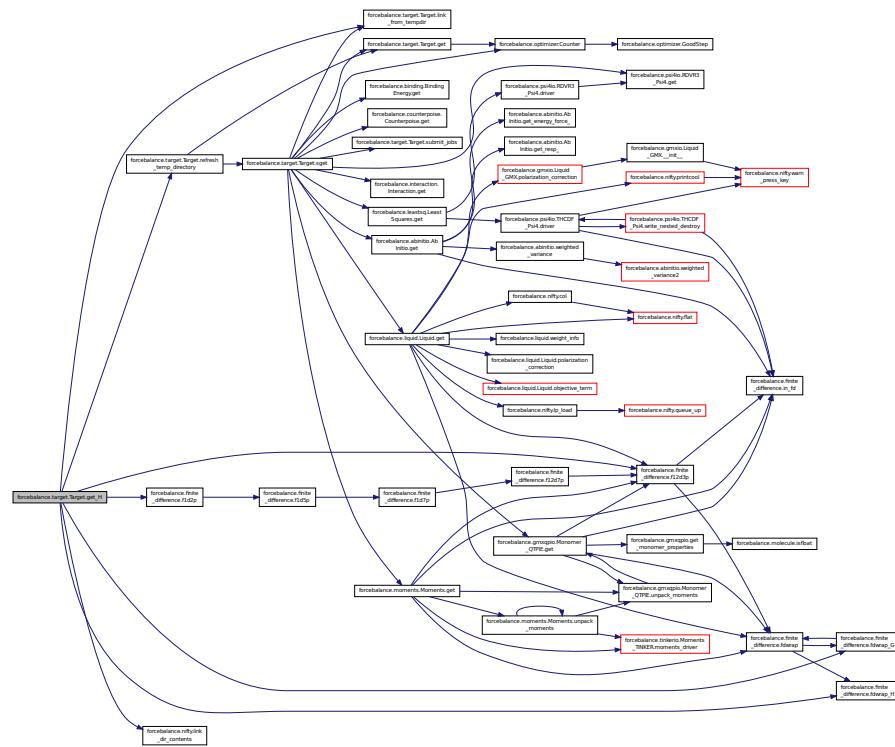
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

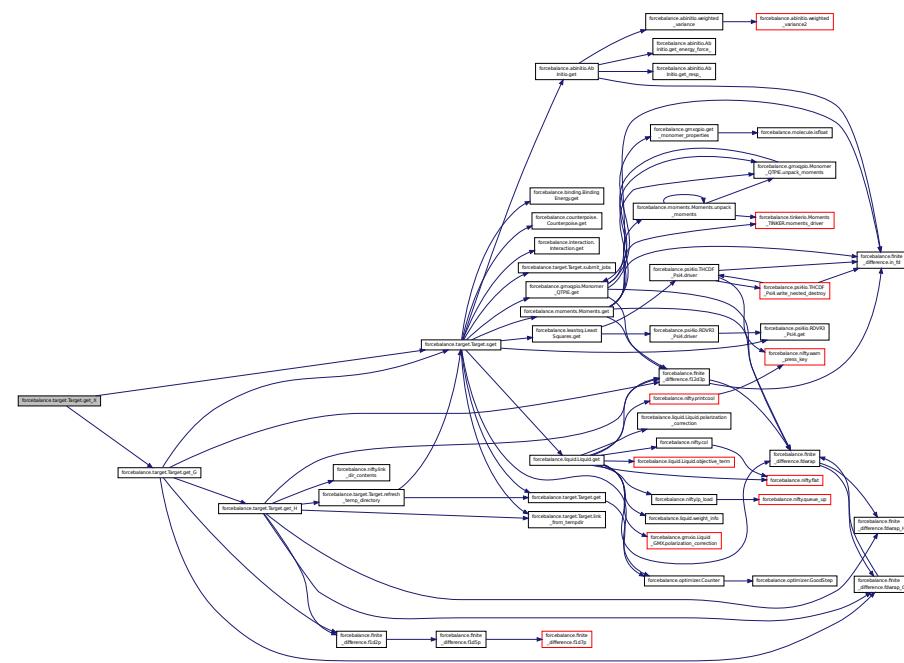
Here is the call graph for this function:



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

Definition at line 155 of file target.py.

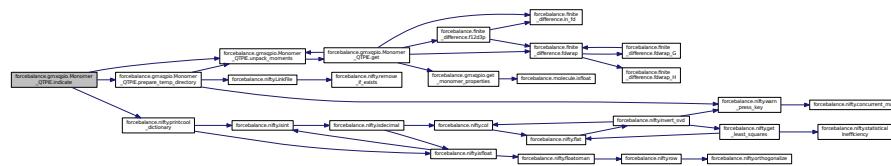
Here is the call graph for this function:



**def forcebalance.gmxqpio.Monomer\_QTPIE.indicate ( self )** Print qualitative indicator.

Definition at line 179 of file gmxqpio.py.

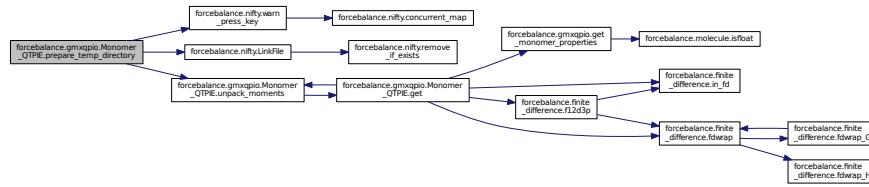
Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

**def forcebalance.gmxqpio.Monomer\_QTPIE.prepare\_temp\_directory ( self, options, tgt\_opts )** Definition at line 200 of file gmxqpio.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.

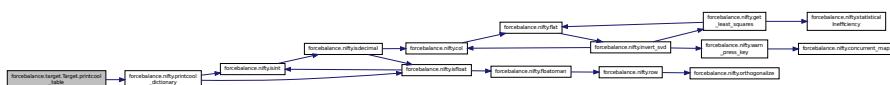
The input should be something like:

Parameters

<i>data</i>	Column contents in the form of an <i>OrderedDict</i> , 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 <i>OrderedDict</i> , 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 367 of file target.py.

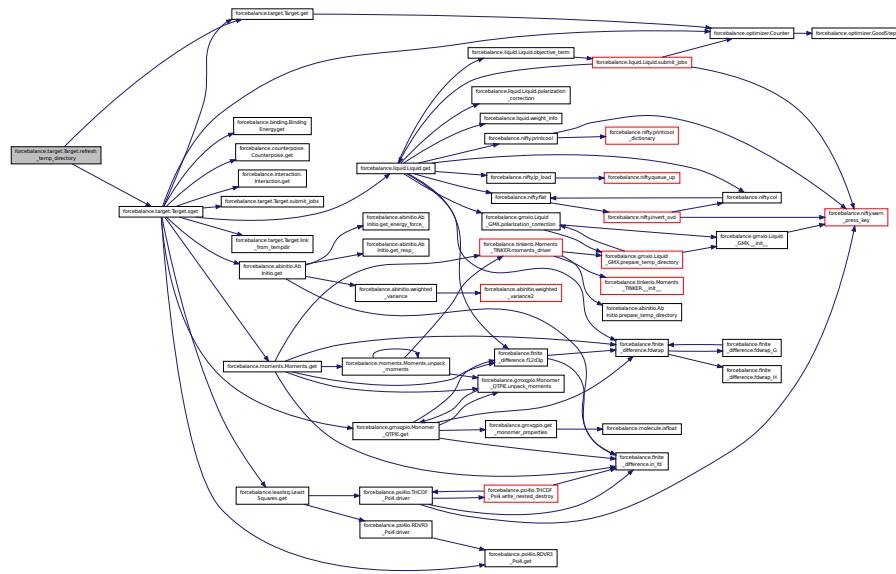
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



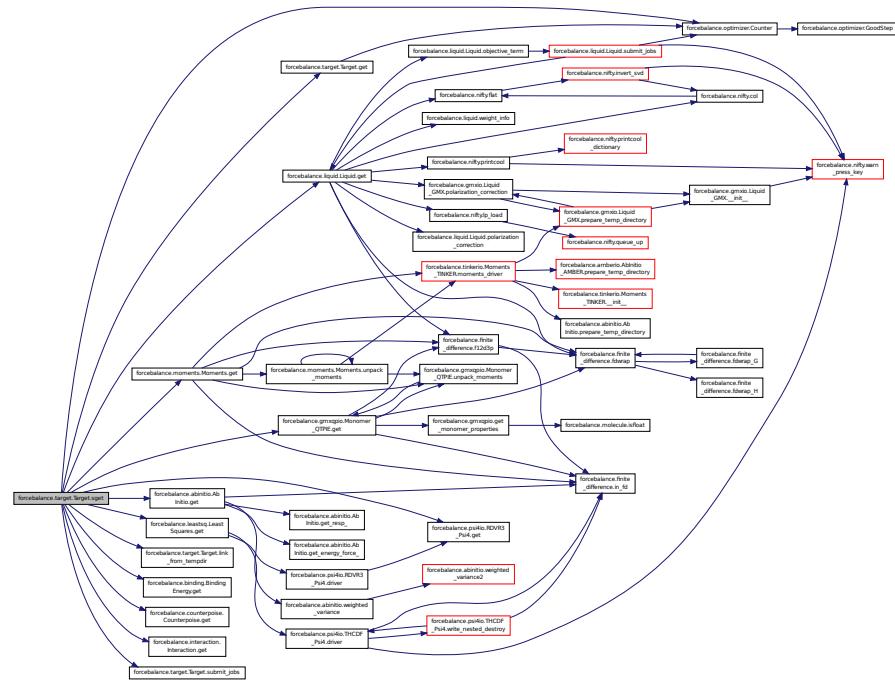
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget( self, mvals, AGrad = False, AHess = False, customdir = None ) [inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

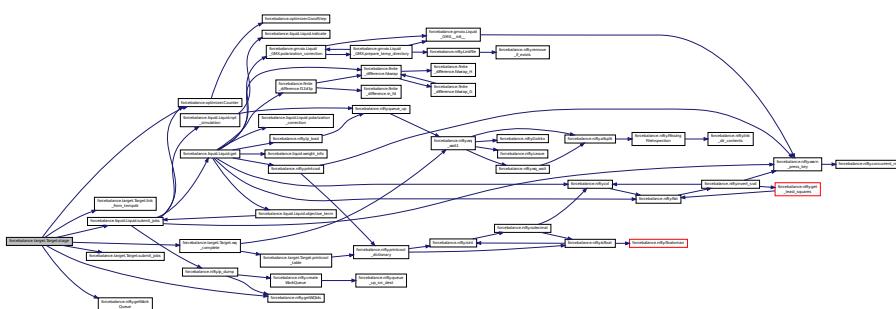


**[inherited]** Stages the directory for the target, and then launches Work Queue processes if any.

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

Definition at line 301 of file target.py.

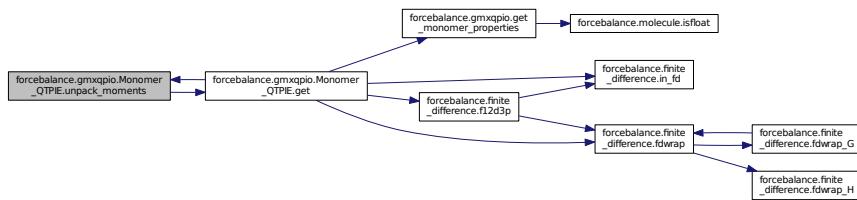
Here is the call graph for this function:



```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )  
[inherited] Definition at line 291 of file target.py.
```

**def forcebalance.gmxqpio.Monomer\_QTPIE.unpack\_moments ( self, moment\_dict )** Definition at line 217 of file gmxqpio.py.

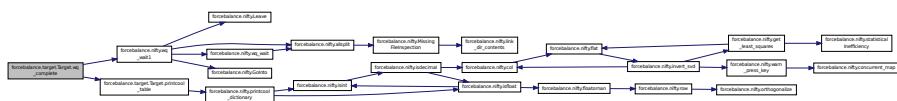
Here is the call graph for this function:



**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 331 of file target.py.

Here is the call graph for this function:



#### **8.44.4 Member Data Documentation**

**forcebalance.gmxqpio.Monomer\_QTPIE.calc\_moments** Definition at line 258 of file gmxqpio.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.

Definition at line 145 of file target.py.

**forcebalance.qmxqpio.Monomer\_QTPIE.objective** Definition at line 259 of file qmxqpio.py.

RECORDED BY: [REDACTED] DATE: [REDACTED] - Definition at line 25 of the LEXICON

**forcebalance.gmxqpio.Monomer.QTPRE.refmoments** Definition at line 151 of file gmxqpio.py.

**forcebalance.target.target.rundir** [inherited] The directory in which the simulation is run.

The 'customdir' is customizable and can go below anything.  
B. [initializing 107 .fil files](#)

Definition at line 137 of file target.py.

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

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.gmxqpio.Monomer\_QTPIE.weights** Definition at line 165 of file gmxqpio.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 141 of file target.py.

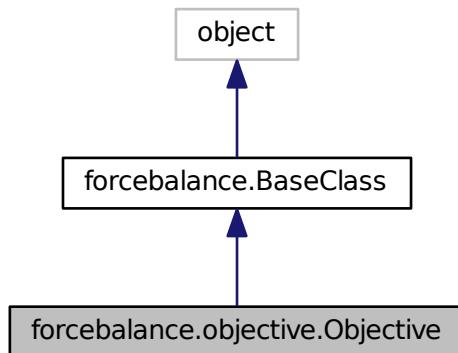
The documentation for this class was generated from the following file:

- [gmxqpio.py](#)

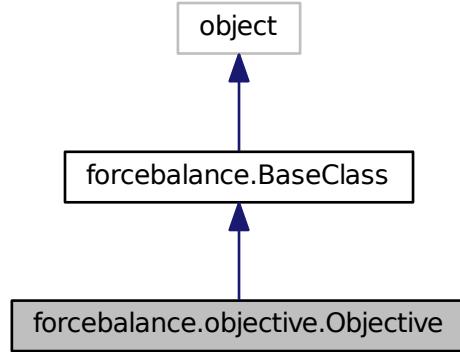
## 8.45 forcebalance.objective.Objective Class Reference

[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 `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`
- `PrintOptionDict`
- `verbose_options`

#### 8.45.1 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 114 of file objective.py.

### 8.45.2 Constructor & Destructor Documentation

```
def forcebalance.objective.Objective.__init__ ( self, options, tgt_opts, forcefield ) Definition at line 115 of file
objective.py.
```

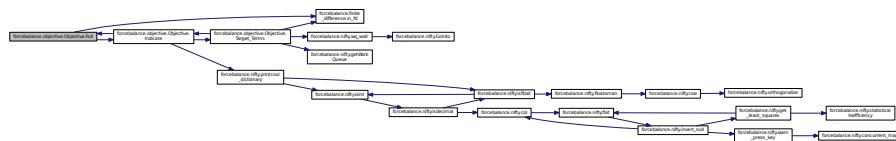
Here is the call graph for this function:



### 8.45.3 Member Function Documentation

```
def forcebalance.objective.Objective.Full ( self, mvals, Order = 0, verbose = False ) Definition at line 260
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 222 of file objective.py.

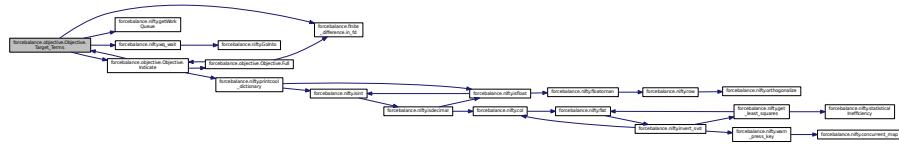
Here is the call graph for this function:



```
def forcebalance.BaseClass.set_option ( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.objective.Objective.Target_Terms ( self, mvals, Order = 0, verbose = False ) Definition at line 162 of file objective.py.
```

Here is the call graph for this function:



#### 8.45.4 Member Data Documentation

**forcebalance.objective.Objective.FF** The force field (it seems to be everywhere)

Definition at line 141 of file objective.py.

**forcebalance.objective.Objective.ObjDict** Definition at line 151 of file objective.py.

**forcebalance.objective.Objective.ObjDict.Last** Definition at line 152 of file objective.py.

**forcebalance.objective.Objective.Penalty** Initialize the penalty function.

Definition at line 143 of file objective.py.

**forcebalance.BaseClass.PrintOptionDict** [[inherited](#)] Definition at line 29 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 130 of file objective.py.

**forcebalance.BaseClass.verbose\_options** [[inherited](#)] Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.objective.Objective.WTot** Obtain the denominator.

Definition at line 148 of file objective.py.

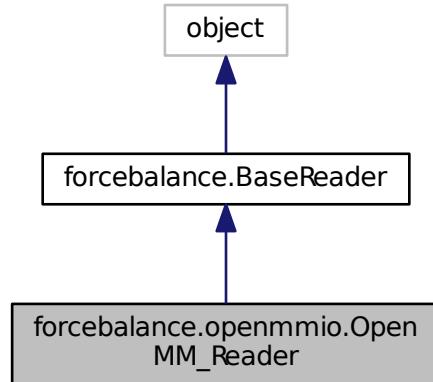
The documentation for this class was generated from the following file:

- [objective.py](#)

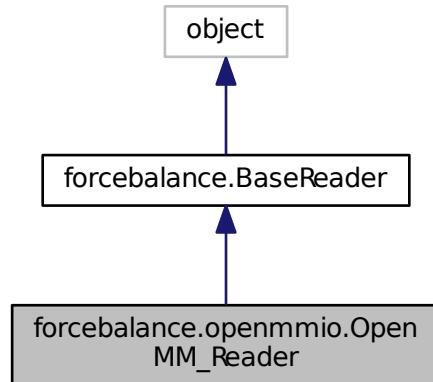
### 8.46 forcebalance.openmmio.OpenMM\_Reader Class Reference

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.*

- [ln](#)

- [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](#)

### 8.46.1 Detailed Description

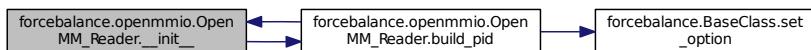
Class for parsing OpenMM force field files.

Definition at line 324 of file openmmio.py.

### 8.46.2 Constructor & Destructor Documentation

**def forcebalance.openmmio.OpenMM\_Reader.\_\_init\_\_ ( self, fnm )** Definition at line 325 of file openmmio.py.

Here is the call graph for this function:



### 8.46.3 Member Function Documentation

**def forcebalance.BaseReader.build\_pid ( self, pfld ) [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'

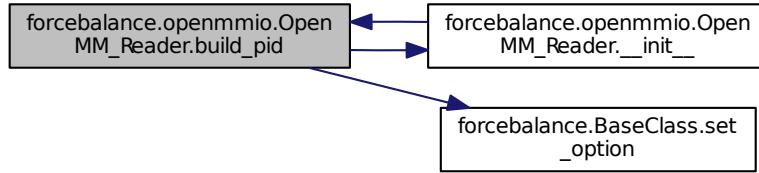
Definition at line 107 of file \_\_init\_\_.py.

**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 334 of file openmmio.py.

Here is the call graph for this function:



**def forcebalance.BaseReader.feed ( self, line ) [inherited]** Definition at line 88 of file \_\_init\_\_.py.

Here is the call graph for this function:



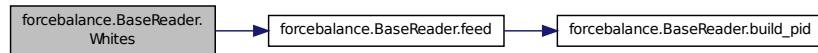
**def forcebalance.BaseReader.Split ( self, line ) [inherited]** Definition at line 82 of file \_\_init\_\_.py.

Here is the call graph for this function:



**def forcebalance.BaseReader.Whites ( self, line ) [inherited]** Definition at line 85 of file \_\_init\_\_.py.

Here is the call graph for this function:



#### 8.46.4 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 72 of file \_\_init\_\_.py.

**forcebalance.BaseReader.AtomTypes** [inherited] Definition at line 80 of file \_\_init\_\_.py.

**forcebalance.BaseReader.itype** [inherited] Definition at line 68 of file \_\_init\_\_.py.

**forcebalance.BaseReader.In** [inherited] Definition at line 67 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 77 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules** [inherited] Definition at line 79 of file \_\_init\_\_.py.

**forcebalance.openmmio.OpenMM.Reader.pdict** Initialize the superclass.

:) The parameter dictionary (defined in this file)

Definition at line 329 of file openmmio.py.

**forcebalance.BaseReader.suffix** [inherited] Definition at line 69 of file \_\_init\_\_.py.

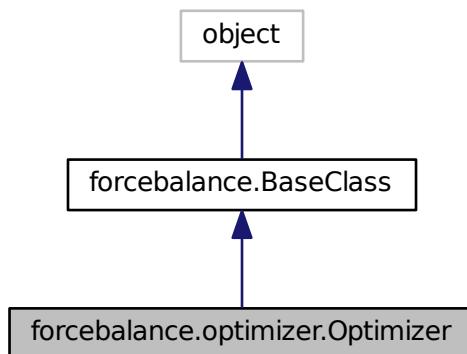
The documentation for this class was generated from the following file:

- [openmmio.py](#)

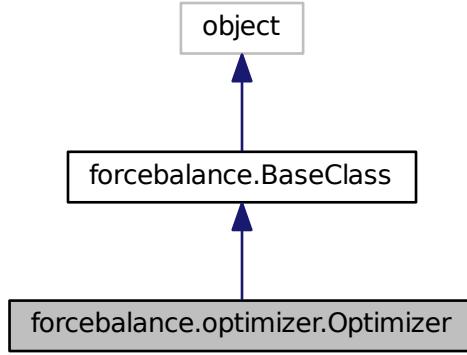
## 8.47 forcebalance.optimizer.Optimizer Class Reference

[Optimizer](#) class.

Inheritance diagram for forcebalance.optimizer.Optimizer:



Collaboration diagram for forcebalance.optimizer.Optimizer:



## Public Member Functions

- def `__init__`  
*Create an [Optimizer](#) object.*
- def `Run`  
*Call the appropriate optimizer.*
- 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 simulated annealing 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 `set_option`

## Public Attributes

- `OptTab`  
*A list of all the things we can ask the optimizer to do.*
- `Objective`  
*The root directory.*
- `bhyp`  
*Whether the penalty function is hyperbolic.*
- `FF`  
*The force field itself.*
- `excision`  
*The indices to be excluded from the Hessian update.*
- `np`  
*Number of parameters.*
- `mvals0`  
*The original parameter values.*
- `chk`  
*Put data into the checkpoint file.*
- `H`
- `dx`
- `Val`
- `Grad`
- `Hess`
- `Penalty`
- `PrintOptionDict`
- `verbose_options`

### 8.47.1 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 44 of file optimizer.py.

### 8.47.2 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 57 of file optimizer.py.

Here is the call graph for this function:



### 8.47.3 Member Function Documentation

**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 784 of file optimizer.py.

**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 605 of file optimizer.py.

**def forcebalance.optimizer.Optimizer.ConjugateGradient( self )** Use SciPy's built-in simulated annealing algorithm to optimize the parameters.

See Also

[Optimizer::ScipyOptimizer](#)

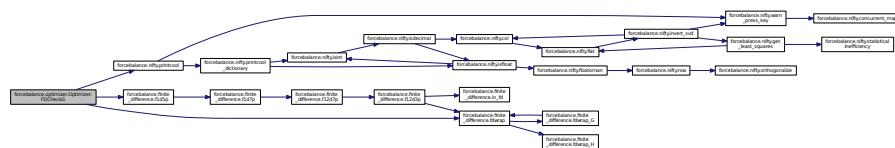
Definition at line 789 of file optimizer.py.

```
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 885 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.

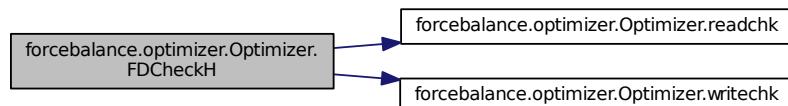
For each element in the Hessian, use a five-point stencil in both parameter indices to compute a finite-difference derivative, and compare it to the analytic result.

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 917 of file optimizer.py.

Here is the call graph for this function:



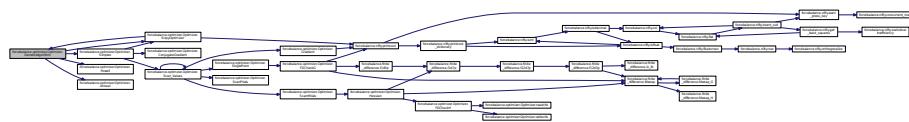
**def forcebalance.optimizer.Optimizer.GeneticAlgorithm ( self )** Genetic algorithm, under development.

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 681 of file optimizer.py.

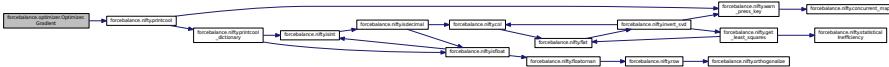
Here is the call graph for this function:



**def forcebalance.optimizer.Optimizer.Gradient ( self )** A single-point gradient computation.

Definition at line 863 of file optimizer.py.

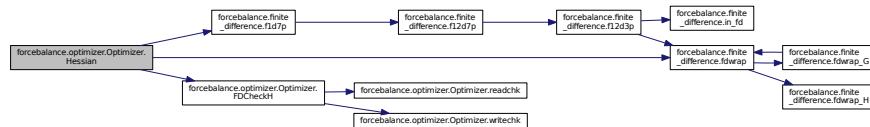
Here is the call graph for this function:



**def forcebalance.optimizer.Optimizer.Hessian ( self )** A single-point Hessian computation.

Definition at line 871 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 is vestigial and currently does not work.

BFGS is a pseudo-Newton method in the sense that it builds an approximate Hessian matrix from the gradient information in previous steps; Newton-Raphson requires the actual Hessian matrix. However, the algorithms are similar in that they both compute the step by inverting the Hessian and multiplying by the gradient.

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 229 of file optimizer.py.

Here is the call graph for this function:



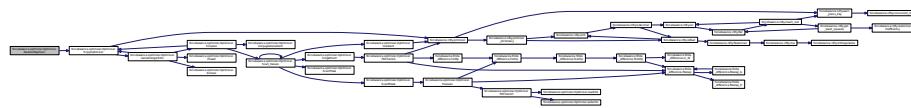
```
def forcebalance.optimizer.Optimizer.NewtonRaphson ( self ) Optimize the force field parameters using the Newton-Raphson method .
```

See Also

[MainOptimizer](#))

Definition at line 600 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 779 of file optimizer.py.

```
def forcebalance.optimizer.Optimizer.readchk ( self ) Read the checkpoint file for the main optimizer.
```

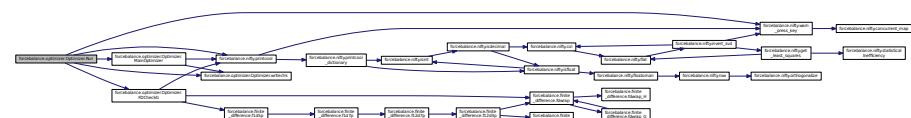
Definition at line 945 of file optimizer.py.

```
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 168 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.

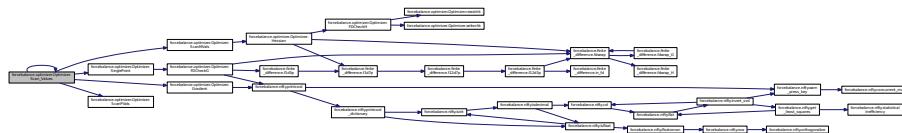
**Todo** Maybe a multidimensional grid can be done.

## Parameters

**in**      *MathPhys*      Switch to use mathematical (True) or physical (False) parameters.

Definition at line 815 of file optimizer.py.

Here is the call graph for this function:



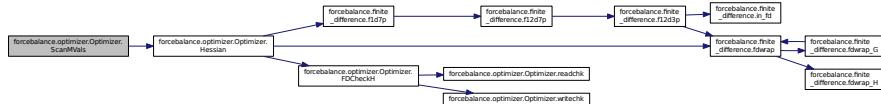
**def forcebalance.optimizer.Optimizer.ScanMVals ( self )** Scan through the mathematical parameter space.

#### See Also

## Optimizer::ScanValues

Definition at line 847 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 852 of file optimizer.py.

```
def forcebalance.optimizer.Optimizer.ScipyOptimizer ( self, Algorithm = "None" ) Driver for SciPy optimizations.
```

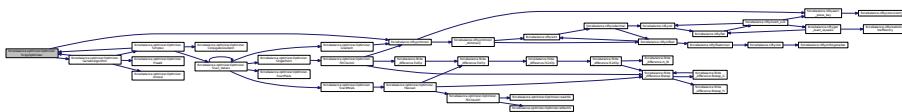
Using any of the SciPy optimizers requires that SciPy is installed. This method first defines several wrappers around the objective function that the SciPy optimizers can use. Then it calls the algorithm itself.

### Parameters

`in`      *Algorithm*      The optimization algorithm to use, for example 'powell', 'simplex' or 'anneal'

Definition at line 618 of file optimizer.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 32 of file __init__.py.
```

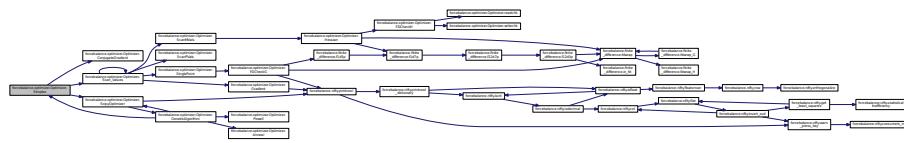
**def forcebalance.optimizer.Optimizer.Simplex ( self )** Use SciPy's built-in simplex algorithm to optimize the parameters.

See Also

[Optimizer::ScipyOptimizer](#)

Definition at line 774 of file optimizer.py.

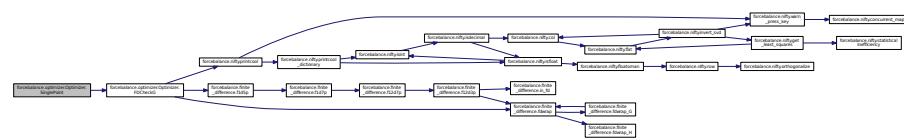
Here is the call graph for this function:



```
def forcebalance.optimizer.Optimizer.SinglePoint ( self ) A single-point objective function computation.
```

Definition at line 857 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 421 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 957 of file optimizer.py.

#### 8.47.4 Member Data Documentation

**forcebalance.optimizer.Optimizer.bhyp** Whether the penalty function is hyperbolic.

Definition at line 141 of file optimizer.py.

**forcebalance.optimizer.Optimizer.chk** Put data into the checkpoint file.

Definition at line 272 of file optimizer.py.

**forcebalance.optimizer.Optimizer.dx** Definition at line 450 of file optimizer.py.

**forcebalance.optimizer.Optimizer.excision** The indices to be excluded from the Hessian update.

Definition at line 149 of file optimizer.py.

**forcebalance.optimizer.Optimizer.FF** The force field itself.

Definition at line 143 of file optimizer.py.

**forcebalance.optimizer.Optimizer.Grad** Definition at line 452 of file optimizer.py.

**forcebalance.optimizer.Optimizer.H** Definition at line 449 of file optimizer.py.

**forcebalance.optimizer.Optimizer.Hess** Definition at line 453 of file optimizer.py.

**forcebalance.optimizer.Optimizer.mvals0** The original parameter values.

Sometimes the optimizer doesn't return anything (i.e.

in the case of a single point calculation) In these situations, don't do anything Check derivatives by finite difference after the optimization is over (for good measure)

Definition at line 155 of file optimizer.py.

**forcebalance.optimizer.Optimizer.np** Number of parameters.

Definition at line 152 of file optimizer.py.

**forcebalance.optimizer.Optimizer.Objective** The root directory.

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) Guess value for Brent Step size for numerical finite difference Number of steps to average over Function value convergence threshold Step size convergence threshold Gradient convergence threshold 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 nonlinear search will stop if the change is below this threshold) The objective function (needs to pass in when I instantiate)

Definition at line 139 of file optimizer.py.

**forcebalance.optimizer.Optimizer.OptTab** A list of all the things we can ask the optimizer to do.

Definition at line 61 of file optimizer.py.

**forcebalance.optimizer.Optimizer.Penalty** Definition at line 454 of file optimizer.py.

**forcebalance.BaseClass.PrintOptionDict** [**inherited**] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.optimizer.Optimizer.Val** Definition at line 451 of file optimizer.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file `__init__.py`.

The documentation for this class was generated from the following file:

- [optimizer.py](#)

## 8.48 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`

#### 8.48.1 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 317 of file objective.py.

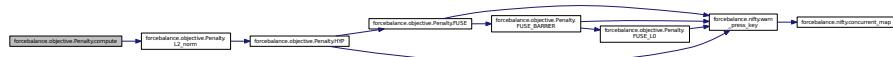
#### 8.48.2 Constructor & Destructor Documentation

```
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 323 of file objective.py.
```

#### 8.48.3 Member Function Documentation

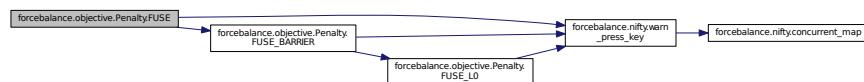
```
def forcebalance.objective.Penalty.compute ( self, mvals, Objective ) Definition at line 349 of file objective.py.
```

Here is the call graph for this function:



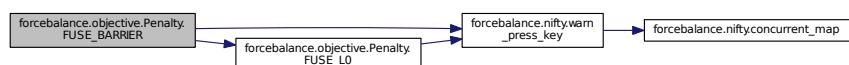
```
def forcebalance.objective.Penalty.FUSE ( self, mvals ) Definition at line 409 of file objective.py.
```

Here is the call graph for this function:



```
def forcebalance.objective.Penalty.FUSE_BARRIER ( self, mvals ) Definition at line 450 of file objective.py.
```

Here is the call graph for this function:



```
def forcebalance.objective.Penalty.FUSE_L0 ( self, mvals ) Definition at line 492 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

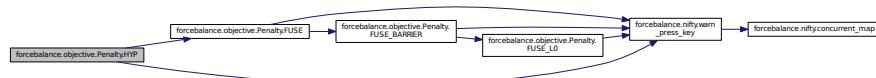
DC0 The hyperbolic penalty

DC1 The gradient

DC2 The Hessian

Definition at line 401 of file objective.py.

Here is the call graph for this function:



**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 381 of file objective.py.

Here is the call graph for this function:



## 8.48.4 Member Data Documentation

**forcebalance.objective.Penalty.a** Definition at line 326 of file objective.py.

**forcebalance.objective.Penalty.b** Definition at line 327 of file objective.py.

**forcebalance.objective.Penalty.fadd** Definition at line 324 of file objective.py.

**forcebalance.objective.Penalty.FF** Definition at line 328 of file objective.py.

**forcebalance.objective.Penalty.fmul** Definition at line 325 of file objective.py.

**dictionary forcebalance.objective.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 318 of file objective.py.

**forcebalance.objective.Pen\_Tab** Definition at line 330 of file objective.py.

**forcebalance.objective.Pen\_ptyp** Definition at line 329 of file objective.py.

**forcebalance.objective.Penalty.spacings** Find exponential spacings.

Definition at line 346 of file objective.py.

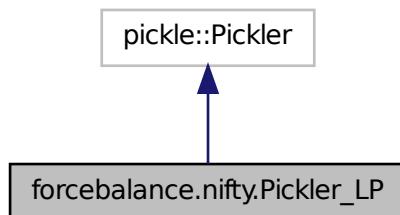
The documentation for this class was generated from the following file:

- [objective.py](#)

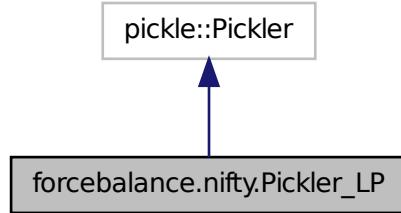
## 8.49 forcebalance.nifty.Pickler\_LP Class Reference

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__`

#### 8.49.1 Detailed Description

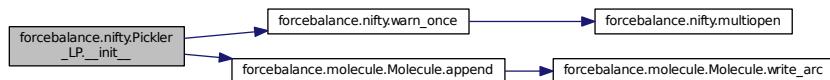
A subclass of the python Pickler that implements pickling of `_ElementTree` types.

Definition at line 500 of file nifty.py.

#### 8.49.2 Constructor & Destructor Documentation

`def forcebalance.nifty.Pickler_LP.__init__( self, file, protocol = None )` Definition at line 501 of file nifty.py.

Here is the call graph for this function:



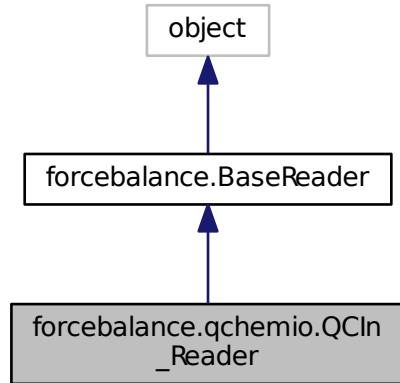
The documentation for this class was generated from the following file:

- [nifty.py](#)

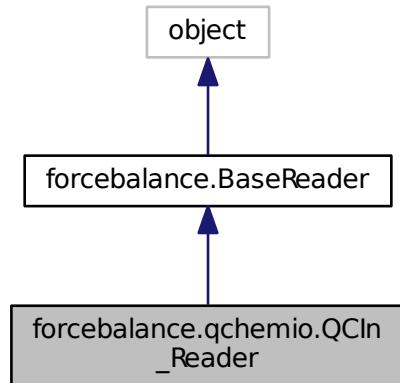
## 8.50 forcebalance.qchemio.QCIn Reader Class Reference

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**  
The mapping of (this residue, atom number) to (atom name) for building atom-specific interactions in [ bonds ], [ angles ] etc.
  - **snum**
  - **cnum**
  - **shell**
  - **pdict**
  - **sec**
  - **itype**
  - **suffix**
  - **In**
  - **adict**
- The mapping of (molecule name) to a dictionary of atom types for the atoms in that residue.
- **Molecules**
  - **AtomTypes**

### 8.50.1 Detailed Description

Finite state machine for parsing Q-Chem input files.

Definition at line 31 of file qchemio.py.

### 8.50.2 Constructor & Destructor Documentation

```
def forcebalance.qchemio.QCIn_Reader.__init__ ( self, fnm )
```

 Definition at line 33 of file qchemio.py.

### 8.50.3 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 107 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]
```

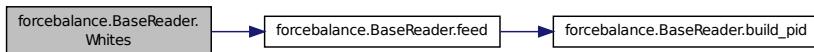
 Definition at line 82 of file \_\_init\_\_.py.

Here is the call graph for this function:



```
def forcebalance.BaseReader.Whites( self, line ) [inherited] Definition at line 85 of file __init__.py.
```

Here is the call graph for this function:



#### 8.50.4 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 72 of file \_\_init\_\_.py.

**forcebalance.qchemio.QCIn\_Reader.atom** Definition at line 36 of file qchemio.py.

**forcebalance.BaseReader.AtomTypes** [inherited] Definition at line 80 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 67 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 77 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules** [inherited] Definition at line 79 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.

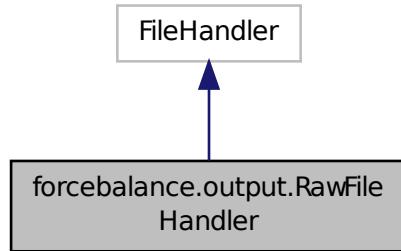
**forcebalance.qchemio.QCIn\_Reader.suffix** Definition at line 72 of file qchemio.py.

The documentation for this class was generated from the following file:

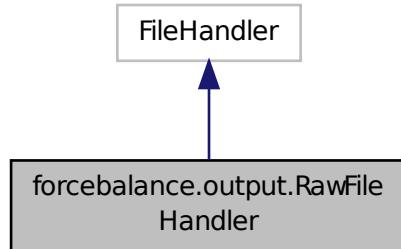
- [qchemio.py](#)

## 8.51 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](#)

#### 8.51.1 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](#).

#### 8.51.2 Member Function Documentation

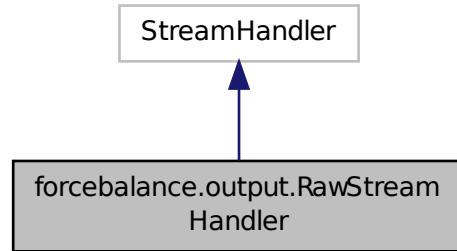
**def forcebalance.output.RawFileHandler.emit ( *self*, *record* )** Definition at line 48 of file [output.py](#).

The documentation for this class was generated from the following file:

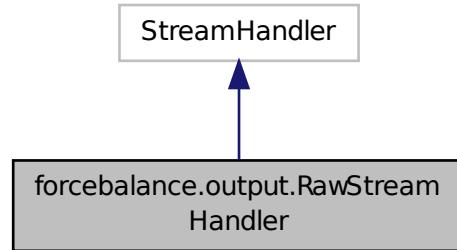
- [output.py](#)

## 8.52 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`

#### 8.52.1 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.

#### 8.52.2 Constructor & Destructor Documentation

```
def forcebalance.output.RawStreamHandler.__init__ ( self, stream = sys.stdout ) Definition at line 35 of file output.py.
```

Here is the call graph for this function:



### 8.52.3 Member Function Documentation

```
def forcebalance.output.RawStreamHandler.emit ( self, record ) Definition at line 38 of file output.py.
```

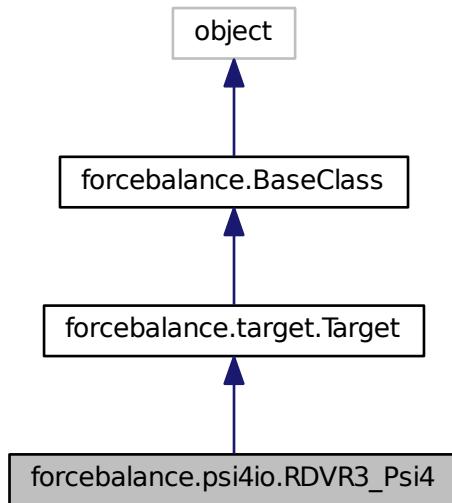
The documentation for this class was generated from the following file:

- [output.py](#)

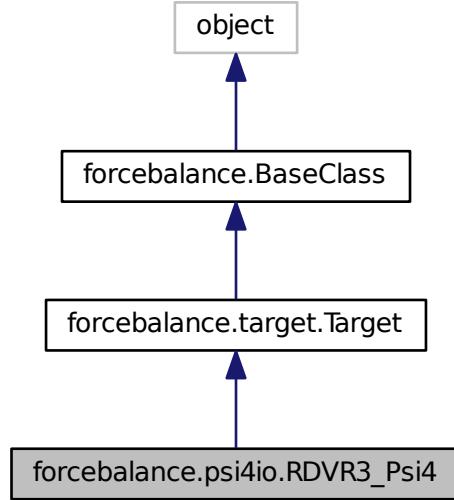
## 8.53 forcebalance.psi4io.RDVR3\_Psi4 Class Reference

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 `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 `sget`

*Stages the directory for the target, and then calls 'get'.*
- 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 `set_option`

## Public Attributes

- `objfiles`

*Which parameters are differentiated?*

- `objvals`

- `elements`

- `molecules`

- `callderivs`

- `factor`

- `bidirect`

- `tdir`

- `objd`

- `gradd`

- `hdiagd`

- `objective`

- `tempdir`

*Root directory of the whole project.*

- `rundir`

*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.*

- `PrintOptionDict`

- `verbose_options`

### 8.53.1 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 296 of file psi4io.py.

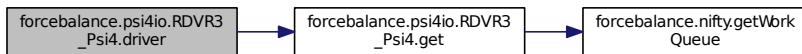
### 8.53.2 Constructor & Destructor Documentation

```
def forcebalance.psi4io.RDVR3.Psi4__init__( self, options, tgt.opts, forcefield ) Definition at line 299 of file  
psi4io.py.
```

### 8.53.3 Member Function Documentation

**def forcebalance.psi4io.RDVR3.Psi4.driver ( self, mvals, d )** Definition at line 412 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

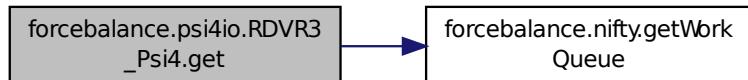
in	mvals	Mathematical parameter values
in	AGrad	Switch to turn on analytic gradient
in	AHess	Switch to turn on analytic Hessian

#### Returns

Answer Contribution to the objective function

Definition at line 449 of file psi4io.py.

Here is the call graph for this function:

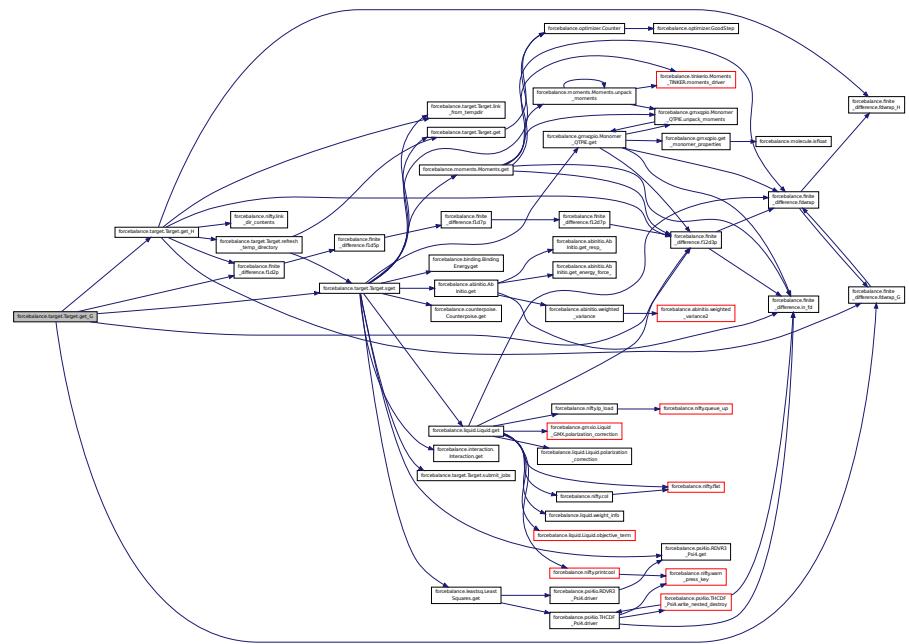


**def forcebalance.target.Target.get\_G ( self, mvals=None ) [inherited]** Computes the objective function contribution and its gradient.

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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



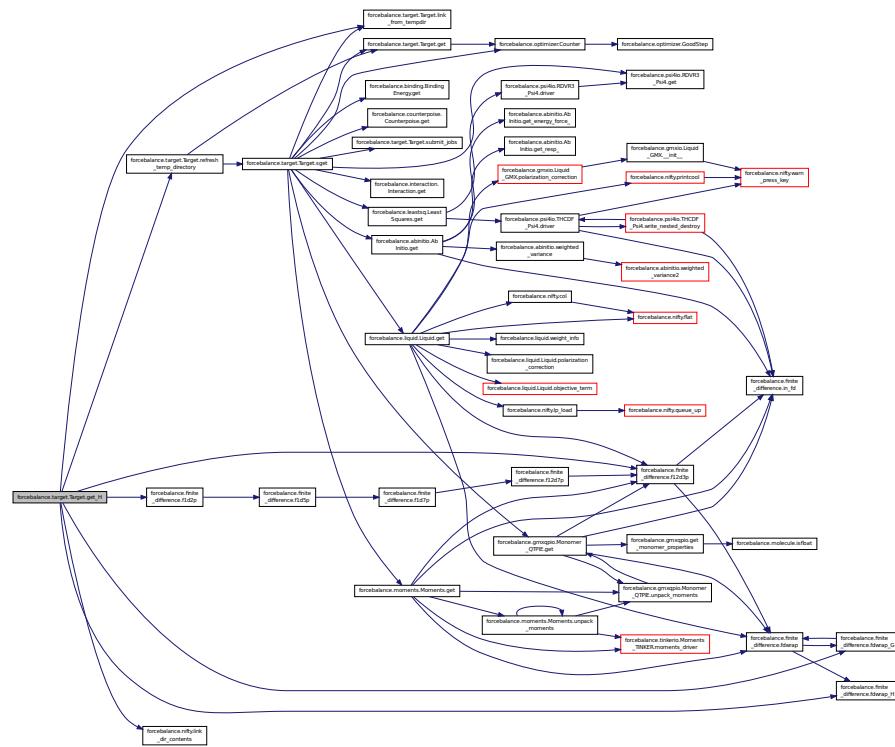
**def forcebalance.target.Target.get.H ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

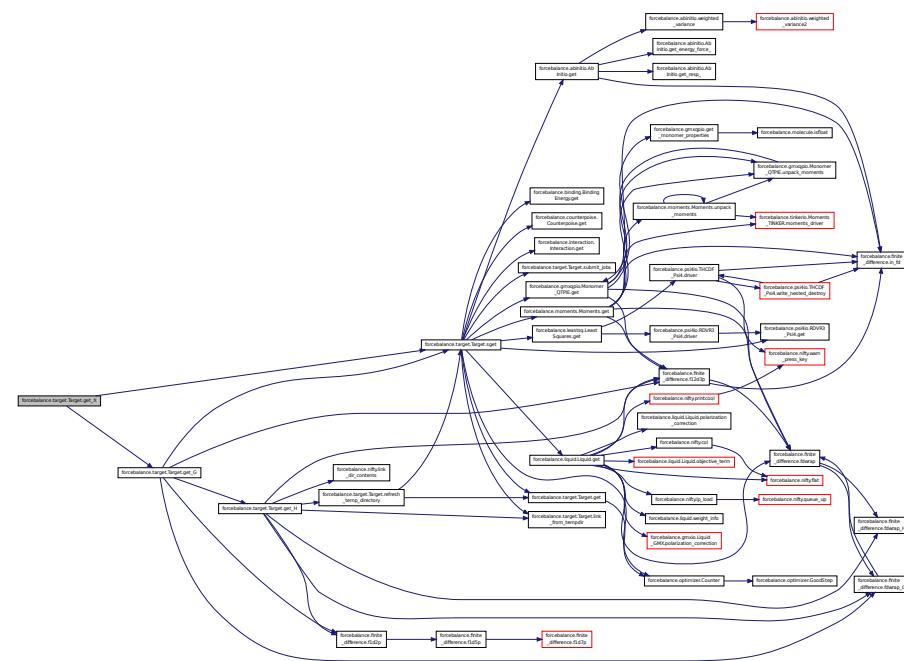
Here is the call graph for this function:



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

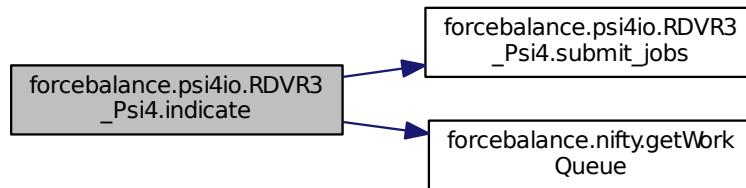
Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.psi4io.RDVR3.Psi4.indicate( self )** Definition at line 339 of file psi4io.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

```
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.

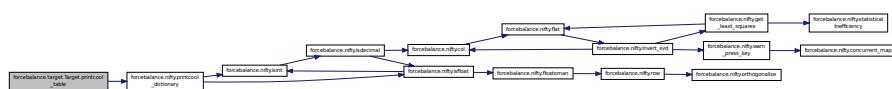
The input should be something like:

## 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 367 of file target.py.

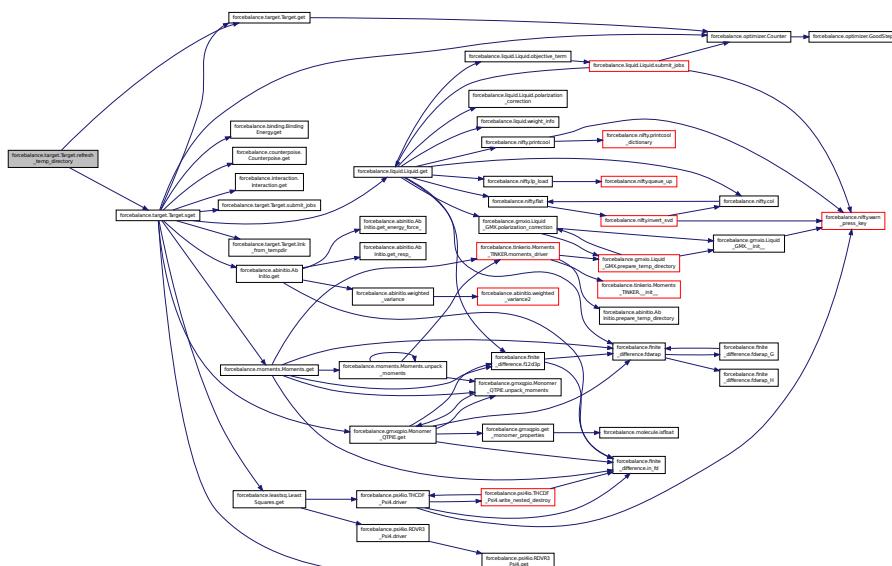
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



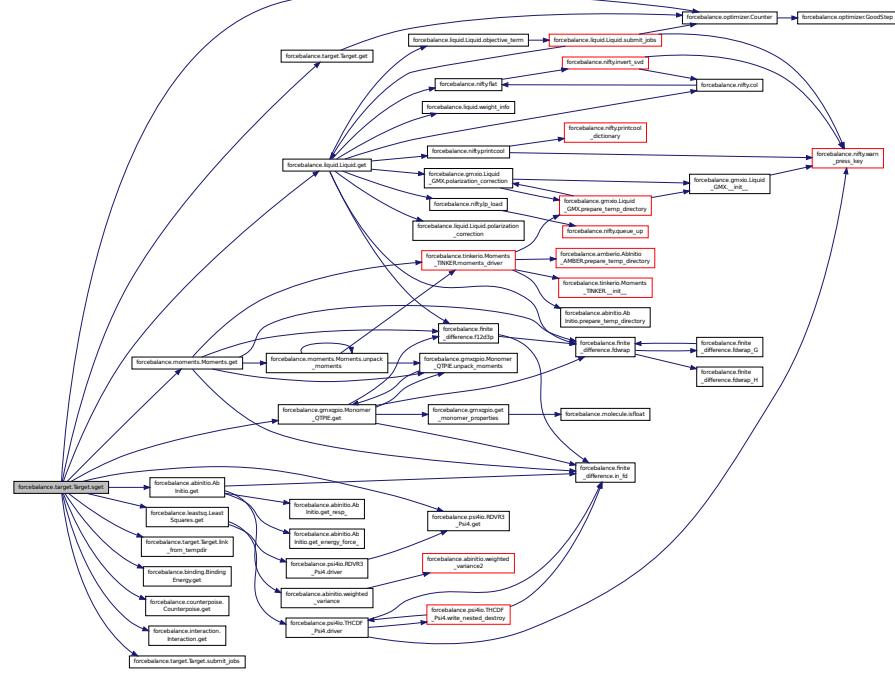
`def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited]` Definition at line 32 of file `_init_.py`.

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
    [inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py

Here is the call graph for this function:

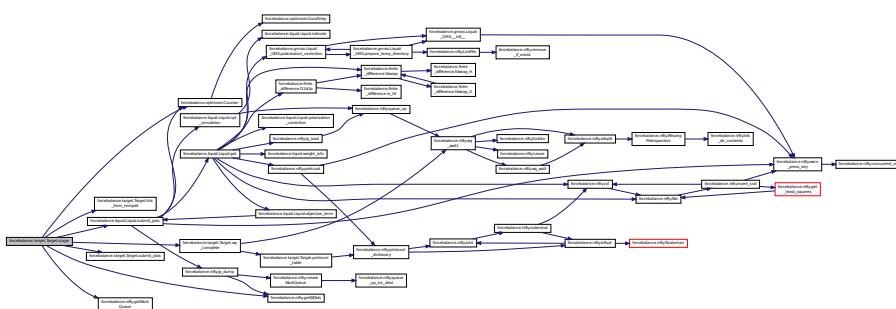


```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

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

Definition at line 301 of file target.py.

Here is the call graph for this function:



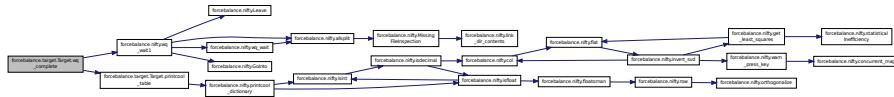
**def forcebalance.psi4io.RDVR3.Psi4.submit\_jobs( self, mvals, AGrad = True, AHess = True )** Create a grid file and a psi4 input file in the absolute path and submit it to the work queue.

Definition at line 348 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 331 of file target.py.

Here is the call graph for this function:



#### 8.53.4 Member Data Documentation

**forcebalance.psi4io.RDVR3\_Psi4.bidirect** Definition at line 311 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.callderivs** Definition at line 309 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.elements** Definition at line 307 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.factor** Definition at line 310 of file psi4io.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.psi4io.RDVR3\_Psi4.gradd** Definition at line 459 of file psi4io.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.

Definition at line 145 of file target.py.

**forcebalance.psi4io.RDVR3\_Psi4.hdiagd** Definition at line 460 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.molecules** Definition at line 308 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.objd** Definition at line 458 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.objective** Definition at line 535 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.objfiles** Which parameters are differentiated?

Definition at line 305 of file psi4io.py.

**forcebalance.psi4io.RDVR3\_Psi4.objvals** Definition at line 306 of file psi4io.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.target.Target.rundir** [inherited] 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.

Definition at line 137 of file target.py.

**forcebalance.psi4io.RDVR3.Psi4.tdir** Definition at line 352 of file psi4io.py.

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

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

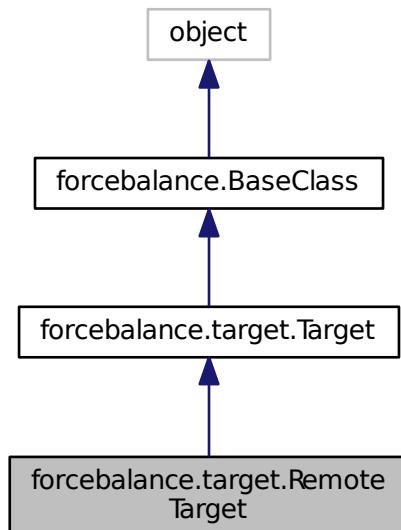
Definition at line 141 of file target.py.

The documentation for this class was generated from the following file:

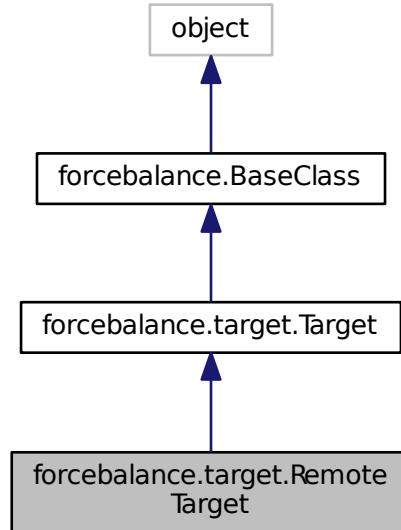
- [psi4io.py](#)

## 8.54 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 `get`
- def `indicate`
- def `get_X`

*Computes the objective function contribution without any parametric derivatives.*
- 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 `sget`

*Stages the directory for the target, and then calls 'get'.*
- 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 `set_option`

## Public Attributes

- `r_options`  
*Root directory of the whole project.*
  - `r_tgt_opts`
  - `remote_indicate`
  - `id_string`
  - `tempdir`
- `rundir`  
*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.*
  - `PrintOptionDict`
  - `verbose_options`

### 8.54.1 Detailed Description

Definition at line 406 of file target.py.

### 8.54.2 Constructor & Destructor Documentation

`def forcebalance.target.RemoteTarget.__init__ ( self, options, tgt_opts, forcefield )` Definition at line 407 of file target.py.

### 8.54.3 Member Function Documentation

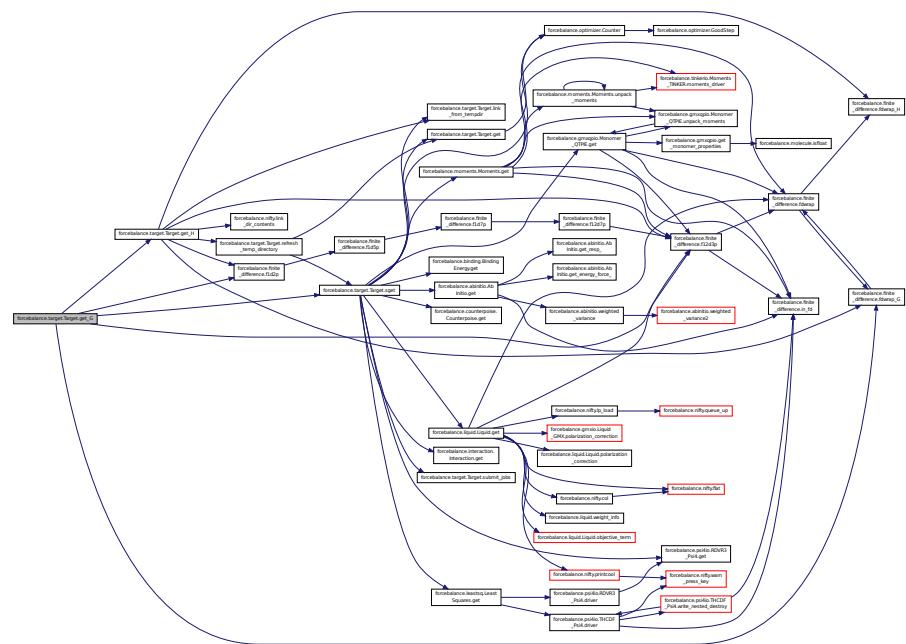
`def forcebalance.target.RemoteTarget.get ( self, mvals, AGrad = False, AHess = False )` Definition at line 452 of file target.py.

`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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



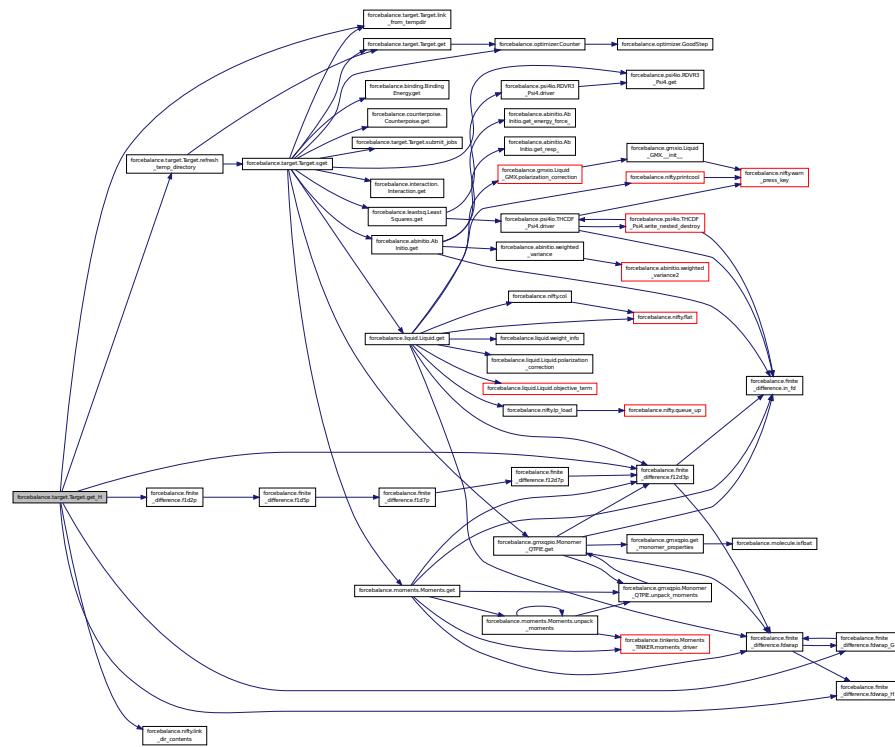
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

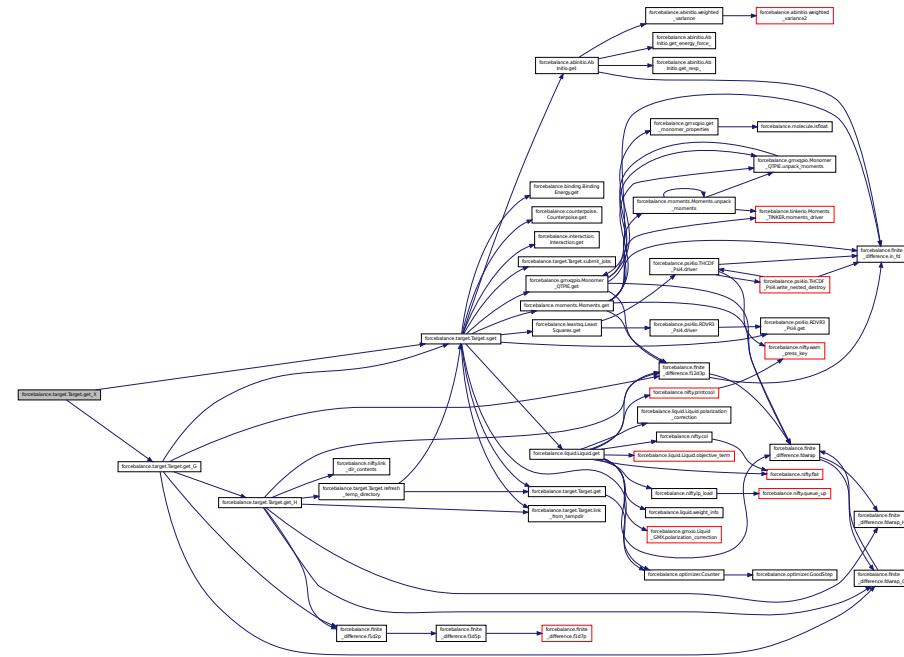
Here is the call graph for this function:



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

Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.RemoteTarget.indicate ( self )** Definition at line 458 of file target.py.

**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir ) [inherited]** Definition at line 213 of file target.py.

```
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.

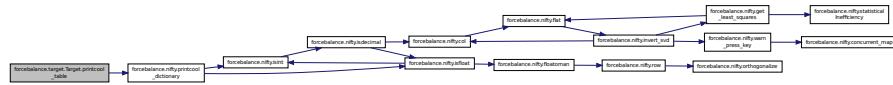
The input should be something like:

## 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 367 of file target.py.

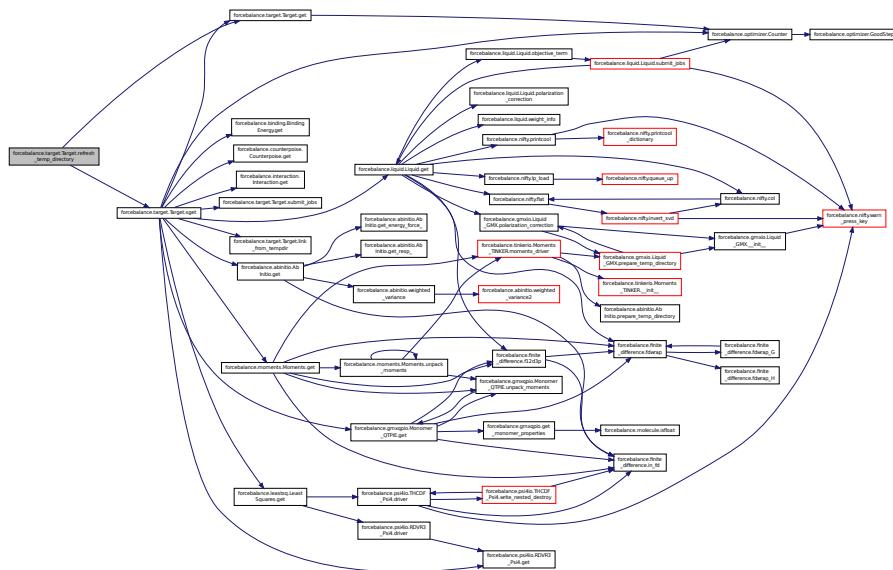
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



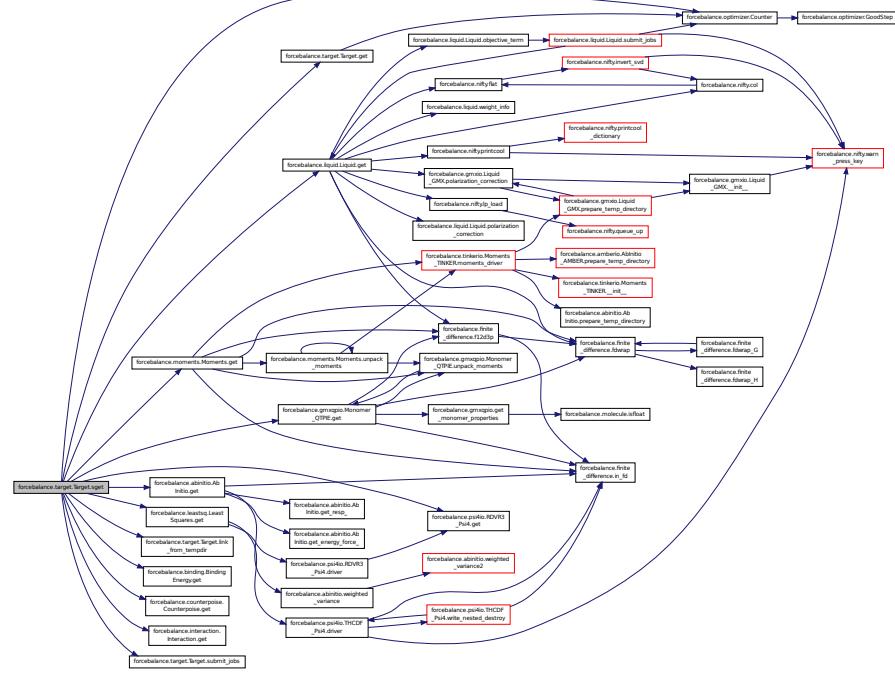
`def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited]` Definition at line 32 of file `__init__.py`.

**def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then calls 'get'.

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

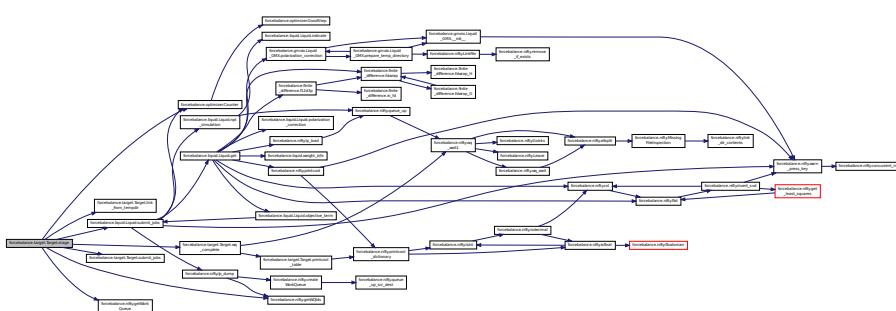


**def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )**  
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.

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

Definition at line 301 of file target.py.

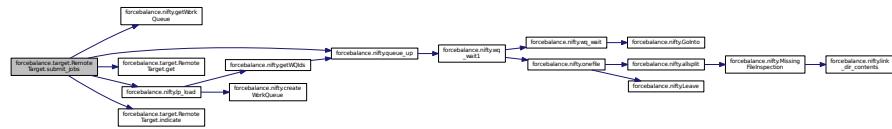
Here is the call graph for this function:



```
def forcebalance.target.RemoteTarget.submit_jobs ( self, mvals, AGrad = False, AHess = False )
```

Definition at line 422 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 331 of file target.py.

Here is the call graph for this function:



#### **8.54.4 Member Data Documentation**

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.

Definition at line 145 of file target.py.

**forcebalance.target.RemoteTarget.id\_string** Definition at line 450 of file target.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 29 of file `__init__.py`.

**forcebalance.target.RemoteTarget.r\_options** Definition at line 410 of file target.py.

**forcebalance.target.RemoteTarget.r\_tgt\_opts** Definition at line 413 of file target.py.

**forcebalance.target.RemoteTarget.remote\_indicate** Definition at line 420 of file target.py.

**forcebalance.target.Target.rundir** [inherited] 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.

Definition at line 137 of file target.py

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

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options [inherited]** Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 141 of file target.py.

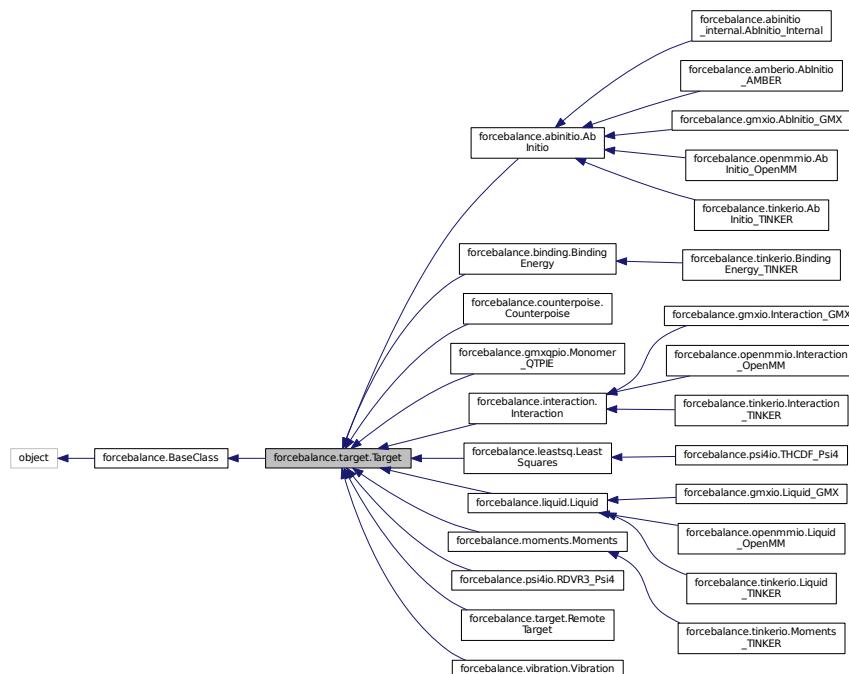
The documentation for this class was generated from the following file:

- [target.py](#)

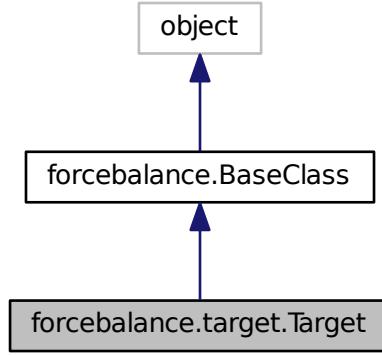
## 8.55 forcebalance.target.Target Class Reference

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 `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 `sget`

*Stages the directory for the target, and then calls 'get'.*
- 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 `set_option`

## Public Attributes

- `tempdir`  
*Root directory of the whole project.*
- `rundir`  
*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.*
- `PrintOptionDict`
- `verbose_options`

### 8.55.1 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': numpy.array(np), 'H': numpy.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`.

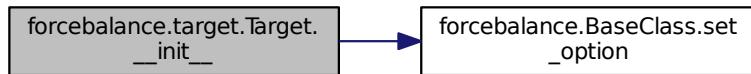
### 8.55.2 Constructor & Destructor Documentation

**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 `AbInitio` that inherits from `Target`.

Definition at line 92 of file `target.py`.

Here is the call graph for this function:



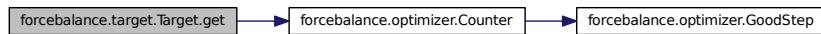
### 8.55.3 Member Function Documentation

**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 254 of file `target.py`.

Here is the call graph for this function:

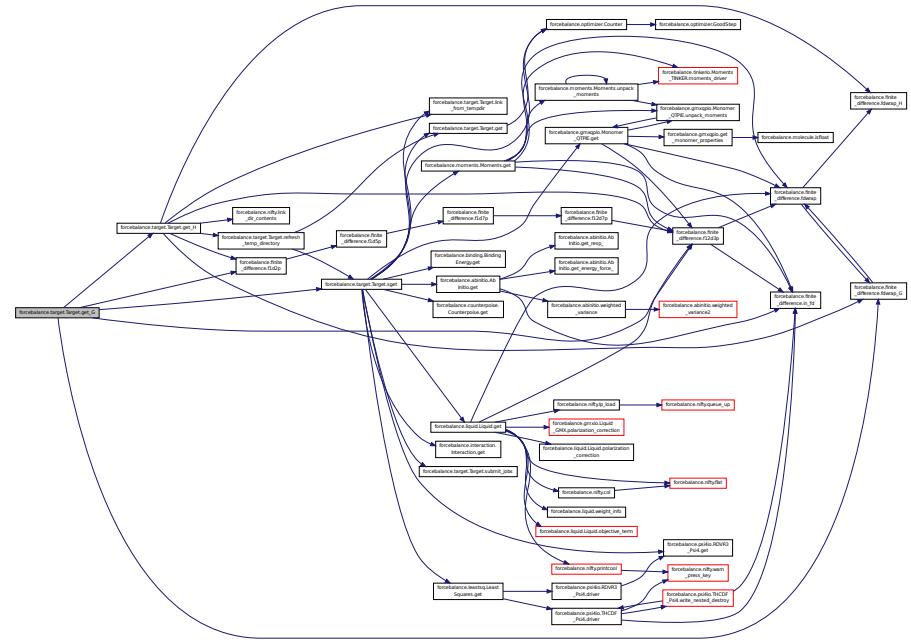


**def forcebalance.target.Target.get\_G ( self, mvals = None )** Computes the objective function contribution and its gradient.

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.

Definition at line 172 of file `target.py`.

Here is the call graph for this function:



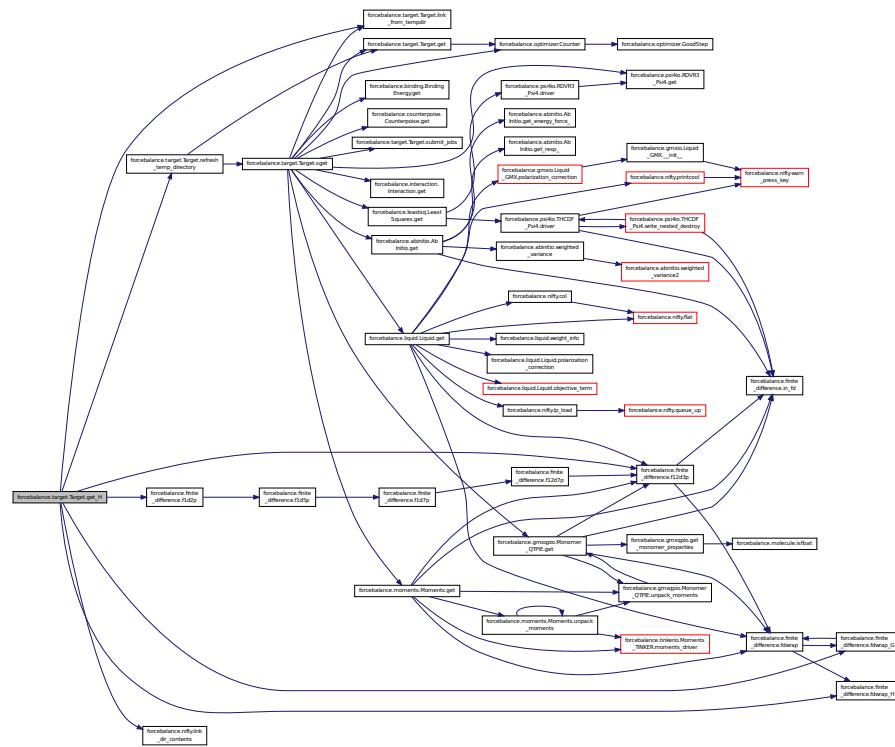
**def forcebalance.target.Target.get\_H ( self, mvals = None )** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

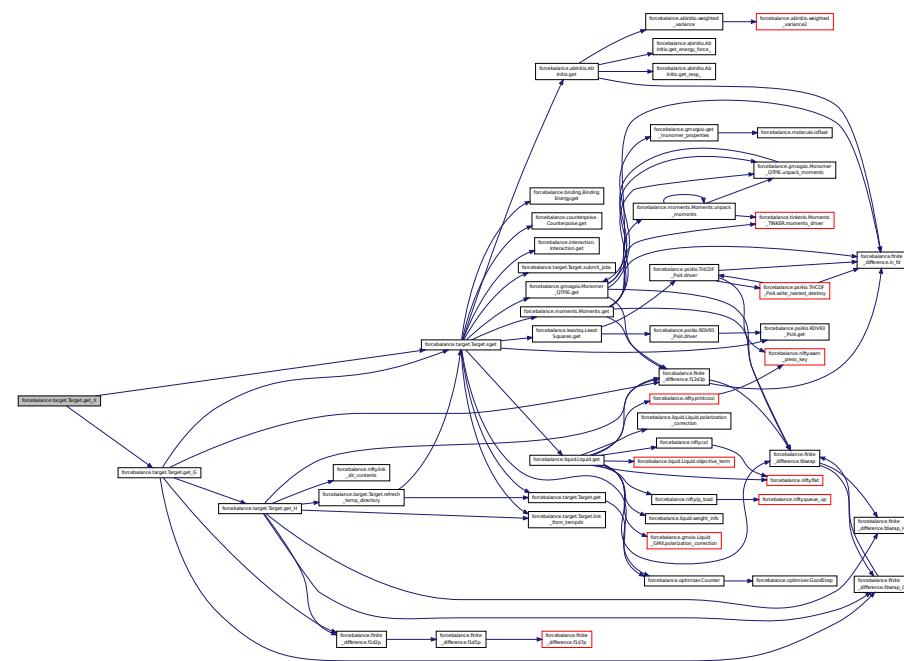
Here is the call graph for this function:



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

Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.target.Target.link\_from\_tempdir ( self, absdestdir )** Definition at line 213 of file target.py.

```
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.

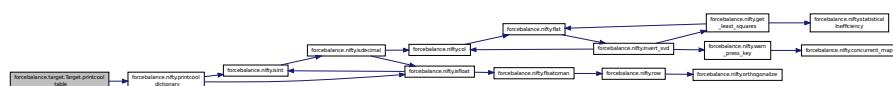
The input should be something like:

## 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 367 of file target.py.

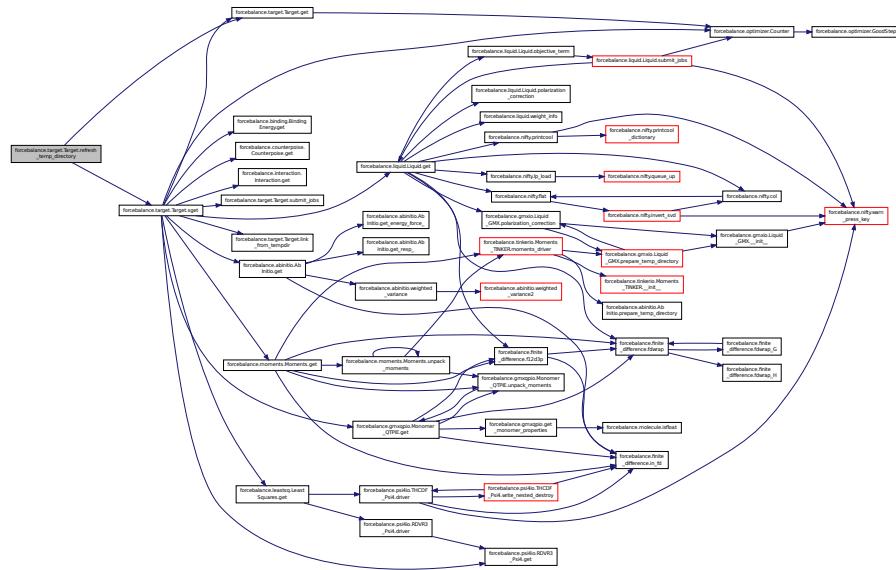
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



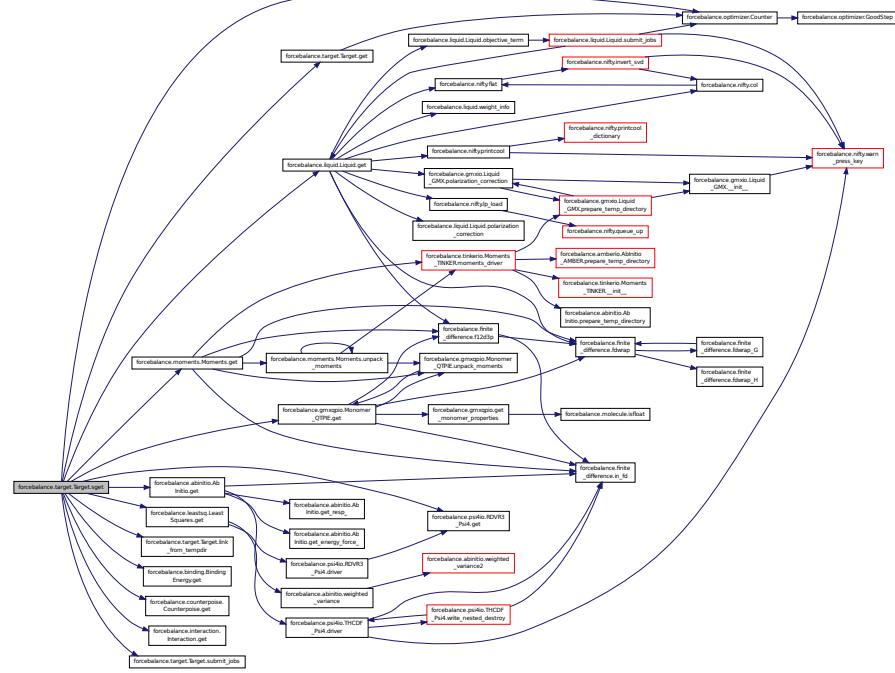
```
def forcebalance.BaseClass.set_option( self, in_dict, src_key, dest_key = None, val = None, default = None, forceprint = False ) [inherited] Definition at line 32 of file __init__.py.
```

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )  
    Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:



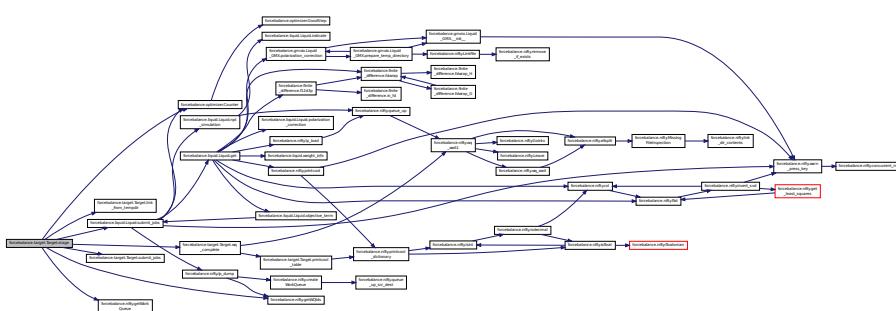
```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )
```

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 301 of file target.py.

Here is the call graph for this function:

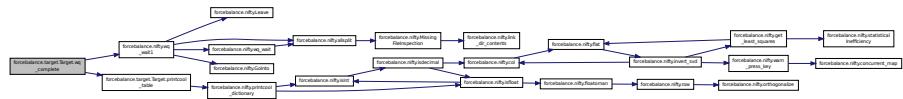


**def forcebalance.target.Target.submit\_jobs ( self, mvals, AGrad = False, AHess = False )** Definition at line 291 of file target.py.

**def forcebalance.target.Target.wq.complete ( self )** This method determines whether the Work Queue tasks for the current target have completed.

Definition at line 331 of file target.py.

Here is the call graph for this function:



#### **8.55.4 Member Data Documentation**

**forcebalance.target.Target.FF** Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.target.Target.gct** Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** Counts how often the Hessian was computed.

Definition at line 145 of file target.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 29 of file `__init__.py`.

**forcebalance.target.Target.rundir** 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.

Definition at line 137 of file target.py.

**forcebalance.target.Target.tempdir** Root directory of the whole project.

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 30 of file \_\_init\_\_.py.

**forcebalance.target.Target.xct** Counts how often the objective function was computed.

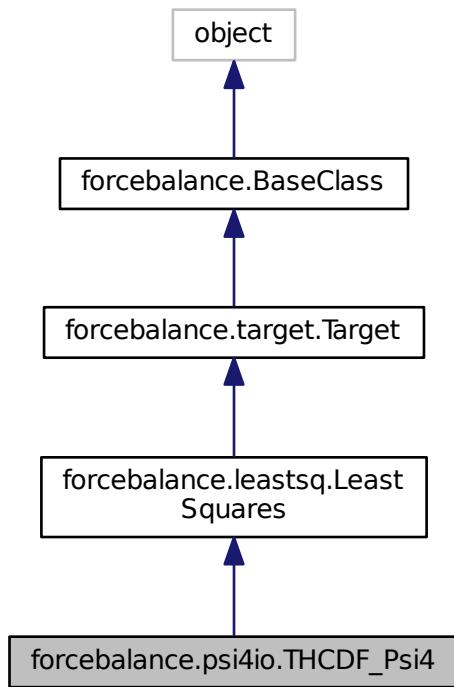
Definition at line 141 of file target.py.

The documentation for this class was generated from the following file:

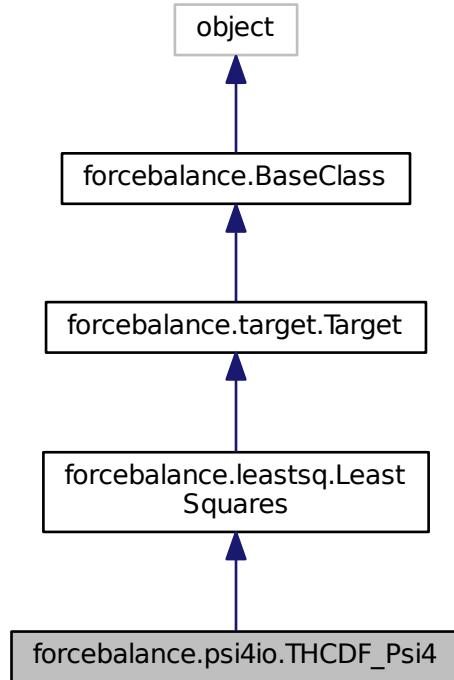
- target nv

## 8.56 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 `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 `sget`  
    *Stages the directory for the target, and then calls 'get'.*

- 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 `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`
- `call_derivatives`  
*Number of snapshots.*
- `MAQ`  
*Dictionary for derivative terms.*
- `D`
- `objective`
- `tempdir`  
*Root directory of the whole project.*
- `rundir`  
*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.*
- `PrintOptionDict`
- `verbose_options`

### 8.56.1 Detailed Description

Definition at line 97 of file psi4io.py.

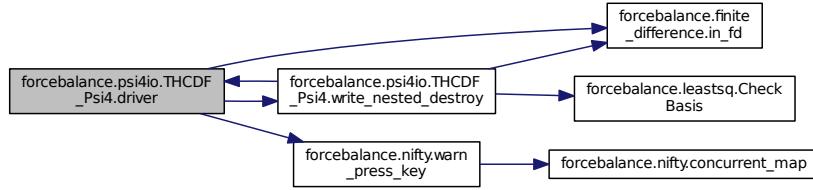
### 8.56.2 Constructor & Destructor Documentation

```
def forcebalance.psi4io.THCDF_Psi4.__init__ ( self, options, tgt_opts, forcefield ) Definition at line 99 of file
psi4io.py.
```

### 8.56.3 Member Function Documentation

**def forcebalance.psi4io.THCDF\_Psi4.driver ( self )** Definition at line 172 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

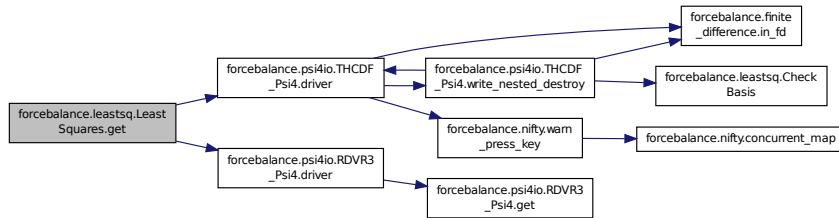
in	mvals	Mathematical parameter values
in	AGrad	Switch to turn on analytic gradient
in	AHess	Switch to turn on analytic Hessian

Returns

Answer Contribution to the objective function

Definition at line 74 of file leastsq.py.

Here is the call graph for this function:

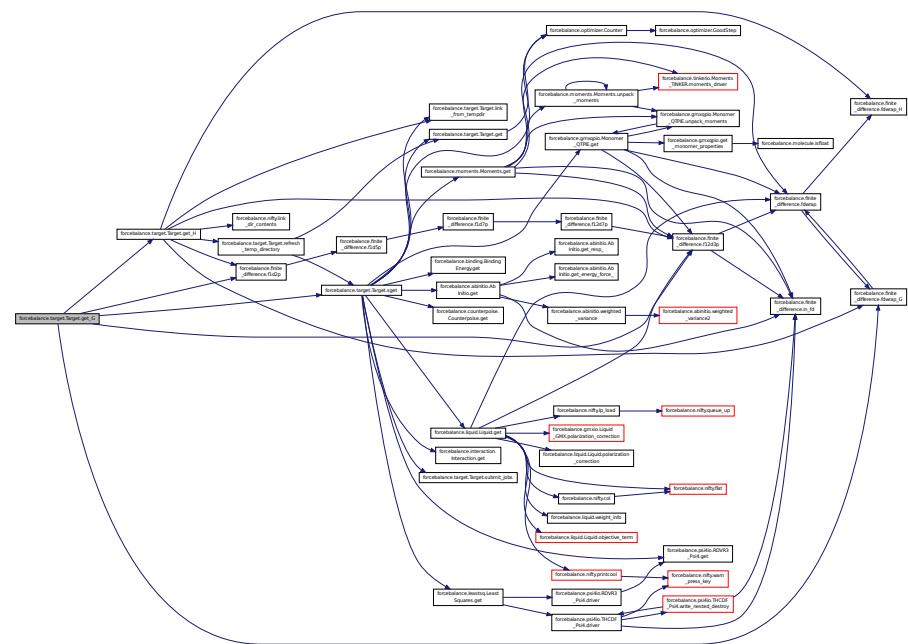


**def forcebalance.target.Target.get\_G ( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



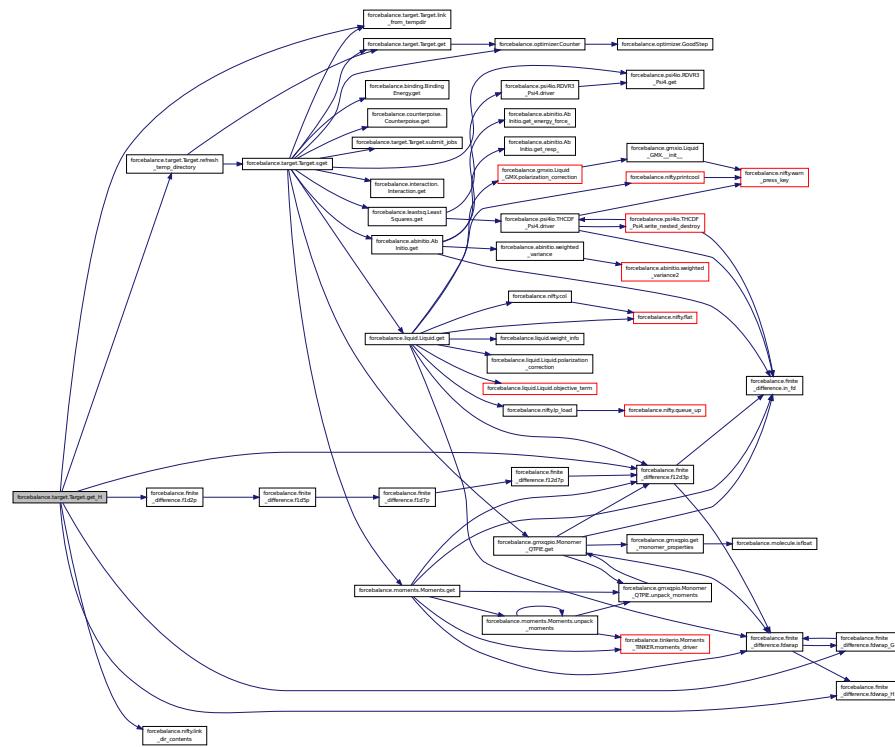
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

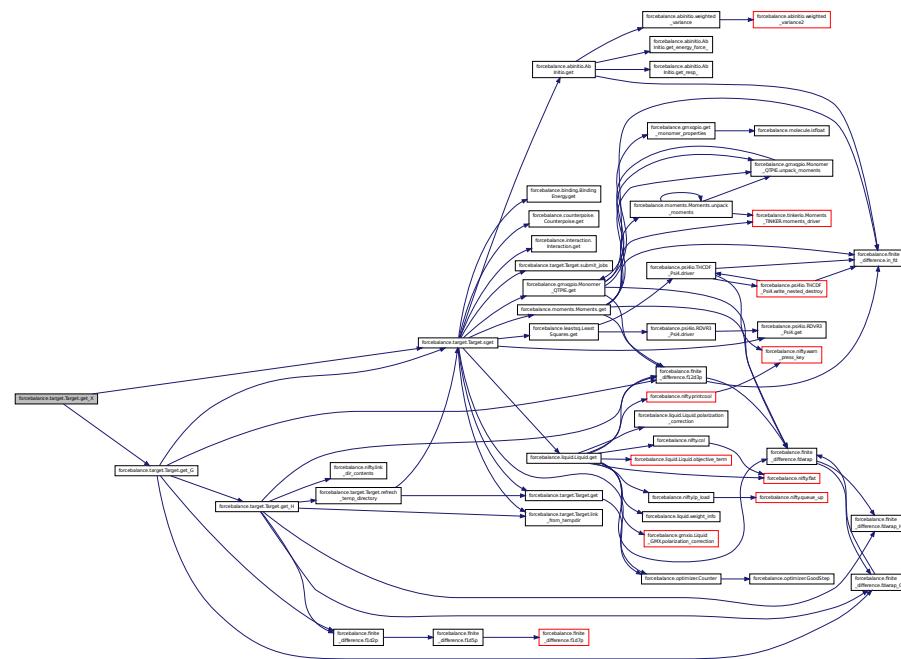
Here is the call graph for this function:



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

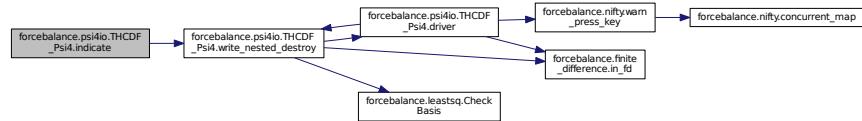
Definition at line 155 of file target.py.

Here is the call graph for this function:



**def forcebalance.psi4io.THCDF\_Psi4.indicate ( self )** Definition at line 152 of file psi4io.py.

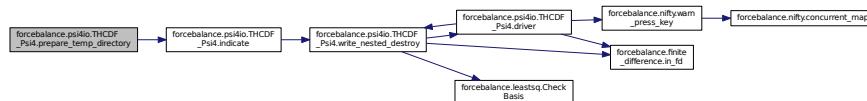
Here is the call graph for this function:



```
def forcebalance.target.Target.link_from_tempdir( self, absdestdir ) [inherited] Definition at line 213 of file target.py.
```

```
def forcebalance.psi4io.THCDF_Psi4.prepare_temp_directory ( self, options, tgt_opts ) Definition at line 141  
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.

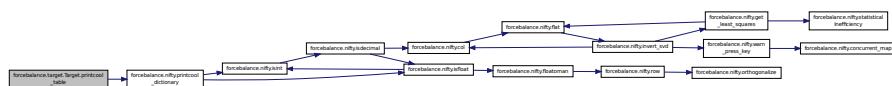
The input should be something like:

## 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 367 of file target.py.

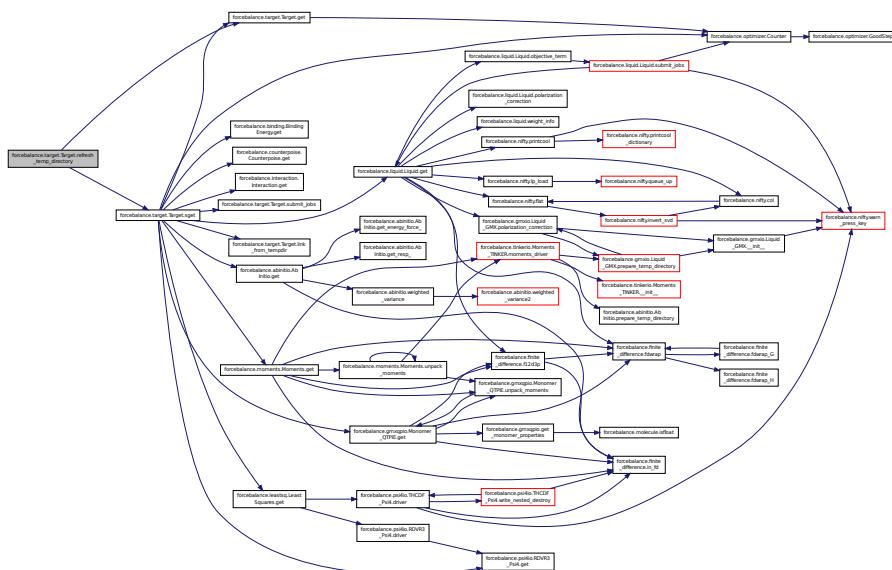
Here is the call graph for this function:



**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 219 of file target.py.

Here is the call graph for this function:



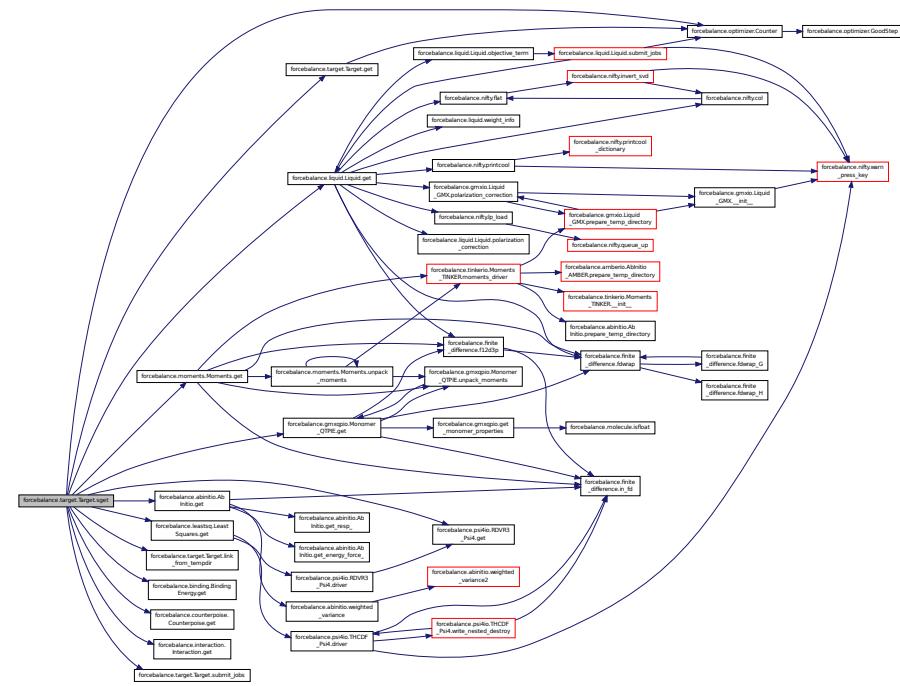
**def forcebalance.BaseClass.set\_option( self, in\_dict, src\_key, dest\_key = None, val = None, default = None, forceprint = False )** [inherited] Definition at line 32 of file `init.py`.

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

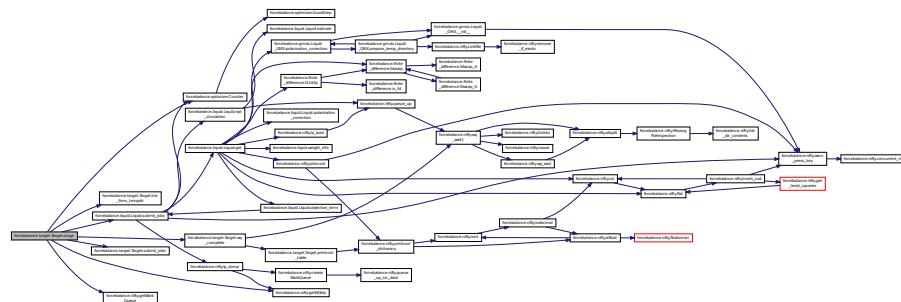


```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

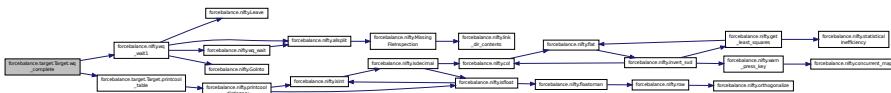


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 291 of file target.py.
```

**def forcebalance.target.Target.wq\_complete( self ) [inherited]** This method determines whether the Work Queue tasks for the current target have completed.

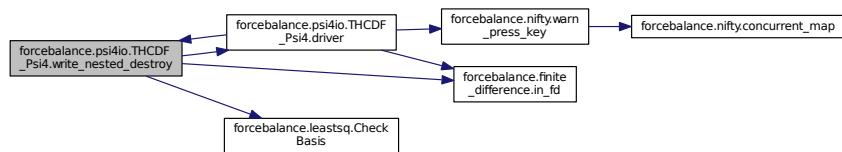
Definition at line 331 of file target.py.

Here is the call graph for this function:



**def forcebalance.psi4io.THCDF\_Psi4.write\_nested\_destroy ( self, fnm, linedestroy )** Definition at line 157 of file psi4io.py.

Here is the call graph for this function:



#### **8.56.4 Member Data Documentation**

**forcebalance.leastsq.LeastSquares.call\_derivatives** [inherited] Number of snapshots.

Which parameters are differentiated?

Definition at line 51 of file leastsq.py

**forcebalance.leastsg.LeastSquares.D** [inherited] Definition at line 137 of file leastsg.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 137 of file psi4io.py.

**forcebalance** psi4io THCDE Psi4 DE Energy Definition at line 238 of file psi4io.py

**forcebalance\_psi4ic THCFD Psi4 Elements** Definition at line 117 of file psi4ic.py

**forcebalance target Target\_EE** [inherited] Need the forcefield (here for now)

Definition at line 129 of file target.h.

for a balanced *Sal1* THORE *Sal1* CRISPR - *Sal1* basic set file.

D:\fisi\W\w\w\w\160\file\141

for each balance target Target<sub>net</sub>,  $\{1, 2, \dots, 11\}$ . Counts how often the quadrant was computed.

Profile was collected 100% full.

**forcebalance.target.Target.hct [inherited]** Counts how often the Hessian was computed.  
Definition at line 145 of file target.py.

**forcebalance.leastsq.LeastSquares.MAQ [inherited]** Dictionary for derivative terms.  
Definition at line 100 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 236 of file psi4io.py.

**forcebalance.leastsq.LeastSquares.objective [inherited]** Definition at line 138 of file leastsq.py.

**forcebalance.BaseClass.PrintOptionDict [inherited]** Definition at line 29 of file \_\_init\_\_.py.

**forcebalance.target.Target.rundir [inherited]** 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.

Definition at line 137 of file target.py.

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

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 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 135 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 30 of file \_\_init\_\_.py.

**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.

Definition at line 141 of file target.py.

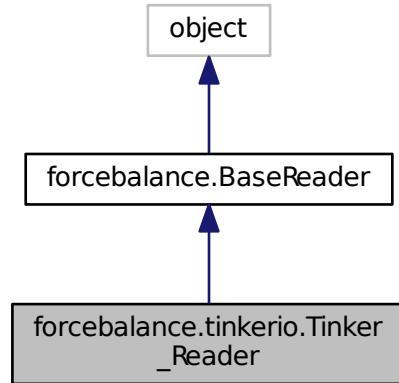
The documentation for this class was generated from the following file:

- [psi4io.py](#)

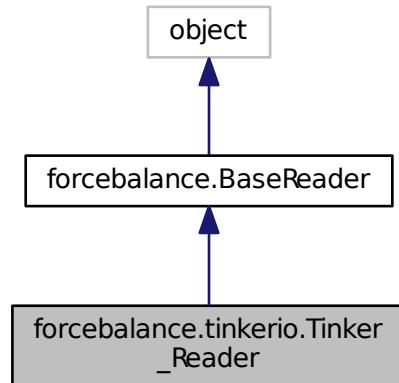
## 8.57 forcebalance.tinkerio.Tinker\_Reader Class Reference

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](#)
- [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](#)

### 8.57.1 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 66 of file `tinkerio.py`.

### 8.57.2 Constructor & Destructor Documentation

`def forcebalance.tinkerio.Tinker_Reader.__init__ ( self, fnm )` Definition at line 68 of file `tinkerio.py`.

### 8.57.3 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 107 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 # PARM 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 '#PARM 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 109 of file tinkerio.py.

**def forcebalance.BaseReader.Split( self, line ) [inherited]** Definition at line 82 of file \_\_init\_\_.py.

Here is the call graph for this function:



**def forcebalance.BaseReader.Whites( self, line ) [inherited]** Definition at line 85 of file \_\_init\_\_.py.

Here is the call graph for this function:



#### 8.57.4 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 72 of file \_\_init\_\_.py.

**forcebalance.tinkerio.Tinker\_Reader.atom** The atom numbers in the interaction (stored in the TINKER parser)

Definition at line 73 of file tinkerio.py.

**forcebalance.BaseReader.AtomTypes [inherited]** Definition at line 80 of file \_\_init\_\_.py.

**forcebalance.tinkerio.Tinker\_Reader.itype** Definition at line 117 of file tinkerio.py.

**forcebalance.BaseReader.In [inherited]** Definition at line 67 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 77 of file \_\_init\_\_.py.

**forcebalance.BaseReader.Molecules** [inherited] Definition at line 79 of file \_\_init\_\_.py.

**forcebalance.tinkerio.Tinker\_Reader.pdict** The parameter dictionary (defined in this file)  
Definition at line 71 of file tinkerio.py.

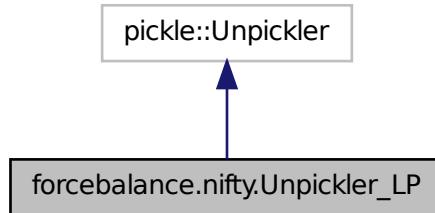
**forcebalance.tinkerio.Tinker\_Reader.suffix** Definition at line 139 of file tinkerio.py.  
The documentation for this class was generated from the following file:

- [tinkerio.py](#)

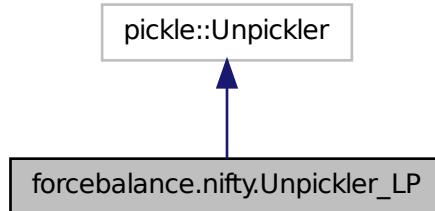
## 8.58 forcebalance.nifty.Unpickler\_LP Class Reference

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\\_\\_](#)

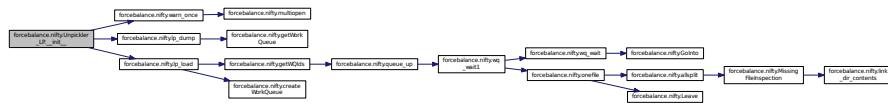
### 8.58.1 Detailed Description

A subclass of the python Unpickler that implements unpickling of \_ElementTree types.  
Definition at line 524 of file nifty.py.

### 8.58.2 Constructor & Destructor Documentation

**def forcebalance.nifty.Unpickler\_LP.\_\_init\_\_( self, file )** Definition at line 525 of file nifty.py.

Here is the call graph for this function:



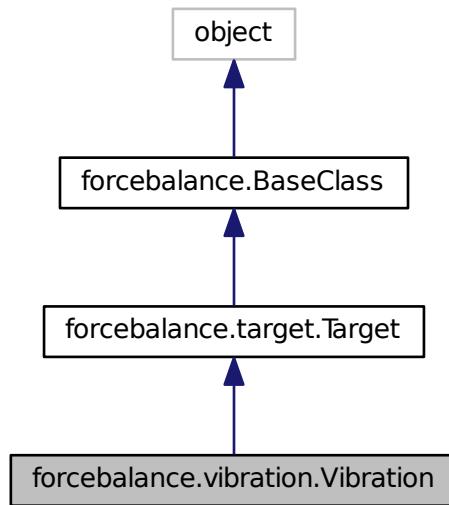
The documentation for this class was generated from the following file:

- [nifty.py](#)

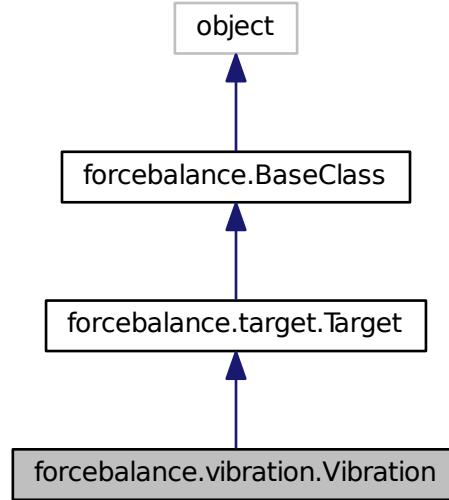
## 8.59 forcebalance.vibration.Vibration Class Reference

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 `prepare_temp_directory`  
*Prepare the temporary directory, by default does nothing.*
- def `indicate`  
*Print qualitative indicator.*
- def `get`  
*Evaluate objective function.*
- def `get_X`  
*Computes the objective function contribution without any parametric derivatives.*
- 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 `sget`  
*Stages the directory for the target, and then calls 'get'.*
- 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 `set_option`

## Public Attributes

- `vfnm`  
*The vdata.txt file that contains the vibrations.*
- `na`  
*Number of atoms.*
- `ref_eigvals`
- `ref_eigvecs`
- `calc_eigvals`
- `objective`
- `tempdir`  
*Root directory of the whole project.*
- `rundir`  
*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.*
- `PrintOptionDict`
- `verbose_options`

### 8.59.1 Detailed Description

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

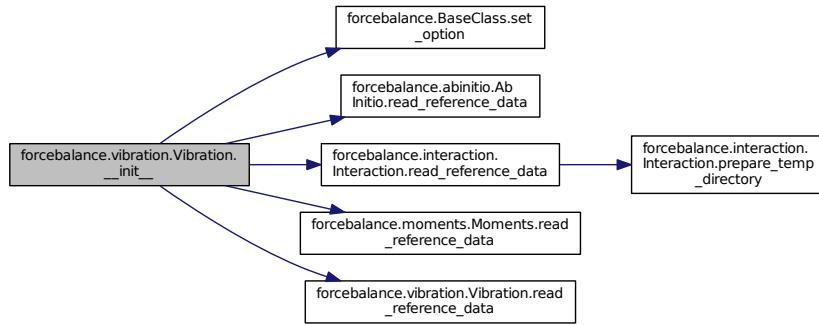
Currently Tinker is supported.

Definition at line 30 of file vibration.py.

### 8.59.2 Constructor & Destructor Documentation

`def forcebalance.vibration.Vibration.__init__( self, options, tgt_opts, forcefield )` Initialization.  
 Definition at line 35 of file vibration.py.

Here is the call graph for this function:

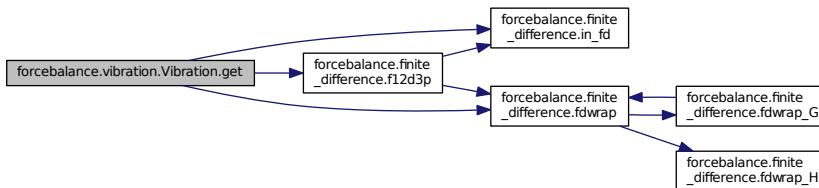


### 8.59.3 Member Function Documentation

**def forcebalance.vibration.Vibration.get( self, mvals, AGrad = False, AHess = False )** Evaluate objective function.

Definition at line 111 of file vibration.py.

Here is the call graph for this function:

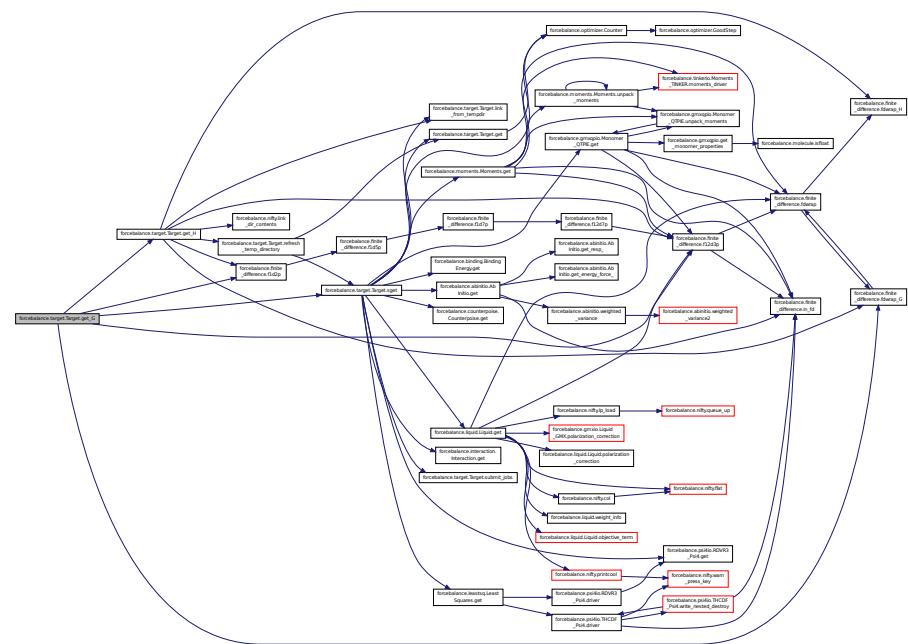


**def forcebalance.target.Target.get\_G( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient.

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.

Definition at line 172 of file target.py.

Here is the call graph for this function:



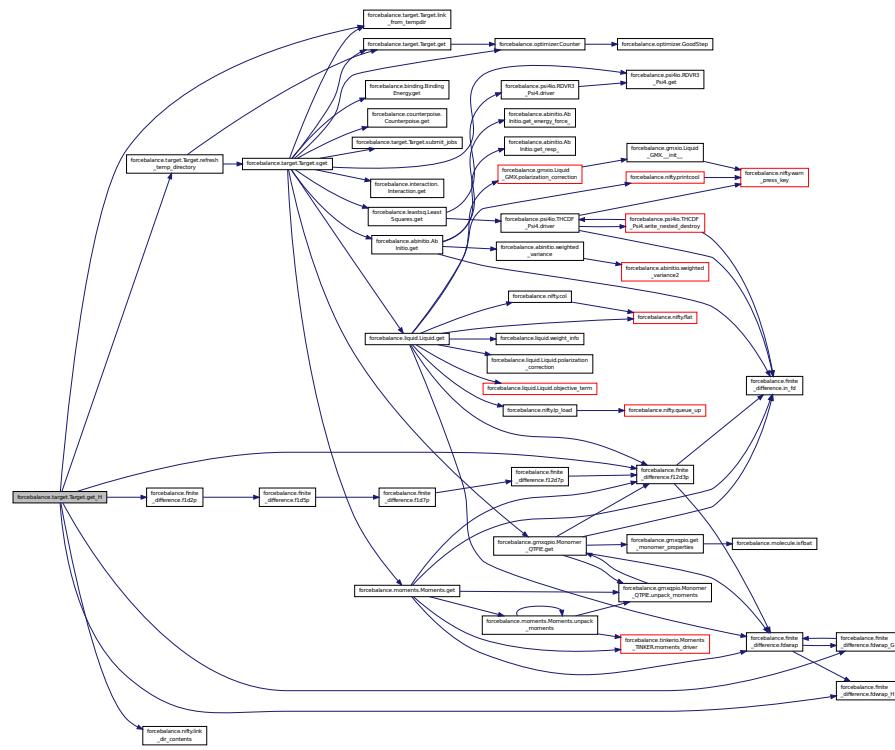
**def forcebalance.target.Target.get\_H( self, mvals = None ) [inherited]** Computes the objective function contribution and its gradient / Hessian.

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 195 of file target.py.

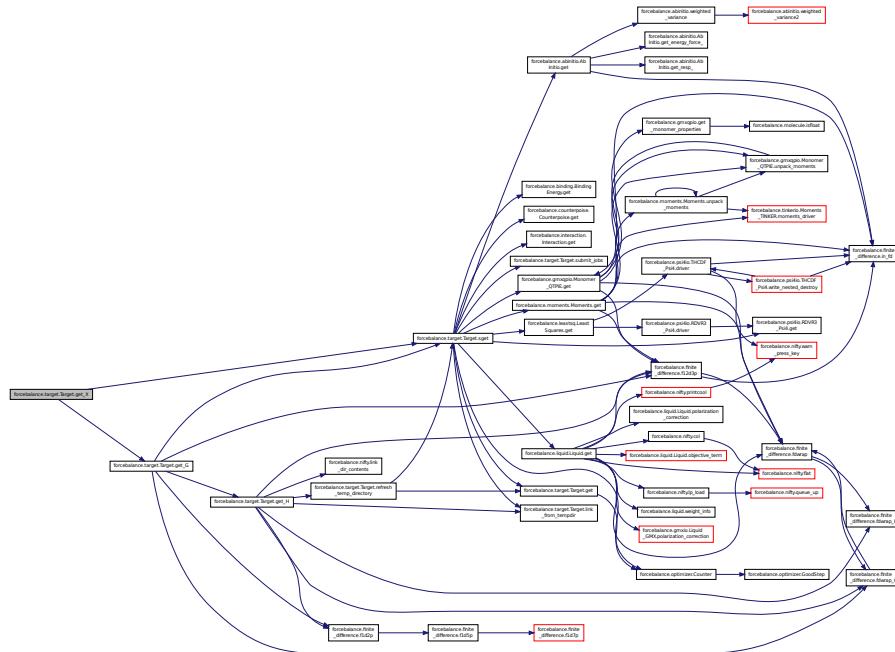
Here is the call graph for this function:



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

Definition at line 155 of file target.py.

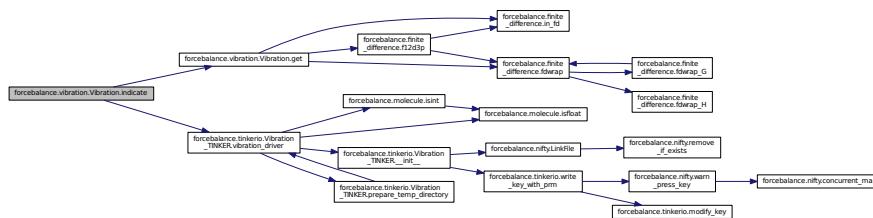
Here is the call graph for this function:



**def forcebalance.vibration.Vibration.indicate ( self )** Print qualitative indicator.

Definition at line 102 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 213 of file target.py.

**def forcebalance.vibration.Vibration.prepare\_temp\_directory( self, options, tgt\_opts )** Prepare the temporary directory, by default does nothing.

Definition at line 97 of file vibration.py.

```
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.

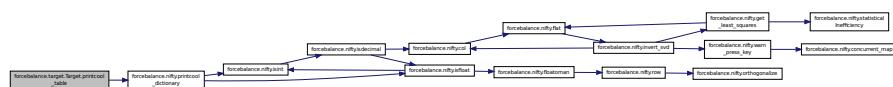
The input should be something like:

## 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 367 of file target.py.

Here is the call graph for this function:



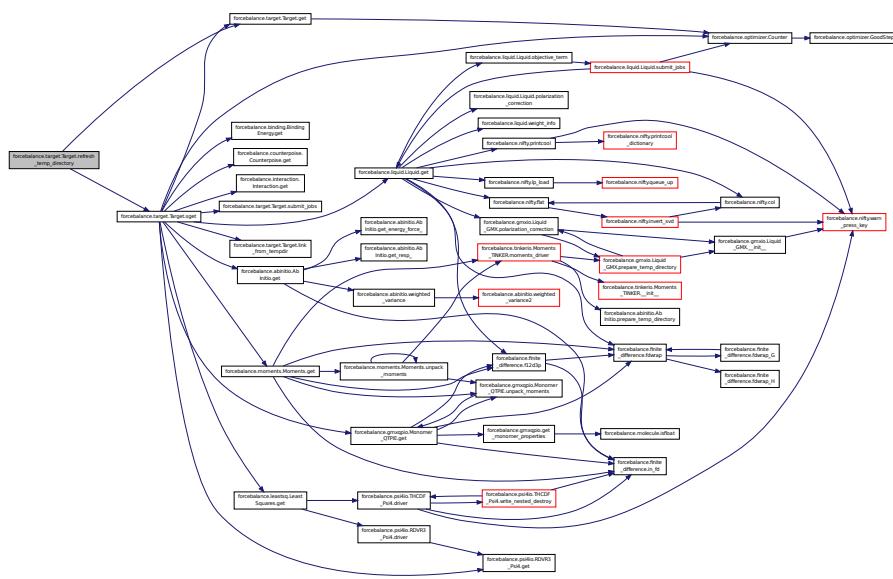
**def forcebalance.vibration.Vibration.read\_reference\_data( self )** Read the reference vibrational data from a file.

Definition at line 57 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 219 of file target.py.

Here is the call graph for this function:



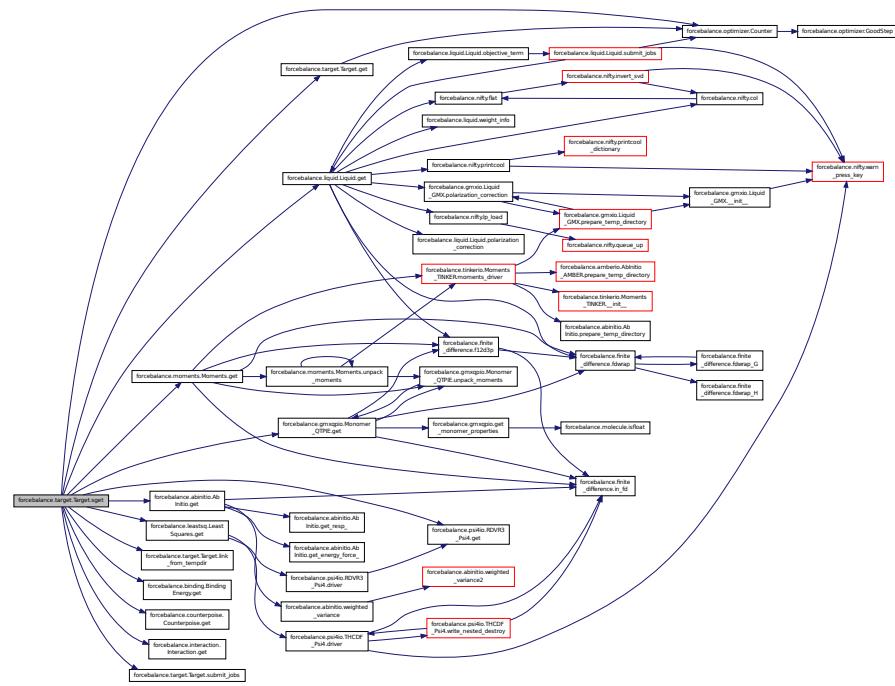
**def forcebalance.BaseClass.set\_option( self, in\_dict, src\_key, dest\_key = None, val = None, default = None, forceprint = False ) [inherited]**  
Definition at line 32 of file \_\_init\_\_.py.

```
def forcebalance.target.Target.sget ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then calls 'get'.
```

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

Definition at line 266 of file target.py.

Here is the call graph for this function:

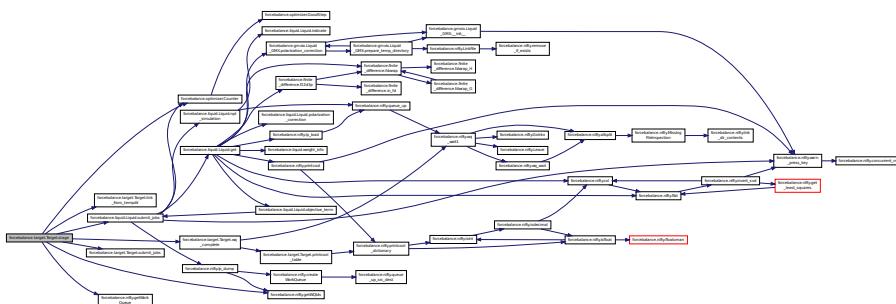


```
def forcebalance.target.Target.stage ( self, mvals, AGrad = False, AHess = False, customdir = None )
[inherited] Stages the directory for the target, and then launches Work Queue processes if any.
```

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

Definition at line 301 of file target.py.

Here is the call graph for this function:

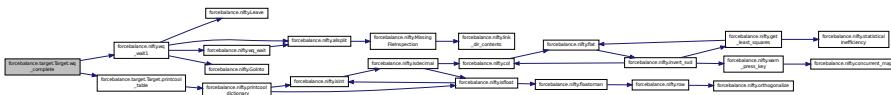


```
def forcebalance.target.Target.submit_jobs ( self, mvals, AGrad = False, AHess = False )
[inherited] Definition at line 291 of file target.py.
```

**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 331 of file target.py.

Here is the call graph for this function:



#### **8.59.4 Member Data Documentation**

**forcebalance.vibration.Vibration.calc\_eigvals** Definition at line 140 of file vibration.py.

**forcebalance.target.Target.FF** [inherited] Need the forcefield (here for now)

Definition at line 139 of file target.py.

**forcebalance.target.Target.gct** [inherited] Counts how often the gradient was computed.

Definition at line 143 of file target.py.

**forcebalance.target.Target.hct** [inherited] Counts how often the Hessian was computed.

Definition at line 145 of file target.py.

**forcebalance.vibration.Vibration.na** Number of atoms.

Definition at line 59 of file vibration.py.

**forcebalance.vibration.Vibration.objective** Definition at line 141 of file vibration.py.

**forcebalance.BaseClass.PrintOptionDict** [inherited] Definition at line 29 of file `__init__.py`.

**forcebalance.vibration.Vibration.ref\_eigvals** Definition at line 60 of file vibration.py.

**forcebalance.vibration.Vibration.ref\_eigvecs** Definition at line 61 of file vibration.py.

**forcebalance.target.Target.rundir** [inherited] 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.

Definition at line 137 of file target.py.

**forcebalance.target.Target.tempdir** [inherited] Boot directory of the whole project.

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 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 135 of file target.py.

**forcebalance.BaseClass.verbose\_options** [inherited] Definition at line 30 of file \_init\_.py.

**forcebalance.vibration.Vibration.vfnm** The vdata.txt file that contains the vibrations.  
Definition at line 49 of file vibration.py.

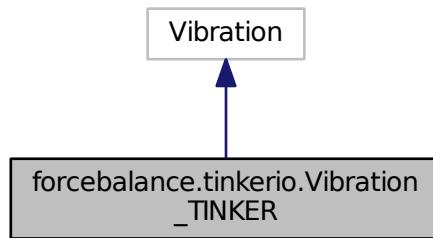
**forcebalance.target.Target.xct [inherited]** Counts how often the objective function was computed.  
Definition at line 141 of file target.py.  
The documentation for this class was generated from the following file:

- [vibration.py](#)

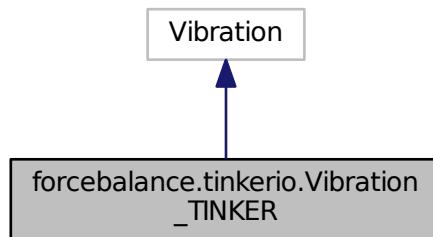
## 8.60 forcebalance.tinkerio.Vibration\_TINKER Class Reference

Subclass of Target for 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 [prepare\\_temp\\_directory](#)
- def [vibration\\_driver](#)

### 8.60.1 Detailed Description

Subclass of Target for vibrational frequency matching using TINKER.

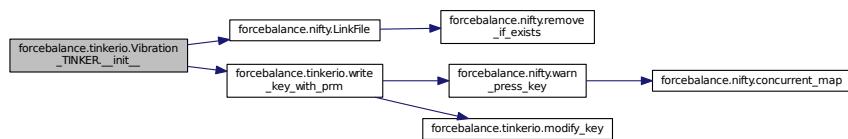
Provides optimized geometry, vibrational frequencies (in cm-1), and eigenvectors.

Definition at line 353 of file tinkerio.py.

### 8.60.2 Constructor & Destructor Documentation

**def forcebalance.tinkerio.Vibration.TINKER.\_\_init\_\_ ( self, options, tgt\_opts, forcefield )** Definition at line 356 of file tinkerio.py.

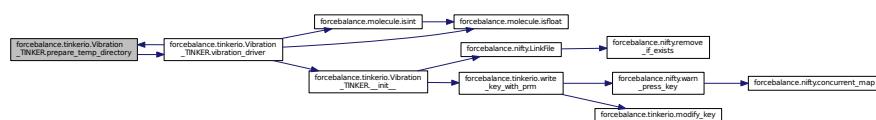
Here is the call graph for this function:



### 8.60.3 Member Function Documentation

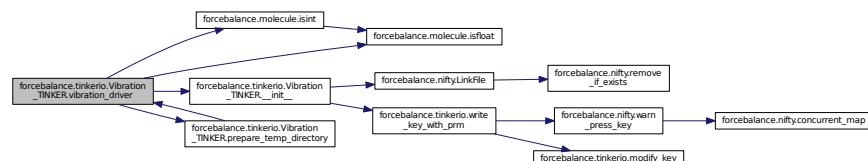
**def forcebalance.tinkerio.Vibration.TINKER.prepare\_temp\_directory ( self, options, tgt\_opts )** Definition at line 361 of file tinkerio.py.

Here is the call graph for this function:



**def forcebalance.tinkerio.Vibration.TINKER.vibration\_driver ( self )** Definition at line 371 of file tinkerio.py.

Here is the call graph for this function:



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.AbInitio`  
*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.AbInitio_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.ABInitio_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.*
- dictionary `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.in_fd`  
*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.AbInitio_GMX`

*Subclass of AbInitio for force and energy matching using normal GROMACS.*

- class `forcebalance.gmxio.Liquid_GMX`
- class `forcebalance.gmxio.Interaction_GMX`

*Subclass of Interaction for interaction energy matching using GROMACS.*

## Namespaces

- `forcebalance.gmxio`

*GROMACS input/output.*

## Functions

- def `forcebalance.gmxio.edit_mdp`  
*Create or edit a Gromacs MDP 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.*
- string `forcebalance.gmxio.shot_mdp`

## 9.15 gmxqpio.py File Reference

### Classes

- class `forcebalance.gmxqpio.Monomer_QTPIE`

*Subclass of Target for monomer properties of QTPIE (implemented within gromacs WCV branch).*

### Namespaces

- `forcebalance.gmxqpio`
- `forcebalance.gmxio`

*GROMACS input/output.*

## Functions

- def `forcebalance.gmxqpio.get_monomer_properties`

## Variables

- tuple `forcebalance.gmxqpio.logger` = getLogger(\_\_name\_\_)

## 9.16 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\_\_)

## 9.17 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

## 9.18 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.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.isdigit`  
*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.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', 'tinkersuf', 'resid', 'resname', 'qcsuf', 'qm\_ghost', 'chain', 'altloc', 'icode'])
- tuple `forcebalance.molecule.MetaVariableNames` = set(['fnm', 'ftype', 'qcrems', 'qctemplate', '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
- `forcebalance.molecule.Alive`

## 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.pmat2d`  
*Printout of a 2-D matrix.*
- def `forcebalance.nifty.encode`
- def `forcebalance.nifty.segments`
- def `forcebalance.nifty.commashash`
- def `forcebalance.nifty.uncommashash`
- def `forcebalance.nifty.printcool`  
*Cool-looking printout for slick formatting of output.*
- def `forcebalance.nifty.printcool_dictionary`  
*See documentation for printcool; this is a nice way to print out keys/values in a dictionary.*
- def `forcebalance.nifty.isint`  
*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.floatornan`  
*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.orthogonalize`  
*Given two vectors vec1 and vec2, project out the component of vec1 that is along the vec2-direction.*
- def `forcebalance.nifty.invert_svd`  
*Invert a matrix using singular value decomposition.*
- def `forcebalance.nifty.get_least_squares`
- def `forcebalance.nifty.statisticalinefficiency`  
*Compute the (cross) statistical inefficiency of (two) timeseries.*

- def `forcebalance.nifty.lp_dump`  
*Use this instead of pickle.dump for pickling anything that contains \_ElementTree types.*
- def `forcebalance.nifty.lp_load`  
*Use this instead of pickle.load for unpickling anything that contains \_ElementTree types.*
- def `forcebalance.nifty.getWorkQueue`
- def `forcebalance.nifty.getWQIds`
- def `forcebalance.nifty.createWorkQueue`
- def `forcebalance.nifty.destroyWorkQueue`
- def `forcebalance.nifty.queue_up`  
*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.onefile`
- def `forcebalance.nifty.GoInto`
- def `forcebalance.nifty.allsplit`
- def `forcebalance.nifty.Leave`
- def `forcebalance.nifty.MissingFileInspection`
- 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().*
- def `forcebalance.nifty.multiopen`  
*This function be given any of several variable types (single file name, file object, or list of lines, or a list of the above) and give a list of files:*

## 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.objectiveImplemented_TTargets`  
*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.Liquid_OpenMM`
- class `forcebalance.openmmio.AbInitio_OpenMM`  
*Subclass of AbInitio for force and energy matching using OpenMM.*
- class `forcebalance.openmmio.Interaction_OpenMM`  
*Subclass of Target for interaction matching using OpenMM.*

### Namespaces

- `forcebalance.openmmio`  
*OpenMM input/output.*

### Functions

- def `forcebalance.openmmio.get_dipole`  
*Return the current dipole moment in Debye.*
- 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.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.GoodStep`

### Variables

- tuple `forcebalance.optimizer.logger` = getLogger(`__name__`)
- int `forcebalance.optimizer.ITERATION_NUMBER` = 0
- int `forcebalance.optimizer.GOODSTEP` = 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.*

### 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` = {"simtype" : "type", "masterfile" : "binding\_txt"}  
*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 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\_\_)

## 9.33 tinkerio.py File Reference

### Classes

- class `forcebalance.tinkerio.Tinker_Reader`  
*Finite state machine for parsing TINKER force field files.*
- class `forcebalance.tinkerio.Liquid_TINKER`
- class `forcebalance.tinkerio.AblInitio_TINKER`  
*Subclass of Target for force and energy matching using TINKER.*
- class `forcebalance.tinkerio.Vibration_TINKER`  
*Subclass of Target for vibrational frequency matching using TINKER.*
- class `forcebalance.tinkerio.Moments_TINKER`  
*Subclass of Target for multipole moment matching using TINKER.*
- class `forcebalance.tinkerio.BindingEnergy_TINKER`  
*Subclass of BindingEnergy for binding energy matching using TINKER.*
- class `forcebalance.tinkerio.Interaction_TINKER`  
*Subclass of Target for interaction matching using TINKER.*

### Namespaces

- `forcebalance.tinkerio`  
*TINKER input/output.*

## Functions

- def `forcebalance.tinkerio.write_key_with_prm`  
*Copies a TINKER .key file but changes the parameter keyword as necessary to reflect the ForceBalance settings.*
- def `forcebalance.tinkerio.modify_key`  
*Performs in-place modification of a TINKER .key file.*

## Variables

- tuple `forcebalance.tinkerio.logger` = getLogger(\_\_name\_\_)
- dictionary `forcebalance.tinkerio.pdict`

## 9.34 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.*

### Variables

- tuple `forcebalance.vibration.logger` = getLogger(\_\_name\_\_)

## Index

\_\_add\_\_  
    forcebalance::molecule::Molecule, 395  
\_\_delitem\_\_  
    forcebalance::molecule::Molecule, 395  
\_\_enter\_\_  
    forcebalance::nifty::LineChunker, 322  
\_\_eq\_\_  
    forcebalance::forcefield::FF, 230  
\_\_exit\_\_  
    forcebalance::nifty::LineChunker, 322  
\_\_getattr\_\_  
    forcebalance::molecule::Molecule, 396  
\_\_getitem\_\_  
    forcebalance::molecule::Molecule, 396  
\_\_iadd\_\_  
    forcebalance::molecule::Molecule, 396  
\_\_init\_\_  
    forcebalance::abinitio::AbInitio, 81  
    forcebalance::abinitio\_internal::AbInitio\_Internal, 136  
    forcebalance::amber::AbInitio\_AMBER, 99  
    forcebalance::amber::FrcMod\_Reader, 242  
    forcebalance::amber::Mol2\_Reader, 386  
    forcebalance::BaseClass, 189  
    forcebalance::BaseReader, 191  
    forcebalance::binding::BindingEnergy, 195  
    forcebalance::counterpoise::Counterpoise, 216  
    forcebalance::custom\_io::Gen\_Reader, 249  
    forcebalance::engine::Engine, 226  
    forcebalance::forcefield::BackedUpDict, 187  
    forcebalance::forcefield::FF, 230  
    forcebalance::gmxio::AbInitio\_GMX, 117  
    forcebalance::gmxio::GMX, 253  
    forcebalance::gmxio::Interaction\_GMX, 274  
    forcebalance::gmxio::ITP\_Reader, 309  
    forcebalance::gmxio::Liquid\_GMX, 337  
    forcebalance::gmxqpio::Monomer\_QTPIE, 439  
    forcebalance::interaction::Interaction, 262  
    forcebalance::leastsq::LeastSquares, 313  
    forcebalance::liquid::Liquid, 325  
    forcebalance::Mol2::mol2, 373  
    forcebalance::Mol2::mol2\_atom, 378  
    forcebalance::Mol2::mol2\_bond, 382  
    forcebalance::Mol2::mol2\_set, 391  
    forcebalance::mol2io::Mol2\_Reader, 389  
    forcebalance::molecule::Molecule, 395  
    forcebalance::moments::Moments, 416  
    forcebalance::nifty::LineChunker, 322  
    forcebalance::nifty::Pickler\_LP, 468  
    forcebalance::nifty::Unpickler\_LP, 518  
    forcebalance::objective::Objective, 449  
    forcebalance::objective::Penalty, 465  
    forcebalance::openmmio::AbInitio\_OpenMM, 153  
    forcebalance::openmmio::Interaction\_OpenMM, 286  
    forcebalance::openmmio::Liquid\_OpenMM, 349  
    forcebalance::openmmio::OpenMM\_Reader, 452  
    forcebalance::optimizer::Optimizer, 457  
    forcebalance::output::CleanStreamHandler, 213  
    forcebalance::output::ForceBalanceLogger, 240  
    forcebalance::output::RawStreamHandler, 473  
    forcebalance::psi4io::GBS\_Reader, 245  
    forcebalance::psi4io::Grid\_Reader, 257  
    forcebalance::psi4io::RDVR3\_Psi4, 476  
    forcebalance::psi4io::THCDF\_Psi4, 505  
    forcebalance::qchemio::QCIn\_Reader, 470  
    forcebalance::target::RemoteTarget, 486  
    forcebalance::target::Target, 495  
    forcebalance::tinkerio::AbInitio\_TINKER, 172  
    forcebalance::tinkerio::BindingEnergy\_TINKER, 204  
    forcebalance::tinkerio::Interaction\_TINKER, 298  
    forcebalance::tinkerio::Liquid\_TINKER, 363  
    forcebalance::tinkerio::Moments\_TINKER, 428  
    forcebalance::tinkerio::Tinker\_Reader, 515  
    forcebalance::tinkerio::Vibration\_TINKER, 529  
    forcebalance::vibration::Vibration, 520  
\_\_init\_\_.py, 530  
\_\_iter\_\_  
    forcebalance::molecule::Molecule, 397  
\_\_len\_\_  
    forcebalance::molecule::Molecule, 397  
\_\_missing\_\_  
    forcebalance::forcefield::BackedUpDict, 187  
\_\_repr\_\_  
    forcebalance::Mol2::mol2, 373  
    forcebalance::Mol2::mol2\_atom, 379  
    forcebalance::Mol2::mol2\_bond, 382  
\_\_setattr\_\_  
    forcebalance::molecule::Molecule, 397  
\_\_version\_\_  
    forcebalance, 13

A  
    forcebalance::chemistry, 17

a  
    forcebalance::objective::Penalty, 466

abinitio.py, 530

abinitio\_internal.py, 530

add\_quantum  
    forcebalance::molecule::Molecule, 397

add\_strip\_to\_mat  
    forcebalance::molecule, 36

add\_virtual\_site  
    forcebalance::molecule::Molecule, 398

```

addHandler
    forcebalance::output::ForceBalanceLogger, 240
addff
    forcebalance::forcefield::FF, 230
addff_txt
    forcebalance::forcefield::FF, 231
addff_xml
    forcebalance::forcefield::FF, 232
adict
    forcebalance::amberio::FrcMod_Reader, 243
    forcebalance::amberio::Mol2_Reader, 387
    forcebalance::BaseReader, 192
    forcebalance::custom_io::Gen_Reader, 250
    forcebalance::gmxio::ITP_Reader, 310
    forcebalance::mol2io::Mol2_Reader, 390
    forcebalance::openmmio::OpenMM_Reader, 453
    forcebalance::psi4io::GBS_Reader, 246
    forcebalance::psi4io::Grid_Reader, 258
    forcebalance::qchemio::QCIn_Reader, 471
    forcebalance::tinkerio::Tinker_Reader, 516
aftypes
    forcebalance::gmxio, 30
align
    forcebalance::molecule::Molecule, 398
align_by_moments
    forcebalance::molecule::Molecule, 398
align_center
    forcebalance::molecule::Molecule, 398
AlignToDensity
    forcebalance::molecule, 36
AlignToMoments
    forcebalance::molecule, 36
Alive
    forcebalance::molecule, 42
all_at_once
    forcebalance::amberio::AbInitio_AMBER, 109
    forcebalance::tinkerio::AbInitio_TINKER, 183
all_opts_names
    forcebalance::parser, 71
all_pairwise_rmsd
    forcebalance::molecule::Molecule, 398
AllVariableNames
    forcebalance::molecule, 42
allsplit
    forcebalance::nifty, 47
amberio.py, 531
amom
    forcebalance::psi4io::GBS_Reader, 246
Anneal
    forcebalance::optimizer::Optimizer, 457
api.dox, 531
append
    forcebalance::molecule::Molecule, 398
assign_field
    forcebalance::forcefield::FF, 232
assign_p0
    forcebalance::forcefield::FF, 233
atom
    forcebalance::amberio::FrcMod_Reader, 243
    forcebalance::amberio::Mol2_Reader, 387
    forcebalance::mol2io::Mol2_Reader, 390
    forcebalance::qchemio::QCIn_Reader, 471
    forcebalance::tinkerio::Tinker_Reader, 516
atom_distances
    forcebalance::contact, 19
atom_id
    forcebalance::Mol2::mol2_atom, 381
atom_name
    forcebalance::Mol2::mol2_atom, 381
atom_select
    forcebalance::molecule::Molecule, 399
atom_stack
    forcebalance::molecule::Molecule, 399
atom_type
    forcebalance::Mol2::mol2_atom, 381
AtomLists
    forcebalance::abinitio::AbInitio, 91
    forcebalance::abinitio_internal::AbInitio_Internal, 145
    forcebalance::amberio::AbInitio_AMBER, 109
    forcebalance::gmxio::AbInitio_GMX, 128
    forcebalance::gmxio::GMX, 255
    forcebalance::openmmio::AbInitio_OpenMM, 164
    forcebalance::tinkerio::AbInitio_TINKER, 183
AtomMask
    forcebalance::gmxio::AbInitio_GMX, 128
    forcebalance::gmxio::GMX, 255
AtomTypes
    forcebalance::amberio::FrcMod_Reader, 243
    forcebalance::amberio::Mol2_Reader, 387
    forcebalance::BaseReader, 192
    forcebalance::custom_io::Gen_Reader, 250
    forcebalance::gmxio::ITP_Reader, 310
    forcebalance::mol2io::Mol2_Reader, 390
    forcebalance::openmmio::OpenMM_Reader, 453
    forcebalance::psi4io::GBS_Reader, 246
    forcebalance::psi4io::Grid_Reader, 258
    forcebalance::qchemio::QCIn_Reader, 471
    forcebalance::tinkerio::Tinker_Reader, 516
AtomVariableNames
    forcebalance::molecule, 42
atomnames
    forcebalance::amberio::Mol2_Reader, 387
    forcebalance::forcefield::FF, 237
    forcebalance::gmxio::ITP_Reader, 310
atoms
    forcebalance::chemistry, 17
    forcebalance::Mol2::mol2, 377
atomtype_to_mass

```

forcebalance::gmxio::ITP\_Reader, 310  
 atomtypes  
     forcebalance::gmxio::ITP\_Reader, 310

B  
     forcebalance::chemistry, 17

b  
     forcebalance::objective::Penalty, 466

BE  
     forcebalance::chemistry, 17

BFGS  
     forcebalance::optimizer::Optimizer, 457

backup\_dict  
     forcebalance::forcefield::BackedUpDict, 188

basis\_number  
     forcebalance::psi4io::GBS\_Reader, 247

bftypes  
     forcebalance::gmxio, 30

bhyp  
     forcebalance::optimizer::Optimizer, 462

bidirect  
     forcebalance::psi4io::RDVR3\_Psi4, 483

binding.py, 531

bkwd  
     forcebalance::parser, 71

bo  
     forcebalance::chemistry, 17

bohrang  
     forcebalance::molecule, 42  
     forcebalance::nifty, 60

bond\_id  
     forcebalance::Mol2::mol2\_bond, 384

bond\_type  
     forcebalance::Mol2::mol2\_bond, 384

BondChars  
     forcebalance::chemistry, 17

BondEnergies  
     forcebalance::chemistry, 17

BondStrengthByLength  
     forcebalance::chemistry, 17

bonds  
     forcebalance::Mol2::mol2, 377

both  
     forcebalance::molecule, 36

Box  
     forcebalance::molecule, 42

boxes  
     forcebalance::molecule::Molecule, 410

buf  
     forcebalance::nifty::LineChunker, 322

build\_invdist  
     forcebalance::abinitio::AbInitio, 82  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 136  
     forcebalance::amberio::AbInitio\_AMBER, 99

forcebalance::gmxio::AbInitio\_GMX, 117  
 forcebalance::openmmio::AbInitio\_OpenMM, 153  
 forcebalance::tinkerio::AbInitio\_TINKER, 172

build\_objective  
     forcebalance::ab initio, 13

build\_pid  
     forcebalance::amberio::FrcMod\_Reader, 242  
     forcebalance::amberio::Mol2\_Reader, 386  
     forcebalance::BaseReader, 192  
     forcebalance::custom\_io::Gen\_Reader, 249  
     forcebalance::gmxio::ITP\_Reader, 309  
     forcebalance::mol2io::Mol2\_Reader, 389  
     forcebalance::openmmio::OpenMM\_Reader, 452  
     forcebalance::psi4io::GBS\_Reader, 245  
     forcebalance::psi4io::Grid\_Reader, 257  
     forcebalance::qchemio::QCIn\_Reader, 470  
     forcebalance::tinkerio::Tinker\_Reader, 515

build\_topology  
     forcebalance::molecule::Molecule, 399

BuildLatticeFromLengthsAngles  
     forcebalance::molecule, 37

BuildLatticeFromVectors  
     forcebalance::molecule, 37

built\_bonds  
     forcebalance::molecule::Molecule, 410

CHECK\_BASIS  
     forcebalance::leastsq, 33

calc\_eigvals  
     forcebalance::vibration::Vibration, 527

calc\_moments  
     forcebalance::gmxqpio::Monomer\_QTPIE, 446  
     forcebalance::moments::Moments, 423  
     forcebalance::tinkerio::Moments\_TINKER, 435

call\_derivatives  
     forcebalance::leastsq::LeastSquares, 320  
     forcebalance::psi4io::THCDF\_Psi4, 512

callback  
     forcebalance::nifty::LineChunker, 322

callderivs  
     forcebalance::psi4io::RDVR3\_Psi4, 483

callgmx  
     forcebalance::gmxio::GMX, 253

cartesian\_product2  
     forcebalance::molecule, 37

center\_of\_mass  
     forcebalance::molecule::Molecule, 400

charge  
     forcebalance::Mol2::mol2\_atom, 381

charge\_type  
     forcebalance::Mol2::mol2, 377

CheckBasis  
     forcebalance::leastsq, 33

chemistry.py, 531

chk  
     forcebalance::optimizer::Optimizer, 462  
 close  
     forcebalance::nifty::LineChunker, 322  
 cnum  
     forcebalance::qchemio::QCIn\_Reader, 471  
 col  
     forcebalance::nifty, 47  
 commadash  
     forcebalance::nifty, 47  
 comments  
     forcebalance::Mol2::mol2, 377  
     forcebalance::Mol2::mol2\_set, 391  
 comms  
     forcebalance::molecule::Molecule, 410  
 compounds  
     forcebalance::Mol2::mol2\_set, 391  
 compute  
     forcebalance::objective::Penalty, 465  
 compute\_netforce\_torque  
     forcebalance::abinitio::AbInitio, 82  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 136  
     forcebalance::amberio::AbInitio\_AMBER, 99  
     forcebalance::gmxio::AbInitio\_GMX, 117  
     forcebalance::openmmio::AbInitio\_OpenMM, 153  
     forcebalance::tinkerio::AbInitio\_TINKER, 172  
 ComputeOverlap  
     forcebalance::molecule, 37  
 concurrent\_map  
     forcebalance::nifty, 47  
 ConjugateGradient  
     forcebalance::optimizer::Optimizer, 457  
 contact.py, 532  
 contraction\_number  
     forcebalance::psi4io::GBS\_Reader, 247  
 CopyAmoebaAngleParameters  
     forcebalance::openmmio, 63  
 CopyAmoebaBondParameters  
     forcebalance::openmmio, 63  
 CopyAmoebaInPlaneAngleParameters  
     forcebalance::openmmio, 63  
 CopyAmoebaMultipoleParameters  
     forcebalance::openmmio, 63  
 CopyAmoebaOutOfPlaneBendParameters  
     forcebalance::openmmio, 63  
 CopyAmoebaVdwParameters  
     forcebalance::openmmio, 63  
 CopyFile  
     forcebalance::nifty, 47  
 CopyHarmonicAngleParameters  
     forcebalance::openmmio, 64  
 CopyHarmonicBondParameters  
     forcebalance::openmmio, 64  
 CopyNonbondedParameters  
     forcebalance::openmmio, 64  
     forcebalance::optimizer, 67  
     forcebalance::counterpoise::Counterpoise, 223  
     forcebalance::custom\_io, 21  
 create\_mvals  
     forcebalance::forcefield::FF, 233  
 create\_pvals  
     forcebalance::forcefield::FF, 234  
 createWorkQueue  
     forcebalance::nifty, 48  
 custom.io.py, 533

D

forcebalance::leastsq::LeastSquares, 320  
     forcebalance::psi4io::THCDF\_Psi4, 512  
 DATfnm  
     forcebalance::psi4io::THCDF\_Psi4, 512  
 DF\_Energy  
     forcebalance::psi4io::THCDF\_Psi4, 512  
 Data  
     forcebalance::molecule::Molecule, 410  
 data  
     forcebalance::Mol2, 34  
 data\_from\_web  
     forcebalance::chemistry, 17  
 defaultHandler  
     forcebalance::output::ForceBalanceLogger, 240  
 denoms  
     forcebalance::moments::Moments, 423  
     forcebalance::tinkerio::Moments\_TINKER, 435  
 destroy  
     forcebalance::psi4io::GBS\_Reader, 247  
 destroyWorkQueue  
     forcebalance::nifty, 48  
 determine\_fftype  
     forcebalance::forcefield, 27  
 dtypes  
     forcebalance::gmxio, 30  
 Diel\_B  
     forcebalance::gmxio::Interaction\_GMX, 281  
 Dielectric  
     forcebalance::gmxio::Interaction\_GMX, 281  
 diff  
     forcebalance::molecule, 37  
 dihe  
     forcebalance::amberio::FrcMod\_Reader, 243

divisor  
 forcebalance::gmxio::Interaction\_GMX, 281  
 forcebalance::interaction::Interaction, 269  
 forcebalance::openmmio::Interaction\_OpenMM, 293  
 forcebalance::tinkerio::Interaction\_TINKER, 305  
 do\_nothing  
 forcebalance::openmmio, 65  
 do\_self\_pol  
 forcebalance::gmxio::Liquid\_GMX, 344  
 forcebalance::liquid::Liquid, 332  
 forcebalance::openmmio::Liquid\_OpenMM, 357  
 forcebalance::tinkerio::Liquid\_TINKER, 370  
 driver  
 forcebalance::psi4io::RDVR3\_Psi4, 477  
 forcebalance::psi4io::THCDF\_Psi4, 506  
 dx  
 forcebalance::optimizer::Optimizer, 463  
 DynDict  
 forcebalance::tinkerio::Liquid\_TINKER, 370  
 DynDict\_New  
 forcebalance::tinkerio::Liquid\_TINKER, 370  
  
 e\_err  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
 forcebalance::amberio::AbInitio\_AMBER, 109  
 forcebalance::gmxio::AbInitio\_GMX, 128  
 forcebalance::gmxio::Interaction\_GMX, 281  
 forcebalance::interaction::Interaction, 269  
 forcebalance::openmmio::AbInitio\_OpenMM, 164  
 forcebalance::openmmio::Interaction\_OpenMM, 293  
 forcebalance::tinkerio::AbInitio\_TINKER, 183  
 forcebalance::tinkerio::Interaction\_TINKER, 305  
  
 e\_err\_pct  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
 forcebalance::amberio::AbInitio\_AMBER, 109  
 forcebalance::gmxio::AbInitio\_GMX, 128  
 forcebalance::gmxio::Interaction\_GMX, 281  
 forcebalance::interaction::Interaction, 269  
 forcebalance::openmmio::AbInitio\_OpenMM, 164  
 forcebalance::openmmio::Interaction\_OpenMM, 293  
 forcebalance::tinkerio::AbInitio\_TINKER, 183  
 forcebalance::tinkerio::Interaction\_TINKER, 305  
  
 e\_ref  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
 forcebalance::amberio::AbInitio\_AMBER, 109  
 forcebalance::gmxio::AbInitio\_GMX, 128  
 forcebalance::openmmio::AbInitio\_OpenMM, 164  
 forcebalance::tinkerio::AbInitio\_TINKER, 183  
  
 edit\_mdp  
 forcebalance::gmxio, 29  
  
 edit\_qcrems

forcebalance::molecule::Molecule, 400  
 either  
 forcebalance::molecule, 37  
 elem\_from\_atomname  
 forcebalance::molecule, 37  
 element  
 forcebalance::psi4io::GBS\_Reader, 247  
 forcebalance::psi4io::Grid\_Reader, 258  
 Elements  
 forcebalance::chemistry, 17  
 forcebalance::molecule, 42  
 forcebalance::psi4io::THCDF\_Psi4, 512  
 forcebalance::PT, 73  
 elements  
 forcebalance::psi4io::RDVR3\_Psi4, 483  
  
 emd0  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
 forcebalance::amberio::AbInitio\_AMBER, 109  
 forcebalance::gmxio::AbInitio\_GMX, 128  
 forcebalance::openmmio::AbInitio\_OpenMM, 164  
 forcebalance::tinkerio::AbInitio\_TINKER, 183  
  
 emit  
 forcebalance::output::CleanFileHandler, 212  
 forcebalance::output::CleanStreamHandler, 214  
 forcebalance::output::RawFileHandler, 472  
 forcebalance::output::RawStreamHandler, 474  
  
 emm  
 forcebalance::gmxio::Interaction\_GMX, 281  
 forcebalance::interaction::Interaction, 269  
 forcebalance::openmmio::Interaction\_OpenMM, 293  
 forcebalance::tinkerio::Interaction\_TINKER, 305  
  
 encode  
 forcebalance::nifty, 48  
  
 energy\_driver\_all  
 forcebalance::openmmio::Interaction\_OpenMM, 286  
 forcebalance::tinkerio::AbInitio\_TINKER, 172  
 forcebalance::tinkerio::Interaction\_TINKER, 298  
  
 energy\_force\_driver  
 forcebalance::gmxio::AbInitio\_GMX, 117  
 forcebalance::gmxio::GMX, 253  
 forcebalance::tinkerio::AbInitio\_TINKER, 173  
  
 energy\_force\_driver\_all  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 136  
 forcebalance::amberio::AbInitio\_AMBER, 99  
 forcebalance::gmxio::AbInitio\_GMX, 118  
 forcebalance::gmxio::GMX, 253  
 forcebalance::openmmio::AbInitio\_OpenMM, 153  
 forcebalance::tinkerio::AbInitio\_TINKER, 173  
  
 energy\_force\_driver\_all\_external  
 forcebalance::amberio::AbInitio\_AMBER, 99  
 forcebalance::openmmio::AbInitio\_OpenMM, 154  
  
 energy\_force\_driver\_all\_internal  
 forcebalance::openmmio::AbInitio\_OpenMM, 154

energy\_force\_transformer  
 forcebalance::abinitio::AbInitio, 82  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 136  
 forcebalance::amberio::AbInitio\_AMBER, 100  
 forcebalance::gmxio::AbInitio\_GMX, 118  
 forcebalance::openmmio::AbInitio\_OpenMM, 154  
 forcebalance::tinkerio::AbInitio\_TINKER, 173  
 energy\_force\_transformer\_all  
 forcebalance::abinitio::AbInitio, 82  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 137  
 forcebalance::amberio::AbInitio\_AMBER, 100  
 forcebalance::gmxio::AbInitio\_GMX, 118  
 forcebalance::openmmio::AbInitio\_OpenMM, 154  
 forcebalance::tinkerio::AbInitio\_TINKER, 173  
 energy\_part  
 forcebalance::binding::BindingEnergy, 201  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 210  
 engine  
 forcebalance::gmxio::AbInitio\_GMX, 128  
 forcebalance::gmxio::Liquid\_GMX, 344  
 forcebalance::openmmio::Liquid\_OpenMM, 357  
 engine.py, 533  
 eqcgmx  
 forcebalance::nifty, 60  
 eqm  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
 forcebalance::amberio::AbInitio\_AMBER, 109  
 forcebalance::gmxio::AbInitio\_GMX, 128  
 forcebalance::gmxio::Interaction\_GMX, 281  
 forcebalance::interaction::Interaction, 269  
 forcebalance::openmmio::AbInitio\_OpenMM, 164  
 forcebalance::openmmio::Interaction\_OpenMM, 293  
 forcebalance::tinkerio::AbInitio\_TINKER, 183  
 forcebalance::tinkerio::Interaction\_TINKER, 305  
 esp\_err  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
 forcebalance::amberio::AbInitio\_AMBER, 109  
 forcebalance::gmxio::AbInitio\_GMX, 128  
 forcebalance::openmmio::AbInitio\_OpenMM, 164  
 forcebalance::tinkerio::AbInitio\_TINKER, 183  
 espval  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
 forcebalance::amberio::AbInitio\_AMBER, 109  
 forcebalance::gmxio::AbInitio\_GMX, 128  
 forcebalance::openmmio::AbInitio\_OpenMM, 164  
 forcebalance::tinkerio::AbInitio\_TINKER, 183  
 espxyz  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
 forcebalance::amberio::AbInitio\_AMBER, 109  
 forcebalance::gmxio::AbInitio\_GMX, 128  
 forcebalance::openmmio::AbInitio\_OpenMM, 164  
 forcebalance::tinkerio::AbInitio\_TINKER, 183  
 FDCheckG  
 forcebalance::optimizer::Optimizer, 457

forcebalance::openmmio::AbInitio\_OpenMM, 164  
 forcebalance::tinkerio::AbInitio\_TINKER, 183  
 EulerMatrix  
 forcebalance::molecule, 37  
 evaluate\_optimized  
 forcebalance::engine::Engine, 226  
 forcebalance::gmxio::GMX, 253  
 evaluate\_snapshots  
 forcebalance::engine::Engine, 226  
 forcebalance::gmxio::GMX, 253  
 even\_list  
 forcebalance::molecule, 38  
 excision  
 forcebalance::forcefield::FF, 237  
 forcebalance::optimizer::Optimizer, 463  
 extra\_output  
 forcebalance::gmxio::Liquid\_GMX, 344  
 forcebalance::liquid::Liquid, 332  
 forcebalance::openmmio::Liquid\_OpenMM, 357  
 forcebalance::tinkerio::Liquid\_TINKER, 370

f12d3p  
 forcebalance::finite\_difference, 22  
 f12d7p  
 forcebalance::finite\_difference, 22  
 f1d2p  
 forcebalance::finite\_difference, 23  
 f1d5p  
 forcebalance::finite\_difference, 23  
 f1d7p  
 forcebalance::finite\_difference, 24  
 f\_err  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
 forcebalance::amberio::AbInitio\_AMBER, 109  
 forcebalance::gmxio::AbInitio\_GMX, 128  
 forcebalance::openmmio::AbInitio\_OpenMM, 164  
 forcebalance::tinkerio::AbInitio\_TINKER, 183  
 f\_err\_pct  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
 forcebalance::amberio::AbInitio\_AMBER, 109  
 forcebalance::gmxio::AbInitio\_GMX, 128  
 forcebalance::openmmio::AbInitio\_OpenMM, 164  
 forcebalance::tinkerio::AbInitio\_TINKER, 183  
 f\_ref  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
 forcebalance::amberio::AbInitio\_AMBER, 109  
 forcebalance::gmxio::AbInitio\_GMX, 128  
 forcebalance::openmmio::AbInitio\_OpenMM, 164  
 forcebalance::tinkerio::AbInitio\_TINKER, 183

FDCheckH  
     forcebalance::optimizer::Optimizer, 458  
**FF**  
     forcebalance::abinitio::AbInitio, 91  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
     forcebalance::amberio::AbInitio\_AMBER, 109  
     forcebalance::binding::BindingEnergy, 201  
     forcebalance::counterpoise::Counterpoise, 223  
     forcebalance::gmxio::AbInitio\_GMX, 128  
     forcebalance::gmxio::Interaction\_GMX, 281  
     forcebalance::gmxio::Liquid\_GMX, 344  
     forcebalance::gmxpath::Monomer\_QTPIE, 446  
     forcebalance::interaction::Interaction, 269  
     forcebalance::leastsq::LeastSquares, 320  
     forcebalance::liquid::Liquid, 332  
     forcebalance::moments::Moments, 423  
     forcebalance::objective::Objective, 450  
     forcebalance::objective::Penalty, 467  
     forcebalance::openmm::AbInitio\_OpenMM, 164  
     forcebalance::openmm::Interaction\_OpenMM, 293  
     forcebalance::openmm::Liquid\_OpenMM, 357  
     forcebalance::optimizer::Optimizer, 463  
     forcebalance::psi4io::RDVR3\_Psi4, 483  
     forcebalance::psi4io::THCDF\_Psi4, 512  
     forcebalance::target::RemoteTarget, 492  
     forcebalance::target::Target, 502  
     forcebalance::tinkerio::AbInitio\_TINKER, 183  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 210  
     forcebalance::tinkerio::Interaction\_TINKER, 305  
     forcebalance::tinkerio::Liquid\_TINKER, 370  
     forcebalance::tinkerio::Moments\_TINKER, 435  
     forcebalance::vibration::Vibration, 527  
**FF.Extensions**  
     forcebalance::forcefield, 27  
**FF.IOModules**  
     forcebalance::forcefield, 27  
**FFAtomTypes**  
     forcebalance::forcefield::FF, 237  
**FFMolecules**  
     forcebalance::forcefield::FF, 237  
**FUSE**  
     forcebalance::objective::Penalty, 465  
**FUSE\_BARRIER**  
     forcebalance::objective::Penalty, 465  
**FUSE\_L0**  
     forcebalance::objective::Penalty, 465  
**factor**  
     forcebalance::psi4io::RDVR3\_Psi4, 483  
**fadd**  
     forcebalance::objective::Penalty, 467  
**fdict**  
     forcebalance::custom\_io, 21  
     forcebalance::gmxio, 30  
**fdwrap**  
     forcebalance::finite\_difference, 24  
**fdwrap\_G**  
     forcebalance::finite\_difference, 25  
**fdwrap\_H**  
     forcebalance::finite\_difference, 25  
**feed**  
     forcebalance::amberio::FrcMod\_Reader, 242  
     forcebalance::amberio::Mol2\_Reader, 386  
     forcebalance::BaseReader, 192  
     forcebalance::custom\_io::Gen\_Reader, 249  
     forcebalance::gmxio::ITP\_Reader, 309  
     forcebalance::mol2io::Mol2\_Reader, 389  
     forcebalance::openmm::OpenMM\_Reader, 453  
     forcebalance::psi4io::GBS\_Reader, 245, 246  
     forcebalance::psi4io::Grid\_Reader, 257  
     forcebalance::qchemio::QCIn\_Reader, 470  
     forcebalance::tinkerio::Tinker\_Reader, 515  
**ffdata**  
     forcebalance::forcefield::FF, 237  
**ffdata\_isxml**  
     forcebalance::forcefield::FF, 237  
**find\_spacings**  
     forcebalance::forcefield::FF, 234  
**finite\_difference.py**, 533  
**fitatoms**  
     forcebalance::abinitio::AbInitio, 91  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 109  
     forcebalance::gmxio::AbInitio\_GMX, 128  
     forcebalance::openmm::AbInitio\_OpenMM, 164  
     forcebalance::tinkerio::AbInitio\_TINKER, 183  
**flat**  
     forcebalance::nifty, 48  
**floatornan**  
     forcebalance::nifty, 49  
**fmul**  
     forcebalance::objective::Penalty, 467  
**force**  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmm::AbInitio\_OpenMM, 164  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
**force\_map**  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmm::AbInitio\_OpenMM, 164  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
**forcebalance**, 11  
     \_\_version\_\_, 13  
**forcebalance.abinitio**, 13

forcebalance.abinitio.ABInitio, 77  
forcebalance.abinitio\_internal, 14  
forcebalance.abinitio\_internal.ABInitio\_Internal, 131  
forcebalance.amberio, 14  
forcebalance.amberio.ABInitio\_AMBER, 94  
forcebalance.amberio.FrcMod\_Reader, 240  
forcebalance.amberio.Mol2\_Reader, 384  
forcebalance.BaseClass, 188  
forcebalance.BaseReader, 190  
forcebalance.binding, 16  
forcebalance.binding.BindingEnergy, 193  
forcebalance.chemistry, 17  
forcebalance.contact, 19  
forcebalance.counterpoise, 19  
forcebalance.counterpoise.Counterpoise, 214  
forcebalance.custom\_io, 20  
forcebalance.custom\_io.Gen\_Reader, 247  
forcebalance.engine, 21  
forcebalance.engine.Engine, 224  
forcebalance.finite\_difference, 21  
forcebalance.forcefield, 25  
forcebalance.forcefield.BackedUpDict, 186  
forcebalance.forcefield.FF, 227  
forcebalance.gmxio, 28  
forcebalance.gmxio.ABInitio\_GMX, 112  
forcebalance.gmxio.GMX, 250  
forcebalance.gmxio.ITP\_Reader, 306  
forcebalance.gmxio.Interaction\_GMX, 270  
forcebalance.gmxio.Liquid\_GMX, 334  
forcebalance.gmxqpio, 31  
forcebalance.gmxqpio.Monomer\_QTPIE, 436  
forcebalance.interaction, 32  
forcebalance.interaction.Interaction, 259  
forcebalance.leastsq, 32  
forcebalance.leastsq.LeastSquares, 311  
forcebalance.liquid, 33  
forcebalance.liquid.Liquid, 322  
forcebalance.Mol2, 34  
forcebalance.Mol2.mol2, 372  
forcebalance.Mol2.mol2\_atom, 378  
forcebalance.Mol2.mol2\_bond, 382  
forcebalance.Mol2.mol2\_set, 390  
forcebalance.mol2io, 34  
forcebalance.mol2io.Mol2\_Reader, 388  
forcebalance.molecule, 35  
forcebalance.molecule.Molecule, 391  
forcebalance.molecule.MolfileTimestep, 411  
forcebalance.moments, 44  
forcebalance.moments.Moments, 412  
forcebalance.nifty, 44  
forcebalance.nifty.LineChunker, 321  
forcebalance.nifty.Pickler\_LP, 467  
forcebalance.nifty.Unpickler\_LP, 517  
forcebalance.objective, 61  
forcebalance.objective.Objective, 447  
forcebalance.objective.Penalty, 464  
forcebalance.openmmio, 62  
forcebalance.openmmio.ABInitio\_OpenMM, 148  
forcebalance.openmmio.Interaction\_OpenMM, 282  
forcebalance.openmmio.Liquid\_OpenMM, 346  
forcebalance.openmmio.OpenMM\_Reader, 450  
forcebalance.optimizer, 67  
forcebalance.optimizer.Optimizer, 454  
forcebalance.output, 68  
forcebalance.output.CleanFileHandler, 211  
forcebalance.output.CleanStreamHandler, 212  
forcebalance.output.ForceBalanceLogger, 239  
forcebalance.output.RawFileHandler, 472  
forcebalance.output.RawStreamHandler, 473  
forcebalance.PT, 73  
forcebalance.parser, 68  
forcebalance.psi4io, 72  
forcebalance.psi4io.GBS\_Reader, 243  
forcebalance.psi4io.Grid\_Reader, 255  
forcebalance.psi4io.RDVR3\_Psi4, 474  
forcebalance.psi4io.THCDF\_Psi4, 503  
forcebalance.qchemio, 74  
forcebalance.qchemio.QCIn\_Reader, 468  
forcebalance.target, 75  
forcebalance.target.RemoteTarget, 484  
forcebalance.target.Target, 493  
forcebalance.tinkerio, 75  
forcebalance.tinkerio.ABInitio\_TINKER, 167  
forcebalance.tinkerio.BindingEnergy\_TINKER, 202  
forcebalance.tinkerio.Interaction\_TINKER, 295  
forcebalance.tinkerio.Liquid\_TINKER, 360  
forcebalance.tinkerio.Moments\_TINKER, 424  
forcebalance.tinkerio.Tinker\_Reader, 513  
forcebalance.tinkerio.Vibration\_TINKER, 528  
forcebalance.vibration, 77  
forcebalance.vibration.Vibration, 518  
forcebalance::BaseClass  
    \_\_init\_\_, 189  
    PrintOptionDict, 189  
    set\_option, 189  
    verbose\_options, 189  
forcebalance::BaseReader  
    \_\_init\_\_, 191  
    adict, 192  
    AtomTypes, 192  
    build\_pid, 192  
    feed, 192  
    itype, 192  
    In, 192  
    molatom, 193  
    Molecules, 193  
    pdict, 193  
    Split, 192

suffix, 193  
 Whites, 192  
**forcebalance::Mol2**  
 data, 34  
**forcebalance::Mol2::mol2**  
`__init__`, 373  
`__repr__`, 373  
 atoms, 377  
 bonds, 377  
 charge\_type, 377  
 comments, 377  
`get_atom`, 373  
`get_bonded_atoms`, 374  
 mol\_name, 377  
 mol\_type, 377  
 num\_atoms, 377  
 num\_bonds, 377  
 num\_feat, 377  
 num\_sets, 377  
 num\_subst, 377  
 out, 374  
`parse`, 374  
`set_charge_type`, 374  
`set_donor_acceptor_atoms`, 375  
`set_mol_name`, 375  
`set_mol_type`, 375  
`set_num_atoms`, 376  
`set_num_bonds`, 376  
`set_num_feat`, 376  
`set_num_sets`, 376  
`set_num_subst`, 377  
**forcebalance::Mol2::mol2\_atom**  
`__init__`, 378  
`__repr__`, 379  
`atom_id`, 381  
`atom_name`, 381  
`atom_type`, 381  
`charge`, 381  
`parse`, 379  
`set_atom_id`, 379  
`set_atom_name`, 379  
`set_atom_type`, 379  
`set_charge`, 380  
`set_crds`, 380  
`set_status_bit`, 380  
`set_subst_id`, 380  
`set_subst_name`, 381  
`status_bit`, 381  
`subst_id`, 381  
`subst_name`, 381  
`x`, 381  
`y`, 381  
`z`, 381  
**forcebalance::Mol2::mol2\_bond**  
`__init__`, 382  
`__repr__`, 382  
`bond_id`, 384  
`bond_type`, 384  
`origin_atom_id`, 384  
`parse`, 383  
`set_bond_id`, 383  
`set_bond_type`, 383  
`set_origin_atom_id`, 383  
`set_status_bit`, 383  
`set_target_atom_id`, 384  
`status_bit`, 384  
`target_atom_id`, 384  
**forcebalance::Mol2::mol2\_set**  
`__init__`, 391  
 comments, 391  
 compounds, 391  
 num\_compounds, 391  
`parse`, 391  
**forcebalance::PT**  
 Elements, 73  
 PeriodicTable, 73  
**forcebalance::ab initio**  
`build_objective`, 13  
`logger`, 14  
`weighted_variance`, 13  
`weighted_variance2`, 14  
**forcebalance::ab initio::AbInitio**  
`__init__`, 81  
`AtomLists`, 91  
`build_invdist`, 82  
`compute_netforce_torque`, 82  
`e_err`, 91  
`e_err_pct`, 91  
`e_ref`, 91  
`emd0`, 91  
`energy_force_transformer`, 82  
`energy_force_transformer_all`, 82  
`eqm`, 91  
`esp_err`, 91  
`espval`, 91  
`espxyz`, 91  
`f_err`, 91  
`f_err_pct`, 91  
`f_ref`, 91  
`FF`, 91  
`fitatoms`, 91  
`force`, 92  
`force_map`, 92  
`fqm`, 92  
`fref`, 92  
`gct`, 92  
`get`, 83  
`get_G`, 84

get\_H, 85  
 get\_X, 86  
 get\_energy\_force\_, 83  
 get\_resp\_, 86  
 hct, 92  
 indicate, 87  
 invdists, 92  
 link\_from\_tempdir, 87  
 nesp, 92  
 new\_vsites, 92  
 nf\_err, 92  
 nf\_err\_pct, 92  
 nf\_ref, 92  
 nftqm, 92  
 nnf, 92  
 nparticles, 92  
 ns, 92  
 ntq, 92  
 objective, 92  
 prepare\_temp\_directory, 87  
 PrintOptionDict, 92  
 printcool\_table, 87  
 qfnm, 92  
 qmatoms, 93  
 qmboltz\_wts, 93  
 read\_reference\_data, 88  
 read\_topology, 89  
 refresh\_temp\_directory, 89  
 respterm, 93  
 rundir, 93  
 save\_vmvalls, 93  
 set\_option, 89  
 sget, 90  
 stage, 90  
 submit\_jobs, 90  
 tempdir, 93  
 topology\_flag, 93  
 tq\_err, 93  
 tq\_err\_pct, 93  
 tq\_ref, 93  
 traj, 93  
 use\_nft, 93  
 verbose\_options, 93  
 w\_energy, 93  
 w\_force, 93  
 w\_netforce, 93  
 w\_resp, 93  
 w\_torque, 93  
 whamboltz, 94  
 whamboltz\_wts, 94  
 wq\_complete, 91  
 xct, 94  
 forcebalance::abinitio\_internal::AbInitio\_Internal  
 \_init\_, 136

AtomLists, 145  
 build\_invdist, 136  
 compute\_netforce\_torque, 136  
 e\_err, 145  
 e\_err\_pct, 145  
 e\_ref, 145  
 emd0, 145  
 energy\_force\_driver\_all, 136  
 energy\_force\_transformer, 136  
 energy\_force\_transformer\_all, 137  
 eqm, 145  
 esp\_err, 145  
 espval, 145  
 espxyz, 145  
 f\_err, 145  
 f\_err\_pct, 145  
 f\_ref, 145  
 FF, 145  
 fitatoms, 146  
 force, 146  
 force\_map, 146  
 fqm, 146  
 fref, 146  
 gct, 146  
 get, 137  
 get\_G, 138  
 get\_H, 139  
 get\_X, 140  
 get\_energy\_force\_, 137  
 get\_resp\_, 140  
 hct, 146  
 indicate, 141  
 invdists, 146  
 link\_from\_tempdir, 141  
 nesp, 146  
 new\_vsites, 146  
 nf\_err, 146  
 nf\_err\_pct, 146  
 nf\_ref, 146  
 nftqm, 146  
 nnf, 146  
 nparticles, 146  
 ns, 146  
 ntq, 146  
 objective, 146  
 prepare\_temp\_directory, 141  
 PrintOptionDict, 146  
 printcool\_table, 141  
 qfnm, 147  
 qmatoms, 147  
 qmboltz\_wts, 147  
 read\_reference\_data, 142  
 read\_topology, 143  
 refresh\_temp\_directory, 143

respterm, 147  
 rundir, 147  
 save\_vmvalls, 147  
 set\_option, 143  
 sget, 144  
 stage, 144  
 submit\_jobs, 144  
 tempdir, 147  
 topology\_flag, 147  
 tq\_err, 147  
 tq\_err\_pct, 147  
 tq\_ref, 147  
 traj, 147  
 trajfnm, 147  
 use\_nft, 147  
 verbose\_options, 147  
 w\_energy, 147  
 w\_force, 148  
 w\_netforce, 148  
 w\_resp, 148  
 w\_torque, 148  
 whamboltz, 148  
 whamboltz\_wts, 148  
 wq\_complete, 145  
 xct, 148  
**forcebalance::amberio**  
 frmod\_pdct, 15  
 is\_mol2\_atom, 15  
 logger, 15  
 mol2\_pdct, 16  
**forcebalance::amberio::AbInitio\_AMBER**  
 \_\_init\_\_, 99  
 all\_at\_once, 109  
 AtomLists, 109  
 build\_invdist, 99  
 compute\_netforce\_torque, 99  
 e\_err, 109  
 e\_err\_pct, 109  
 e\_ref, 109  
 emd0, 109  
 energy\_force\_driver\_all, 99  
 energy\_force\_transformer, 100  
 energy\_force\_transformer\_all, 100  
 eqm, 109  
 esp\_err, 109  
 espval, 109  
 espxyz, 109  
 f\_err, 109  
 f\_err\_pct, 109  
 f\_ref, 109  
 FF, 109  
 fitatoms, 109  
 force, 110  
 force\_map, 110  
 fqm, 110  
 fref, 110  
 gct, 110  
 get, 100  
 get\_G, 102  
 get\_H, 102  
 get\_X, 103  
 get\_energy\_force\_, 101  
 get\_resp\_, 103  
 hct, 110  
 indicate, 104  
 invdists, 110  
 link\_from\_tempdir, 104  
 nesp, 110  
 new\_vsites, 110  
 nf\_err, 110  
 nf\_err\_pct, 110  
 nf\_ref, 110  
 nftqm, 110  
 nnf, 110  
 nparticles, 110  
 ns, 110  
 ntq, 110  
 objective, 110  
 prepare\_temp\_directory, 104  
 PrintOptionDict, 110  
 printcool\_table, 105  
 qfnm, 110  
 qmatoms, 111  
 qmboltz\_wts, 111  
 read\_reference\_data, 105  
 read\_topology, 106  
 refresh\_temp\_directory, 106  
 respterm, 111  
 rundir, 111  
 save\_vmvalls, 111  
 set\_option, 107  
 sget, 107  
 stage, 108  
 submit\_jobs, 108  
 tempdir, 111  
 topology\_flag, 111  
 tq\_err, 111  
 tq\_err\_pct, 111  
 tq\_ref, 111  
 traj, 111  
 trajfnm, 111  
 use\_nft, 111  
 verbose\_options, 111  
 w\_energy, 111  
 w\_force, 111  
 w\_netforce, 112  
 w\_resp, 112  
 w\_torque, 112

whamboltz, 112  
 whamboltz\_wts, 112  
 wq\_complete, 108  
 xct, 112  
**forcebalance::amberio::FrcMod\_Reader**  
 \_\_init\_\_, 242  
 adict, 243  
 atom, 243  
 AtomTypes, 243  
 build\_pid, 242  
 dihe, 243  
 feed, 242  
 itype, 243  
 In, 243  
 molatom, 243  
 Molecules, 243  
 pdict, 243  
 Split, 242  
 suffix, 243  
 Whites, 242  
**forcebalance::amberio::Mol2\_Reader**  
 \_\_init\_\_, 386  
 adict, 387  
 atom, 387  
 AtomTypes, 387  
 atomnames, 387  
 build\_pid, 386  
 feed, 386  
 itype, 387  
 In, 387  
 mol, 387  
 molatom, 387  
 Molecules, 387  
 pdict, 387  
 section, 387  
 Split, 386  
 suffix, 387  
 Whites, 386  
**forcebalance::binding**  
 logger, 16  
 parse\_interactions, 16  
**forcebalance::binding::BindingEnergy**  
 \_\_init\_\_, 195  
 energy\_part, 201  
 FF, 201  
 gct, 201  
 get, 195  
 get\_G, 195  
 get\_H, 196  
 get\_X, 197  
 hct, 201  
 indicate, 198  
 inter\_opts, 201  
 link\_from\_tempdir, 198  
 objective, 201  
 PrintDict, 201  
 PrintOptionDict, 201  
 printcool\_table, 198  
 RMSDDict, 201  
 refresh\_temp\_directory, 199  
 rmsd\_part, 201  
 rundir, 201  
 set\_option, 199  
 sget, 199  
 stage, 200  
 submit\_jobs, 200  
 tempdir, 201  
 verbose\_options, 201  
 wq\_complete, 200  
 xct, 201  
**forcebalance::chemistry**  
 A, 17  
 atoms, 17  
 B, 17  
 BE, 17  
 bo, 17  
 BondChars, 17  
 BondEnergies, 17  
 BondStrengthByLength, 17  
 data\_from\_web, 17  
 Elements, 17  
 L, 18  
 line, 18  
 LookupByMass, 17  
 PeriodicTable, 18  
 Radii, 18  
**forcebalance::contact**  
 atom\_distances, 19  
 residue\_distances, 19  
**forcebalance::counterpoise**  
 logger, 20  
**forcebalance::counterpoise::Counterpoise**  
 \_\_init\_\_, 216  
 cpqm, 223  
 FF, 223  
 gct, 223  
 get, 217  
 get\_G, 217  
 get\_H, 218  
 get\_X, 219  
 hct, 223  
 link\_from\_tempdir, 220  
 load\_cp, 220  
 loadxyz, 220  
 na, 223  
 ns, 223  
 PrintOptionDict, 223  
 printcool\_table, 220

```

refresh_temp_directory, 221
rundir, 223
set_option, 221
sget, 221
stage, 222
submit_jobs, 222
tempdir, 223
verbose_options, 223
wq_complete, 222
xct, 223
xyzs, 223
forcebalance::custom_io
    cotypes, 21
    fdict, 21
    ndtypes, 21
    pdict, 21
forcebalance::custom_io::Gen_Reader
    __init__, 249
    adict, 250
    AtomTypes, 250
    build_pid, 249
    feed, 249
    itype, 250
    In, 250
    molatom, 250
    Molecules, 250
    pdict, 250
    sec, 250
    Split, 249
    suffix, 250
    Whites, 250
forcebalance::engine
    logger, 21
forcebalance::engine::Engine
    __init__, 226
    evaluate_optimized, 226
    evaluate_snapshots, 226
    name, 227
    prepare, 226
    PrintOptionDict, 227
    root, 227
    set_option, 227
    target, 227
    verbose_options, 227
forcebalance::finite_difference
    f12d3p, 22
    f12d7p, 22
    f1d2p, 23
    f1d5p, 23
    f1d7p, 24
    fdwrap, 24
    fdwrap_G, 25
    fdwrap_H, 25
    in_fd, 25
                                logger, 25
forcebalance::forcefield
    determine_fftype, 27
    FF_Extensions, 27
    FF_IOModules, 27
    logger, 28
    rs_override, 27
forcebalance::forcefield::BackedUpDict
    __init__, 187
    __missing__, 187
    backup_dict, 188
forcebalance::forcefield::FF
    __eq__, 230
    __init__, 230
    addff, 230
    addff_txt, 231
    addff_xml, 232
    assign_field, 232
    assign_p0, 233
    atomnames, 237
    create_mvals, 233
    create_pvalls, 234
    excision, 237
    FFAAtomTypes, 237
    FFMolecules, 237
    ffldata, 237
    ffldata_isxml, 237
    find_spacings, 234
    linedestroy_save, 237
    linedestroy_this, 237
    list_map, 234
    make, 235
    make_redirect, 235
    map, 237
    mktransmat, 235
    np, 237
    openmmxml, 237
    parmdestroy_save, 237
    parmdestroy_this, 237
    patoms, 238
    pfields, 238
    plist, 238
    print_map, 236
    PrintOptionDict, 238
    pvalls0, 238
    qid, 238
    qid2, 238
    qmap, 238
    Readers, 238
    redirect, 238
    rs, 238
    rsmake, 236
    set_option, 237
    tinkerprm, 238

```

tm, 238  
 tml, 238  
 verbose\_options, 238  
**forcebalance::gmxio**  
 aftytypes, 30  
 bftytypes, 30  
 dftytypes, 30  
 edit\_mdp, 29  
 fdict, 30  
 logger, 31  
 nftytypes, 31  
 parse\_atomtype\_line, 29  
 pdict, 31  
 pftytypes, 31  
 rm\_gmx\_baks, 30  
 shot\_mdp, 31  
**forcebalance::gmxio::AbInitio\_GMX**  
 \_\_init\_\_, 117  
 AtomLists, 128  
 AtomMask, 128  
 build\_invdist, 117  
 compute\_netforce\_torque, 117  
 e\_err, 128  
 e\_err\_pct, 128  
 e\_ref, 128  
 emd0, 128  
 energy\_force\_driver, 117  
 energy\_force\_driver\_all, 118  
 energy\_force\_transformer, 118  
 energy\_force\_transformer\_all, 118  
 engine, 128  
 eqm, 128  
 esp\_err, 128  
 espval, 128  
 espxyz, 128  
 f\_err, 128  
 f\_err\_pct, 128  
 f\_ref, 128  
 FF, 128  
 fitatoms, 128  
 force, 129  
 force\_map, 129  
 fqm, 129  
 fref, 129  
 gct, 129  
 generate\_vsites\_positions, 118  
 get, 119  
 get\_G, 120  
 get\_H, 121  
 get\_X, 122  
 get\_energy\_force, 119  
 get\_resp, 122  
 hct, 129  
 indicate, 123  
 invdists, 129  
 link\_from\_tempdir, 123  
 nesp, 129  
 new\_vsites, 129  
 nf\_err, 129  
 nf\_err\_pct, 129  
 nf\_ref, 129  
 nftqm, 129  
 nnf, 129  
 nparticles, 129  
 ns, 129  
 ntq, 129  
 objective, 129  
 prepare\_temp\_directory, 123  
 PrintOptionDict, 129  
 printcool\_table, 124  
 qfnm, 129  
 qmatoms, 130  
 qmboltz\_wts, 130  
 read\_reference\_data, 124  
 read\_topology, 125  
 refresh\_temp\_directory, 125  
 respterm, 130  
 rundir, 130  
 save\_vmvalls, 130  
 set\_option, 126  
 sget, 126  
 stage, 127  
 submit\_jobs, 127  
 tempdir, 130  
 topology\_flag, 130  
 tq\_err, 130  
 tq\_err\_pct, 130  
 tq\_ref, 130  
 traj, 130  
 use\_nft, 130  
 verbose\_options, 130  
 w\_energy, 130  
 w\_force, 130  
 w\_netforce, 130  
 w\_resp, 130  
 w\_torque, 131  
 whamboltz, 131  
 whamboltz\_wts, 131  
 wq\_complete, 127  
 xct, 131  
**forcebalance::gmxio::GMX**  
 \_\_init\_\_, 253  
 AtomLists, 255  
 AtomMask, 255  
 callgmx, 253  
 energy\_force\_driver, 253  
 energy\_force\_driver\_all, 253  
 evaluate\_optimized, 253

```

evaluate_snapshots, 253
generate_vsites_positions, 254
gmxpath, 255
gmxsuffix, 255
mdp, 255
mol, 255
name, 255
prepare, 254
PrintOptionDict, 255
root, 255
set_option, 254
srcdir, 255
target, 255
top, 255
verbose_options, 255
forcebalance::gmlio::ITP_Reader
    __init__, 309
    adict, 310
    AtomTypes, 310
    atomnames, 310
    atomtype_to_mass, 310
    atomtypes, 310
    build_pid, 309
    feed, 309
    itype, 310
    In, 310
    mol, 310
    molatom, 310
    Molecules, 310
    nbtype, 310
    pdict, 310
    sec, 310
    Split, 309
    suffix, 310
    Whites, 309
forcebalance::gmlio::Interaction_GMX
    __init__, 274
    Diel_B, 281
    Dielectric, 281
    divisor, 281
    e_err, 281
    e_err_pct, 281
    emm, 281
    eqm, 281
    FF, 281
    gct, 281
    get, 274
    get_G, 274
    get_H, 275
    get_X, 276
    hct, 281
    indicate, 277
    interaction_driver, 277
    interaction_driver_all, 277
    label, 281
    link_from_tempdir, 278
    ns, 281
    objective, 281
    prefactor, 281
    prepare_temp_directory, 278
    PrintOptionDict, 281
    printcool_table, 278
    qfnm, 281
    read_reference_data, 278
    refresh_temp_directory, 279
    rundir, 282
    select1, 282
    select2, 282
    set_option, 279
    sget, 279
    stage, 280
    submit_jobs, 280
    tempdir, 282
    topfnm, 282
    traj, 282
    trajfnm, 282
    verbose_options, 282
    weight, 282
    wq_complete, 280
    xct, 282
forcebalance::gmlio::Liquid_GMX
    __init__, 337
    do_self_pol, 344
    engine, 344
    extra_output, 344
    FF, 344
    gas_fnm, 344
    gct, 344
    get, 337
    get_G, 338
    get_H, 338
    get_X, 339
    hct, 344
    indicate, 340
    Labels, 344
    last_traj, 344
    link_from_tempdir, 340
    liquid_conf, 344
    liquid_fnm, 344
    liquid_traj, 345
    MBarEnergy, 345
    npt_simulation, 340
    nptfiles, 345
    nptpx, 345
    nptsfx, 345
    objective_term, 340
    PhasePoints, 345
    polarization_correction, 341

```

```

prepare_temp_directory, 341
PrintOptionDict, 345
printcool_table, 341
read_data, 342
RefData, 345
refresh_temp_directory, 342
rundir, 345
SavedMVal, 345
SavedTraj, 345
set_option, 342
sget, 342
stage, 343
submit_jobs, 343
tempdir, 345
verbose_options, 345
w_alpha, 345
w_cp, 345
w_eps0, 345
w_hvap, 345
w_kappa, 345
w_rho, 346
wq_complete, 344
xct, 346
forcebalance::gmxqpio
    get_monomer_properties, 32
    logger, 32
forcebalance::gmxqpio::Monomer_QTPIE
    __init__, 439
    calc_moments, 446
    FF, 446
    gct, 446
    get, 439
    get_G, 439
    get_H, 440
    get_X, 441
    hct, 446
    indicate, 442
    link_from_tempdir, 442
    objective, 446
    prepare_temp_directory, 442
    PrintOptionDict, 446
    printcool_table, 443
    ref_moments, 446
    refresh_temp_directory, 443
    rundir, 446
    set_option, 444
    sget, 444
    stage, 445
    submit_jobs, 445
    tempdir, 446
    unpack_moments, 445
    verbose_options, 447
    weights, 447
    wq_complete, 446
                                xct, 447
forcebalance::interaction
    logger, 32
forcebalance::interaction::Interaction
    __init__, 262
    divisor, 269
    e_err, 269
    e_err_pct, 269
    emm, 269
    eqm, 269
    FF, 269
    gct, 269
    get, 262
    get_G, 262
    get_H, 263
    get_X, 264
    hct, 269
    indicate, 265
    label, 269
    link_from_tempdir, 265
    ns, 269
    objective, 269
    prefactor, 269
    prepare_temp_directory, 265
    PrintOptionDict, 269
    printcool_table, 265
    qfnm, 269
    read_reference_data, 266
    refresh_temp_directory, 266
    rundir, 269
    select1, 270
    select2, 270
    set_option, 267
    sget, 267
    stage, 268
    submit_jobs, 268
    tempdir, 270
    traj, 270
    verbose_options, 270
    weight, 270
    wq_complete, 268
    xct, 270
forcebalance::leastsq
    CHECK_BASIS, 33
    CheckBasis, 33
    LAST_MVALS, 33
    LastMvals, 33
    logger, 33
forcebalance::leastsq::LeastSquares
    __init__, 313
    call_derivatives, 320
    D, 320
    FF, 320
    gct, 320

```

get, 313  
 get\_G, 314  
 get\_H, 315  
 get\_X, 316  
 hct, 320  
 indicate, 317  
 link\_from\_tempdir, 317  
 MAQ, 320  
 objective, 320  
 PrintOptionDict, 320  
 printcool\_table, 317  
 refresh\_temp\_directory, 318  
 rundir, 320  
 set\_option, 318  
 sget, 318  
 stage, 319  
 submit\_jobs, 319  
 tempdir, 320  
 verbose\_options, 320  
 wq\_complete, 319  
 xct, 320  
**forcebalance::liquid**  
 logger, 34  
 weight\_info, 33  
**forcebalance::liquid::Liquid**  
 \_\_init\_\_, 325  
 do\_self\_pol, 332  
 extra\_output, 332  
 FF, 332  
 gct, 332  
 get, 325  
 get\_G, 326  
 get\_H, 326  
 get\_X, 327  
 hct, 332  
 indicate, 328  
 Labels, 332  
 last\_traj, 332  
 link\_from\_tempdir, 328  
 MBarEnergy, 332  
 npt\_simulation, 328  
 nptfiles, 332  
 nptpx, 332  
 nptsfx, 333  
 objective\_term, 328  
 PhasePoints, 333  
 polarization\_correction, 329  
 PrintOptionDict, 333  
 printcool\_table, 329  
 read\_data, 329  
 RefData, 333  
 refresh\_temp\_directory, 329  
 rundir, 333  
 SavedMVal, 333  
 SavedTraj, 333  
 set\_option, 330  
 sget, 330  
 stage, 331  
 submit\_jobs, 331  
 tempdir, 333  
 verbose\_options, 333  
 w\_alpha, 333  
 w\_cp, 333  
 w\_eps0, 333  
 w\_hvap, 333  
 w\_kappa, 333  
 w\_rho, 333  
 wq\_complete, 332  
 xct, 333  
**forcebalance::mol2io**  
 mol2\_pdict, 34  
**forcebalance::mol2io::Mol2\_Reader**  
 \_\_init\_\_, 389  
 adict, 390  
 atom, 390  
 AtomTypes, 390  
 build\_pid, 389  
 feed, 389  
 itype, 390  
 In, 390  
 molatom, 390  
 Molecules, 390  
 pdict, 390  
 Split, 389  
 suffix, 390  
 Whites, 389  
**forcebalance::molecule**  
 add\_strip\_to\_mat, 36  
 AlignToDensity, 36  
 AlignToMoments, 36  
 Alive, 42  
 AllVariableNames, 42  
 AtomVariableNames, 42  
 bohrang, 42  
 both, 36  
 Box, 42  
 BuildLatticeFromLengthsAngles, 37  
 BuildLatticeFromVectors, 37  
 cartesian\_product2, 37  
 ComputeOverlap, 37  
 diff, 37  
 either, 37  
 elem\_from\_atomname, 37  
 Elements, 42  
 EulerMatrix, 37  
 even\_list, 38  
 format\_gro\_box, 38  
 format\_gro\_coord, 38

format\_xyz\_coord, 39  
 format\_xyzgen\_coord, 39  
 FrameVariableNames, 43  
 get\_rotate\_translate, 40  
 getElement, 40  
 grouper, 40  
 is\_charmm\_coord, 40  
 is\_gro\_box, 41  
 is\_gro\_coord, 41  
 isfloat, 41  
 isint, 42  
 main, 42  
 MetaVariableNames, 43  
 nodematch, 42  
 PeriodicTable, 43  
 pvec, 42  
 QuantumVariableNames, 43  
 radian, 43  
 Radii, 43  
 splitter, 44  
 forcebalance::molecule::Molecule  
     \_\_add\_\_, 395  
     \_\_delitem\_\_, 395  
     \_\_getattr\_\_, 396  
     \_\_getitem\_\_, 396  
     \_\_iadd\_\_, 396  
     \_\_init\_\_, 395  
     \_\_iter\_\_, 397  
     \_\_len\_\_, 397  
     \_\_setattr\_\_, 397  
     add\_quantum, 397  
     add\_virtual\_site, 398  
     align, 398  
     align\_by\_moments, 398  
     align\_center, 398  
     all\_pairwise\_rmsd, 398  
     append, 398  
     atom\_select, 399  
     atom\_stack, 399  
     boxes, 410  
     build\_topology, 399  
     built\_bonds, 410  
     center\_of\_mass, 400  
     comms, 410  
     Data, 410  
     edit\_qcrems, 400  
     fout, 410  
     Funnel, 411  
     load\_frames, 400  
     measure\_dihedrals, 400  
     molecules, 411  
     openmm\_boxes, 400  
     openmm\_positions, 401  
     pathwise\_rmsd, 401  
     positive\_resid, 411  
     radius\_of\_gyration, 401  
     Read\_Tab, 411  
     read\_arc, 401  
     read\_charmm, 402  
     read\_com, 402  
     read\_dcd, 402  
     read\_gro, 403  
     read\_mdcrd, 403  
     read\_mol2, 403  
     read\_pdb, 404  
     read\_qcesp, 404  
     read\_qcin, 404  
     read\_qcout, 405  
     read\_qdata, 405  
     read\_xyz, 406  
     ref\_rmsd, 406  
     replace\_peratom, 406  
     replace\_peratom\_conditional, 406  
     require, 407  
     require\_boxes, 407  
     require\_resid, 407  
     require\_resname, 407  
     resid, 411  
     resname, 411  
     split, 407  
     topology, 411  
     write, 407  
     Write\_Tab, 411  
     write\_arc, 408  
     write\_dcd, 408  
     write\_gro, 408  
     write\_mdcrd, 408  
     write\_molproq, 408  
     write\_pdb, 409  
     write\_qcin, 409  
     write\_qdata, 410  
     write\_xyz, 410  
 forcebalance::moments  
     logger, 44  
 forcebalance::moments::Moments  
     \_\_init\_\_, 416  
     calc\_moments, 423  
     denoms, 423  
     FF, 423  
     gct, 423  
     get, 416  
     get\_G, 416  
     get\_H, 417  
     get\_X, 418  
     hct, 423  
     indicate, 419  
     link\_from\_tempdir, 419  
     mfnm, 423

na, 423  
 objective, 423  
 prepare\_temp\_directory, 419  
 PrintOptionDict, 423  
 printcool\_table, 420  
 read\_reference\_data, 420  
 ref\_eigvals, 423  
 ref\_eigvecs, 424  
 ref\_moments, 424  
 refresh\_temp\_directory, 420  
 rundir, 424  
 set\_option, 421  
 sget, 421  
 stage, 422  
 submit\_jobs, 422  
 tempdir, 424  
 unpack\_moments, 422  
 verbose\_options, 424  
 wq\_complete, 423  
 xct, 424  
**forcebalance::nifty**  
     allsplit, 47  
     bohrang, 60  
     col, 47  
     commadash, 47  
     concurrent\_map, 47  
     CopyFile, 47  
     createWorkQueue, 48  
     destroyWorkQueue, 48  
     encode, 48  
     eqcgmx, 60  
     flat, 48  
     floatornan, 49  
     fqcgmx, 60  
     get\_least\_squares, 49  
     getWQIds, 50  
     getWorkQueue, 50  
     GoInto, 50  
     invert\_svd, 50  
     isdecimal, 50  
     isfloat, 52  
     isint, 52  
     kb, 60  
     Leave, 52  
     link\_dir\_contents, 53  
     LinkFile, 53  
     logger, 60  
     lp\_dump, 53  
     lp\_load, 53  
     MissingFileInspection, 53  
     multiopen, 54  
     onefile, 54  
     orthogonalize, 54  
     pmat2d, 54  
     printcool, 55  
     printcool\_dictionary, 55  
     pvec1d, 55  
     queue\_up, 57  
     queue\_up\_src\_dest, 57  
     remove\_if\_exists, 57  
     row, 57  
     segments, 58  
     specific\_dct, 60  
     specific\_lst, 60  
     statistical inefficiency, 58  
     uncommadash, 58  
     WORK\_QUEUE, 60  
     WQIDS, 60  
     warn\_once, 59  
     warn\_press\_key, 59  
     which, 59  
     wq\_wait, 59  
     wq\_wait1, 59  
     XMLFILE, 60  
**forcebalance::nifty::LineChunker**  
     \_\_enter\_\_, 322  
     \_\_exit\_\_, 322  
     \_\_init\_\_, 322  
     buf, 322  
     callback, 322  
     close, 322  
     nomnom, 322  
     push, 322  
**forcebalance::nifty::Pickler\_LP**  
     \_\_init\_\_, 468  
**forcebalance::nifty::Unpickler\_LP**  
     \_\_init\_\_, 518  
**forcebalance::objective**  
     Implemented\_Targets, 61  
     Letters, 61  
     logger, 61  
**forcebalance::objective::Objective**  
     \_\_init\_\_, 449  
     FF, 450  
     Full, 449  
     Indicate, 449  
     ObjDict, 450  
     ObjDict\_Last, 450  
     Penalty, 450  
     PrintOptionDict, 450  
     set\_option, 449  
     Target\_Terms, 449  
     Targets, 450  
     verbose\_options, 450  
     WTot, 450  
**forcebalance::objective::Penalty**  
     \_\_init\_\_, 465  
     a, 466

b, 466  
 compute, 465  
 FF, 467  
 FUSE, 465  
 FUSE\_BARRIER, 465  
 FUSE\_L0, 465  
 fadd, 467  
 fmul, 467  
 HYP, 466  
 L2\_norm, 466  
 Pen\_Names, 467  
 Pen\_Tab, 467  
 ptyp, 467  
 spacings, 467  
 forcebalance::openmmio  
   CopyAmoebaAngleParameters, 63  
   CopyAmoebaBondParameters, 63  
   CopyAmoebaInPlaneAngleParameters, 63  
   CopyAmoebaMultipoleParameters, 63  
   CopyAmoebaOutOfPlaneBendParameters, 63  
   CopyAmoebaVdwParameters, 63  
   CopyHarmonicAngleParameters, 64  
   CopyHarmonicBondParameters, 64  
   CopyNonbondedParameters, 64  
   CopyPeriodicTorsionParameters, 64  
   CopySystemParameters, 65  
   do\_nothing, 65  
   get\_dipole, 65  
   logger, 66  
   MTSVVRIntegrator, 65  
   pdict, 66  
   ResetVirtualSites, 65  
   SetAmoebaVirtualExclusions, 66  
   suffix\_dict, 66  
   UpdateSimulationParameters, 66  
 forcebalance::openmmio::AbInitio\_OpenMM  
   \_\_init\_\_, 153  
   AtomLists, 164  
   build\_invdist, 153  
   compute\_netforce\_torque, 153  
   e\_err, 164  
   e\_err\_pct, 164  
   e\_ref, 164  
   emd0, 164  
   energy\_force\_driver\_all, 153  
   energy\_force\_driver\_all\_external, 154  
   energy\_force\_driver\_all\_internal, 154  
   energy\_force\_transformer, 154  
   energy\_force\_transformer\_all, 154  
   eqm, 164  
   esp\_err, 164  
   espval, 164  
   espxyz, 164  
   f\_err, 164  
   f\_err\_pct, 164  
   f\_ref, 164  
   FF, 164  
   fitatoms, 164  
   force, 164  
   force\_map, 164  
   fqm, 165  
   fref, 165  
   gct, 165  
   get, 155  
   get\_G, 156  
   get\_H, 157  
   get\_X, 158  
   get\_energy\_force\_, 155  
   get\_resp\_, 158  
   hct, 165  
   indicate, 159  
   invdists, 165  
   link\_from\_tempdir, 159  
   nesp, 165  
   new\_vsites, 165  
   nf\_err, 165  
   nf\_err\_pct, 165  
   nf\_ref, 165  
   nftqm, 165  
   nnf, 165  
   nparticles, 165  
   ns, 165  
   ntq, 165  
   objective, 165  
   platform, 165  
   prepare\_temp\_directory, 159  
   PrintOptionDict, 165  
   printcool\_table, 160  
   qfnm, 165  
   qmatoms, 166  
   qmboltz\_wts, 166  
   read\_reference\_data, 160  
   read\_topology, 161  
   refresh\_temp\_directory, 161  
   respterm, 166  
   rundir, 166  
   save\_vmvalls, 166  
   set\_option, 162  
   sget, 162  
   simulation, 166  
   stage, 163  
   submit\_jobs, 163  
   tempdir, 166  
   topology\_flag, 166  
   tq\_err, 166  
   tq\_err\_pct, 166  
   tq\_ref, 166  
   traj, 166

trajfnm, 166  
 use\_nft, 166  
 verbose\_options, 166  
 w\_energy, 167  
 w\_force, 167  
 w\_netforce, 167  
 w\_resp, 167  
 w\_torque, 167  
 whamboltz, 167  
 whamboltz\_wts, 167  
 wq\_complete, 163  
 xct, 167  
 xyz\_omms, 167  
 forcebalance::openmmio::Interaction\_OpenMM  
     \_\_init\_\_, 286  
     divisor, 293  
     e\_err, 293  
     e\_err\_pct, 293  
     emm, 293  
     energy\_driver\_all, 286  
     eqm, 293  
     FF, 293  
     gct, 293  
     get, 286  
     get\_G, 286  
     get\_H, 287  
     get\_X, 288  
     hct, 293  
     indicate, 289  
     interaction\_driver\_all, 289  
     label, 293  
     link\_from\_tempdir, 289  
     ns, 293  
     objective, 293  
     platform, 293  
     prefactor, 293  
     prepare\_temp\_directory, 289  
     PrintOptionDict, 293  
     printcool\_table, 289  
     qfnm, 293  
     read\_reference\_data, 290  
     refresh\_temp\_directory, 290  
     rundir, 294  
     select1, 294  
     select2, 294  
     set\_option, 291  
     sget, 291  
     simulations, 294  
     stage, 292  
     submit\_jobs, 292  
     tempdir, 294  
     traj, 294  
     trajfnm, 294  
     verbose\_options, 294  
     weight, 294  
     wq\_complete, 292  
     xct, 294  
 forcebalance::openmmio::Liquid\_OpenMM  
     \_\_init\_\_, 349  
     do\_self\_pol, 357  
     engine, 357  
     extra\_output, 357  
     FF, 357  
     gas\_fnm, 357  
     gct, 357  
     get, 349  
     get\_G, 350  
     get\_H, 351  
     get\_X, 352  
     hct, 357  
     indicate, 353  
     Labels, 357  
     last\_traj, 357  
     link\_from\_tempdir, 353  
     liquid\_conf, 357  
     liquid\_fnm, 358  
     liquid\_traj, 358  
     MBarEnergy, 358  
     mpdb, 358  
     msim, 358  
     npt\_simulation, 353  
     nptfiles, 358  
     nptpx, 358  
     nptsfx, 358  
     objective\_term, 353  
     PhasePoints, 358  
     platform, 358  
     polarization\_correction, 354  
     prepare\_temp\_directory, 354  
     PrintOptionDict, 358  
     printcool\_table, 354  
     read\_data, 355  
     RefData, 358  
     refresh\_temp\_directory, 355  
     rundir, 358  
     SavedMVal, 358  
     SavedTraj, 358  
     set\_option, 355  
     sget, 355  
     stage, 356  
     submit\_jobs, 356  
     tempdir, 358  
     verbose\_options, 358  
     w\_alpha, 358  
     w\_cp, 359  
     w\_eps0, 359  
     w\_hvap, 359  
     w\_kappa, 359

```

w_rho, 359
wq_complete, 357
xct, 359
forcebalance::openmmio::OpenMM_Reader
    __init__, 452
    adict, 453
    AtomTypes, 453
    build_pid, 452
    feed, 453
    itype, 454
    In, 454
    molatom, 454
    Molecules, 454
    pdict, 454
    Split, 453
    suffix, 454
    Whites, 453
forcebalance::optimizer
    Counter, 67
    GOODSTEP, 67
    GoodStep, 67
    ITERATION_NUMBER, 67
    logger, 67
forcebalance::optimizer::Optimizer
    __init__, 457
    Anneal, 457
    BFGS, 457
    bhyp, 462
    chk, 462
    ConjugateGradient, 457
    dx, 463
    excision, 463
    FDCheckG, 457
    FDCheckH, 458
    FF, 463
    GeneticAlgorithm, 458
    Grad, 463
    Gradient, 458
    H, 463
    Hess, 463
    Hessian, 459
    MainOptimizer, 459
    mvals0, 463
    NewtonRaphson, 459
    np, 463
    Objective, 463
    OptTab, 463
    Penalty, 463
    Powell, 460
    PrintOptionDict, 463
    readchk, 460
    Run, 460
    Scan_Values, 460
    ScanMVals, 461
    ScanPVals, 461
    ScipyOptimizer, 461
    set_option, 461
    Simplex, 462
    SinglePoint, 462
    step, 462
    Val, 463
    verbose_options, 463
    writechk, 462
forcebalance::output::CleanFileHandler
    emit, 212
forcebalance::output::CleanStreamHandler
    __init__, 213
    emit, 214
forcebalance::output::ForceBalanceLogger
    __init__, 240
    addHandler, 240
    defaultHandler, 240
    removeHandler, 240
forcebalance::output::RawFileHandler
    emit, 472
forcebalance::output::RawStreamHandler
    __init__, 473
    emit, 474
forcebalance::parser
    all_opts_names, 71
    bkwrd, 71
    gen_opts_defaults, 71
    gen_opts_types, 71
    iocc, 72
    logger, 72
    mainsections, 72
    ParsTab, 72
    parse_inputs, 70
    printsection, 70
    read_internals, 70
    read_mvals, 71
    read_priors, 71
    read_pvals, 71
    subdict, 72
    tgt_opts_defaults, 72
    tgt_opts_types, 72
forcebalance::psi4io
    logger, 73
forcebalance::psi4io::GBS_Reader
    __init__, 245
    adict, 246
    amom, 246
    AtomTypes, 246
    basis_number, 247
    build_pid, 245
    contraction_number, 247
    destroy, 247
    element, 247

```

feed, 245, 246  
 isdata, 247  
 itype, 247  
 last\_amom, 247  
 ln, 247  
 molatom, 247  
 Molecules, 247  
 pdict, 247  
 Split, 246  
 suffix, 247  
 Whites, 246  
**forcebalance::psi4io::Grid\_Reader**  
 \_\_init\_\_, 257  
 adict, 258  
 AtomTypes, 258  
 build\_pid, 257  
 element, 258  
 feed, 257  
 isdata, 258  
 itype, 258  
 ln, 258  
 molatom, 258  
 Molecules, 259  
 pdict, 259  
 point, 259  
 radii, 259  
 Split, 258  
 suffix, 259  
 Whites, 258  
**forcebalance::psi4io::RDVR3\_Psi4**  
 \_\_init\_\_, 476  
 bidirect, 483  
 callderivs, 483  
 driver, 477  
 elements, 483  
 FF, 483  
 factor, 483  
 gct, 483  
 get, 477  
 get\_G, 477  
 get\_H, 478  
 get\_X, 479  
 gradd, 483  
 hct, 483  
 hdiagd, 483  
 indicate, 480  
 link\_from\_tempdir, 480  
 molecules, 483  
 objd, 483  
 objective, 483  
 objfiles, 483  
 objvals, 483  
 PrintOptionDict, 483  
 printcool\_table, 480  
 refresh\_temp\_directory, 481  
 rundir, 483  
 set\_option, 481  
 sget, 481  
 stage, 482  
 submit\_jobs, 482  
 tdir, 483  
 tempdir, 484  
 verbose\_options, 484  
 wq\_complete, 482  
 xct, 484  
**forcebalance::psi4io::THCDF\_Psi4**  
 \_\_init\_\_, 505  
 call\_derivatives, 512  
 D, 512  
 DATfnm, 512  
 DF\_Energy, 512  
 driver, 506  
 Elements, 512  
 FF, 512  
 GBSfnm, 512  
 gct, 512  
 get, 506  
 get\_G, 506  
 get\_H, 507  
 get\_X, 508  
 hct, 512  
 indicate, 509  
 link\_from\_tempdir, 509  
 MAQ, 513  
 MP2\_Energy, 513  
 Molecules, 513  
 objective, 513  
 prepare\_temp\_directory, 509  
 PrintOptionDict, 513  
 printcool\_table, 509  
 refresh\_temp\_directory, 510  
 rundir, 513  
 set\_option, 510  
 sget, 510  
 stage, 511  
 submit\_jobs, 511  
 tempdir, 513  
 throw\_outs, 513  
 verbose\_options, 513  
 wq\_complete, 511  
 write\_nested\_destroy, 512  
 xct, 513  
**forcebalance::qchemio**  
 logger, 74  
 ndtypes, 74  
 pdict, 74  
 QChem\_Dielectric\_Energy, 74  
**forcebalance::qchemio::QCIn\_Reader**

`__init__`, 470  
`adict`, 471  
`atom`, 471  
`AtomTypes`, 471  
`build_pid`, 470  
`cnum`, 471  
`feed`, 470  
`itype`, 471  
`In`, 471  
`molatom`, 471  
`Molecules`, 471  
`pdict`, 471  
`sec`, 471  
`shell`, 471  
`snum`, 471  
`Split`, 470  
`suffix`, 471  
`Whites`, 470  
`forcebalance::target`  
    `logger`, 75  
`forcebalance::target::RemoteTarget`  
    `__init__`, 486  
    `FF`, 492  
    `gct`, 492  
    `get`, 486  
    `get.G`, 486  
    `get.H`, 487  
    `get.X`, 488  
    `hct`, 492  
    `id_string`, 492  
    `indicate`, 489  
    `link_from_tempdir`, 489  
    `PrintOptionDict`, 492  
    `printcool.table`, 489  
    `r_options`, 492  
    `r_tgt_opts`, 492  
    `refresh_temp_directory`, 490  
    `remote_indicate`, 492  
    `rundir`, 492  
    `set_option`, 490  
    `sget`, 490  
    `stage`, 491  
    `submit_jobs`, 491  
    `tempdir`, 492  
    `verbose_options`, 493  
    `wq_complete`, 492  
    `xct`, 493  
`forcebalance::target::Target`  
    `__init__`, 495  
    `FF`, 502  
    `gct`, 502  
    `get`, 496  
    `get.G`, 496  
    `get.H`, 497  
`get_X`, 498  
`hct`, 502  
`link_from_tempdir`, 499  
`PrintOptionDict`, 502  
`printcool.table`, 499  
`refresh_temp_directory`, 499  
`rundir`, 502  
`set_option`, 500  
`sget`, 500  
`stage`, 501  
`submit_jobs`, 501  
`tempdir`, 502  
`verbose_options`, 502  
`wq_complete`, 501  
`xct`, 502  
`forcebalance::tinkerio`  
    `logger`, 76  
    `modify_key`, 76  
    `pdict`, 76  
    `write_key_with_prm`, 76  
`forcebalance::tinkerio::AbInitio_TINKER`  
    `__init__`, 172  
    `all_at_once`, 183  
    `AtomLists`, 183  
    `build_invdist`, 172  
    `compute_netforce_torque`, 172  
    `e_err`, 183  
    `e_err_pct`, 183  
    `e_ref`, 183  
    `emd0`, 183  
    `energy_driver_all`, 172  
    `energy_force_driver`, 173  
    `energy_force_driver_all`, 173  
    `energy_force_transformer`, 173  
    `energy_force_transformer_all`, 173  
    `eqm`, 183  
    `esp_err`, 183  
    `espval`, 183  
    `espxyz`, 183  
    `f_err`, 183  
    `f_err_pct`, 183  
    `f_ref`, 183  
    `FF`, 183  
    `fitatoms`, 183  
    `force`, 184  
    `force_map`, 184  
    `fqm`, 184  
    `fref`, 184  
    `gct`, 184  
    `get`, 174  
    `get_G`, 175  
    `get_H`, 176  
    `get_X`, 177  
    `get_energy_force_`, 174

get\_resp\_, 177  
 hct, 184  
 indicate, 178  
 invdists, 184  
 link\_from\_tempdir, 178  
 nesp, 184  
 new\_vsites, 184  
 nf\_err, 184  
 nf\_err\_pct, 184  
 nf\_ref, 184  
 nftqm, 184  
 nnf, 184  
 nparticles, 184  
 ns, 184  
 ntq, 184  
 objective, 184  
 prepare\_temp\_directory, 178  
 PrintOptionDict, 184  
 printcool\_table, 179  
 qfnm, 184  
 qmatoms, 185  
 qmboltz\_wts, 185  
 read\_reference\_data, 179  
 read\_topology, 180  
 refresh\_temp\_directory, 180  
 respterm, 185  
 rundir, 185  
 save\_vmvalls, 185  
 set\_option, 181  
 sget, 181  
 stage, 182  
 submit\_jobs, 182  
 tempdir, 185  
 topology\_flag, 185  
 tq\_err, 185  
 tq\_err\_pct, 185  
 tq\_ref, 185  
 traj, 185  
 trajfnm, 185  
 use\_nft, 185  
 verbose\_options, 185  
 w\_energy, 185  
 w\_force, 185  
 w\_netforce, 186  
 w\_resp, 186  
 w\_torque, 186  
 whamboltz, 186  
 whamboltz\_wts, 186  
 wq\_complete, 182  
 xct, 186  
**forcebalance::tinkerio::BindingEnergy\_TINKER**  
 \_\_init\_\_, 204  
 energy\_part, 210  
 FF, 210  
 gct, 210  
 get, 205  
 get\_G, 205  
 get\_H, 205  
 get\_X, 206  
 hct, 210  
 indicate, 207  
 inter\_opts, 210  
 link\_from\_tempdir, 207  
 objective, 210  
 optprog, 210  
 prepare\_temp\_directory, 207  
 PrintDict, 210  
 PrintOptionDict, 210  
 printcool\_table, 207  
 RMSDDict, 210  
 refresh\_temp\_directory, 208  
 rmsd\_part, 210  
 rundir, 210  
 set\_option, 208  
 sget, 208  
 stage, 209  
 submit\_jobs, 209  
 system\_driver, 209  
 tempdir, 211  
 verbose\_options, 211  
 wq\_complete, 210  
 xct, 211  
**forcebalance::tinkerio::Interaction\_TINKER**  
 \_\_init\_\_, 298  
 divisor, 305  
 e\_err, 305  
 e\_err\_pct, 305  
 emm, 305  
 energy\_driver\_all, 298  
 eqm, 305  
 FF, 305  
 gct, 305  
 get, 298  
 get\_G, 298  
 get\_H, 299  
 get\_X, 300  
 hct, 305  
 indicate, 301  
 interaction\_driver\_all, 301  
 label, 305  
 link\_from\_tempdir, 301  
 ns, 305  
 objective, 305  
 prefactor, 305  
 prepare\_temp\_directory, 301  
 PrintOptionDict, 305  
 printcool\_table, 301  
 qfnm, 305

```

read_reference_data, 302
refresh_temp_directory, 302
rundir, 305
select1, 306
select2, 306
set_option, 303
sget, 303
stage, 304
submit_jobs, 304
tempdir, 306
traj, 306
trajfnm, 306
verbose_options, 306
weight, 306
wq_complete, 304
xct, 306
forcebalance::tinkerio::Liquid_TINKER
    __init__, 363
    do_self_pol, 370
    DynDict, 370
    DynDict_New, 370
    extra_output, 370
    FF, 370
    gct, 370
    get, 363
    get_G, 364
    get_H, 364
    get_X, 365
    hct, 370
    indicate, 366
    Labels, 370
    last_traj, 370
    link_from_tempdir, 366
    MBarEnergy, 370
    npt_simulation, 366
    nptfiles, 371
    nptpx, 371
    nptsfx, 371
    objective_term, 366
    PhasePoints, 371
    polarization_correction, 367
    prepare_temp_directory, 367
    PrintOptionDict, 371
    printcool_table, 367
    read_data, 368
    RefData, 371
    refresh_temp_directory, 368
    rundir, 371
    SavedMVal, 371
    SavedTraj, 371
    set_option, 368
    sget, 368
    stage, 369
    submit_jobs, 369
tempdir, 371
verbose_options, 371
w_alpha, 371
w_cp, 371
w_eps0, 371
w_hvap, 371
w_kappa, 371
w_rho, 371
wq_complete, 370
xct, 372
forcebalance::tinkerio::Moments_TINKER
    __init__, 428
    calc_moments, 435
    denoms, 435
    FF, 435
    gct, 435
    get, 428
    get_G, 428
    get_H, 429
    get_X, 430
    hct, 435
    indicate, 431
    link_from_tempdir, 431
    mfnm, 435
    moments_driver, 431
    na, 435
    objective, 435
    prepare_temp_directory, 432
    PrintOptionDict, 435
    printcool_table, 432
    read_reference_data, 432
    ref_eigvals, 435
    ref_eigvecs, 436
    ref_moments, 436
    refresh_temp_directory, 433
    rundir, 436
    set_option, 433
    sget, 433
    stage, 434
    submit_jobs, 434
    tempdir, 436
    unpack_moments, 434
    verbose_options, 436
    wq_complete, 435
    xct, 436
forcebalance::tinkerio::Tinker_Reader
    __init__, 515
    adict, 516
    atom, 516
    AtomTypes, 516
    build_pid, 515
    feed, 515
    itype, 516
    In, 516

```

molatom, 516  
 Molecules, 516  
 pdict, 517  
 Split, 516  
 suffix, 517  
 Whites, 516  
 forcebalance::tinkerio::Vibration\_TINKER  
     \_\_init\_\_, 529  
     prepare\_temp\_directory, 529  
     vibration\_driver, 529  
 forcebalance::vibration  
     logger, 77  
 forcebalance::vibration::Vibration  
     \_\_init\_\_, 520  
     calc\_eigvals, 527  
     FF, 527  
     gct, 527  
     get, 521  
     get\_G, 521  
     get\_H, 522  
     get\_X, 523  
     hct, 527  
     indicate, 524  
     link\_from\_tempdir, 524  
     na, 527  
     objective, 527  
     prepare\_temp\_directory, 524  
     PrintOptionDict, 527  
     printcool\_table, 524  
     read\_reference\_data, 525  
     ref\_eigvals, 527  
     ref\_eigvecs, 527  
     refresh\_temp\_directory, 525  
     rundir, 527  
     set\_option, 525  
     sget, 525  
     stage, 526  
     submit\_jobs, 526  
     tempdir, 527  
     verbose\_options, 527  
     vfnm, 527  
     wq\_complete, 526  
     xct, 528  
 forcefield.py, 534  
 format\_gro\_box  
     forcebalance::molecule, 38  
 format\_gro\_coord  
     forcebalance::molecule, 38  
 format\_xyz\_coord  
     forcebalance::molecule, 39  
 format\_xyzgen\_coord  
     forcebalance::molecule, 39  
 fout  
     forcebalance::molecule::Molecule, 410

fqcgmx  
     forcebalance::nifty, 60  
 fqm  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
 FrameVariableNames  
     forcebalance::molecule, 43  
 frmod\_pdict  
     forcebalance::amberio, 15  
 fref  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
 Full  
     forcebalance::objective::Objective, 449  
 Funnel  
     forcebalance::molecule::Molecule, 411  
 GBSfnm  
     forcebalance::psi4io::THCDF\_Psi4, 512  
 GOODSTEP  
     forcebalance::optimizer, 67  
 gas\_fnm  
     forcebalance::gmxio::Liquid\_GMX, 344  
     forcebalance::openmmio::Liquid\_OpenMM, 357  
 gct  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::binding::BindingEnergy, 201  
     forcebalance::counterpoise::Counterpoise, 223  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::gmxio::Interaction\_GMX, 281  
     forcebalance::gmxio::Liquid\_GMX, 344  
     forcebalance::gmxpqio::Monomer\_QTPIE, 446  
     forcebalance::interaction::Interaction, 269  
     forcebalance::leastsq::LeastSquares, 320  
     forcebalance::liquid::Liquid, 332  
     forcebalance::moments::Moments, 423  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::openmmio::Interaction\_OpenMM, 293  
     forcebalance::openmmio::Liquid\_OpenMM, 357  
     forcebalance::psi4io::RDVR3\_Psi4, 483  
     forcebalance::psi4io::THCDF\_Psi4, 512  
     forcebalance::target::RemoteTarget, 492  
     forcebalance::target::Target, 502  
     forcebalance::tinkerio::AbInitio\_TINKER, 184

forcebalance::tinkerio::BindingEnergy\_TINKER, 210  
 forcebalance::tinkerio::Interaction\_TINKER, 305  
 forcebalance::tinkerio::Liquid\_TINKER, 370  
 forcebalance::tinkerio::Moments\_TINKER, 435  
 forcebalance::vibration::Vibration, 527  
 gen\_opts\_defaults  
     forcebalance::parser, 71  
 gen\_opts\_types  
     forcebalance::parser, 71  
 generate\_vsite\_positions  
     forcebalance::gmxio::AbInitio\_GMX, 118  
     forcebalance::gmxio::GMX, 254  
 GeneticAlgorithm  
     forcebalance::optimizer::Optimizer, 458  
 get  
     forcebalance::abinitio::AbInitio, 83  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 137  
     forcebalance::amberio::AbInitio\_AMBER, 100  
     forcebalance::binding::BindingEnergy, 195  
     forcebalance::counterpoise::Counterpoise, 217  
     forcebalance::gmxio::AbInitio\_GMX, 119  
     forcebalance::gmxio::Interaction\_GMX, 274  
     forcebalance::gmxio::Liquid\_GMX, 337  
     forcebalance::gmxqpio::Monomer\_QTPIE, 439  
     forcebalance::interaction::Interaction, 262  
     forcebalance::leastsq::LeastSquares, 313  
     forcebalance::liquid::Liquid, 325  
     forcebalance::moments::Moments, 416  
     forcebalance::openmmio::AbInitio\_OpenMM, 155  
     forcebalance::openmmio::Interaction\_OpenMM, 286  
     forcebalance::openmmio::Liquid\_OpenMM, 349  
     forcebalance::psi4io::RDVR3\_Psi4, 477  
     forcebalance::psi4io::THCDF\_Psi4, 506  
     forcebalance::target::RemoteTarget, 486  
     forcebalance::target::Target, 496  
     forcebalance::tinkerio::AbInitio\_TINKER, 174  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 205  
     forcebalance::tinkerio::Interaction\_TINKER, 298  
     forcebalance::tinkerio::Liquid\_TINKER, 363  
     forcebalance::tinkerio::Moments\_TINKER, 428  
     forcebalance::vibration::Vibration, 521  
 get\_G  
     forcebalance::abinitio::AbInitio, 84  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 138  
     forcebalance::amberio::AbInitio\_AMBER, 102  
     forcebalance::binding::BindingEnergy, 195  
     forcebalance::counterpoise::Counterpoise, 217  
     forcebalance::gmxio::AbInitio\_GMX, 120  
     forcebalance::gmxio::Interaction\_GMX, 274  
     forcebalance::gmxio::Liquid\_GMX, 338  
     forcebalance::gmxqpio::Monomer\_QTPIE, 439  
     forcebalance::interaction::Interaction, 262  
     forcebalance::leastsq::LeastSquares, 314  
     forcebalance::liquid::Liquid, 326  
 get\_H  
     forcebalance::abinitio::AbInitio, 85  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 139  
     forcebalance::amberio::AbInitio\_AMBER, 102  
     forcebalance::binding::BindingEnergy, 196  
     forcebalance::counterpoise::Counterpoise, 218  
     forcebalance::gmxio::AbInitio\_GMX, 121  
     forcebalance::gmxio::Interaction\_GMX, 275  
     forcebalance::gmxio::Liquid\_GMX, 338  
     forcebalance::gmxqpio::Monomer\_QTPIE, 440  
     forcebalance::interaction::Interaction, 263  
     forcebalance::leastsq::LeastSquares, 315  
     forcebalance::liquid::Liquid, 326  
     forcebalance::moments::Moments, 417  
     forcebalance::openmmio::AbInitio\_OpenMM, 157  
     forcebalance::openmmio::Interaction\_OpenMM, 287  
     forcebalance::openmmio::Liquid\_OpenMM, 351  
     forcebalance::psi4io::RDVR3\_Psi4, 478  
     forcebalance::psi4io::THCDF\_Psi4, 507  
     forcebalance::target::RemoteTarget, 487  
     forcebalance::target::Target, 497  
     forcebalance::tinkerio::AbInitio\_TINKER, 176  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 205  
     forcebalance::tinkerio::Interaction\_TINKER, 299  
     forcebalance::tinkerio::Liquid\_TINKER, 364  
     forcebalance::tinkerio::Moments\_TINKER, 429  
     forcebalance::vibration::Vibration, 522  
 get\_X  
     forcebalance::abinitio::AbInitio, 86  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 140  
     forcebalance::amberio::AbInitio\_AMBER, 103  
     forcebalance::binding::BindingEnergy, 197  
     forcebalance::counterpoise::Counterpoise, 219  
     forcebalance::gmxio::AbInitio\_GMX, 122  
     forcebalance::gmxio::Interaction\_GMX, 276  
     forcebalance::gmxio::Liquid\_GMX, 339  
     forcebalance::gmxqpio::Monomer\_QTPIE, 441  
     forcebalance::interaction::Interaction, 264  
     forcebalance::leastsq::LeastSquares, 316  
     forcebalance::liquid::Liquid, 327

forcebalance::moments::Moments, 418  
 forcebalance::openmmio::AbInitio\_OpenMM, 158  
 forcebalance::openmmio::Interaction\_OpenMM, 288  
 forcebalance::openmmio::Liquid\_OpenMM, 352  
 forcebalance::psi4io::RDVR3\_Psi4, 479  
 forcebalance::psi4io::THCDF\_Psi4, 508  
 forcebalance::target::RemoteTarget, 488  
 forcebalance::target::Target, 498  
 forcebalance::tinkerio::AbInitio\_TINKER, 177  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 206  
 forcebalance::tinkerio::Interaction\_TINKER, 300  
 forcebalance::tinkerio::Liquid\_TINKER, 365  
 forcebalance::tinkerio::Moments\_TINKER, 430  
 forcebalance::vibration::Vibration, 523  
**get\_atom**  
 forcebalance::Mol2::mol2, 373  
**get\_bonded\_atoms**  
 forcebalance::Mol2::mol2, 374  
**get\_dipole**  
 forcebalance::openmmio, 65  
**get\_energy\_force**  
 forcebalance::abinitio::AbInitio, 83  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 137  
 forcebalance::amberio::AbInitio\_AMBER, 101  
 forcebalance::gmxio::AbInitio\_GMX, 119  
 forcebalance::openmmio::AbInitio\_OpenMM, 155  
 forcebalance::tinkerio::AbInitio\_TINKER, 174  
**get\_least\_squares**  
 forcebalance::nifty, 49  
**get\_monomer\_properties**  
 forcebalance::gmxqpio, 32  
**get\_resp**  
 forcebalance::abinitio::AbInitio, 86  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 140  
 forcebalance::amberio::AbInitio\_AMBER, 103  
 forcebalance::gmxio::AbInitio\_GMX, 122  
 forcebalance::openmmio::AbInitio\_OpenMM, 158  
 forcebalance::tinkerio::AbInitio\_TINKER, 177  
**get\_rotate\_translate**  
 forcebalance::molecule, 40  
**getElement**  
 forcebalance::molecule, 40  
**getWQIds**  
 forcebalance::nifty, 50  
**getWorkQueue**  
 forcebalance::nifty, 50  
 gmxio.py, 534  
 gmxpath  
 forcebalance::gmxio::GMX, 255  
 gmxqpio.py, 535  
 gmxsuffix  
 forcebalance::gmxio::GMX, 255  
**GolInt**  
 forcebalance::nifty, 50  
**GoodStep**  
 forcebalance::optimizer, 67  
**Grad**  
 forcebalance::optimizer::Optimizer, 463  
**gradd**  
 forcebalance::psi4io::RDVR3\_Psi4, 483  
**Gradient**  
 forcebalance::optimizer::Optimizer, 458  
**grouper**  
 forcebalance::molecule, 40  
**H**  
 forcebalance::optimizer::Optimizer, 463  
**HYP**  
 forcebalance::objective::Penalty, 466  
**hct**  
 forcebalance::abinitio::AbInitio, 92  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
 forcebalance::amberio::AbInitio\_AMBER, 110  
 forcebalance::binding::BindingEnergy, 201  
 forcebalance::counterpoise::Counterpoise, 223  
 forcebalance::gmxio::AbInitio\_GMX, 129  
 forcebalance::gmxio::Interaction\_GMX, 281  
 forcebalance::gmxio::Liquid\_GMX, 344  
 forcebalance::gmxqpio::Monomer\_QTPIE, 446  
 forcebalance::interaction::Interaction, 269  
 forcebalance::leastsq::LeastSquares, 320  
 forcebalance::liquid::Liquid, 332  
 forcebalance::moments::Moments, 423  
 forcebalance::openmmio::AbInitio\_OpenMM, 165  
 forcebalance::openmmio::Interaction\_OpenMM, 293  
 forcebalance::openmmio::Liquid\_OpenMM, 357  
 forcebalance::psi4io::RDVR3\_Psi4, 483  
 forcebalance::psi4io::THCDF\_Psi4, 512  
 forcebalance::target::RemoteTarget, 492  
 forcebalance::target::Target, 502  
 forcebalance::tinkerio::AbInitio\_TINKER, 184  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 210  
 forcebalance::tinkerio::Interaction\_TINKER, 305  
 forcebalance::tinkerio::Liquid\_TINKER, 370  
 forcebalance::tinkerio::Moments\_TINKER, 435  
 forcebalance::vibration::Vibration, 527  
**hdiadg**  
 forcebalance::psi4io::RDVR3\_Psi4, 483  
**Hess**  
 forcebalance::optimizer::Optimizer, 463  
**Hessian**  
 forcebalance::optimizer::Optimizer, 459  
**ITERATION\_NUMBER**  
 forcebalance::optimizer, 67  
**id\_string**  
 forcebalance::target::RemoteTarget, 492  
**Implemented\_TTargets**

```

        forcebalance::objective, 61
in_fd
        forcebalance::finite_difference, 25
Indicate
        forcebalance::objective::Objective, 449
indicate
        forcebalance::abinitio::AbInitio, 87
        forcebalance::abinitio_internal::AbInitio_Internal, 141
        forcebalance::amberio::AbInitio_AMBER, 104
        forcebalance::binding::BindingEnergy, 198
        forcebalance::gmxio::AbInitio_GMX, 123
        forcebalance::gmxio::Interaction_GMX, 277
        forcebalance::gmxio::Liquid_GMX, 340
        forcebalance::gmxqpio::Monomer_QTPIE, 442
        forcebalance::interaction::Interaction, 265
        forcebalance::leastsq::LeastSquares, 317
        forcebalance::liquid::Liquid, 328
        forcebalance::moments::Moments, 419
        forcebalance::openmmio::AbInitio_OpenMM, 159
        forcebalance::openmmio::Interaction_OpenMM, 289
        forcebalance::openmmio::Liquid_OpenMM, 353
        forcebalance::psi4io::RDVR3_Psi4, 480
        forcebalance::psi4io::THCDF_Psi4, 509
        forcebalance::target::RemoteTarget, 489
forcebalance::tinkerio::AbInitio_TINKER, 178
forcebalance::tinkerio::BindingEnergy_TINKER, 207
forcebalance::tinkerio::Interaction_TINKER, 301
forcebalance::tinkerio::Liquid_TINKER, 366
forcebalance::tinkerio::Moments_TINKER, 431
forcebalance::vibration::Vibration, 524
inter_opts
        forcebalance::binding::BindingEnergy, 201
        forcebalance::tinkerio::BindingEnergy_TINKER, 210
interaction.py, 536
interaction_driver
        forcebalance::gmxio::Interaction_GMX, 277
interaction_driver_all
        forcebalance::gmxio::Interaction_GMX, 277
        forcebalance::openmmio::Interaction_OpenMM, 289
        forcebalance::tinkerio::Interaction_TINKER, 301
invdists
        forcebalance::abinitio::AbInitio, 92
        forcebalance::abinitio_internal::AbInitio_Internal, 146
        forcebalance::amberio::AbInitio_AMBER, 110
        forcebalance::gmxio::AbInitio_GMX, 129
        forcebalance::openmmio::AbInitio_OpenMM, 165
        forcebalance::tinkerio::AbInitio_TINKER, 184
invert_svd
        forcebalance::nifty, 50
iocc
        forcebalance::parser, 72
is_charmm_coord
        forcebalance::molecule, 40
is_gro_box
        forcebalance::molecule, 41
is_gro_coord
        forcebalance::molecule, 41
is_mol2_atom
        forcebalance::amberio, 15
isdata
        forcebalance::psi4io::GBS_Reader, 247
        forcebalance::psi4io::Grid_Reader, 258
isdecimal
        forcebalance::nifty, 50
isfloat
        forcebalance::molecule, 41
        forcebalance::nifty, 52
isint
        forcebalance::molecule, 42
        forcebalance::nifty, 52
itype
        forcebalance::amberio::FrcMod_Reader, 243
        forcebalance::amberio::Mol2_Reader, 387
        forcebalance::BaseReader, 192
        forcebalance::custom_io::Gen_Reader, 250
        forcebalance::gmxio::ITP_Reader, 310
        forcebalance::mol2io::Mol2_Reader, 390
        forcebalance::openmmio::OpenMM_Reader, 454
        forcebalance::psi4io::GBS_Reader, 247
        forcebalance::psi4io::Grid_Reader, 258
        forcebalance::qchemio::QCIn_Reader, 471
        forcebalance::tinkerio::Tinker_Reader, 516
kb
        forcebalance::nifty, 60
L
        forcebalance::chemistry, 18
L2_norm
        forcebalance::objective::Penalty, 466
LAST_MVALS
        forcebalance::leastsq, 33
label
        forcebalance::gmxio::Interaction_GMX, 281
        forcebalance::interaction::Interaction, 269
        forcebalance::openmmio::Interaction_OpenMM, 293
        forcebalance::tinkerio::Interaction_TINKER, 305
Labels
        forcebalance::gmxio::Liquid_GMX, 344
        forcebalance::liquid::Liquid, 332
        forcebalance::openmmio::Liquid_OpenMM, 357
        forcebalance::tinkerio::Liquid_TINKER, 370
last_amom
        forcebalance::psi4io::GBS_Reader, 247
last_traj
        forcebalance::gmxio::Liquid_GMX, 344
        forcebalance::liquid::Liquid, 332
        forcebalance::openmmio::Liquid_OpenMM, 357

```

forcebalance::tinkerio::Liquid\_TINKER, 370  
**LastMvals**  
 forcebalance::leastsq, 33  
**leastsq.py**, 536  
**Leave**  
 forcebalance::nifty, 52  
**Letters**  
 forcebalance::objective, 61  
**line**  
 forcebalance::chemistry, 18  
**linedestroy\_save**  
 forcebalance::forcefield::FF, 237  
**linedestroy\_this**  
 forcebalance::forcefield::FF, 237  
**link\_dir\_contents**  
 forcebalance::nifty, 53  
**link\_from\_tempdir**  
 forcebalance::abinitio::AbInitio, 87  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 141  
 forcebalance::amberio::AbInitio\_AMBER, 104  
 forcebalance::binding::BindingEnergy, 198  
 forcebalance::counterpoise::Counterpoise, 220  
 forcebalance::gmxio::AbInitio\_GMX, 123  
 forcebalance::gmxio::Interaction\_GMX, 278  
 forcebalance::gmxio::Liquid\_GMX, 340  
 forcebalance::gmxqpio::Monomer\_QTPIE, 442  
 forcebalance::interaction::Interaction, 265  
 forcebalance::leastsq::LeastSquares, 317  
 forcebalance::liquid::Liquid, 328  
 forcebalance::moments::Moments, 419  
 forcebalance::openmmio::AbInitio\_OpenMM, 159  
 forcebalance::openmmio::Interaction\_OpenMM, 289  
 forcebalance::openmmio::Liquid\_OpenMM, 353  
 forcebalance::psi4io::RDVR3\_Psi4, 480  
 forcebalance::psi4io::THCDF\_Psi4, 509  
 forcebalance::target::RemoteTarget, 489  
 forcebalance::target::Target, 499  
 forcebalance::tinkerio::AbInitio\_TINKER, 178  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 207  
 forcebalance::tinkerio::Interaction\_TINKER, 301  
 forcebalance::tinkerio::Liquid\_TINKER, 366  
 forcebalance::tinkerio::Moments\_TINKER, 431  
 forcebalance::vibration::Vibration, 524  
**LinkFile**  
 forcebalance::nifty, 53  
**liquid.py**, 536  
**liquid\_conf**  
 forcebalance::gmxio::Liquid\_GMX, 344  
 forcebalance::openmmio::Liquid\_OpenMM, 357  
**liquid\_fnm**  
 forcebalance::gmxio::Liquid\_GMX, 344  
 forcebalance::openmmio::Liquid\_OpenMM, 358  
**liquid\_traj**  
 forcebalance::gmxio::Liquid\_GMX, 345  
 forcebalance::openmmio::Liquid\_OpenMM, 358  
**list\_map**  
 forcebalance::forcefield::FF, 234  
**In**  
 forcebalance::amberio::FrcMod\_Reader, 243  
 forcebalance::amberio::Mol2\_Reader, 387  
 forcebalance::BaseReader, 192  
 forcebalance::custom\_io::Gen\_Reader, 250  
 forcebalance::gmxio::ITP\_Reader, 310  
 forcebalance::mol2io::Mol2\_Reader, 390  
 forcebalance::openmmio::OpenMM\_Reader, 454  
 forcebalance::psi4io::GBS\_Reader, 247  
 forcebalance::psi4io::Grid\_Reader, 258  
 forcebalance::qchemio::QCIn\_Reader, 471  
 forcebalance::tinkerio::Tinker\_Reader, 516  
**load\_cp**  
 forcebalance::counterpoise::Counterpoise, 220  
**load\_frames**  
 forcebalance::molecule::Molecule, 400  
**loadxyz**  
 forcebalance::counterpoise::Counterpoise, 220  
**logger**  
 forcebalance::abinitio, 14  
 forcebalance::amberio, 15  
 forcebalance::binding, 16  
 forcebalance::counterpoise, 20  
 forcebalance::engine, 21  
 forcebalance::finite\_difference, 25  
 forcebalance::forcefield, 28  
 forcebalance::gmxio, 31  
 forcebalance::gmxqpio, 32  
 forcebalance::interaction, 32  
 forcebalance::leastsq, 33  
 forcebalance::liquid, 34  
 forcebalance::moments, 44  
 forcebalance::nifty, 60  
 forcebalance::objective, 61  
 forcebalance::openmmio, 66  
 forcebalance::optimizer, 67  
 forcebalance::parser, 72  
 forcebalance::psi4io, 73  
 forcebalance::qchemio, 74  
 forcebalance::target, 75  
 forcebalance::tinkerio, 76  
 forcebalance::vibration, 77  
**LookupByMass**  
 forcebalance::chemistry, 17  
**lp\_dump**  
 forcebalance::nifty, 53  
**lp\_load**  
 forcebalance::nifty, 53  
**MAQ**  
 forcebalance::leastsq::LeastSquares, 320

forcebalance::psi4io::THCDF\_Psi4, 513  
**MBarEnergy**  
 forcebalance::gmxio::Liquid\_GMX, 345  
 forcebalance::liquid::Liquid, 332  
 forcebalance::openmmio::Liquid\_OpenMM, 358  
 forcebalance::tinkerio::Liquid\_TINKER, 370  
**MP2\_Energy**  
 forcebalance::psi4io::THCDF\_Psi4, 513  
**MTSVVVRIntegrator**  
 forcebalance::openmmio, 65  
**main**  
 forcebalance::molecule, 42  
**MainOptimizer**  
 forcebalance::optimizer::Optimizer, 459  
**mainsections**  
 forcebalance::parser, 72  
**make**  
 forcebalance::forcefield::FF, 235  
**make\_redirect**  
 forcebalance::forcefield::FF, 235  
**map**  
 forcebalance::forcefield::FF, 237  
**mdp**  
 forcebalance::gmxio::GMX, 255  
**measure\_dihedrals**  
 forcebalance::molecule::Molecule, 400  
**MetaVariableNames**  
 forcebalance::molecule, 43  
**mfnm**  
 forcebalance::moments::Moments, 423  
 forcebalance::tinkerio::Moments\_TINKER, 435  
**MissingFileInspection**  
 forcebalance::nifty, 53  
**mktransmat**  
 forcebalance::forcefield::FF, 235  
**modify\_key**  
 forcebalance::tinkerio, 76  
**mol**  
 forcebalance::amberio::Mol2\_Reader, 387  
 forcebalance::gmxio::GMX, 255  
 forcebalance::gmxio::ITP\_Reader, 310  
**Mol2.py**, 537  
**mol2\_pdct**  
 forcebalance::amberio, 16  
 forcebalance::mol2io, 34  
**mol2io.py**, 537  
**mol\_name**  
 forcebalance::Mol2::mol2, 377  
**mol\_type**  
 forcebalance::Mol2::mol2, 377  
**molatom**  
 forcebalance::amberio::FrcMod\_Reader, 243  
 forcebalance::amberio::Mol2\_Reader, 387  
 forcebalance::BaseReader, 193  
 forcebalance::custom\_io::Gen\_Reader, 250  
 forcebalance::gmxio::ITP\_Reader, 310  
 forcebalance::mol2io::Mol2\_Reader, 390  
 forcebalance::openmmio::OpenMM\_Reader, 454  
 forcebalance::psi4io::GBS\_Reader, 247  
 forcebalance::psi4io::Grid\_Reader, 258  
 forcebalance::qchemio::QCIn\_Reader, 471  
 forcebalance::tinkerio::Tinker\_Reader, 516  
**molecule.py**, 538  
**Molecules**  
 forcebalance::amberio::FrcMod\_Reader, 243  
 forcebalance::amberio::Mol2\_Reader, 387  
 forcebalance::BaseReader, 193  
 forcebalance::custom\_io::Gen\_Reader, 250  
 forcebalance::gmxio::ITP\_Reader, 310  
 forcebalance::mol2io::Mol2\_Reader, 390  
 forcebalance::openmmio::OpenMM\_Reader, 454  
 forcebalance::psi4io::GBS\_Reader, 247  
 forcebalance::psi4io::Grid\_Reader, 259  
 forcebalance::psi4io::THCDF\_Psi4, 513  
 forcebalance::qchemio::QCIn\_Reader, 471  
 forcebalance::tinkerio::Tinker\_Reader, 516  
**molecules**  
 forcebalance::molecule::Molecule, 411  
 forcebalance::psi4io::RDVR3\_Psi4, 483  
**moments.py**, 539  
**moments\_driver**  
 forcebalance::tinkerio::Moments\_TINKER, 431  
**mpdb**  
 forcebalance::openmmio::Liquid\_OpenMM, 358  
**msim**  
 forcebalance::openmmio::Liquid\_OpenMM, 358  
**multiopen**  
 forcebalance::nifty, 54  
**mvals0**  
 forcebalance::optimizer::Optimizer, 463  
**na**  
 forcebalance::counterpoise::Counterpoise, 223  
 forcebalance::moments::Moments, 423  
 forcebalance::tinkerio::Moments\_TINKER, 435  
 forcebalance::vibration::Vibration, 527  
**name**  
 forcebalance::engine::Engine, 227  
 forcebalance::gmxio::GMX, 255  
**nbtype**  
 forcebalance::gmxio::ITP\_Reader, 310  
**ndtypes**  
 forcebalance::custom\_io, 21  
 forcebalance::qchemio, 74  
**nesp**  
 forcebalance::abinitio::AbInitio, 92  
 forcebalance::abinitio.internal::AbInitio\_Internal, 146  
 forcebalance::amberio::AbInitio\_AMBER, 110

forcebalance::gmxio::AbInitio\_GMX, 129  
 forcebalance::openmmio::AbInitio\_OpenMM, 165  
 forcebalance::tinkerio::AbInitio\_TINKER, 184  
**new\_vsites**  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
**NewtonRaphson**  
     forcebalance::optimizer::Optimizer, 459  
**nf\_err**  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
**nf\_err\_pct**  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
**nf\_ref**  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
**nftqm**  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
**nftytes**  
     forcebalance::gmxio, 31  
**nifty.py**, 540  
**nnf**  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
**nodematch**  
     forcebalance::molecule, 42  
**nomnom**  
     forcebalance::nifty::LineChunker, 322  
**np**  
     forcebalance::forcefield::FF, 237  
     forcebalance::optimizer::Optimizer, 463  
**nparticles**  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
**npt\_simulation**  
     forcebalance::gmxio::Liquid\_GMX, 340  
     forcebalance::liquid::Liquid, 328  
     forcebalance::openmmio::Liquid\_OpenMM, 353  
     forcebalance::tinkerio::Liquid\_TINKER, 366  
**nptfiles**  
     forcebalance::gmxio::Liquid\_GMX, 345  
     forcebalance::liquid::Liquid, 332  
     forcebalance::openmmio::Liquid\_OpenMM, 358  
     forcebalance::tinkerio::Liquid\_TINKER, 371  
**nptpx**  
     forcebalance::gmxio::Liquid\_GMX, 345  
     forcebalance::liquid::Liquid, 332  
     forcebalance::openmmio::Liquid\_OpenMM, 358  
     forcebalance::tinkerio::Liquid\_TINKER, 371  
**nptsfx**  
     forcebalance::gmxio::Liquid\_GMX, 345  
     forcebalance::liquid::Liquid, 333  
     forcebalance::openmmio::Liquid\_OpenMM, 358  
     forcebalance::tinkerio::Liquid\_TINKER, 371  
**ns**  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::counterpoise::Counterpoise, 223  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::gmxio::Interaction\_GMX, 281  
     forcebalance::interaction::Interaction, 269  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::openmmio::Interaction\_OpenMM, 293  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
     forcebalance::tinkerio::Interaction\_TINKER, 305  
**ntq**  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
**num\_atoms**  
     forcebalance::Mol2::mol2, 377  
**num\_bonds**  
     forcebalance::Mol2::mol2, 377  
**num\_compounds**

```

    forcebalance::Mol2::mol2_set, 391
num_feat
    forcebalance::Mol2::mol2, 377
num_sets
    forcebalance::Mol2::mol2, 377
num_subst
    forcebalance::Mol2::mol2, 377

ObjDict
    forcebalance::objective::Objective, 450
ObjDict_Last
    forcebalance::objective::Objective, 450
objd
    forcebalance::psi4io::RDVR3_Psi4, 483
Objective
    forcebalance::optimizer::Optimizer, 463
objective
    forcebalance::abinitio::AbInitio, 92
    forcebalance::abinitio_internal::AbInitio_Internal, 146
    forcebalance::amberio::AbInitio_AMBER, 110
    forcebalance::binding::BindingEnergy, 201
    forcebalance::gmxio::AbInitio_GMX, 129
    forcebalance::gmxio::Interaction_GMX, 281
    forcebalance::gmxqpio::Monomer_QTPIE, 446
    forcebalance::interaction::Interaction, 269
    forcebalance::leastsq::LeastSquares, 320
    forcebalance::moments::Moments, 423
    forcebalance::openmmio::AbInitio_OpenMM, 165
    forcebalance::openmmio::Interaction_OpenMM, 293
    forcebalance::psi4io::RDVR3_Psi4, 483
    forcebalance::psi4io::THCDF_Psi4, 513
    forcebalance::tinkerio::AbInitio_TINKER, 184
    forcebalance::tinkerio::BindingEnergy_TINKER, 210
    forcebalance::tinkerio::Interaction_TINKER, 305
    forcebalance::tinkerio::Moments_TINKER, 435
    forcebalance::vibration::Vibration, 527
objective.py, 542
objective_term
    forcebalance::gmxio::Liquid_GMX, 340
    forcebalance::liquid::Liquid, 328
    forcebalance::openmmio::Liquid_OpenMM, 353
    forcebalance::tinkerio::Liquid_TINKER, 366
objfiles
    forcebalance::psi4io::RDVR3_Psi4, 483
objvals
    forcebalance::psi4io::RDVR3_Psi4, 483
onefile
    forcebalance::nifty, 54
openmm_boxes
    forcebalance::molecule::Molecule, 400
openmm_positions
    forcebalance::molecule::Molecule, 401
openmmio.py, 542
openmmxml

```

```

    forcebalance::forcefield::FF, 237
OptTab
    forcebalance::optimizer::Optimizer, 463
optimizer.py, 543
optprog
    forcebalance::tinkerio::BindingEnergy_TINKER, 210
origin_atom_id
    forcebalance::Mol2::mol2_bond, 384
orthogonalize
    forcebalance::nifty, 54
out
    forcebalance::Mol2::mol2, 374
output.py, 544
PT.py, 545
parmdestroy_save
    forcebalance::forcefield::FF, 237
parmdestroy_this
    forcebalance::forcefield::FF, 237
ParsTab
    forcebalance::parser, 72
parse
    forcebalance::Mol2::mol2, 374
    forcebalance::Mol2::mol2_atom, 379
    forcebalance::Mol2::mol2_bond, 383
    forcebalance::Mol2::mol2_set, 391
parse_atomtype_line
    forcebalance::gmxio, 29
parse_inputs
    forcebalance::parser, 70
parse_interactions
    forcebalance::binding, 16
parser.py, 544
pathwise_rmsd
    forcebalance::molecule::Molecule, 401
patoms
    forcebalance::forcefield::FF, 238
pdict
    forcebalance::amberio::FrcMod_Reader, 243
    forcebalance::amberio::Mol2_Reader, 387
    forcebalance::BaseReader, 193
    forcebalance::custom_io, 21
    forcebalance::custom_io::Gen_Reader, 250
    forcebalance::gmxio, 31
    forcebalance::gmxio::ITP_Reader, 310
    forcebalance::mol2io::Mol2_Reader, 390
    forcebalance::openmmio, 66
    forcebalance::openmmio::OpenMM_Reader, 454
    forcebalance::psi4io::GBS_Reader, 247
    forcebalance::psi4io::Grid_Reader, 259
    forcebalance::qchemio, 74
    forcebalance::qchemio::QCIn_Reader, 471
    forcebalance::tinkerio, 76
    forcebalance::tinkerio::Tinker_Reader, 517

```

Pen\_Names  
     forcebalance::objective::Penalty, 467  
 Pen\_Tab  
     forcebalance::objective::Penalty, 467  
 Penalty  
     forcebalance::objective::Objective, 450  
     forcebalance::optimizer::Optimizer, 463  
 PeriodicTable  
     forcebalance::chemistry, 18  
     forcebalance::molecule, 43  
     forcebalance::PT, 73  
 pfields  
     forcebalance::forcefield::FF, 238  
 pftypes  
     forcebalance::gmxio, 31  
 PhasePoints  
     forcebalance::gmxio::Liquid\_GMX, 345  
     forcebalance::liquid::Liquid, 333  
     forcebalance::openmmio::Liquid\_OpenMM, 358  
     forcebalance::tinkerio::Liquid\_TINKER, 371  
 platform  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::openmmio::Interaction\_OpenMM, 293  
     forcebalance::openmmio::Liquid\_OpenMM, 358  
 plist  
     forcebalance::forcefield::FF, 238  
 pmat2d  
     forcebalance::nifty, 54  
 point  
     forcebalance::psi4io::Grid\_Reader, 259  
 polarization\_correction  
     forcebalance::gmxio::Liquid\_GMX, 341  
     forcebalance::liquid::Liquid, 329  
     forcebalance::openmmio::Liquid\_OpenMM, 354  
     forcebalance::tinkerio::Liquid\_TINKER, 367  
 positive\_resid  
     forcebalance::molecule::Molecule, 411  
 Powell  
     forcebalance::optimizer::Optimizer, 460  
 prefactor  
     forcebalance::gmxio::Interaction\_GMX, 281  
     forcebalance::interaction::Interaction, 269  
     forcebalance::openmmio::Interaction\_OpenMM, 293  
     forcebalance::tinkerio::Interaction\_TINKER, 305  
 prepare  
     forcebalance::engine::Engine, 226  
     forcebalance::gmxio::GMX, 254  
 prepare\_temp\_directory  
     forcebalance::abinitio::AbInitio, 87  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 141  
     forcebalance::amberio::AbInitio\_AMBER, 104  
     forcebalance::gmxio::AbInitio\_GMX, 123  
     forcebalance::gmxio::Interaction\_GMX, 278  
     forcebalance::gmxio::Liquid\_GMX, 341  
     forcebalance::gmxqpio::Monomer\_QTPIE, 442  
     forcebalance::interaction::Interaction, 265  
     forcebalance::moments::Moments, 419  
     forcebalance::openmmio::AbInitio\_OpenMM, 159  
     forcebalance::openmmio::Interaction\_OpenMM, 289  
     forcebalance::openmmio::Liquid\_OpenMM, 354  
     forcebalance::psi4io::THCDF\_Psi4, 509  
     forcebalance::tinkerio::AbInitio\_TINKER, 178  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 207  
     forcebalance::tinkerio::Interaction\_TINKER, 301  
     forcebalance::tinkerio::Liquid\_TINKER, 367  
     forcebalance::tinkerio::Moments\_TINKER, 432  
     forcebalance::tinkerio::Vibration\_TINKER, 529  
     forcebalance::vibration::Vibration, 524  
 print\_map  
     forcebalance::forcefield::FF, 236  
 PrintDict  
     forcebalance::binding::BindingEnergy, 201  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 210  
 PrintOptionDict  
     forcebalance::abinitio::AbInitio, 92  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 146  
     forcebalance::amberio::AbInitio\_AMBER, 110  
     forcebalance::BaseClass, 189  
     forcebalance::binding::BindingEnergy, 201  
     forcebalance::counterpoise::Counterpoise, 223  
     forcebalance::engine::Engine, 227  
     forcebalance::forcefield::FF, 238  
     forcebalance::gmxio::AbInitio\_GMX, 129  
     forcebalance::gmxio::GMX, 255  
     forcebalance::gmxio::Interaction\_GMX, 281  
     forcebalance::gmxio::Liquid\_GMX, 345  
     forcebalance::gmxqpio::Monomer\_QTPIE, 446  
     forcebalance::interaction::Interaction, 269  
     forcebalance::leastsq::LeastSquares, 320  
     forcebalance::liquid::Liquid, 333  
     forcebalance::moments::Moments, 423  
     forcebalance::objective::Objective, 450  
     forcebalance::openmmio::AbInitio\_OpenMM, 165  
     forcebalance::openmmio::Interaction\_OpenMM, 293  
     forcebalance::openmmio::Liquid\_OpenMM, 358  
     forcebalance::optimizer::Optimizer, 463  
     forcebalance::psi4io::RDVR3\_Psi4, 483  
     forcebalance::psi4io::THCDF\_Psi4, 513  
     forcebalance::target::RemoteTarget, 492  
     forcebalance::target::Target, 502  
     forcebalance::tinkerio::AbInitio\_TINKER, 184  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 210  
     forcebalance::tinkerio::Interaction\_TINKER, 305  
     forcebalance::tinkerio::Liquid\_TINKER, 371  
     forcebalance::tinkerio::Moments\_TINKER, 435  
     forcebalance::vibration::Vibration, 527  
 printcool  
     forcebalance::nifty, 55

**printcool\_dictionary**  
 forcebalance::nifty, 55  
**printcool\_table**  
 forcebalance::abinitio::AbInitio, 87  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 141  
 forcebalance::amberio::AbInitio\_AMBER, 105  
 forcebalance::binding::BindingEnergy, 198  
 forcebalance::counterpoise::Counterpoise, 220  
 forcebalance::gmxio::AbInitio\_GMX, 124  
 forcebalance::gmxio::Interaction\_GMX, 278  
 forcebalance::gmxio::Liquid\_GMX, 341  
 forcebalance::gmxqpio::Monomer\_QTPIE, 443  
 forcebalance::interaction::Interaction, 265  
 forcebalance::leastsq::LeastSquares, 317  
 forcebalance::liquid::Liquid, 329  
 forcebalance::moments::Moments, 420  
 forcebalance::openmmio::AbInitio\_OpenMM, 160  
 forcebalance::openmmio::Interaction\_OpenMM, 289  
 forcebalance::openmmio::Liquid\_OpenMM, 354  
 forcebalance::psi4io::RDVR3\_Psi4, 480  
 forcebalance::psi4io::THCDF\_Psi4, 509  
 forcebalance::target::RemoteTarget, 489  
 forcebalance::target::Target, 499  
 forcebalance::tinkerio::AbInitio\_TINKER, 179  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 207  
 forcebalance::tinkerio::Interaction\_TINKER, 301  
 forcebalance::tinkerio::Liquid\_TINKER, 367  
 forcebalance::tinkerio::Moments\_TINKER, 432  
 forcebalance::vibration::Vibration, 524  
**printsection**  
 forcebalance::parser, 70  
**psi4io.py**, 545  
**ptyp**  
 forcebalance::objective::Penalty, 467  
**push**  
 forcebalance::nifty::LineChunker, 322  
**pvals0**  
 forcebalance::forcefield::FF, 238  
**pvec**  
 forcebalance::molecule, 42  
**pvec1d**  
 forcebalance::nifty, 55  
**QChem\_Dielectric\_Energy**  
 forcebalance::qchemio, 74  
**qchemio.py**, 545  
**qfnm**  
 forcebalance::abinitio::AbInitio, 92  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 110  
 forcebalance::gmxio::AbInitio\_GMX, 129  
 forcebalance::gmxio::Interaction\_GMX, 281  
 forcebalance::interaction::Interaction, 269  
 forcebalance::openmmio::AbInitio\_OpenMM, 165  
 forcebalance::openmmio::Interaction\_OpenMM, 293  
 forcebalance::tinkerio::AbInitio\_TINKER, 184  
 forcebalance::tinkerio::Interaction\_TINKER, 305  
**qid**  
 forcebalance::forcefield::FF, 238  
**qid2**  
 forcebalance::forcefield::FF, 238  
**qmap**  
 forcebalance::forcefield::FF, 238  
**qmatoms**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::openmmio::AbInitio\_OpenMM, 166  
 forcebalance::tinkerio::AbInitio\_TINKER, 185  
**qmboltz\_wts**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::openmmio::AbInitio\_OpenMM, 166  
 forcebalance::tinkerio::AbInitio\_TINKER, 185  
**QuantumVariableNames**  
 forcebalance::molecule, 43  
**queue\_up**  
 forcebalance::nifty, 57  
**queue\_up\_src\_dest**  
 forcebalance::nifty, 57  
**r\_options**  
 forcebalance::target::RemoteTarget, 492  
**r\_tgt\_opts**  
 forcebalance::target::RemoteTarget, 492  
**RMSDDict**  
 forcebalance::binding::BindingEnergy, 201  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 210  
**radian**  
 forcebalance::molecule, 43  
**Radii**  
 forcebalance::chemistry, 18  
 forcebalance::molecule, 43  
**radii**  
 forcebalance::psi4io::Grid\_Reader, 259  
**radius\_of\_gyration**  
 forcebalance::molecule::Molecule, 401  
**Read\_Tab**  
 forcebalance::molecule::Molecule, 411  
**read\_arc**  
 forcebalance::molecule::Molecule, 401  
**read\_charmm**  
 forcebalance::molecule::Molecule, 402  
**read\_com**  
 forcebalance::molecule::Molecule, 402

**read\_data**  
 forcebalance::gmxio::Liquid\_GMX, 342  
 forcebalance::liquid::Liquid, 329  
 forcebalance::openmmio::Liquid\_OpenMM, 355  
 forcebalance::tinkerio::Liquid\_TINKER, 368  
**read\_dcd**  
 forcebalance::molecule::Molecule, 402  
**read\_gro**  
 forcebalance::molecule::Molecule, 403  
**read\_internals**  
 forcebalance::parser, 70  
**read\_mdcrd**  
 forcebalance::molecule::Molecule, 403  
**read\_mol2**  
 forcebalance::molecule::Molecule, 403  
**read\_mvals**  
 forcebalance::parser, 71  
**read\_pdb**  
 forcebalance::molecule::Molecule, 404  
**read\_priors**  
 forcebalance::parser, 71  
**read\_pvals**  
 forcebalance::parser, 71  
**read\_qcesp**  
 forcebalance::molecule::Molecule, 404  
**read\_qcin**  
 forcebalance::molecule::Molecule, 404  
**read\_qcout**  
 forcebalance::molecule::Molecule, 405  
**read\_qdata**  
 forcebalance::molecule::Molecule, 405  
**read\_reference\_data**  
 forcebalance::abinitio::AbInitio, 88  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 142  
 forcebalance::amberio::AbInitio\_AMBER, 105  
 forcebalance::gmxio::AbInitio\_GMX, 124  
 forcebalance::gmxio::Interaction\_GMX, 278  
 forcebalance::interaction::Interaction, 266  
 forcebalance::moments::Moments, 420  
 forcebalance::openmmio::AbInitio\_OpenMM, 160  
 forcebalance::openmmio::Interaction\_OpenMM, 290  
 forcebalance::tinkerio::AbInitio\_TINKER, 179  
 forcebalance::tinkerio::Interaction\_TINKER, 302  
 forcebalance::tinkerio::Moments\_TINKER, 432  
 forcebalance::vibration::Vibration, 525  
**read\_topology**  
 forcebalance::abinitio::AbInitio, 89  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 143  
 forcebalance::amberio::AbInitio\_AMBER, 106  
 forcebalance::gmxio::AbInitio\_GMX, 125  
 forcebalance::openmmio::AbInitio\_OpenMM, 161  
 forcebalance::tinkerio::AbInitio\_TINKER, 180  
**read\_xyz**  
 forcebalance::molecule::Molecule, 406

**readchk**  
 forcebalance::optimizer::Optimizer, 460  
**Readers**  
 forcebalance::forcefield::FF, 238  
**redirect**  
 forcebalance::forcefield::FF, 238  
**ref\_eigvals**  
 forcebalance::moments::Moments, 423  
 forcebalance::tinkerio::Moments\_TINKER, 435  
 forcebalance::vibration::Vibration, 527  
**ref\_eigvecs**  
 forcebalance::moments::Moments, 424  
 forcebalance::tinkerio::Moments\_TINKER, 436  
 forcebalance::vibration::Vibration, 527  
**ref\_moments**  
 forcebalance::gmxqpio::Monomer\_QTPIE, 446  
 forcebalance::moments::Moments, 424  
 forcebalance::tinkerio::Moments\_TINKER, 436  
**ref\_rmsd**  
 forcebalance::molecule::Molecule, 406  
**RefData**  
 forcebalance::gmxio::Liquid\_GMX, 345  
 forcebalance::liquid::Liquid, 333  
 forcebalance::openmmio::Liquid\_OpenMM, 358  
 forcebalance::tinkerio::Liquid\_TINKER, 371  
**refresh\_temp\_directory**  
 forcebalance::abinitio::AbInitio, 89  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 143  
 forcebalance::amberio::AbInitio\_AMBER, 106  
 forcebalance::binding::BindingEnergy, 199  
 forcebalance::counterpoise::Counterpoise, 221  
 forcebalance::gmxio::AbInitio\_GMX, 125  
 forcebalance::gmxio::Interaction\_GMX, 279  
 forcebalance::gmxio::Liquid\_GMX, 342  
 forcebalance::gmxqpio::Monomer\_QTPIE, 443  
 forcebalance::interaction::Interaction, 266  
 forcebalance::leastsq::LeastSquares, 318  
 forcebalance::liquid::Liquid, 329  
 forcebalance::moments::Moments, 420  
 forcebalance::openmmio::AbInitio\_OpenMM, 161  
 forcebalance::openmmio::Interaction\_OpenMM, 290  
 forcebalance::openmmio::Liquid\_OpenMM, 355  
 forcebalance::psi4io::RDVR3\_Psi4, 481  
 forcebalance::psi4io::THCDF\_Psi4, 510  
 forcebalance::target::RemoteTarget, 490  
 forcebalance::target::Target, 499  
 forcebalance::tinkerio::AbInitio\_TINKER, 180  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 208  
 forcebalance::tinkerio::Interaction\_TINKER, 302  
 forcebalance::tinkerio::Liquid\_TINKER, 368  
 forcebalance::tinkerio::Moments\_TINKER, 433  
 forcebalance::vibration::Vibration, 525  
**remote\_indicate**  
 forcebalance::target::RemoteTarget, 492

```

remove_if_exists
    forcebalance::nifty, 57
removeHandler
    forcebalance::output::ForceBalanceLogger, 240
replace_peratom
    forcebalance::molecule::Molecule, 406
replace_peratom_conditional
    forcebalance::molecule::Molecule, 406
require
    forcebalance::molecule::Molecule, 407
require_boxes
    forcebalance::molecule::Molecule, 407
require_resid
    forcebalance::molecule::Molecule, 407
require_resname
    forcebalance::molecule::Molecule, 407
ResetVirtualSites
    forcebalance::openmmio, 65
resid
    forcebalance::molecule::Molecule, 411
residue_distances
    forcebalance::contact, 19
renname
    forcebalance::molecule::Molecule, 411
respterm
    forcebalance::abinitio::AbInitio, 93
    forcebalance::abinitio_internal::AbInitio_Internal, 147
    forcebalance::amberio::AbInitio_AMBER, 111
    forcebalance::gmxio::AbInitio_GMX, 130
    forcebalance::openmmio::AbInitio_OpenMM, 166
    forcebalance::tinkerio::AbInitio_TINKER, 185
rm_gmx_baks
    forcebalance::gmxio, 30
rmsd_part
    forcebalance::binding::BindingEnergy, 201
    forcebalance::tinkerio::BindingEnergy_TINKER, 210
root
    forcebalance::engine::Engine, 227
    forcebalance::gmxio::GMX, 255
row
    forcebalance::nifty, 57
rs
    forcebalance::forcefield::FF, 238
rs_override
    forcebalance::forcefield, 27
rsmake
    forcebalance::forcefield::FF, 236
Run
    forcebalance::optimizer::Optimizer, 460
rundir
    forcebalance::abinitio::AbInitio, 93
    forcebalance::abinitio_internal::AbInitio_Internal, 147
    forcebalance::amberio::AbInitio_AMBER, 111
    forcebalance::binding::BindingEnergy, 201
forcebalance::counterpoise::Counterpoise, 223
forcebalance::gmxio::AbInitio_GMX, 130
forcebalance::gmxio::Interaction_GMX, 282
forcebalance::gmxio::Liquid_GMX, 345
forcebalance::gmxqpio::Monomer_QTPIE, 446
forcebalance::interaction::Interaction, 269
forcebalance::leastsq::LeastSquares, 320
forcebalance::liquid::Liquid, 333
forcebalance::moments::Moments, 424
forcebalance::openmmio::AbInitio_OpenMM, 166
forcebalance::openmmio::Interaction_OpenMM, 294
forcebalance::openmmio::Liquid_OpenMM, 358
forcebalance::psi4io::RDVR3_Psi4, 483
forcebalance::psi4io::THCDF_Psi4, 513
forcebalance::target::RemoteTarget, 492
forcebalance::target::Target, 502
forcebalance::tinkerio::AbInitio_TINKER, 185
forcebalance::tinkerio::BindingEnergy_TINKER, 210
forcebalance::tinkerio::Interaction_TINKER, 305
forcebalance::tinkerio::Liquid_TINKER, 371
forcebalance::tinkerio::Moments_TINKER, 436
forcebalance::vibration::Vibration, 527

save_vmvalls
    forcebalance::abinitio::AbInitio, 93
    forcebalance::abinitio_internal::AbInitio_Internal, 147
    forcebalance::amberio::AbInitio_AMBER, 111
    forcebalance::gmxio::AbInitio_GMX, 130
    forcebalance::openmmio::AbInitio_OpenMM, 166
    forcebalance::tinkerio::AbInitio_TINKER, 185
SavedMVal
    forcebalance::gmxio::Liquid_GMX, 345
    forcebalance::liquid::Liquid, 333
    forcebalance::openmmio::Liquid_OpenMM, 358
    forcebalance::tinkerio::Liquid_TINKER, 371
SavedTraj
    forcebalance::gmxio::Liquid_GMX, 345
    forcebalance::liquid::Liquid, 333
    forcebalance::openmmio::Liquid_OpenMM, 358
    forcebalance::tinkerio::Liquid_TINKER, 371
Scan_Values
    forcebalance::optimizer::Optimizer, 460
ScanMVals
    forcebalance::optimizer::Optimizer, 461
ScanPVals
    forcebalance::optimizer::Optimizer, 461
ScipyOptimizer
    forcebalance::optimizer::Optimizer, 461
sec
    forcebalance::custom_io::Gen_Reader, 250
    forcebalance::gmxio::ITP_Reader, 310
    forcebalance::qchemio::QCIn_Reader, 471
section
    forcebalance::amberio::Mol2_Reader, 387

```

```

segments
    forcebalance::nifty, 58
select1
    forcebalance::gmxio::Interaction_GMX, 282
    forcebalance::interaction::Interaction, 270
    forcebalance::openmmio::Interaction_OpenMM, 294
    forcebalance::tinkerio::Interaction_TINKER, 306
select2
    forcebalance::gmxio::Interaction_GMX, 282
    forcebalance::interaction::Interaction, 270
    forcebalance::openmmio::Interaction_OpenMM, 294
    forcebalance::tinkerio::Interaction_TINKER, 306
set_atom_id
    forcebalance::Mol2::mol2_atom, 379
set_atom_name
    forcebalance::Mol2::mol2_atom, 379
set_atom_type
    forcebalance::Mol2::mol2_atom, 379
set_bond_id
    forcebalance::Mol2::mol2_bond, 383
set_bond_type
    forcebalance::Mol2::mol2_bond, 383
set_charge
    forcebalance::Mol2::mol2_atom, 380
set_charge_type
    forcebalance::Mol2::mol2, 374
set_crds
    forcebalance::Mol2::mol2_atom, 380
set_donor_acceptor_atoms
    forcebalance::Mol2::mol2, 375
set_mol_name
    forcebalance::Mol2::mol2, 375
set_mol_type
    forcebalance::Mol2::mol2, 375
set_num_atoms
    forcebalance::Mol2::mol2, 376
set_num_bonds
    forcebalance::Mol2::mol2, 376
set_num_feat
    forcebalance::Mol2::mol2, 376
set_num_sets
    forcebalance::Mol2::mol2, 376
set_num_subst
    forcebalance::Mol2::mol2, 377
set_option
    forcebalance::abinitio::AbInitio, 89
    forcebalance::abinitio_internal::AbInitio_Internal, 143
    forcebalance::amberio::AbInitio_AMBER, 107
    forcebalance::BaseClass, 189
    forcebalance::binding::BindingEnergy, 199
    forcebalance::counterpoise::Counterpoise, 221
    forcebalance::engine::Engine, 227
    forcebalance::forcefield::FF, 237
    forcebalance::gmxio::AbInitio_GMX, 126
forcebalance::gmxio::GMX, 254
forcebalance::gmxio::Interaction_GMX, 279
forcebalance::gmxio::Liquid_GMX, 342
forcebalance::gmxqpio::Monomer_QTPIE, 444
forcebalance::interaction::Interaction, 267
forcebalance::leastsq::LeastSquares, 318
forcebalance::liquid::Liquid, 330
forcebalance::moments::Moments, 421
forcebalance::objective::Objective, 449
forcebalance::openmmio::AbInitio_OpenMM, 162
forcebalance::openmmio::Interaction_OpenMM, 291
forcebalance::openmmio::Liquid_OpenMM, 355
forcebalance::optimizer::Optimizer, 461
forcebalance::psi4io::RDVR3_Psi4, 481
forcebalance::psi4io::THCDF_Psi4, 510
forcebalance::target::RemoteTarget, 490
forcebalance::target::Target, 500
forcebalance::tinkerio::AbInitio_TINKER, 181
forcebalance::tinkerio::BindingEnergy_TINKER, 208
forcebalance::tinkerio::Interaction_TINKER, 303
forcebalance::tinkerio::Liquid_TINKER, 368
forcebalance::tinkerio::Moments_TINKER, 433
forcebalance::vibration::Vibration, 525
set_origin_atom_id
    forcebalance::Mol2::mol2_bond, 383
set_status_bit
    forcebalance::Mol2::mol2_atom, 380
    forcebalance::Mol2::mol2_bond, 383
set_subst_id
    forcebalance::Mol2::mol2_atom, 380
set_subst_name
    forcebalance::Mol2::mol2_atom, 381
set_target_atom_id
    forcebalance::Mol2::mol2_bond, 384
SetAmoebaVirtualExclusions
    forcebalance::openmmio, 66
sget
    forcebalance::abinitio::AbInitio, 90
    forcebalance::abinitio_internal::AbInitio_Internal, 144
    forcebalance::amberio::AbInitio_AMBER, 107
    forcebalance::binding::BindingEnergy, 199
    forcebalance::counterpoise::Counterpoise, 221
    forcebalance::gmxio::AbInitio_GMX, 126
    forcebalance::gmxio::Interaction_GMX, 279
    forcebalance::gmxio::Liquid_GMX, 342
    forcebalance::gmxqpio::Monomer_QTPIE, 444
    forcebalance::interaction::Interaction, 267
    forcebalance::leastsq::LeastSquares, 318
    forcebalance::liquid::Liquid, 330
    forcebalance::moments::Moments, 421
    forcebalance::openmmio::AbInitio_OpenMM, 162
    forcebalance::openmmio::Interaction_OpenMM, 291
    forcebalance::openmmio::Liquid_OpenMM, 355
    forcebalance::psi4io::RDVR3_Psi4, 481

```

forcebalance::psi4io::THCDF\_Psi4, 510  
 forcebalance::target::RemoteTarget, 490  
 forcebalance::target::Target, 500  
 forcebalance::tinkerio::AbInitio\_TINKER, 181  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 208  
 forcebalance::tinkerio::Interaction\_TINKER, 303  
 forcebalance::tinkerio::Liquid\_TINKER, 368  
 forcebalance::tinkerio::Moments\_TINKER, 433  
 forcebalance::vibration::Vibration, 525  
 shell  
     forcebalance::qchemio::QCIn\_Reader, 471  
 shot\_mdp  
     forcebalance::gmxio, 31  
 Simplex  
     forcebalance::optimizer::Optimizer, 462  
 simulation  
     forcebalance::openmmio::AbInitio\_OpenMM, 166  
 simulations  
     forcebalance::openmmio::Interaction\_OpenMM, 294  
 SinglePoint  
     forcebalance::optimizer::Optimizer, 462  
 snum  
     forcebalance::qchemio::QCIn\_Reader, 471  
 spacings  
     forcebalance::objective::Penalty, 467  
 specific\_dct  
     forcebalance::nifty, 60  
 specific\_lst  
     forcebalance::nifty, 60  
 Split  
     forcebalance::amberio::FrcMod\_Reader, 242  
     forcebalance::amberio::Mol2\_Reader, 386  
     forcebalance::BaseReader, 192  
     forcebalance::custom\_io::Gen\_Reader, 249  
     forcebalance::gmxio::ITP\_Reader, 309  
     forcebalance::mol2io::Mol2\_Reader, 389  
     forcebalance::openmmio::OpenMM\_Reader, 453  
     forcebalance::psi4io::GBS\_Reader, 246  
     forcebalance::psi4io::Grid\_Reader, 258  
     forcebalance::qchemio::QCIn\_Reader, 470  
     forcebalance::tinkerio::Tinker\_Reader, 516  
 split  
     forcebalance::molecule::Molecule, 407  
 splitter  
     forcebalance::molecule, 44  
 srmdir  
     forcebalance::gmxio::GMX, 255  
 stage  
     forcebalance::abinitio::AbInitio, 90  
     forcebalance::abinitio\_internal::AbInitio\_Internal, 144  
     forcebalance::amberio::AbInitio\_AMBER, 108  
     forcebalance::binding::BindingEnergy, 200  
     forcebalance::counterpoise::Counterpoise, 222  
     forcebalance::gmxio::AbInitio\_GMX, 127  
     forcebalance::gmxio::Interaction\_GMX, 280  
     forcebalance::gmxio::Liquid\_GMX, 343  
     forcebalance::gmxqpio::Monomer\_QTPIE, 445  
     forcebalance::interaction::Interaction, 268  
     forcebalance::leastsq::LeastSquares, 319  
     forcebalance::liquid::Liquid, 331  
     forcebalance::moments::Moments, 422  
     forcebalance::openmmio::AbInitio\_OpenMM, 163  
     forcebalance::openmmio::Interaction\_OpenMM, 292  
     forcebalance::openmmio::Liquid\_OpenMM, 356  
     forcebalance::psi4io::RDVR3\_Psi4, 482  
     forcebalance::psi4io::THCDF\_Psi4, 511  
     forcebalance::target::RemoteTarget, 491  
     forcebalance::target::Target, 501  
     forcebalance::tinkerio::AbInitio\_TINKER, 182  
     forcebalance::tinkerio::BindingEnergy\_TINKER, 209  
     forcebalance::tinkerio::Interaction\_TINKER, 304  
     forcebalance::tinkerio::Liquid\_TINKER, 369

forcebalance::tinkerio::Moments\_TINKER, 434  
 forcebalance::vibration::Vibration, 526  
**subst\_id**  
 forcebalance::Mol2::mol2\_atom, 381  
**subst\_name**  
 forcebalance::Mol2::mol2\_atom, 381  
**suffix**  
 forcebalance::amberio::FrcMod\_Reader, 243  
 forcebalance::amberio::Mol2\_Reader, 387  
 forcebalance::BaseReader, 193  
 forcebalance::custom\_io::Gen\_Reader, 250  
 forcebalance::gmxio::ITP\_Reader, 310  
 forcebalance::mol2io::Mol2\_Reader, 390  
 forcebalance::openmmio::OpenMM\_Reader, 454  
 forcebalance::psi4io::GBS\_Reader, 247  
 forcebalance::psi4io::Grid\_Reader, 259  
 forcebalance::qchemio::QCIn\_Reader, 471  
 forcebalance::tinkerio::Tinker\_Reader, 517  
**suffix\_dict**  
 forcebalance::openmmio, 66  
**system\_driver**  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 209  
**target**  
 forcebalance::engine::Engine, 227  
 forcebalance::gmxio::GMX, 255  
**target.py**, 546  
**Target\_Terms**  
 forcebalance::objective::Objective, 449  
**target\_atom\_id**  
 forcebalance::Mol2::mol2\_bond, 384  
**Targets**  
 forcebalance::objective::Objective, 450  
**tdir**  
 forcebalance::psi4io::RDVR3\_Psi4, 483  
**tempdir**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::binding::BindingEnergy, 201  
 forcebalance::counterpoise::Counterpoise, 223  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::gmxio::Interaction\_GMX, 282  
 forcebalance::gmxio::Liquid\_GMX, 345  
 forcebalance::gmxqpio::Monomer\_QTPIE, 446  
 forcebalance::interaction::Interaction, 270  
 forcebalance::leastsq::LeastSquares, 320  
 forcebalance::liquid::Liquid, 333  
 forcebalance::moments::Moments, 424  
 forcebalance::openmmio::AbInitio\_OpenMM, 166  
 forcebalance::openmmio::Interaction\_OpenMM, 294  
 forcebalance::openmmio::Liquid\_OpenMM, 358  
 forcebalance::psi4io::RDVR3\_Psi4, 484  
 forcebalance::psi4io::THCDF\_Psi4, 513  
  
 forcebalance::target::RemoteTarget, 492  
 forcebalance::target::Target, 502  
 forcebalance::tinkerio::AbInitio\_TINKER, 185  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 211  
 forcebalance::tinkerio::Interaction\_TINKER, 306  
 forcebalance::tinkerio::Liquid\_TINKER, 371  
 forcebalance::tinkerio::Moments\_TINKER, 436  
 forcebalance::vibration::Vibration, 527  
**tgt\_opts\_defaults**  
 forcebalance::parser, 72  
**tgt\_opts\_types**  
 forcebalance::parser, 72  
**throw\_outs**  
 forcebalance::psi4io::THCDF\_Psi4, 513  
**tinkerio.py**, 546  
**tinkerprm**  
 forcebalance::forcefield::FF, 238  
**tm**  
 forcebalance::forcefield::FF, 238  
**tml**  
 forcebalance::forcefield::FF, 238  
**top**  
 forcebalance::gmxio::GMX, 255  
**topfnm**  
 forcebalance::gmxio::Interaction\_GMX, 282  
**topology**  
 forcebalance::molecule::Molecule, 411  
**topology\_flag**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::openmmio::AbInitio\_OpenMM, 166  
 forcebalance::tinkerio::AbInitio\_TINKER, 185  
**tq\_err**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::openmmio::AbInitio\_OpenMM, 166  
 forcebalance::tinkerio::AbInitio\_TINKER, 185  
**tq\_err\_pct**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::openmmio::AbInitio\_OpenMM, 166  
 forcebalance::tinkerio::AbInitio\_TINKER, 185  
**tq\_ref**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::openmmio::AbInitio\_OpenMM, 166

forcebalance::tinkerio::AbInitio\_TINKER, 185  
**traj**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::gmxio::Interaction\_GMX, 282  
 forcebalance::interaction::Interaction, 270  
 forcebalance::openmmio::AbInitio\_OpenMM, 166  
 forcebalance::openmmio::Interaction\_OpenMM, 294  
 forcebalance::tinkerio::AbInitio\_TINKER, 185  
 forcebalance::tinkerio::Interaction\_TINKER, 306  
**trajfnm**  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::gmxio::Interaction\_GMX, 282  
 forcebalance::openmmio::AbInitio\_OpenMM, 166  
 forcebalance::openmmio::Interaction\_OpenMM, 294  
 forcebalance::tinkerio::AbInitio\_TINKER, 185  
 forcebalance::tinkerio::Interaction\_TINKER, 306  
**uncommadash**  
 forcebalance::nifty, 58  
**unpack\_moments**  
 forcebalance::gmxqpio::Monomer\_QTPIE, 445  
 forcebalance::moments::Moments, 422  
 forcebalance::tinkerio::Moments\_TINKER, 434  
**UpdateSimulationParameters**  
 forcebalance::openmmio, 66  
**use\_nft**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::openmmio::AbInitio\_OpenMM, 166  
 forcebalance::tinkerio::AbInitio\_TINKER, 185  
**Val**  
 forcebalance::optimizer::Optimizer, 463  
**verbose\_options**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::BaseClass, 189  
 forcebalance::binding::BindingEnergy, 201  
 forcebalance::counterpoise::Counterpoise, 223  
 forcebalance::engine::Engine, 227  
 forcebalance::forcefield::FF, 238  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::gmxio::GMX, 255  
 forcebalance::gmxio::Interaction\_GMX, 282  
 forcebalance::gmxio::Liquid\_GMX, 345  
 forcebalance::gmxqpio::Monomer\_QTPIE, 447  
 forcebalance::interaction::Interaction, 270  
 forcebalance::leastsq::LeastSquares, 320  
 forcebalance::liquid::Liquid, 333  
 forcebalance::moments::Moments, 424  
 forcebalance::objective::Objective, 450  
 forcebalance::openmmio::AbInitio\_OpenMM, 166  
 forcebalance::openmmio::Interaction\_OpenMM, 294  
 forcebalance::openmmio::Liquid\_OpenMM, 358  
 forcebalance::optimizer::Optimizer, 463  
 forcebalance::psi4io::RDVR3\_Psi4, 484  
 forcebalance::psi4io::THCDF\_Psi4, 513  
 forcebalance::target::RemoteTarget, 493  
 forcebalance::target::Target, 502  
 forcebalance::tinkerio::AbInitio\_TINKER, 185  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 211  
 forcebalance::tinkerio::Interaction\_TINKER, 306  
 forcebalance::tinkerio::Liquid\_TINKER, 371  
 forcebalance::tinkerio::Moments\_TINKER, 436  
 forcebalance::vibration::Vibration, 527  
**vfnm**  
 forcebalance::vibration::Vibration, 527  
**vibration.py**, 547  
**vibration\_driver**  
 forcebalance::tinkerio::Vibration\_TINKER, 529  
**w\_alpha**  
 forcebalance::gmxio::Liquid\_GMX, 345  
 forcebalance::liquid::Liquid, 333  
 forcebalance::openmmio::Liquid\_OpenMM, 358  
 forcebalance::tinkerio::Liquid\_TINKER, 371  
**w\_cp**  
 forcebalance::gmxio::Liquid\_GMX, 345  
 forcebalance::liquid::Liquid, 333  
 forcebalance::openmmio::Liquid\_OpenMM, 359  
 forcebalance::tinkerio::Liquid\_TINKER, 371  
**w\_energy**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 147  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::openmmio::AbInitio\_OpenMM, 167  
 forcebalance::tinkerio::AbInitio\_TINKER, 185  
**w\_eps0**  
 forcebalance::gmxio::Liquid\_GMX, 345  
 forcebalance::liquid::Liquid, 333  
 forcebalance::openmmio::Liquid\_OpenMM, 359  
 forcebalance::tinkerio::Liquid\_TINKER, 371  
**w\_force**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 148  
 forcebalance::amberio::AbInitio\_AMBER, 111  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::openmmio::AbInitio\_OpenMM, 167  
 forcebalance::tinkerio::AbInitio\_TINKER, 185  
**w\_hvap**

forcebalance::gmxio::Liquid\_GMX, 345  
 forcebalance::liquid::Liquid, 333  
 forcebalance::openmmio::Liquid\_OpenMM, 359  
 forcebalance::tinkerio::Liquid\_TINKER, 371  
**w\_kappa**  
 forcebalance::gmxio::Liquid\_GMX, 345  
 forcebalance::liquid::Liquid, 333  
 forcebalance::openmmio::Liquid\_OpenMM, 359  
 forcebalance::tinkerio::Liquid\_TINKER, 371  
**w\_netforce**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 148  
 forcebalance::amberio::AbInitio\_AMBER, 112  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::openmmio::AbInitio\_OpenMM, 167  
 forcebalance::tinkerio::AbInitio\_TINKER, 186  
**w\_resp**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 148  
 forcebalance::amberio::AbInitio\_AMBER, 112  
 forcebalance::gmxio::AbInitio\_GMX, 130  
 forcebalance::openmmio::AbInitio\_OpenMM, 167  
 forcebalance::tinkerio::AbInitio\_TINKER, 186  
**w\_rho**  
 forcebalance::gmxio::Liquid\_GMX, 346  
 forcebalance::liquid::Liquid, 333  
 forcebalance::openmmio::Liquid\_OpenMM, 359  
 forcebalance::tinkerio::Liquid\_TINKER, 371  
**w\_torque**  
 forcebalance::abinitio::AbInitio, 93  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 148  
 forcebalance::amberio::AbInitio\_AMBER, 112  
 forcebalance::gmxio::AbInitio\_GMX, 131  
 forcebalance::openmmio::AbInitio\_OpenMM, 167  
 forcebalance::tinkerio::AbInitio\_TINKER, 186  
**WORK\_QUEUE**  
 forcebalance::nifty, 60  
**WQIDS**  
 forcebalance::nifty, 60  
**WTot**  
 forcebalance::objective::Objective, 450  
**warn\_once**  
 forcebalance::nifty, 59  
**warn\_press\_key**  
 forcebalance::nifty, 59  
**weight**  
 forcebalance::gmxio::Interaction\_GMX, 282  
 forcebalance::interaction::Interaction, 270  
 forcebalance::openmmio::Interaction\_OpenMM, 294  
 forcebalance::tinkerio::Interaction\_TINKER, 306  
**weight\_info**  
 forcebalance::liquid, 33  
**weighted\_variance**  
 forcebalance::abinitio, 13  
 weighted\_variance2  
 forcebalance::abinitio, 14  
**weights**  
 forcebalance::gmxqpio::Monomer\_QTPIE, 447  
**whamboltz**  
 forcebalance::abinitio::AbInitio, 94  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 148  
 forcebalance::amberio::AbInitio\_AMBER, 112  
 forcebalance::gmxio::AbInitio\_GMX, 131  
 forcebalance::openmmio::AbInitio\_OpenMM, 167  
 forcebalance::tinkerio::AbInitio\_TINKER, 186  
**whamboltz\_wts**  
 forcebalance::abinitio::AbInitio, 94  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 148  
 forcebalance::amberio::AbInitio\_AMBER, 112  
 forcebalance::gmxio::AbInitio\_GMX, 131  
 forcebalance::openmmio::AbInitio\_OpenMM, 167  
 forcebalance::tinkerio::AbInitio\_TINKER, 186  
**which**  
 forcebalance::nifty, 59  
**Whites**  
 forcebalance::amberio::FrcMod\_Reader, 242  
 forcebalance::amberio::Mol2\_Reader, 386  
 forcebalance::BaseReader, 192  
 forcebalance::custom\_io::Gen\_Reader, 250  
 forcebalance::gmxio::ITP\_Reader, 309  
 forcebalance::mol2io::Mol2\_Reader, 389  
 forcebalance::openmmio::OpenMM\_Reader, 453  
 forcebalance::psi4io::GBS\_Reader, 246  
 forcebalance::psi4io::Grid\_Reader, 258  
 forcebalance::qcchemio::QCIn\_Reader, 470  
 forcebalance::tinkerio::Tinker\_Reader, 516  
**wq\_complete**  
 forcebalance::abinitio::AbInitio, 91  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 145  
 forcebalance::amberio::AbInitio\_AMBER, 108  
 forcebalance::binding::BindingEnergy, 200  
 forcebalance::counterpoise::Counterpoise, 222  
 forcebalance::gmxio::AbInitio\_GMX, 127  
 forcebalance::gmxio::Interaction\_GMX, 280  
 forcebalance::gmxio::Liquid\_GMX, 344  
 forcebalance::gmxqpio::Monomer\_QTPIE, 446  
 forcebalance::interaction::Interaction, 268  
 forcebalance::leastsq::LeastSquares, 319  
 forcebalance::liquid::Liquid, 332  
 forcebalance::moments::Moments, 423  
 forcebalance::openmmio::AbInitio\_OpenMM, 163  
 forcebalance::openmmio::Interaction\_OpenMM, 292  
 forcebalance::openmmio::Liquid\_OpenMM, 357  
 forcebalance::psi4io::RDVR3\_Psi4, 482  
 forcebalance::psi4io::THCDF\_Psi4, 511  
 forcebalance::target::RemoteTarget, 492  
 forcebalance::target::Target, 501  
 forcebalance::tinkerio::AbInitio\_TINKER, 182

forcebalance::tinkerio::BindingEnergy\_TINKER, 210  
 forcebalance::tinkerio::Interaction\_TINKER, 304  
 forcebalance::tinkerio::Liquid\_TINKER, 370  
 forcebalance::tinkerio::Moments\_TINKER, 435  
 forcebalance::vibration::Vibration, 526  
**wq\_wait**  
     forcebalance::nifty, 59  
**wq\_wait1**  
     forcebalance::nifty, 59  
**write**  
     forcebalance::molecule::Molecule, 407  
**Write\_Tab**  
     forcebalance::molecule::Molecule, 411  
**write\_arc**  
     forcebalance::molecule::Molecule, 408  
**write\_dcd**  
     forcebalance::molecule::Molecule, 408  
**write\_gro**  
     forcebalance::molecule::Molecule, 408  
**write\_key\_with\_prm**  
     forcebalance::tinkerio, 76  
**write\_mdcrd**  
     forcebalance::molecule::Molecule, 408  
**write\_molproq**  
     forcebalance::molecule::Molecule, 408  
**write\_nested\_destroy**  
     forcebalance::psi4io::THCDF\_Psi4, 512  
**write\_pdb**  
     forcebalance::molecule::Molecule, 409  
**write\_qcin**  
     forcebalance::molecule::Molecule, 409  
**write\_qdata**  
     forcebalance::molecule::Molecule, 410  
**write\_xyz**  
     forcebalance::molecule::Molecule, 410  
**writechk**  
     forcebalance::optimizer::Optimizer, 462

**x**

forcebalance::Mol2::mol2\_atom, 381

**XMLFILE**

forcebalance::nifty, 60

**xct**

forcebalance::abinitio::AbInitio, 94  
 forcebalance::abinitio\_internal::AbInitio\_Internal, 148  
 forcebalance::amberio::AbInitio\_AMBER, 112  
 forcebalance::binding::BindingEnergy, 201  
 forcebalance::counterpoise::Counterpoise, 223  
 forcebalance::gmxio::AbInitio\_GMX, 131  
 forcebalance::gmxio::Interaction\_GMX, 282  
 forcebalance::gmxio::Liquid\_GMX, 346  
 forcebalance::gmxqpio::Monomer\_QTPIE, 447  
 forcebalance::interaction::Interaction, 270  
 forcebalance::leastsq::LeastSquares, 320

forcebalance::liquid::Liquid, 333  
 forcebalance::moments::Moments, 424  
 forcebalance::openmmio::AbInitio\_OpenMM, 167  
 forcebalance::openmmio::Interaction\_OpenMM, 294  
 forcebalance::openmmio::Liquid\_OpenMM, 359  
 forcebalance::psi4io::RDVR3\_Psi4, 484  
 forcebalance::psi4io::THCDF\_Psi4, 513  
 forcebalance::target::RemoteTarget, 493  
 forcebalance::target::Target, 502  
 forcebalance::tinkerio::AbInitio\_TINKER, 186  
 forcebalance::tinkerio::BindingEnergy\_TINKER, 211  
 forcebalance::tinkerio::Interaction\_TINKER, 306  
 forcebalance::tinkerio::Liquid\_TINKER, 372  
 forcebalance::tinkerio::Moments\_TINKER, 436  
 forcebalance::vibration::Vibration, 528  
**xyz\_omms**  
     forcebalance::openmmio::AbInitio\_OpenMM, 167  
**xyzs**  
     forcebalance::counterpoise::Counterpoise, 223  
**y**  
     forcebalance::Mol2::mol2\_atom, 381  
**z**  
     forcebalance::Mol2::mol2\_atom, 381