

GoSUM core

3.0

Generated by Doxygen 1.8.1.2

Sat Jan 4 2014 21:47:16

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Class Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	9
4.1	File List	9
5	Namespace Documentation	11
5.1	boost Namespace Reference	11
5.2	boost::serialization Namespace Reference	11
5.3	GoSUM Namespace Reference	11
5.3.1	Detailed Description	11
6	Class Documentation	13
6.1	GoSUM::CA analyticalModel Class Reference	13
6.1.1	Detailed Description	14
6.1.2	Member Enumeration Documentation	14
6.1.2.1	svmtyp e	14
6.1.3	Constructor & Destructor Documentation	15
6.1.3.1	CA analyticalModel	15
6.1.3.2	CA analyticalModel	15
6.1.3.3	~CA analyticalModel	15
6.1.4	Member Function Documentation	15
6.1.4.1	clear	15
6.1.4.2	empty	15
6.1.4.3	exportExpectedCurve	15
6.1.4.4	exportIntersectionCurve	15
6.1.4.5	inputSamples	15

6.1.4.6	learn	15
6.1.4.7	outputSamples	15
6.1.4.8	predict	15
6.1.4.9	predictDerivative	15
6.1.4.10	predictDerivatives	15
6.1.4.11	predictValue	16
6.1.4.12	predictValues	16
6.1.4.13	serialize	16
6.1.4.14	setInputSamples	16
6.1.4.15	setProgressSlot	16
6.1.4.16	setSvmType	16
6.1.4.17	svmType	16
6.1.5	Friends And Related Function Documentation	16
6.1.5.1	boost::serialization::access	16
6.1.6	Member Data Documentation	16
6.1.6.1	etype	16
6.1.6.2	pIP	16
6.1.6.3	pOS	16
6.1.6.4	progressSlot	16
6.1.6.5	sam	17
6.1.6.6	X	17
6.1.6.7	Y	17
6.2	GoSUM::CA analyticalOptimizationProblem Class Reference	17
6.2.1	Detailed Description	18
6.2.2	Constructor & Destructor Documentation	18
6.2.2.1	CA analyticalOptimizationProblem	18
6.2.2.2	CA analyticalOptimizationProblem	18
6.2.2.3	~CA analyticalOptimizationProblem	18
6.2.3	Member Function Documentation	18
6.2.3.1	inputPoint2ModelPoint	18
6.2.3.2	serialize	18
6.2.4	Friends And Related Function Documentation	18
6.2.4.1	boost::serialization::access	18
6.2.5	Member Data Documentation	18
6.2.5.1	pAM	18
6.3	CBaseRNG Class Reference	19
6.3.1	Detailed Description	19
6.3.2	Member Function Documentation	19
6.3.2.1	rnd	19
6.3.2.2	rnd	19

6.3.2.3	rndi	19
6.3.2.4	setSeed	19
6.4	CCategoricalDRV Class Reference	20
6.4.1	Detailed Description	21
6.4.2	Constructor & Destructor Documentation	21
6.4.2.1	CCategoricalDRV	21
6.4.2.2	~CCategoricalDRV	21
6.4.2.3	CCategoricalDRV	21
6.4.3	Member Function Documentation	21
6.4.3.1	cumulative	21
6.4.3.2	distributionName	21
6.4.3.3	distributionType	21
6.4.3.4	doQuantile	22
6.4.3.5	expandedSize	22
6.4.3.6	expectedValue	22
6.4.3.7	exportDomain	22
6.4.3.8	isDistributionDefined	22
6.4.3.9	maxValue	22
6.4.3.10	minValue	22
6.4.3.11	probability	22
6.4.3.12	serialize	22
6.4.3.13	setDistribution	22
6.4.3.14	variance	23
6.4.4	Friends And Related Function Documentation	23
6.4.4.1	boost::serialization::access	23
6.4.5	Member Data Documentation	23
6.4.5.1	cdf	23
6.4.5.2	p	23
6.5	CCategoricalSample Class Reference	23
6.5.1	Detailed Description	24
6.5.2	Constructor & Destructor Documentation	24
6.5.2.1	CCategoricalSample	24
6.5.2.2	CCategoricalSample	24
6.5.2.3	~CCategoricalSample	24
6.5.3	Member Function Documentation	24
6.5.3.1	maxValue	24
6.5.3.2	minValue	24
6.5.3.3	readSampleValue	25
6.5.3.4	serialize	25
6.5.3.5	setSampleSize	25

6.5.3.6	variance	25
6.5.3.7	writeSampleValue	25
6.5.4	Friends And Related Function Documentation	25
6.5.4.1	boost::serialization::access	25
6.5.5	Member Data Documentation	25
6.5.5.1	categories	25
6.6	CConstantCRV Class Reference	25
6.6.1	Detailed Description	27
6.6.2	Constructor & Destructor Documentation	27
6.6.2.1	CConstantCRV	27
6.6.2.2	~CConstantCRV	27
6.6.2.3	CConstantCRV	27
6.6.3	Member Function Documentation	27
6.6.3.1	constantValue	27
6.6.3.2	cumulative	27
6.6.3.3	distributionName	27
6.6.3.4	distributionType	27
6.6.3.5	doQuantile	27
6.6.3.6	expectedValue	28
6.6.3.7	exportDomain	28
6.6.3.8	maxValue	28
6.6.3.9	minValue	28
6.6.3.10	probability	28
6.6.3.11	serialize	28
6.6.3.12	setConstantValue	28
6.6.3.13	setDistribution	28
6.6.3.14	setDistribution	28
6.6.3.15	variance	28
6.6.4	Friends And Related Function Documentation	28
6.6.4.1	boost::serialization::access	28
6.6.5	Member Data Documentation	29
6.6.5.1	xc	29
6.7	CConstantDRV Class Reference	29
6.7.1	Detailed Description	30
6.7.2	Constructor & Destructor Documentation	30
6.7.2.1	CConstantDRV	30
6.7.2.2	~CConstantDRV	30
6.7.2.3	CConstantDRV	30
6.7.3	Member Function Documentation	30
6.7.3.1	constantValue	30

6.7.3.2	cumulative	30
6.7.3.3	distributionName	31
6.7.3.4	distributionType	31
6.7.3.5	doQuantile	31
6.7.3.6	expectedValue	31
6.7.3.7	exportDomain	31
6.7.3.8	maxValue	31
6.7.3.9	minValue	31
6.7.3.10	probability	31
6.7.3.11	serialize	31
6.7.3.12	setConstantValue	31
6.7.3.13	setDistribution	31
6.7.3.14	setDistribution	32
6.7.3.15	variance	32
6.7.4	Friends And Related Function Documentation	32
6.7.4.1	boost::serialization::access	32
6.7.5	Member Data Documentation	32
6.7.5.1	c	32
6.8	GoSUM::CConstraints Class Reference	32
6.8.1	Detailed Description	33
6.8.2	Constructor & Destructor Documentation	34
6.8.2.1	CConstraints	34
6.8.2.2	~CConstraints	34
6.8.3	Member Function Documentation	34
6.8.3.1	addExpression	34
6.8.3.2	clear	34
6.8.3.3	constraintsSatisfied	34
6.8.3.4	eraseExpression	34
6.8.3.5	evaluate	34
6.8.3.6	expression	34
6.8.3.7	findInExpressions	34
6.8.3.8	parseExpressions	34
6.8.3.9	roundoffEquality	34
6.8.3.10	serialize	34
6.8.3.11	setExpression	35
6.8.3.12	setVariableNames	35
6.8.3.13	size	35
6.8.3.14	validateExpressions	35
6.8.4	Friends And Related Function Documentation	35
6.8.4.1	boost::serialization::access	35

6.8.5	Member Data Documentation	35
6.8.5.1	f	35
6.8.5.2	g	35
6.8.5.3	gexpr	35
6.8.5.4	names	35
6.9	GoSUM::CContainer Class Reference	35
6.9.1	Detailed Description	43
6.9.2	Member Enumeration Documentation	43
6.9.2.1	optimizationmethodtype	43
6.9.2.2	projecttype	43
6.9.3	Constructor & Destructor Documentation	43
6.9.3.1	CContainer	43
6.9.3.2	~CContainer	43
6.9.4	Member Function Documentation	43
6.9.4.1	addDefaultVariable	43
6.9.4.2	addInput	44
6.9.4.3	addInputs	44
6.9.4.4	addOptimizationConstraint	44
6.9.4.5	addOutput	44
6.9.4.6	addOutputs	44
6.9.4.7	addVariable	44
6.9.4.8	addVariable	44
6.9.4.9	addVariables	44
6.9.4.10	addVariables	44
6.9.4.11	analyticalModel	44
6.9.4.12	clear	44
6.9.4.13	clearForReducing	45
6.9.4.14	clearLearningResults	45
6.9.4.15	clearOptimizationResults	45
6.9.4.16	clearResults	45
6.9.4.17	clearSamplingResults	45
6.9.4.18	clearSensitivityResults	45
6.9.4.19	cloneVariable	45
6.9.4.20	computeSensitivities	45
6.9.4.21	containsDeclaredEmpiricalViarables	45
6.9.4.22	containsEmpiricalViarables	45
6.9.4.23	containsNamedDeclaredEmpiricalViarables	45
6.9.4.24	containsNamedEmpiricalViarables	46
6.9.4.25	containsNamedTheoreticalViarables	46
6.9.4.26	containsPredictionSamples	46

6.9.4.27	containsTheoreticalVariables	46
6.9.4.28	emptyLearningResults	46
6.9.4.29	emptyOptimizationResults	46
6.9.4.30	emptyResults	46
6.9.4.31	emptySamplingResults	46
6.9.4.32	emptySelectedSamples	46
6.9.4.33	emptySensitivityResults	46
6.9.4.34	eraseSelectedSamples	46
6.9.4.35	eraseVariable	46
6.9.4.36	evaluateOutputs	47
6.9.4.37	exportAbsoluteAverageDerivative	47
6.9.4.38	exportAverageDerivative	47
6.9.4.39	exportDerivativeSensitivity	47
6.9.4.40	exportFirstOrderANOVA	47
6.9.4.41	exportInputSamples	47
6.9.4.42	exportOptimizationHistory	47
6.9.4.43	exportOptimizationMethod	47
6.9.4.44	exportOutputSamples	47
6.9.4.45	exportPredictionOutputSamples	47
6.9.4.46	exportSamples	47
6.9.4.47	exportVarianceSensitivity	47
6.9.4.48	gaOptimizer	48
6.9.4.49	hyperCube	48
6.9.4.50	importDeclaredEmpiricalVariables	48
6.9.4.51	importEmpiricalVariables	48
6.9.4.52	importInitialValue	48
6.9.4.53	importInputs	48
6.9.4.54	importLowerBound	48
6.9.4.55	importModelConstraints	48
6.9.4.56	importNamedDeclaredEmpiricalVariables	48
6.9.4.57	importNamedEmpiricalVariables	48
6.9.4.58	importNamedTheoreticalVariables	48
6.9.4.59	importOptimizationConstraints	49
6.9.4.60	importOutputs	49
6.9.4.61	importPredictionInputSamples	49
6.9.4.62	importTheoreticalVariables	49
6.9.4.63	importUpperBound	49
6.9.4.64	importVariables	49
6.9.4.65	inputParameter	49
6.9.4.66	inputParameters	49

6.9.4.67	inputsSize	49
6.9.4.68	isSampleSelected	49
6.9.4.69	join	49
6.9.4.70	learnEvaluationSize	49
6.9.4.71	learningStepsSize	50
6.9.4.72	learnModel	50
6.9.4.73	load	50
6.9.4.74	loadTxt	50
6.9.4.75	loadXml	50
6.9.4.76	longProjectName	50
6.9.4.77	maxOptimizer	50
6.9.4.78	maximize	50
6.9.4.79	minimize	50
6.9.4.80	modelConstraints	50
6.9.4.81	modelEvaluator	50
6.9.4.82	optimizationMethod	50
6.9.4.83	OptimizationMethodType	51
6.9.4.84	optimizationProblem	51
6.9.4.85	optimizationStepsSize	51
6.9.4.86	optimize	51
6.9.4.87	outputsSize	51
6.9.4.88	outputState	51
6.9.4.89	outputStates	51
6.9.4.90	predict	51
6.9.4.91	predictMean	51
6.9.4.92	projectName	51
6.9.4.93	projectPath	51
6.9.4.94	ProjectType	51
6.9.4.95	projectType	52
6.9.4.96	readyToComputeSensitivities	52
6.9.4.97	readyToEvaluateOutputs	52
6.9.4.98	readyToLearnModel	52
6.9.4.99	readyToOptimize	52
6.9.4.100	readyToReduceModel	52
6.9.4.101	readyToSampleInputs	52
6.9.4.102	readyToSetOptimization	52
6.9.4.103	reduce	52
6.9.4.104	reducer	52
6.9.4.105	resampleInputs	52
6.9.4.106	resetOptimizationVariable	52

6.9.4.107	save	53
6.9.4.108	saveTxt	53
6.9.4.109	saveXml	53
6.9.4.110	selectedSamples	53
6.9.4.111	selectSamples	53
6.9.4.112	sensitivityAnalysis	53
6.9.4.113	separateBySelection	53
6.9.4.114	serialize	53
6.9.4.115	setInitialValue	53
6.9.4.116	setLowerBound	53
6.9.4.117	setMadsInitMeshSize	53
6.9.4.118	setMadsLHSearch	53
6.9.4.119	setMadsMaxEvaluation	53
6.9.4.120	setMadsMinPollSize	54
6.9.4.121	setMatLabPath	54
6.9.4.122	setModelEvaluator	54
6.9.4.123	setObjective	54
6.9.4.124	setOptimizationMethod	54
6.9.4.125	setProjectName	54
6.9.4.126	setProjectPath	54
6.9.4.127	setProjectType	54
6.9.4.128	setReductionCutoffSize	54
6.9.4.129	setReductionCutoffValue	54
6.9.4.130	setReductionOutputs	54
6.9.4.131	setReductionType	54
6.9.4.132	setResampleSize	55
6.9.4.133	setResampleType	55
6.9.4.134	setRNG	55
6.9.4.135	setSensitivityOptions	55
6.9.4.136	setThreadSize	55
6.9.4.137	setUpperBound	55
6.9.4.138	setVoronoiOptions	55
6.9.4.139	startEvaluating	55
6.9.4.140	startLearning	55
6.9.4.141	startOptimizing	55
6.9.4.142	startResampling	55
6.9.4.143	startSensitivityComputing	55
6.9.5	Friends And Related Function Documentation	56
6.9.5.1	boost::serialization::access	56
6.9.6	Member Data Documentation	56

6.9.6.1	analytical	56
6.9.6.2	evaluatingFinished	56
6.9.6.3	evaluator	56
6.9.6.4	GAMO	56
6.9.6.5	hycube	56
6.9.6.6	inconsts	56
6.9.6.7	inputs	56
6.9.6.8	learningFinished	56
6.9.6.9	MADS	56
6.9.6.10	omType	56
6.9.6.11	optimizingFinished	57
6.9.6.12	outputs	57
6.9.6.13	pOP	57
6.9.6.14	prjName	57
6.9.6.15	prjPath	57
6.9.6.16	prjType	57
6.9.6.17	reduced	57
6.9.6.18	resamplingFinished	57
6.9.6.19	selectedsamples	57
6.9.6.20	sensitivity	57
6.9.6.21	sensitivityComputingFinished	57
6.9.6.22	thrd	57
6.10	GoSUM::CContinuousMV Class Reference	58
6.10.1	Constructor & Destructor Documentation	58
6.10.1.1	CContinuousMV	58
6.10.1.2	CContinuousMV	59
6.10.1.3	CContinuousMV	59
6.10.2	Member Function Documentation	59
6.10.2.1	hasDiscreteData	59
6.10.2.2	isContinuous	59
6.10.2.3	isDiscrete	59
6.10.2.4	serialize	59
6.10.2.5	setEmpiricalDistribution	59
6.10.2.6	setTheoreticalDistribution	59
6.10.3	Friends And Related Function Documentation	59
6.10.3.1	boost::serialization::access	59
6.11	CContinuousRV Class Reference	59
6.11.1	Constructor & Destructor Documentation	60
6.11.1.1	CContinuousRV	60
6.11.1.2	CContinuousRV	60

6.11.2	Member Function Documentation	60
6.11.2.1	probability	60
6.11.2.2	quantile	60
6.11.2.3	serialize	60
6.11.3	Friends And Related Function Documentation	60
6.11.3.1	boost::serialization::access	60
6.12	CContinuousSample Class Reference	61
6.12.1	Constructor & Destructor Documentation	62
6.12.1.1	CContinuousSample	62
6.12.1.2	~CContinuousSample	62
6.12.1.3	CContinuousSample	62
6.12.2	Member Function Documentation	62
6.12.2.1	computeMeanAndStandardDeviation	62
6.12.2.2	computeNormalizedHistogram	62
6.12.2.3	computeNormalizedSubHistogram	62
6.12.2.4	computeStatistics	62
6.12.2.5	exportHistogram	62
6.12.2.6	exportSubHistogram	63
6.12.2.7	hasDiscreteData	63
6.12.2.8	mean	63
6.12.2.9	serialize	63
6.12.2.10	standardDeviation	63
6.12.2.11	variance	63
6.12.3	Friends And Related Function Documentation	63
6.12.3.1	boost::serialization::access	63
6.12.4	Member Data Documentation	63
6.12.4.1	mu	63
6.12.4.2	stdv	63
6.13	GoSUM::CCSvcSAM Class Reference	63
6.13.1	Detailed Description	64
6.13.2	Constructor & Destructor Documentation	64
6.13.2.1	CCSvcSAM	64
6.13.2.2	~CCSvcSAM	64
6.13.3	Member Function Documentation	64
6.13.3.1	serialize	64
6.13.4	Friends And Related Function Documentation	64
6.13.4.1	boost::serialization::access	64
6.14	GoSUM::CCVoronoiHC Class Reference	65
6.14.1	Constructor & Destructor Documentation	65
6.14.1.1	CCVoronoiHC	65

6.14.1.2	CCVoronoiHC	65
6.14.1.3	~CCVoronoiHC	65
6.14.2	Member Function Documentation	65
6.14.2.1	centralize	65
6.14.2.2	doGenerate	66
6.14.2.3	serialize	66
6.14.3	Friends And Related Function Documentation	66
6.14.3.1	boost::serialization::access	66
6.15	GoSUM::CDiscreteMV Class Reference	66
6.15.1	Constructor & Destructor Documentation	67
6.15.1.1	CDiscreteMV	67
6.15.1.2	CDiscreteMV	67
6.15.1.3	CDiscreteMV	67
6.15.1.4	CDiscreteMV	67
6.15.2	Member Function Documentation	67
6.15.2.1	isContinuous	67
6.15.2.2	isDiscrete	67
6.15.2.3	serialize	67
6.15.2.4	setEmpiricalDistribution	67
6.15.2.5	setTheoreticalDistribution	67
6.15.3	Friends And Related Function Documentation	68
6.15.3.1	boost::serialization::access	68
6.16	CDiscreteRV Class Reference	68
6.16.1	Constructor & Destructor Documentation	68
6.16.1.1	CDiscreteRV	68
6.16.1.2	CDiscreteRV	68
6.16.2	Member Function Documentation	68
6.16.2.1	quantile	69
6.16.2.2	serialize	69
6.16.3	Friends And Related Function Documentation	69
6.16.3.1	boost::serialization::access	69
6.17	CDiscreteSample Class Reference	69
6.17.1	Constructor & Destructor Documentation	70
6.17.1.1	CDiscreteSample	70
6.17.1.2	~CDiscreteSample	70
6.17.1.3	CDiscreteSample	70
6.17.2	Member Function Documentation	70
6.17.2.1	computeNormalizedHistogram	70
6.17.2.2	computeNormalizedSubHistogram	70
6.17.2.3	computeStatistics	70

6.17.2.4	exportHistogram	70
6.17.2.5	exportSubHistogram	70
6.17.2.6	serialize	71
6.17.2.7	variance	71
6.17.3	Friends And Related Function Documentation	71
6.17.3.1	boost::serialization::access	71
6.18	GoSUM::CDSampleHC Class Reference	71
6.18.1	Constructor & Destructor Documentation	72
6.18.1.1	CDSampleHC	72
6.18.1.2	CDSampleHC	72
6.18.1.3	~CDSampleHC	72
6.18.2	Member Function Documentation	72
6.18.2.1	doGenerate	72
6.18.2.2	serialize	72
6.18.3	Friends And Related Function Documentation	72
6.18.3.1	boost::serialization::access	72
6.19	CDynamicSystemRNG Class Reference	72
6.19.1	Detailed Description	73
6.19.2	Member Typedef Documentation	73
6.19.2.1	Long	73
6.19.3	Constructor & Destructor Documentation	73
6.19.3.1	CDynamicSystemRNG	73
6.19.3.2	CDynamicSystemRNG	73
6.19.3.3	~CDynamicSystemRNG	73
6.19.4	Member Function Documentation	74
6.19.4.1	prepare	74
6.19.4.2	rnd	74
6.19.4.3	rndi	74
6.19.4.4	setSeed	74
6.19.5	Member Data Documentation	74
6.19.5.1	i	74
6.19.5.2	j	74
6.19.5.3	k	74
6.19.5.4	n	74
6.19.5.5	w	74
6.19.5.6	x	74
6.20	CEmpiricalCRV Class Reference	75
6.20.1	Detailed Description	76
6.20.2	Constructor & Destructor Documentation	76
6.20.2.1	CEmpiricalCRV	76

6.20.2.2	~CEmpiricalCRV	76
6.20.2.3	CEmpiricalCRV	76
6.20.3	Member Function Documentation	76
6.20.3.1	cumulative	76
6.20.3.2	distributionName	76
6.20.3.3	distributionType	77
6.20.3.4	doQuantile	77
6.20.3.5	expectedValue	77
6.20.3.6	exportDomain	77
6.20.3.7	isDistributionDefined	77
6.20.3.8	maxValue	77
6.20.3.9	mean	77
6.20.3.10	minValue	77
6.20.3.11	probability	77
6.20.3.12	serialize	77
6.20.3.13	setDistribution	78
6.20.3.14	standardDeviation	78
6.20.3.15	variance	78
6.20.4	Friends And Related Function Documentation	78
6.20.4.1	boost::serialization::access	78
6.20.5	Member Data Documentation	78
6.20.5.1	cdf	78
6.20.5.2	mu	78
6.20.5.3	p	78
6.20.5.4	pbandwidth	78
6.20.5.5	qbandwidth	78
6.20.5.6	stdv	78
6.20.5.7	x	78
6.21	CEmpiricalDRV Class Reference	78
6.21.1	Detailed Description	80
6.21.2	Constructor & Destructor Documentation	80
6.21.2.1	CEmpiricalDRV	80
6.21.2.2	~CEmpiricalDRV	80
6.21.2.3	CEmpiricalDRV	80
6.21.3	Member Function Documentation	80
6.21.3.1	cumulative	80
6.21.3.2	distributionName	80
6.21.3.3	distributionType	80
6.21.3.4	doQuantile	80
6.21.3.5	expandedSize	80

6.21.3.6	expectedValue	81
6.21.3.7	exportDomain	81
6.21.3.8	isDistributionDefined	81
6.21.3.9	maxValue	81
6.21.3.10	minValue	81
6.21.3.11	probability	81
6.21.3.12	serialize	81
6.21.3.13	setDistribution	81
6.21.3.14	variance	81
6.21.4	Friends And Related Function Documentation	81
6.21.4.1	boost::serialization::access	81
6.21.5	Member Data Documentation	82
6.21.5.1	cdf	82
6.21.5.2	p	82
6.22	GoSUM::CEpsSvrSAM Class Reference	82
6.22.1	Detailed Description	83
6.22.2	Constructor & Destructor Documentation	83
6.22.2.1	CEpsSvrSAM	83
6.22.2.2	~CEpsSvrSAM	83
6.22.3	Member Function Documentation	83
6.22.3.1	openOptimization	83
6.22.3.2	optimizationPoint2SVMParam	83
6.22.3.3	serialize	83
6.22.4	Friends And Related Function Documentation	83
6.22.4.1	boost::serialization::access	83
6.23	GoSUM::CEvaluator Class Reference	83
6.23.1	Detailed Description	85
6.23.2	Member Enumeration Documentation	85
6.23.2.1	evaluatorType	85
6.23.3	Constructor & Destructor Documentation	85
6.23.3.1	CEvaluator	85
6.23.3.2	~CEvaluator	85
6.23.4	Member Function Documentation	85
6.23.4.1	clear	85
6.23.4.2	ExternalEvaluator	85
6.23.4.3	ExternalEvaluatorFolder	86
6.23.4.4	New	86
6.23.4.5	serialize	86
6.23.4.6	SetExternalEvaluator	86
6.23.4.7	SetThreadSize	86

6.23.4.8	SetType	86
6.23.4.9	ThreadSize	86
6.23.4.10	Type	86
6.23.4.11	Type	86
6.23.5	Friends And Related Function Documentation	86
6.23.5.1	boost::serialization::access	86
6.23.6	Member Data Documentation	86
6.23.6.1	evaluatingProgressed	86
6.23.6.2	exe	87
6.23.6.3	ext	87
6.23.6.4	in	87
6.23.6.5	out	87
6.23.6.6	path	87
6.23.6.7	trdN	87
6.23.6.8	type	87
6.24	GoSUM::CExeAsciiEvaluator Class Reference	87
6.24.1	Detailed Description	88
6.24.2	Constructor & Destructor Documentation	88
6.24.2.1	CExeAsciiEvaluator	88
6.24.2.2	CExeAsciiEvaluator	88
6.24.2.3	~CExeAsciiEvaluator	88
6.24.3	Member Function Documentation	88
6.24.3.1	readFrom	88
6.24.3.2	readSizeFrom	88
6.24.3.3	serialize	88
6.24.3.4	writeTo	88
6.24.4	Friends And Related Function Documentation	88
6.24.4.1	boost::serialization::access	88
6.25	GoSUM::CExeEvaluator Class Reference	89
6.25.1	Detailed Description	90
6.25.2	Constructor & Destructor Documentation	90
6.25.2.1	CExeEvaluator	90
6.25.2.2	CExeEvaluator	90
6.25.2.3	~CExeEvaluator	90
6.25.3	Member Function Documentation	90
6.25.3.1	closeEvaluation	90
6.25.3.2	evaluateOutputsSize	90
6.25.3.3	openEvaluation	90
6.25.3.4	operator()	90
6.25.3.5	readFrom	90

6.25.3.6	readSizeFrom	91
6.25.3.7	serialize	91
6.25.3.8	systemCommand	91
6.25.3.9	writeTo	91
6.25.4	Friends And Related Function Documentation	91
6.25.4.1	boost::serialization::access	91
6.25.5	Member Data Documentation	91
6.25.5.1	pP	91
6.26	GoSUM::CExeMatEvaluator Class Reference	91
6.26.1	Detailed Description	92
6.26.2	Constructor & Destructor Documentation	92
6.26.2.1	CExeMatEvaluator	92
6.26.2.2	CExeMatEvaluator	92
6.26.2.3	~CExeMatEvaluator	92
6.26.3	Member Function Documentation	93
6.26.3.1	readFrom	93
6.26.3.2	readSizeFrom	93
6.26.3.3	serialize	93
6.26.3.4	writeTo	93
6.26.4	Friends And Related Function Documentation	93
6.26.4.1	boost::serialization::access	93
6.26.5	Member Data Documentation	93
6.26.5.1	matlab	93
6.27	CExponentialCRV Class Reference	93
6.27.1	Detailed Description	95
6.27.2	Constructor & Destructor Documentation	95
6.27.2.1	CExponentialCRV	95
6.27.2.2	~CExponentialCRV	95
6.27.2.3	CExponentialCRV	95
6.27.3	Member Function Documentation	95
6.27.3.1	cumulative	95
6.27.3.2	distributionName	95
6.27.3.3	distributionType	95
6.27.3.4	doQuantile	95
6.27.3.5	expectedValue	95
6.27.3.6	exportDomain	96
6.27.3.7	maxValue	96
6.27.3.8	minValue	96
6.27.3.9	probability	96
6.27.3.10	rateParameter	96

6.27.3.11	serialize	96
6.27.3.12	setDistribution	96
6.27.3.13	setDistribution	96
6.27.3.14	setRateParameter	96
6.27.3.15	variance	96
6.27.4	Friends And Related Function Documentation	96
6.27.4.1	boost::serialization::access	96
6.27.5	Member Data Documentation	97
6.27.5.1	beta	97
6.27.5.2	lambda	97
6.28	CFFTW Class Reference	97
6.28.1	Detailed Description	97
6.28.2	Constructor & Destructor Documentation	97
6.28.2.1	CFFTW	97
6.28.2.2	~CFFTW	97
6.28.3	Member Function Documentation	97
6.28.3.1	discreteCosineTransform	97
6.28.3.2	inverseDiscreteCosineTransform	97
6.29	CGAModelOptimization Class Reference	98
6.29.1	Detailed Description	99
6.29.2	Constructor & Destructor Documentation	99
6.29.2.1	CGAModelOptimization	99
6.29.2.2	~CGAModelOptimization	99
6.29.3	Member Function Documentation	99
6.29.3.1	Fitness	99
6.29.3.2	GAGenome2ArrayXd	99
6.29.3.3	optimize	99
6.29.3.4	populationSize	99
6.29.3.5	progressStepsSize	99
6.29.3.6	serialize	99
6.29.3.7	setPopulationSize	99
6.29.4	Friends And Related Function Documentation	99
6.29.4.1	boost::serialization::access	99
6.29.5	Member Data Documentation	99
6.29.5.1	C	99
6.29.5.2	pOP	100
6.29.5.3	popsiz	100
6.29.5.4	Xbest	100
6.29.5.5	ybest	100
6.30	CGaussianCRV Class Reference	100

6.30.1 Detailed Description	101
6.30.2 Constructor & Destructor Documentation	102
6.30.2.1 CGaussianCRV	102
6.30.2.2 ~CGaussianCRV	102
6.30.2.3 CGaussianCRV	102
6.30.3 Member Function Documentation	102
6.30.3.1 cumulative	102
6.30.3.2 distributionName	102
6.30.3.3 distributionType	102
6.30.3.4 doQuantile	102
6.30.3.5 expectedValue	102
6.30.3.6 exportDomain	102
6.30.3.7 maxValue	102
6.30.3.8 mean	102
6.30.3.9 minValue	103
6.30.3.10 probability	103
6.30.3.11 serialize	103
6.30.3.12 setDistribution	103
6.30.3.13 setDistribution	103
6.30.3.14 setMean	103
6.30.3.15 setStandardDeviation	103
6.30.3.16 standardDeviation	103
6.30.3.17 variance	103
6.30.4 Friends And Related Function Documentation	103
6.30.4.1 boost::serialization::access	103
6.30.5 Member Data Documentation	103
6.30.5.1 mu	103
6.30.5.2 sigma	103
6.30.5.3 sigma2	104
6.30.5.4 sigmasqrt2	104
6.31 GoSUM::CHypercube Class Reference	104
6.31.1 Member Enumeration Documentation	105
6.31.1.1 hctype	105
6.31.2 Constructor & Destructor Documentation	106
6.31.2.1 CHypercube	106
6.31.2.2 ~CHypercube	106
6.31.3 Member Function Documentation	106
6.31.3.1 CvtIterationSize	106
6.31.3.2 CvtNewCenterCoefficient	106
6.31.3.3 CvtOldCenterCoefficient	106

6.31.3.4	CvtPointsSize	106
6.31.3.5	New	106
6.31.3.6	ProgressSize	106
6.31.3.7	serialize	106
6.31.3.8	SetCvtIterationSize	106
6.31.3.9	SetCvtNewCenterCoefficient	106
6.31.3.10	SetCvtOldCenterCoefficient	106
6.31.3.11	SetCvtPointsSize	107
6.31.3.12	SetType	107
6.31.3.13	SetVoronoiOptions	107
6.31.3.14	Type	107
6.31.3.15	Type	107
6.31.3.16	VoronoiOptions	107
6.31.4	Friends And Related Function Documentation	107
6.31.4.1	boost::serialization::access	107
6.31.5	Member Data Documentation	107
6.31.5.1	alpha2	107
6.31.5.2	beta2	107
6.31.5.3	etype	107
6.31.5.4	generatingProgressed	108
6.31.5.5	maxiter	108
6.31.5.6	q	108
6.32	GoSUM::CInputParameters Class Reference	108
6.32.1	Detailed Description	109
6.32.2	Constructor & Destructor Documentation	109
6.32.2.1	CInputParameters	109
6.32.2.2	CInputParameters	109
6.32.2.3	~CInputParameters	109
6.32.3	Member Function Documentation	109
6.32.3.1	constraints	109
6.32.3.2	generateSamples	109
6.32.3.3	generateSamples	109
6.32.3.4	resampleSize	109
6.32.3.5	resampleStepsSize	110
6.32.3.6	serialize	110
6.32.3.7	setProgressSlot	110
6.32.3.8	setResampleSize	110
6.32.4	Friends And Related Function Documentation	110
6.32.4.1	boost::serialization::access	110
6.32.5	Member Data Documentation	110

6.32.5.1	plC	110
6.32.5.2	progressSlot	110
6.32.5.3	RSize	110
6.33	CLargePeriodRNG Class Reference	110
6.33.1	Detailed Description	111
6.33.2	Member Typedef Documentation	111
6.33.2.1	Ullong	111
6.33.3	Constructor & Destructor Documentation	111
6.33.3.1	CLargePeriodRNG	111
6.33.3.2	CLargePeriodRNG	111
6.33.3.3	~CLargePeriodRNG	111
6.33.4	Member Function Documentation	111
6.33.4.1	int64	111
6.33.4.2	rnd	111
6.33.4.3	rndi	112
6.33.4.4	setSeed	112
6.33.5	Member Data Documentation	112
6.33.5.1	u	112
6.33.5.2	v	112
6.33.5.3	w	112
6.34	GoSUM::CLCVoronoiHC Class Reference	112
6.34.1	Constructor & Destructor Documentation	113
6.34.1.1	CLCVoronoiHC	113
6.34.1.2	CLCVoronoiHC	113
6.34.1.3	~CLCVoronoiHC	113
6.34.2	Member Function Documentation	113
6.34.2.1	compcoo	113
6.34.2.2	doGenerate	113
6.34.2.3	latinize	113
6.34.2.4	serialize	114
6.34.3	Friends And Related Function Documentation	114
6.34.3.1	boost::serialization::access	114
6.34.4	Member Data Documentation	114
6.34.4.1	coo	114
6.35	CMADS Class Reference	114
6.35.1	Detailed Description	115
6.35.2	Constructor & Destructor Documentation	115
6.35.2.1	CMADS	115
6.35.2.2	~CMADS	115
6.35.3	Member Function Documentation	115

6.35.3.1	ArrayXd2NOMADPoint	115
6.35.3.2	evaluationSize	116
6.35.3.3	initaIMeshSize	116
6.35.3.4	initialLHSearch	116
6.35.3.5	iterationLHSearch	116
6.35.3.6	minimalPollSize	116
6.35.3.7	NOMADPoint2ArrayXd	116
6.35.3.8	optimize	116
6.35.3.9	progressStepsSize	116
6.35.3.10	serialize	116
6.35.3.11	setEvaluationSize	116
6.35.3.12	setInitialLHSearch	116
6.35.3.13	setInitialMeshSize	116
6.35.3.14	setIterationLHSearch	116
6.35.3.15	setMinimalPollSize	117
6.35.4	Friends And Related Function Documentation	117
6.35.4.1	boost::serialization::access	117
6.35.5	Member Data Documentation	117
6.35.5.1	ims	117
6.35.5.2	lh0	117
6.35.5.3	lhi	117
6.35.5.4	maxeval	117
6.35.5.5	mps	117
6.35.5.6	pOP	117
6.35.5.7	Xbest	117
6.35.5.8	ybest	117
6.36	CMADSEvaluator Class Reference	117
6.36.1	Detailed Description	118
6.36.2	Constructor & Destructor Documentation	118
6.36.2.1	CMADSEvaluator	118
6.36.2.2	~CMADSEvaluator	118
6.36.3	Member Function Documentation	118
6.36.3.1	eval_x	118
6.36.4	Member Data Documentation	118
6.36.4.1	pOP	118
6.37	CMATLAB Class Reference	118
6.37.1	Detailed Description	120
6.37.2	Constructor & Destructor Documentation	120
6.37.2.1	CMATLAB	120
6.37.2.2	~CMATLAB	120

6.37.3	Member Function Documentation	120
6.37.3.1	engClose	120
6.37.3.2	engEvalString	121
6.37.3.3	engGetVariable	121
6.37.3.4	engGetVisible	121
6.37.3.5	engOpen	121
6.37.3.6	engOpenSingleUse	121
6.37.3.7	engOutputBuffer	121
6.37.3.8	engPutVariable	121
6.37.3.9	engSetVisible	121
6.37.3.10	matClose	121
6.37.3.11	matGet	121
6.37.3.12	matGetVariable	121
6.37.3.13	matOpen	121
6.37.3.14	matPut	122
6.37.3.15	matPutVariable	122
6.37.3.16	mxCreateDoubleMatrix	122
6.37.3.17	mxDestroyArray	122
6.37.3.18	mxGetNumberOfElements	122
6.37.3.19	mxGetPr	122
6.37.3.20	Path	122
6.37.3.21	SetPath	122
6.37.4	Member Data Documentation	122
6.37.4.1	libeng	122
6.37.4.2	libmat	122
6.37.4.3	libmx	122
6.37.4.4	path	122
6.37.4.5	pEngClose	123
6.37.4.6	pEngEvalString	123
6.37.4.7	pEngGetVariable	123
6.37.4.8	pEngGetVisible	123
6.37.4.9	pEngOpen	123
6.37.4.10	pEngOpenSingleUse	123
6.37.4.11	pEngOutputBuffer	123
6.37.4.12	pEngPutVariable	123
6.37.4.13	pEngSetVisible	123
6.37.4.14	pMatClose	123
6.37.4.15	pMatGetVariable	123
6.37.4.16	pMatOpen	123
6.37.4.17	pMatPutVariable	124

6.37.4.18	pMxCreateDoubleMatrix	124
6.37.4.19	pMxDestroyArray	124
6.37.4.20	pMxGetNumberOfElements	124
6.37.4.21	pMxGetPr	124
6.37.4.22	pMxSetPr	124
6.38	GoSUM::CMatlabEngineEvaluator Class Reference	124
6.38.1	Detailed Description	125
6.38.2	Constructor & Destructor Documentation	125
6.38.2.1	CMatlabEngineEvaluator	125
6.38.2.2	CMatlabEngineEvaluator	125
6.38.2.3	~CMatlabEngineEvaluator	125
6.38.3	Member Function Documentation	125
6.38.3.1	closeEvaluation	125
6.38.3.2	evaluateOutputsSize	125
6.38.3.3	openEvaluation	126
6.38.3.4	operator()	126
6.38.3.5	serialize	126
6.38.4	Friends And Related Function Documentation	126
6.38.4.1	boost::serialization::access	126
6.38.5	Member Data Documentation	126
6.38.5.1	ep	126
6.38.5.2	matlab	126
6.39	GoSUM::CMatlabShellEvaluator Class Reference	126
6.39.1	Detailed Description	127
6.39.2	Constructor & Destructor Documentation	127
6.39.2.1	CMatlabShellEvaluator	127
6.39.2.2	CMatlabShellEvaluator	127
6.39.2.3	~CMatlabShellEvaluator	127
6.39.3	Member Function Documentation	127
6.39.3.1	serialize	127
6.39.3.2	systemCommand	128
6.39.4	Friends And Related Function Documentation	128
6.39.4.1	boost::serialization::access	128
6.40	CMediumPeriodRNG Class Reference	128
6.40.1	Constructor & Destructor Documentation	129
6.40.1.1	CMediumPeriodRNG	129
6.40.1.2	CMediumPeriodRNG	129
6.40.1.3	~CMediumPeriodRNG	129
6.40.2	Member Function Documentation	129
6.40.2.1	rnd	129

6.40.2.2	rndi	129
6.40.2.3	setSeed	129
6.40.3	Member Data Documentation	129
6.40.3.1	idum	129
6.40.3.2	idum2	129
6.40.3.3	iv2	129
6.40.3.4	iy2	129
6.41	GoSUM::CModelConstraints Class Reference	130
6.41.1	Constructor & Destructor Documentation	130
6.41.1.1	CModelConstraints	130
6.41.1.2	~CModelConstraints	130
6.41.2	Member Function Documentation	130
6.41.2.1	clear	130
6.41.2.2	isConstrained	131
6.41.2.3	serialize	131
6.41.2.4	set	131
6.41.2.5	setVariableNames	131
6.41.3	Friends And Related Function Documentation	131
6.41.3.1	boost::serialization::access	131
6.41.4	Member Data Documentation	131
6.41.4.1	pIP	131
6.41.4.2	status	131
6.42	GoSUM::CModelEvaluator Class Reference	131
6.42.1	Detailed Description	132
6.42.2	Constructor & Destructor Documentation	133
6.42.2.1	CModelEvaluator	133
6.42.2.2	CModelEvaluator	133
6.42.2.3	~CModelEvaluator	133
6.42.3	Member Function Documentation	133
6.42.3.1	closeEvaluation	133
6.42.3.2	evaluateOutputsSize	133
6.42.3.3	openEvaluation	133
6.42.3.4	operator()	133
6.42.3.5	operator()	133
6.42.3.6	serialize	133
6.42.4	Friends And Related Function Documentation	133
6.42.4.1	boost::serialization::access	133
6.42.5	Member Data Documentation	134
6.42.5.1	I	134
6.42.5.2	pIP	134

6.42.5.3	pOS	134
6.42.5.4	trdl	134
6.43	GoSUM::CModelHypercube Class Reference	134
6.43.1	Constructor & Destructor Documentation	135
6.43.1.1	CModelHypercube	135
6.43.1.2	CModelHypercube	135
6.43.1.3	~CModelHypercube	135
6.43.2	Member Function Documentation	135
6.43.2.1	checkHCPoints	135
6.43.2.2	doGenerate	135
6.43.2.3	generate	135
6.43.2.4	generateHCPPoint	136
6.43.2.5	generateHCPoints	136
6.43.2.6	generateModelPoints	136
6.43.2.7	hcPoints2ModelPoints	136
6.43.2.8	serialize	136
6.43.3	Friends And Related Function Documentation	136
6.43.3.1	boost::serialization::access	136
6.43.4	Member Data Documentation	136
6.43.4.1	maxtries	136
6.43.4.2	pIC	136
6.43.4.3	pIP	136
6.44	GoSUM::CModelOptimizationVariable Class Reference	137
6.44.1	Constructor & Destructor Documentation	137
6.44.1.1	CModelOptimizationVariable	137
6.44.1.2	CModelOptimizationVariable	137
6.44.1.3	CModelOptimizationVariable	137
6.44.1.4	~CModelOptimizationVariable	137
6.44.2	Member Function Documentation	137
6.44.2.1	inputParameter	137
6.44.2.2	reset	138
6.44.2.3	serialize	138
6.44.3	Friends And Related Function Documentation	138
6.44.3.1	boost::serialization::access	138
6.44.4	Member Data Documentation	138
6.44.4.1	pip	138
6.45	GoSUM::CModelVariable Class Reference	138
6.45.1	Detailed Description	140
6.45.2	Constructor & Destructor Documentation	140
6.45.2.1	CModelVariable	140

6.45.2.2	~CModelVariable	140
6.45.3	Member Function Documentation	140
6.45.3.1	cast2ContinuousMV	140
6.45.3.2	cast2DiscreteMV	140
6.45.3.3	castExportSample	140
6.45.3.4	clearSample	140
6.45.3.5	clone	140
6.45.3.6	computeStatistics	141
6.45.3.7	distributionName	141
6.45.3.8	distributionType	141
6.45.3.9	expandedSize	141
6.45.3.10	expectedValue	141
6.45.3.11	generateSampleValue	141
6.45.3.12	generateSampleValue	141
6.45.3.13	isContinuous	141
6.45.3.14	isDiscrete	141
6.45.3.15	isDistributionDefined	142
6.45.3.16	maxValue	142
6.45.3.17	minValue	142
6.45.3.18	name	142
6.45.3.19	randomVariable	142
6.45.3.20	readSampleValue	142
6.45.3.21	sample	142
6.45.3.22	sampleSize	142
6.45.3.23	sampleValue	142
6.45.3.24	serialize	142
6.45.3.25	setDistribution	143
6.45.3.26	setEmpiricalDistribution	143
6.45.3.27	setName	143
6.45.3.28	setSampleSize	143
6.45.3.29	setSampleValue	143
6.45.3.30	setTheoreticalDistribution	143
6.45.3.31	writeSampleValue	143
6.45.4	Friends And Related Function Documentation	143
6.45.4.1	boost::serialization::access	143
6.45.5	Member Data Documentation	143
6.45.5.1	uname	143
6.46	GoSUM::CModelVariables Class Reference	144
6.46.1	Detailed Description	146
6.46.2	Constructor & Destructor Documentation	146

6.46.2.1	CModelVariables	146
6.46.2.2	~CModelVariables	146
6.46.3	Member Function Documentation	146
6.46.3.1	add	146
6.46.3.2	baseName	146
6.46.3.3	clear	146
6.46.3.4	clearSamples	146
6.46.3.5	computeStatistics	146
6.46.3.6	computeStatistics	147
6.46.3.7	dDenormalize	147
6.46.3.8	dDenormalize	147
6.46.3.9	denormalize	147
6.46.3.10	denormalize	147
6.46.3.11	dNormalize	147
6.46.3.12	dNormalize	147
6.46.3.13	empty	147
6.46.3.14	emptySamples	147
6.46.3.15	erase	147
6.46.3.16	eraseSelectedSamples	147
6.46.3.17	expandedNTuple	148
6.46.3.18	expandedSize	148
6.46.3.19	expandNTuple	148
6.46.3.20	find	148
6.46.3.21	hcPoint2ModelPoint	148
6.46.3.22	insert	148
6.46.3.23	isModelVariableName	148
6.46.3.24	nextName	148
6.46.3.25	normalize	148
6.46.3.26	normalize	148
6.46.3.27	nTuple	148
6.46.3.28	operator()	149
6.46.3.29	operator()	149
6.46.3.30	rename	149
6.46.3.31	replace	149
6.46.3.32	samples	149
6.46.3.33	sampleSize	149
6.46.3.34	serialize	149
6.46.3.35	setBaseName	149
6.46.3.36	setDistributions	149
6.46.3.37	setDistributions	149

6.46.3.38	setNormalization	149
6.46.3.39	setNTuple	149
6.46.3.40	setSamples	150
6.46.3.41	size	150
6.46.4	Friends And Related Function Documentation	150
6.46.4.1	boost::serialization::access	150
6.46.5	Member Data Documentation	150
6.46.5.1	basename	150
6.46.5.2	mvs	150
6.46.5.3	nameToFind	150
6.46.5.4	scl	150
6.46.5.5	trn	150
6.47	GoSUM::CMonteCarloHC Class Reference	150
6.47.1	Constructor & Destructor Documentation	151
6.47.1.1	CMonteCarloHC	151
6.47.1.2	CMonteCarloHC	151
6.47.1.3	~CMonteCarloHC	151
6.47.2	Member Function Documentation	151
6.47.2.1	doGenerate	151
6.47.2.2	serialize	151
6.47.3	Friends And Related Function Documentation	152
6.47.3.1	boost::serialization::access	152
6.48	CNLCRNG Class Reference	152
6.48.1	Constructor & Destructor Documentation	152
6.48.1.1	CNLCRNG	152
6.48.1.2	CNLCRNG	152
6.48.1.3	~CNLCRNG	152
6.48.2	Member Function Documentation	152
6.48.2.1	rnd	153
6.48.2.2	rndi	153
6.48.2.3	setSeed	153
6.48.3	Member Data Documentation	153
6.48.3.1	inext	153
6.48.3.2	inextp	153
6.48.3.3	ma	153
6.49	CNomadRNG Class Reference	153
6.49.1	Detailed Description	154
6.49.2	Constructor & Destructor Documentation	154
6.49.2.1	CNomadRNG	154
6.49.2.2	CNomadRNG	154

6.49.2.3	~CNomadRNG	154
6.49.3	Member Function Documentation	154
6.49.3.1	rnd	154
6.49.3.2	rmdi	154
6.49.3.3	setSeed	154
6.49.4	Member Data Documentation	154
6.49.4.1	x	154
6.49.4.2	y	154
6.49.4.3	z	154
6.50	CNumericalSample Class Reference	155
6.50.1	Detailed Description	156
6.50.2	Constructor & Destructor Documentation	156
6.50.2.1	CNumericalSample	156
6.50.2.2	CNumericalSample	156
6.50.2.3	CNumericalSample	156
6.50.2.4	~CNumericalSample	156
6.50.3	Member Function Documentation	156
6.50.3.1	maxValue	156
6.50.3.2	minValue	156
6.50.3.3	readSampleValue	156
6.50.3.4	serialize	156
6.50.3.5	setSampleSize	156
6.50.3.6	variance	156
6.50.3.7	writeSampleValue	157
6.50.4	Friends And Related Function Documentation	157
6.50.4.1	boost::serialization::access	157
6.50.5	Member Data Documentation	157
6.50.5.1	values	157
6.51	GoSUM::CNuSvcSAM Class Reference	157
6.51.1	Detailed Description	158
6.51.2	Constructor & Destructor Documentation	158
6.51.2.1	CNuSvcSAM	158
6.51.2.2	~CNuSvcSAM	158
6.51.3	Member Function Documentation	158
6.51.3.1	serialize	158
6.51.4	Friends And Related Function Documentation	158
6.51.4.1	boost::serialization::access	158
6.52	GoSUM::CNuSvrSAM Class Reference	158
6.52.1	Detailed Description	159
6.52.2	Constructor & Destructor Documentation	159

6.52.2.1	CNuSvrSAM	159
6.52.2.2	~CNuSvrSAM	159
6.52.3	Member Function Documentation	159
6.52.3.1	openOptimization	159
6.52.3.2	optimizationPoint2SVMParam	159
6.52.3.3	serialize	159
6.52.4	Friends And Related Function Documentation	159
6.52.4.1	boost::serialization::access	159
6.53	GoSUM::COneClassSAM Class Reference	160
6.53.1	Detailed Description	160
6.53.2	Constructor & Destructor Documentation	160
6.53.2.1	COneClassSAM	160
6.53.2.2	~COneClassSAM	160
6.53.3	Member Function Documentation	160
6.53.3.1	serialize	160
6.53.4	Friends And Related Function Documentation	161
6.53.4.1	boost::serialization::access	161
6.54	COptimizationProblem Class Reference	161
6.54.1	Detailed Description	163
6.54.2	Constructor & Destructor Documentation	163
6.54.2.1	COptimizationProblem	163
6.54.2.2	~COptimizationProblem	163
6.54.3	Member Function Documentation	163
6.54.3.1	addVariable	163
6.54.3.2	clear	163
6.54.3.3	clearHistory	163
6.54.3.4	closeOptimization	163
6.54.3.5	constraint	164
6.54.3.6	constraintsSize	164
6.54.3.7	dimension	164
6.54.3.8	emptyHistory	164
6.54.3.9	eraseVariable	164
6.54.3.10	evaluate	164
6.54.3.11	exportObjectiveHistory	164
6.54.3.12	historySize	164
6.54.3.13	initialValue	164
6.54.3.14	initialValue	164
6.54.3.15	isFeasible	164
6.54.3.16	isMaximization	165
6.54.3.17	isMinimization	165

6.54.3.18	lowerBound	165
6.54.3.19	lowerBound	165
6.54.3.20	objective	165
6.54.3.21	objectiveHistory	165
6.54.3.22	openOptimization	165
6.54.3.23	serialize	165
6.54.3.24	setInitialValue	165
6.54.3.25	setInitialValue	165
6.54.3.26	setLowerBound	165
6.54.3.27	setLowerBound	166
6.54.3.28	setMaximization	166
6.54.3.29	setMinimization	166
6.54.3.30	setUpperBound	166
6.54.3.31	setUpperBound	166
6.54.3.32	upperBound	166
6.54.3.33	upperBound	166
6.54.3.34	variableHistory	166
6.54.3.35	writeHistory	166
6.54.4	Friends And Related Function Documentation	166
6.54.4.1	boost::serialization::access	166
6.54.5	Member Data Documentation	166
6.54.5.1	history	166
6.54.5.2	minimize	167
6.54.5.3	ovs	167
6.55	COptimizationVariable Class Reference	167
6.55.1	Constructor & Destructor Documentation	168
6.55.1.1	COptimizationVariable	168
6.55.1.2	COptimizationVariable	168
6.55.1.3	COptimizationVariable	168
6.55.1.4	~COptimizationVariable	168
6.55.2	Member Function Documentation	168
6.55.2.1	initialValue	168
6.55.2.2	lowerBound	168
6.55.2.3	serialize	168
6.55.2.4	set	168
6.55.2.5	setInitialValue	168
6.55.2.6	setLowerBound	168
6.55.2.7	setUpperBound	168
6.55.2.8	upperBound	169
6.55.3	Friends And Related Function Documentation	169

6.55.3.1	boost::serialization::access	169
6.55.4	Member Data Documentation	169
6.55.4.1	x0	169
6.55.4.2	xL	169
6.55.4.3	xU	169
6.56	GoSUM::COriginalOptimizationProblem Class Reference	169
6.56.1	Detailed Description	170
6.56.2	Constructor & Destructor Documentation	170
6.56.2.1	COriginalOptimizationProblem	170
6.56.2.2	COriginalOptimizationProblem	170
6.56.2.3	~COriginalOptimizationProblem	170
6.56.3	Member Function Documentation	170
6.56.3.1	closeOptimization	170
6.56.3.2	inputPoint2ModelPoint	170
6.56.3.3	openOptimization	171
6.56.3.4	serialize	171
6.56.3.5	setVariableNames	171
6.56.4	Friends And Related Function Documentation	171
6.56.4.1	boost::serialization::access	171
6.56.5	Member Data Documentation	171
6.56.5.1	pOS	171
6.57	GoSUM::COutputStates Class Reference	171
6.57.1	Detailed Description	172
6.57.2	Constructor & Destructor Documentation	172
6.57.2.1	COutputStates	172
6.57.2.2	~COutputStates	172
6.57.3	Member Function Documentation	172
6.57.3.1	closeEvaluation	172
6.57.3.2	evaluate	172
6.57.3.3	evaluate	172
6.57.3.4	openEvaluation	173
6.57.3.5	serialize	173
6.57.3.6	setEvaluatorCompatible	173
6.57.3.7	setProgressSlot	173
6.57.4	Friends And Related Function Documentation	173
6.57.4.1	boost::serialization::access	173
6.57.5	Member Data Documentation	173
6.57.5.1	pE	173
6.57.5.2	progressSlot	173
6.58	GoSUM::CParserOptimizationProblem Class Reference	173

6.58.1	Detailed Description	175
6.58.2	Constructor & Destructor Documentation	175
6.58.2.1	CParserOptimizationProblem	175
6.58.2.2	~CParserOptimizationProblem	175
6.58.3	Member Function Documentation	175
6.58.3.1	addConstraintExpression	175
6.58.3.2	clearConstraintExpressions	175
6.58.3.3	closeOptimization	175
6.58.3.4	constraint	175
6.58.3.5	constraintExpression	176
6.58.3.6	constraintsSize	176
6.58.3.7	eraseConstraintExpression	176
6.58.3.8	objective	176
6.58.3.9	objectiveExpression	176
6.58.3.10	openOptimization	176
6.58.3.11	parseExpressions	176
6.58.3.12	roundoffEquality	176
6.58.3.13	serialize	176
6.58.3.14	setConstraintExpression	176
6.58.3.15	setObjectiveExpression	177
6.58.3.16	setVariableNames	177
6.58.3.17	validateExpressions	177
6.58.4	Friends And Related Function Documentation	177
6.58.4.1	boost::serialization::access	177
6.58.5	Member Data Documentation	177
6.58.5.1	f	177
6.58.5.2	fexpr	177
6.58.5.3	g	177
6.58.5.4	gexpr	177
6.58.5.5	names	177
6.58.5.6	optimizingProgressed	177
6.59	CPlot2D Class Reference	178
6.59.1	Constructor & Destructor Documentation	179
6.59.1.1	CPlot2D	179
6.59.1.2	~CPlot2D	179
6.59.2	Member Function Documentation	179
6.59.2.1	canvas	179
6.59.2.2	clear	179
6.59.2.3	getPlot	179
6.59.2.4	newColorBarPlot	179

6.59.2.5	newColorMap	179
6.59.2.6	newPlot	179
6.59.2.7	plotWithHistogram	179
6.59.2.8	plotWithHistogram	179
6.59.2.9	plotWithPen	179
6.59.2.10	plotWithSpectrogram	179
6.59.2.11	plotWithSymbol	180
6.59.2.12	plotWithSymbol	180
6.59.2.13	replot	180
6.59.2.14	setTitles	180
6.59.3	Member Data Documentation	180
6.59.3.1	plot	180
6.60	CRandomGenerator Class Reference	180
6.60.1	Detailed Description	181
6.60.2	Member Enumeration Documentation	181
6.60.2.1	rngtype	181
6.60.3	Member Function Documentation	182
6.60.3.1	Rnd	182
6.60.3.2	Rnd	182
6.60.3.3	RndInt	182
6.60.3.4	Set	182
6.60.3.5	SetDynamicSystem	182
6.60.3.6	SetLargePeriod	182
6.60.3.7	SetMediumPeriod	182
6.60.3.8	SetNomads	182
6.60.3.9	SetNonlinearCongruential	182
6.60.3.10	SetSeed	182
6.60.3.11	SetSmallPeriod	182
6.60.3.12	Type	182
6.60.3.13	Type	183
6.60.4	Member Data Documentation	183
6.60.4.1	ds	183
6.60.4.2	lp	183
6.60.4.3	mad	183
6.60.4.4	mp	183
6.60.4.5	nlc	183
6.60.4.6	p	183
6.60.4.7	sp	183
6.60.4.8	type	183
6.61	CRandomVariable Class Reference	183

6.61.1	Detailed Description	185
6.61.2	Member Enumeration Documentation	185
6.61.2.1	distributiontype	185
6.61.3	Constructor & Destructor Documentation	186
6.61.3.1	CRandomVariable	186
6.61.3.2	~CRandomVariable	186
6.61.3.3	CRandomVariable	186
6.61.4	Member Function Documentation	186
6.61.4.1	distributionName	186
6.61.4.2	DistributionParametersSize	186
6.61.4.3	distributionType	186
6.61.4.4	doQuantile	186
6.61.4.5	expandedSize	186
6.61.4.6	expectedValue	186
6.61.4.7	IsContinuous	186
6.61.4.8	IsDiscrete	186
6.61.4.9	isDistributionDefined	187
6.61.4.10	IsDistributionType	187
6.61.4.11	IsEmpirical	187
6.61.4.12	IsTheoretical	187
6.61.4.13	maxValue	187
6.61.4.14	minValue	187
6.61.4.15	quantile	187
6.61.4.16	serialize	187
6.61.4.17	Type	187
6.61.4.18	variance	187
6.61.5	Friends And Related Function Documentation	188
6.61.5.1	boost::serialization::access	188
6.62	GoSUM::CReduction Class Reference	188
6.62.1	Detailed Description	190
6.62.2	Member Enumeration Documentation	190
6.62.2.1	cutofftype	190
6.62.2.2	reductiontype	190
6.62.3	Constructor & Destructor Documentation	190
6.62.3.1	CReduction	190
6.62.3.2	CReduction	190
6.62.3.3	~CReduction	190
6.62.4	Member Function Documentation	190
6.62.4.1	clear	190
6.62.4.2	cutoffSize	191

6.62.4.3	CutoffType	191
6.62.4.4	cutoffValue	191
6.62.4.5	eraseNonSelectedVariables	191
6.62.4.6	evaluateMaximalSensitivities	191
6.62.4.7	initialize	191
6.62.4.8	isSelectedInputParameter	191
6.62.4.9	isSelectedOutputState	191
6.62.4.10	maximalSensitivity	191
6.62.4.11	maximalSensitivity	191
6.62.4.12	ReductionType	191
6.62.4.13	ReductionType	191
6.62.4.14	selectInputParameters	192
6.62.4.15	selectOutputs	192
6.62.4.16	selectOutputState	192
6.62.4.17	sensitivityIndex	192
6.62.4.18	serialize	192
6.62.4.19	setCutoffSize	192
6.62.4.20	SetCutoffType	192
6.62.4.21	setCutoffValue	192
6.62.4.22	SetReductionType	192
6.62.4.23	setReductionType	192
6.62.5	Friends And Related Function Documentation	192
6.62.5.1	boost::serialization::access	192
6.62.6	Member Data Documentation	193
6.62.6.1	cutip	193
6.62.6.2	cutos	193
6.62.6.3	cuttype	193
6.62.6.4	cutval	193
6.62.6.5	pAM	193
6.62.6.6	pIP	193
6.62.6.7	pOS	193
6.62.6.8	pSA	193
6.62.6.9	rankIP	193
6.62.6.10	rankOS	193
6.62.6.11	redtype	193
6.63	CSample Class Reference	194
6.63.1	Detailed Description	195
6.63.2	Constructor & Destructor Documentation	195
6.63.2.1	CSample	195
6.63.2.2	~CSample	195

6.63.2.3	CSample	195
6.63.3	Member Function Documentation	195
6.63.3.1	clear	195
6.63.3.2	computeStatistics	195
6.63.3.3	eraseSelected	195
6.63.3.4	readSampleValue	195
6.63.3.5	sampleSize	195
6.63.3.6	select	195
6.63.3.7	serialize	196
6.63.3.8	setSampleSize	196
6.63.3.9	setSampleValue	196
6.63.3.10	variance	196
6.63.3.11	writeSampleValue	196
6.63.4	Friends And Related Function Documentation	196
6.63.4.1	boost::serialization::access	196
6.63.5	Member Data Documentation	196
6.63.5.1	maxdiscretesize	196
6.64	GoSUM::CScript Class Reference	196
6.64.1	Detailed Description	198
6.64.2	Constructor & Destructor Documentation	198
6.64.2.1	CScript	198
6.64.2.2	~CScript	198
6.64.3	Member Function Documentation	198
6.64.3.1	addInput	199
6.64.3.2	addInputs	199
6.64.3.3	addOptimizationConstraint	199
6.64.3.4	addOutput	199
6.64.3.5	addOutputs	199
6.64.3.6	addVariable	200
6.64.3.7	addVariables	200
6.64.3.8	computeSensitivities	200
6.64.3.9	evaluateOutputs	200
6.64.3.10	exportAbsoluteAverageDerivative	200
6.64.3.11	exportANOVA1	200
6.64.3.12	exportAverageDerivative	200
6.64.3.13	exportDerivativeSensitivity	200
6.64.3.14	exportInputSamples	201
6.64.3.15	exportOptimizationHistory	201
6.64.3.16	exportOptimizationMethod	201
6.64.3.17	exportOutputSamples	201

6.64.3.18 exportPredictionOutputSamples	201
6.64.3.19 exportVarianceSensitivity	201
6.64.3.20 importInputs	201
6.64.3.21 importOutputs	202
6.64.3.22 importPredictionInputSamples	202
6.64.3.23 importVariables	202
6.64.3.24 learnModel	202
6.64.3.25 line2Path	202
6.64.3.26 load	202
6.64.3.27 loadTxt	202
6.64.3.28 loadXml	202
6.64.3.29 maximize	203
6.64.3.30 minimize	203
6.64.3.31 predict	203
6.64.3.32 read	203
6.64.3.33 reduce	203
6.64.3.34 resampleInputs	203
6.64.3.35 save	203
6.64.3.36 saveTxt	203
6.64.3.37 saveXml	203
6.64.3.38 setConstInitialValue	204
6.64.3.39 setConstLowerBound	204
6.64.3.40 setConstUpperBound	204
6.64.3.41 setCoreSize	204
6.64.3.42 setInitialValue	204
6.64.3.43 setLowerBound	204
6.64.3.44 setMadsInitMeshSize	204
6.64.3.45 setMadsLHSearch	204
6.64.3.46 setMadsMaxEvaluation	205
6.64.3.47 setMadsMinPollSize	205
6.64.3.48 setMatLabPath	205
6.64.3.49 setModelEvaluator	205
6.64.3.50 setObjective	205
6.64.3.51 setOptimizationMethod	205
6.64.3.52 setProjectName	205
6.64.3.53 setProjectPath	206
6.64.3.54 setProjectType	206
6.64.3.55 setReductionCutoffSize	206
6.64.3.56 setReductionCutoffValue	206
6.64.3.57 setReductionOutputs	206

6.64.3.58	setReductionType	206
6.64.3.59	setResampleSize	206
6.64.3.60	setResampleType	207
6.64.3.61	setRNG	207
6.64.3.62	setSensitivityOptions	207
6.64.3.63	setUpperBound	207
6.64.3.64	setVoronoiOptions	207
6.64.4	Member Data Documentation	207
6.64.4.1	cmds	207
6.64.4.2	noeq1	207
6.64.4.3	padd	207
6.64.4.4	padds	208
6.64.4.5	pContainer	208
6.64.4.6	pimport	208
6.65	GoSUM::CSensitivityAnalysis Class Reference	208
6.65.1	Detailed Description	210
6.65.2	Constructor & Destructor Documentation	210
6.65.2.1	CSensitivityAnalysis	210
6.65.2.2	CSensitivityAnalysis	210
6.65.2.3	~CSensitivityAnalysis	211
6.65.3	Member Function Documentation	211
6.65.3.1	absoluteAverageDerivative	211
6.65.3.2	averageDerivative	211
6.65.3.3	clear	211
6.65.3.4	computeDerivativeSensitivities	211
6.65.3.5	computeSensitivities	211
6.65.3.6	computeVarianceSensitivities	211
6.65.3.7	computeVarianceSensitivities1	211
6.65.3.8	couplingError	211
6.65.3.9	derivativeSensitivity	211
6.65.3.10	empty	211
6.65.3.11	emptyAbsoluteAverageDerivativeSensitivity	211
6.65.3.12	emptyAverageDerivativeSensitivity	211
6.65.3.13	emptyDerivativeSensitivity	212
6.65.3.14	emptyFirstOrderANOVA	212
6.65.3.15	emptyVarianceSensitivity	212
6.65.3.16	firstOrderANOVA	212
6.65.3.17	prepare	212
6.65.3.18	progressStepsSize	212
6.65.3.19	resampleSize	212

6.65.3.20	secondOrderANOVA	212
6.65.3.21	secondOrderANOVAPairs	212
6.65.3.22	sensitivityError	212
6.65.3.23	separabilityError	212
6.65.3.24	serialize	212
6.65.3.25	setCouplingError	212
6.65.3.26	setResampleSize	213
6.65.3.27	setSensitivityError	213
6.65.3.28	setSensitivityOptions	213
6.65.3.29	setSeparabilityError	213
6.65.3.30	varianceSensitivity	213
6.65.4	Friends And Related Function Documentation	213
6.65.4.1	boost::serialization::access	213
6.65.5	Member Data Documentation	213
6.65.5.1	aader	213
6.65.5.2	ader	213
6.65.5.3	computingProgressed	213
6.65.5.4	eps1	213
6.65.5.5	eps2	213
6.65.5.6	eps3	214
6.65.5.7	pAM	214
6.65.5.8	pIP	214
6.65.5.9	pOS	214
6.65.5.10	RSize	214
6.65.5.11	sample1	214
6.65.5.12	sample2	214
6.65.5.13	si1	214
6.65.5.14	si1der	214
6.65.5.15	si2	214
6.65.5.16	siT	214
6.65.5.17	y	214
6.65.5.18	ymu	214
6.65.5.19	yvar	214
6.66	GoSUM::CSimpleOptimizationProblem Class Reference	215
6.66.1	Detailed Description	216
6.66.2	Constructor & Destructor Documentation	216
6.66.2.1	CSimpleOptimizationProblem	216
6.66.2.2	CSimpleOptimizationProblem	216
6.66.2.3	~CSimpleOptimizationProblem	216
6.66.3	Member Function Documentation	216

6.66.3.1	clear	216
6.66.3.2	clearHistory	216
6.66.3.3	closeOptimization	217
6.66.3.4	evaluate	217
6.66.3.5	findConnectedTo	217
6.66.3.6	inputPoint2ModelPoint	217
6.66.3.7	isFeasible	217
6.66.3.8	isModelVariable	217
6.66.3.9	openOptimization	217
6.66.3.10	serialize	217
6.66.3.11	setVariableNames	217
6.66.3.12	variable	218
6.66.3.13	variable	218
6.66.3.14	variableName	218
6.66.4	Friends And Related Function Documentation	218
6.66.4.1	boost::serialization::access	218
6.66.5	Member Data Documentation	218
6.66.5.1	ovind	218
6.66.5.2	pIP	218
6.66.5.3	pipToFind	218
6.66.5.4	samples	218
6.66.5.5	uvind	218
6.67	GoSUM::CSingleAM Class Reference	218
6.67.1	Detailed Description	221
6.67.2	Constructor & Destructor Documentation	221
6.67.2.1	CSingleAM	221
6.67.2.2	~CSingleAM	221
6.67.3	Member Function Documentation	221
6.67.3.1	CloseSVM	221
6.67.3.2	constraint	221
6.67.3.3	constraintsSize	221
6.67.3.4	df	221
6.67.3.5	f	221
6.67.3.6	learn	221
6.67.3.7	LearningFraction	221
6.67.3.8	MaximalLearningSize	221
6.67.3.9	modelVariables	221
6.67.3.10	New	222
6.67.3.11	objective	222
6.67.3.12	openOptimization	222

6.67.3.13	OpenSVM	222
6.67.3.14	optimizationPoint2SVMParam	222
6.67.3.15	rbfKernel	222
6.67.3.16	serialize	222
6.67.3.17	SetLearningFraction	222
6.67.3.18	SetMaximalLearningSize	222
6.67.4	Friends And Related Function Documentation	222
6.67.4.1	boost::serialization::access	222
6.67.5	Member Data Documentation	223
6.67.5.1	C	223
6.67.5.2	coef0	223
6.67.5.3	coeff	223
6.67.5.4	degree	223
6.67.5.5	gamma	223
6.67.5.6	learnfrac	223
6.67.5.7	learningProgressed	223
6.67.5.8	maxlearnsize	223
6.67.5.9	model	223
6.67.5.10	Nsvm	223
6.67.5.11	nu	223
6.67.5.12	osi	223
6.67.5.13	p	223
6.67.5.14	param	223
6.67.5.15	perm	223
6.67.5.16	pIP	224
6.67.5.17	pOS	224
6.67.5.18	prob	224
6.67.5.19	xspace	224
6.67.5.20	Xsv	224
6.68	CSmallPeriodRNG Class Reference	224
6.68.1	Constructor & Destructor Documentation	225
6.68.1.1	CSmallPeriodRNG	225
6.68.1.2	CSmallPeriodRNG	225
6.68.1.3	~CSmallPeriodRNG	225
6.68.2	Member Function Documentation	225
6.68.2.1	rnd	225
6.68.2.2	rndi	225
6.68.2.3	setSeed	225
6.68.3	Member Data Documentation	225
6.68.3.1	idum1	225

6.68.3.2	iv1	225
6.68.3.3	iy1	225
6.69	CUniformCRV Class Reference	226
6.69.1	Detailed Description	227
6.69.2	Constructor & Destructor Documentation	227
6.69.2.1	CUniformCRV	227
6.69.2.2	~CUniformCRV	227
6.69.2.3	CUniformCRV	227
6.69.3	Member Function Documentation	227
6.69.3.1	cumulative	227
6.69.3.2	distributionName	227
6.69.3.3	distributionType	227
6.69.3.4	doQuantile	228
6.69.3.5	expectedValue	228
6.69.3.6	exportDomain	228
6.69.3.7	lowerBound	228
6.69.3.8	maxValue	228
6.69.3.9	minValue	228
6.69.3.10	probability	228
6.69.3.11	serialize	228
6.69.3.12	setDistribution	228
6.69.3.13	setDistribution	228
6.69.3.14	setLowerBound	228
6.69.3.15	setUpperBound	229
6.69.3.16	upperBound	229
6.69.3.17	variance	229
6.69.4	Friends And Related Function Documentation	229
6.69.4.1	boost::serialization::access	229
6.69.5	Member Data Documentation	229
6.69.5.1	a	229
6.69.5.2	b	229
6.69.5.3	d	229
6.70	CUniformDRV Class Reference	229
6.70.1	Detailed Description	231
6.70.2	Constructor & Destructor Documentation	231
6.70.2.1	CUniformDRV	231
6.70.2.2	~CUniformDRV	231
6.70.2.3	CUniformDRV	231
6.70.3	Member Function Documentation	231
6.70.3.1	cumulative	231

6.70.3.2	distributionName	231
6.70.3.3	distributionType	231
6.70.3.4	doQuantile	231
6.70.3.5	expectedValue	231
6.70.3.6	exportDomain	231
6.70.3.7	lowerBound	232
6.70.3.8	maxValue	232
6.70.3.9	minValue	232
6.70.3.10	probability	232
6.70.3.11	serialize	232
6.70.3.12	setDistribution	232
6.70.3.13	setLowerBound	232
6.70.3.14	setUpperBound	232
6.70.3.15	upperBound	232
6.70.3.16	variance	232
6.70.4	Friends And Related Function Documentation	232
6.70.4.1	boost::serialization::access	232
6.70.5	Member Data Documentation	233
6.70.5.1	a	233
6.70.5.2	b	233
6.70.5.3	n	233
6.71	Cx_ZetaGammax Class Reference	233
6.71.1	Detailed Description	233
6.71.2	Constructor & Destructor Documentation	233
6.71.2.1	Cx_ZetaGammax	233
6.71.2.2	Cx_ZetaGammax	233
6.71.3	Member Function Documentation	233
6.71.3.1	evaluate	233
6.71.4	Member Data Documentation	233
6.71.4.1	a2	234
6.71.4.2	i2	234
6.71.4.3	N	234
6.72	DSn Class Reference	234
6.72.1	Member Typedef Documentation	234
6.72.1.1	Long	234
6.72.2	Constructor & Destructor Documentation	234
6.72.2.1	DSn	234
6.72.2.2	~DSn	234
6.72.3	Member Function Documentation	235
6.72.3.1	doub	235

6.72.4	Member Data Documentation	235
6.72.4.1	i	235
6.72.4.2	j	235
6.72.4.3	k	235
6.72.4.4	n	235
6.72.4.5	w	235
6.72.4.6	x	235
6.73	GoSUM Struct Reference	235
6.73.1	Member Typedef Documentation	237
6.73.1.1	CTContinuousMV	237
6.73.1.2	CTDiscreteMV	237
6.73.1.3	paction	237
6.73.2	Member Data Documentation	237
6.73.2.1	NULL	237
6.74	Ran Class Reference	237
6.74.1	Constructor & Destructor Documentation	237
6.74.1.1	Ran	237
6.74.2	Member Function Documentation	237
6.74.2.1	doub	237
6.74.2.2	int32	237
6.74.2.3	int64	238
6.74.3	Member Data Documentation	238
6.74.3.1	u	238
6.74.3.2	v	238
6.74.3.3	w	238
6.75	GoSUM::TModelVariable< R, S, t > Class Template Reference	238
6.75.1	Constructor & Destructor Documentation	240
6.75.1.1	TModelVariable	240
6.75.1.2	~TModelVariable	240
6.75.2	Member Function Documentation	240
6.75.2.1	clearSample	240
6.75.2.2	computeStatistics	240
6.75.2.3	distributionName	240
6.75.2.4	distributionType	240
6.75.2.5	expandedSize	240
6.75.2.6	expectedValue	240
6.75.2.7	exportCumulative	241
6.75.2.8	exportDomain	241
6.75.2.9	exportHistogram	241
6.75.2.10	exportProbability	241

6.75.2.11	exportQuantile	241
6.75.2.12	exportSample	241
6.75.2.13	exportSubHistogram	241
6.75.2.14	generateSampleValue	241
6.75.2.15	generateSampleValue	241
6.75.2.16	isDistributionDefined	242
6.75.2.17	maxValue	242
6.75.2.18	minValue	242
6.75.2.19	normalizedHistogram	242
6.75.2.20	randomVariable	242
6.75.2.21	readSampleValue	242
6.75.2.22	sample	242
6.75.2.23	sampleSize	242
6.75.2.24	sampleValue	243
6.75.2.25	serialize	243
6.75.2.26	setDistributionParameters	243
6.75.2.27	setEmpiricalDistribution	243
6.75.2.28	setSampleSize	243
6.75.2.29	setSampleValue	243
6.75.2.30	setTheoreticalDistribution	243
6.75.2.31	writeSampleValue	243
6.75.3	Friends And Related Function Documentation	244
6.75.3.1	boost::serialization::access	244
6.75.4	Member Data Documentation	244
6.75.4.1	pRV	244
6.75.4.2	pS	244
6.76	TRandomVariable< t, T > Class Template Reference	244
6.76.1	Detailed Description	245
6.76.2	Constructor & Destructor Documentation	245
6.76.2.1	TRandomVariable	245
6.76.2.2	~TRandomVariable	245
6.76.2.3	TRandomVariable	245
6.76.3	Member Function Documentation	245
6.76.3.1	cumulative	245
6.76.3.2	exportDomain	246
6.76.3.3	probability	246
6.76.3.4	rvExportMacro	246
6.76.3.5	rvExportMacro	246
6.76.3.6	serialize	246
6.76.3.7	setDistribution	246

6.76.3.8	setDistribution	246
6.76.4	Friends And Related Function Documentation	246
6.76.4.1	boost::serialization::access	246
6.76.5	Member Data Documentation	246
6.76.5.1	cumulative	246
6.76.5.2	Dynamic	246
6.77	TSample< t > Class Template Reference	246
6.77.1	Detailed Description	248
6.77.2	Constructor & Destructor Documentation	248
6.77.2.1	TSample	248
6.77.2.2	~TSample	248
6.77.2.3	TSample	248
6.77.3	Member Function Documentation	248
6.77.3.1	clear	248
6.77.3.2	computeNormalizedHistogram	249
6.77.3.3	computeNormalizedSubHistogram	249
6.77.3.4	cummulativeHistogram	249
6.77.3.5	eraseSelected	249
6.77.3.6	exportHistogram	249
6.77.3.7	exportSample	249
6.77.3.8	exportSubHistogram	249
6.77.3.9	isConstant	249
6.77.3.10	maxValue	249
6.77.3.11	minValue	250
6.77.3.12	normalizedHistogram	250
6.77.3.13	readSampleValue	250
6.77.3.14	sampleSize	250
6.77.3.15	sampleValue	250
6.77.3.16	select	250
6.77.3.17	serialize	250
6.77.3.18	setSampleSize	250
6.77.3.19	setSampleValue	251
6.77.3.20	writeSampleValue	251
6.77.4	Friends And Related Function Documentation	251
6.77.4.1	boost::serialization::access	251
6.77.5	Member Data Documentation	251
6.77.5.1	cH	251
6.77.5.2	H	251
6.77.5.3	range	251
6.77.5.4	sample	251

6.77.5.5	x	251
7	File Documentation	253
7.1	C:/Development/core/AnalyticalModel.cpp File Reference	253
7.2	C:/Development/core/AnalyticalModel.h File Reference	253
7.3	C:/Development/core/Constraints.cpp File Reference	254
7.4	C:/Development/core/Constraints.h File Reference	254
7.5	C:/Development/core/Container.cpp File Reference	254
7.6	C:/Development/core/Container.h File Reference	254
7.7	C:/Development/core/Ds.cpp File Reference	255
7.7.1	Function Documentation	255
7.7.1.1	prepareDSample	255
7.8	C:/Development/core/Ds.h File Reference	255
7.8.1	Function Documentation	255
7.8.1.1	prepareDSample	255
7.9	C:/Development/core/FFTWLibrary.cpp File Reference	255
7.10	C:/Development/core/FFTWLibrary.h File Reference	255
7.11	C:/Development/core/GAOptimization.cpp File Reference	256
7.11.1	Macro Definition Documentation	256
7.11.1.1	INstantiate_REAL_GENOME	256
7.12	C:/Development/core/GAOptimization.h File Reference	256
7.13	C:/Development/core/Hypercube.cpp File Reference	256
7.14	C:/Development/core/Hypercube.h File Reference	256
7.15	C:/Development/core/MADSOptimization.cpp File Reference	257
7.16	C:/Development/core/MADSOptimization.h File Reference	257
7.17	C:/Development/core/MatlabLibrary.cpp File Reference	257
7.18	C:/Development/core/MatlabLibrary.h File Reference	257
7.18.1	Typedef Documentation	258
7.18.1.1	engftype1	258
7.18.1.2	engftype2	258
7.18.1.3	engftype3	258
7.18.1.4	engftype4	258
7.18.1.5	engftype5	258
7.18.1.6	engftype6	258
7.18.1.7	engftype7	258
7.18.1.8	engftype8	258
7.18.1.9	engftype9	258
7.18.1.10	matftype1	258
7.18.1.11	matftype2	259
7.18.1.12	matftype3	259

7.18.1.13	matftype4	259
7.18.1.14	mxftype1	259
7.18.1.15	mxftype2	259
7.18.1.16	mxftype3	259
7.18.1.17	mxftype4	259
7.18.1.18	mxftype5	259
7.19	C:/Development/core/Model.cpp File Reference	259
7.20	C:/Development/core/Model.h File Reference	259
7.21	C:/Development/core/ModelVariable.cpp File Reference	259
7.22	C:/Development/core/ModelVariable.h File Reference	260
7.23	C:/Development/core/OptimizationProblem.cpp File Reference	260
7.24	C:/Development/core/OptimizationProblem.h File Reference	260
7.25	C:/Development/core/OriginalModel.cpp File Reference	260
7.25.1	Variable Documentation	261
7.25.1.1	GoSUM	261
7.26	C:/Development/core/OriginalModel.h File Reference	261
7.27	C:/Development/core/ParserOptimizationProblem.cpp File Reference	261
7.28	C:/Development/core/ParserOptimizationProblem.h File Reference	261
7.29	C:/Development/core/Plot2D.cpp File Reference	262
7.30	C:/Development/core/Plot2D.h File Reference	262
7.31	C:/Development/core/Probability.cpp File Reference	263
7.31.1	Function Documentation	263
7.31.1.1	chiSquareTest	263
7.31.1.2	KolmogorovSmirnovTest	263
7.32	C:/Development/core/Probability.h File Reference	263
7.32.1	Function Documentation	263
7.32.1.1	chiSquareTest	263
7.32.1.2	KolmogorovSmirnovTest	263
7.33	C:/Development/core/ran.h File Reference	264
7.33.1	Typedef Documentation	264
7.33.1.1	Ullong	264
7.34	C:/Development/core/RandomGenerators.cpp File Reference	264
7.34.1	Macro Definition Documentation	265
7.34.1.1	AM	265
7.34.1.2	AM	265
7.34.1.3	EPS	265
7.34.1.4	EPS	265
7.34.1.5	FAC	265
7.34.1.6	IA	265
7.34.1.7	IA1	265

7.34.1.8	IA2	265
7.34.1.9	IM	265
7.34.1.10	IM1	265
7.34.1.11	IM2	265
7.34.1.12	IMM1	265
7.34.1.13	IQ	265
7.34.1.14	IQ1	265
7.34.1.15	IQ2	265
7.34.1.16	IR	265
7.34.1.17	IR1	265
7.34.1.18	IR2	265
7.34.1.19	MBIG	265
7.34.1.20	MSEED	265
7.34.1.21	MZ	265
7.34.1.22	NDIV	265
7.34.1.23	NDIV	265
7.34.1.24	NTAB	265
7.34.1.25	NTAB	265
7.34.1.26	RNMX	265
7.34.1.27	RNMX	265
7.35	C:/Development/core/RandomGenerators.h File Reference	266
7.35.1	Macro Definition Documentation	266
7.35.1.1	NTAB	266
7.35.1.2	NTAB	266
7.35.1.3	NTAB	267
7.35.1.4	UINT32_MAX	267
7.36	C:/Development/core/RandomVariable.cpp File Reference	267
7.37	C:/Development/core/RandomVariable.h File Reference	267
7.37.1	Macro Definition Documentation	268
7.37.1.1	rvExportMacro	268
7.37.2	Typedef Documentation	268
7.37.2.1	CTContinuousRV	268
7.37.2.2	CTDiscreteRV	268
7.38	C:/Development/core/Reduction.cpp File Reference	268
7.38.1	Function Documentation	268
7.38.1.1	compareFirst	268
7.38.2	Variable Documentation	269
7.38.2.1	GoSUM	269
7.39	C:/Development/core/Reduction.h File Reference	269
7.40	C:/Development/core/Sample.cpp File Reference	269

7.41 C:/Development/core/Sample.h File Reference	269
7.41.1 Typedef Documentation	270
7.41.1.1 CTContinuousSample	270
7.41.1.2 CTDiscreteSample	270
7.42 C:/Development/core/Script.cpp File Reference	270
7.43 C:/Development/core/Script.h File Reference	270
7.44 C:/Development/core/SensitivityAnalysis.cpp File Reference	270
7.45 C:/Development/core/SensitivityAnalysis.h File Reference	270
7.46 C:/Development/core/Utilities.cpp File Reference	271
7.46.1 Function Documentation	271
7.46.1.1 string2WideString	271
7.47 C:/Development/core/Utilities.h File Reference	271
7.47.1 Macro Definition Documentation	272
7.47.1.1 _USE_MATH_DEFINES	272
7.47.1.2 boostSerializeArrayXdiMacro	273
7.47.1.3 boostSerializeArrayXXdiMacro	273
7.47.1.4 SIGNALSLIB_HPP_INCLUDED	273
7.47.2 Function Documentation	273
7.47.2.1 string2WideString	273
7.48 C:/Development/core/VariousMath.cpp File Reference	273
7.48.1 Function Documentation	274
7.48.1.1 cumulativeSum	274
7.48.1.2 findNearest	274
7.48.1.3 linearInterpolation	274
7.48.1.4 permutation	274
7.48.1.5 permutations	274
7.48.1.6 squareDistance	274
7.48.1.7 variance	274
7.49 C:/Development/core/VariousMath.h File Reference	274
7.49.1 Macro Definition Documentation	275
7.49.1.1 TINY	275
7.49.2 Function Documentation	275
7.49.2.1 cumulativeSum	275
7.49.2.2 findNearest	275
7.49.2.3 findRoot	275
7.49.2.4 linearInterpolation	276
7.49.2.5 permutation	276
7.49.2.6 permutations	276
7.49.2.7 squareDistance	276
7.49.2.8 variance	276

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

boost	11
boost::serialization	11
GoSUM	
Namespace for GoSUM model	11

Chapter 2

Class Index

2.1 Class Hierarchy

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

GoSUM::CA analyticalModel	13
CBaseRNG	19
CDynamicSystemRNG	72
CLargePeriodRNG	110
CMediumPeriodRNG	128
CNLCRNG	152
CNomadRNG	153
CSmallPeriodRNG	224
GoSUM::CConstraints	32
GoSUM::CModelConstraints	130
GoSUM::CContainer	35
GoSUM::CEvaluator	83
GoSUM::CModelEvaluator	131
GoSUM::CExeEvaluator	89
GoSUM::CExeAsciiEvaluator	87
GoSUM::CExeMatEvaluator	91
GoSUM::CMatlabShellEvaluator	126
GoSUM::CMatlabEngineEvaluator	124
CFFTW	97
CGAModelOptimization	98
GoSUM::CHypercube	104
GoSUM::CModelHypercube	134
GoSUM::CCVoronoiHC	65
GoSUM::CLCVoronoiHC	112
GoSUM::CDSampleHC	71
GoSUM::CMonteCarloHC	150
CMADS	114
CMADSEvaluator	117
CMATLAB	118
GoSUM::CModelVariable	138
GoSUM::TModelVariable< R, S, t >	238
GoSUM::CContinuousMV	58
GoSUM::CDiscreteMV	66
GoSUM::CModelVariables	144
GoSUM::CInputParameters	108
GoSUM::COutputStates	171

COptimizationProblem	161
GoSUM::CParserOptimizationProblem	173
GoSUM::CSimpleOptimizationProblem	215
GoSUM::COriginalOptimizationProblem	169
GoSUM::CAnalyticalOptimizationProblem	17
GoSUM::CSingleAM	218
GoSUM::CCSvcSAM	63
GoSUM::CEpsSvrSAM	82
GoSUM::CNUSvcSAM	157
GoSUM::CNUsvrSAM	158
GoSUM::COneClassSAM	160
COptimizationVariable	167
GoSUM::CModelOptimizationVariable	137
CPlot2D	178
CRandomGenerator	180
CRandomVariable	183
TRandomVariable< t, T >	244
CContinuousRV	59
CConstantCRV	25
CEmpiricalCRV	75
CExponentialCRV	93
CGaussianCRV	100
CUniformCRV	226
CDiscreteRV	68
CCategoricalDRV	20
CConstantDRV	29
CEmpiricalDRV	78
CUniformDRV	229
GoSUM::CReduction	188
CSample	194
TSample< t >	246
CContinuousSample	61
CDiscreteSample	69
CCategoricalSample	23
CNumericalSample	155
GoSUM::CScript	196
GoSUM::CSensitivityAnalysis	208
Cx_ZetaGammax	233
DSn	234
GoSUM	235
Ran	237

Chapter 3

Class Index

3.1 Class List

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

GoSUM::CA analyticalModel	13
Class for analytical representation of the model	
GoSUM::CA analyticalOptimizationProblem	17
Class for the optimization problem based on GoSUM analytical model	
CBaseRNG	19
Abstract base class for all uniform random number generators (RNG)	
CCategoricalDRV	20
Class for categorical discrete random variables derived from discrete random variables	
CCategoricalSample	23
Abstract class for categorical discrete samples	
CConstantCRV	25
Class for constant continuous random variables derived from continuous random variables	
CConstantDRV	29
Class for constant discrete random variables derived from discrete random variables	
GoSUM::CConstraints	32
Class for the constraints with parser functions	
GoSUM::CContainer	35
Class for the GoSUM main project	
GoSUM::CContinuousMV	58
CContinuousRV	59
CContinuousSample	61
GoSUM::CCSvcSAM	63
Class for the analytical model for single output state, SVC type	
GoSUM::CCVoronoiHC	65
GoSUM::CDiscreteMV	66
CDiscreteRV	68
CDiscreteSample	69
GoSUM::CDSampleHC	71
CDynamicSystemRNG	72
Class for dynamic system uniform RNG	
CEmpiricalCRV	75
Class for empirical continuous random variables derived from continuous random variables	
CEmpiricalDRV	78
Class for categorical discrete random variables derived from discrete random variables	
GoSUM::CEpsSvrSAM	82
Class for the analytical model for single output state, epsilon-SVR type	
GoSUM::CEvaluator	83
Class for GoSUM model evaluator	

GoSUM::CExeAsciiEvaluator	
Class for the evaluator with ascii i/o and .exe	87
GoSUM::CExeEvaluator	
Template for the evaluator with some file i/o and .exe derived from CModelEvaluator	89
GoSUM::CExeMatEvaluator	
Class for the evaluator with Matlab .mat i/o and .exe	91
CExponentialCRV	
Class for exponential continuous random variables derived from continuous random variables	93
CFFTW	
Class interface for FFTW's libfftw library	97
CGAModelOptimization	
Class for the genetic algorithm, i.e. interface for GALIB	98
CGaussianCRV	
Class for Gaussian continuous random variables derived from continuous random variables	100
GoSUM::CHypercube	104
GoSUM::CInputParameters	
Class for GoSUM input parameters	108
CLargePeriodRNG	
Class for uniform RNG with large period = 3.138×10^{57}	110
GoSUM::CLCVoronoiHC	112
CMADS	
Class for the mesh addaptive direct serach, i.e. interface for NOMAD	114
CMADSEvaluator	
Subclass of the NOMAD::Evaluator class	117
CMATLAB	
Class interface for Matlab's libmat and libmx dynamic libraries	118
GoSUM::CMatlabEngineEvaluator	
Class for the evaluator through Matlab engine derived from CModelEvaluator	124
GoSUM::CMatlabShellEvaluator	
Class for the evaluator through Matlab engine and .mat i/o	126
CMediumPeriodRNG	128
GoSUM::CModelConstraints	130
GoSUM::CModelEvaluator	
Class for GoSUM model evaluator	131
GoSUM::CModelHypercube	134
GoSUM::CModelOptimizationVariable	137
GoSUM::CModelVariable	
Class for any variable in the GoSUM model (input or output)	138
GoSUM::CModelVariables	
Class for the vector of model variables (inputs & outputs)	144
GoSUM::CMonteCarloHC	150
CNLCRNG	152
CNomadRNG	
Class for NOMAD RNG with period = 2^{96-1}	153
CNumericalSample	
Abstract class for numerical discrete samples	155
GoSUM::CNuSvcSAM	
Class for the analytical model for single output state, nu-SVC type	157
GoSUM::CNuSvrSAM	
Class for the analytical model for single output state, nu-SVR type	158
GoSUM::COneClassSAM	
Class for the analytical model for single output state, one class type	160
COptimizationProblem	
Class for the optimization problem	161
COptimizationVariable	167
GoSUM::COriginalOptimizationProblem	
Class for the optimization problem based on GoSUM analytical model	169

GoSUM::COutputStates	
Class for GoSUM output states	171
GoSUM::CParserOptimizationProblem	
Class for the optimization problem with parser functions	173
CPlot2D	178
CRandomGenerator	
Class that holds all implemented uniform RNGs as static. and allows switching between them	180
CRandomVariable	
Abstract base class for all random variables	183
GoSUM::CReduction	
Class for sensitivity analysis of the model	188
CSample	
Abstract class for all samples	194
GoSUM::CScript	
Class for the GoSUM script format	196
GoSUM::CSensitivityAnalysis	
Class for sensitivity analysis of the model	208
GoSUM::CSimpleOptimizationProblem	
Class for the optimization problem based only on GoSUM input parameters	215
GoSUM::CSingleAM	
Base class for the analytical model for single output state, interface between analytical model and SVM	218
CSmallPeriodRNG	224
CUniformCRV	
Class for uniform continuous random variables derived from continuous random variables	226
CUniformDRV	
Class for uniform discrete random variables derived from discrete random variables	229
Cx_ZetaGammax	
Class for the function $f(x)=x-zeta*\gamma^{[7]}(x)$	233
DSn	234
GoSUM	235
Ran	237
GoSUM::TModelVariable< R, S, t >	238
TRandomVariable< t, T >	
Template of an abstract class - covers two types of random variables: discrete (t=int,T= CDiscreteSample) or continuous (t=double,T= CContinuousSample)	244
TSample< t >	
Template of an abstract class - covers two types of samples: discrete (t=int) or continuous (t=double)	246

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

C:/Development/core/ AnalyticalModel.cpp	253
C:/Development/core/ AnalyticalModel.h	253
C:/Development/core/ Constraints.cpp	254
C:/Development/core/ Constraints.h	254
C:/Development/core/ Container.cpp	254
C:/Development/core/ Container.h	254
C:/Development/core/ Ds.cpp	255
C:/Development/core/ Ds.h	255
C:/Development/core/ FFTWLibrary.cpp	255
C:/Development/core/ FFTWLibrary.h	255
C:/Development/core/ GAOptimization.cpp	256
C:/Development/core/ GAOptimization.h	256
C:/Development/core/ Hypercube.cpp	256
C:/Development/core/ Hypercube.h	256
C:/Development/core/ MADSOptimization.cpp	257
C:/Development/core/ MADSOptimization.h	257
C:/Development/core/ MatlabLibrary.cpp	257
C:/Development/core/ MatlabLibrary.h	257
C:/Development/core/ Model.cpp	259
C:/Development/core/ Model.h	259
C:/Development/core/ ModelVariable.cpp	259
C:/Development/core/ ModelVariable.h	260
C:/Development/core/ OptimizationProblem.cpp	260
C:/Development/core/ OptimizationProblem.h	260
C:/Development/core/ OriginalModel.cpp	260
C:/Development/core/ OriginalModel.h	261
C:/Development/core/ ParserOptimizationProblem.cpp	261
C:/Development/core/ ParserOptimizationProblem.h	261
C:/Development/core/ Plot2D.cpp	262
C:/Development/core/ Plot2D.h	262
C:/Development/core/ Probability.cpp	263
C:/Development/core/ Probability.h	263
C:/Development/core/ ran.h	264
C:/Development/core/ RandomGenerators.cpp	264
C:/Development/core/ RandomGenerators.h	266
C:/Development/core/ RandomVariable.cpp	267
C:/Development/core/ RandomVariable.h	267
C:/Development/core/ Reduction.cpp	268

C:/Development/core/ Reduction.h	269
C:/Development/core/ Sample.cpp	269
C:/Development/core/ Sample.h	269
C:/Development/core/ Script.cpp	270
C:/Development/core/ Script.h	270
C:/Development/core/ SensitivityAnalysis.cpp	270
C:/Development/core/ SensitivityAnalysis.h	270
C:/Development/core/ Utilities.cpp	271
C:/Development/core/ Utilities.h	271
C:/Development/core/ VariousMath.cpp	273
C:/Development/core/ VariousMath.h	274

Chapter 5

Namespace Documentation

5.1 boost Namespace Reference

Namespaces

- namespace [serialization](#)

5.2 boost::serialization Namespace Reference

5.3 GoSUM Namespace Reference

Namespace for [GoSUM](#) model.

5.3.1 Detailed Description

Namespace for [GoSUM](#) model.

Chapter 6

Class Documentation

6.1 GoSUM::CA analyticalModel Class Reference

Class for analytical representation of the model.

```
#include <AnalyticalModel.h>
```

Public Types

- enum `svmtype` { `epsilonsvr`, `nusvr` }

Public Member Functions

- `CA analyticalModel` (`CInputParameters` *_pIP, `COutputStates` *_pOS)
- virtual `~CA analyticalModel` ()
- void `setSvmType` (`svmtype` _etype)
Sets chosen svm type.
- `svmtype` `svmType` () const
Returns chosen svm type.
- bool `empty` () const
Returns true if object is empty, true otherwise.
- void `clear` ()
Clears data.
- void `learn` (`CMADS` &_mads, std::ostream &_out=std::cout)
Learns SVM models for all output states.
- void `setInputSamples` (const std::vector< `ArrayXd` > &_X)
Sets internal input parameter samples.
- const std::vector< `ArrayXd` > & `inputSamples` ()
Returns internal input parameter samples.
- const std::vector< `ArrayXd` > & `outputSamples` ()
Returns internal output state samples.
- void `predict` ()
Computes predicted values on internal input/output samples.
- double `predictValue` (const `ArrayXd` &_X, int i) const
Predicts value of the ith (expanded) output state, _X must be unexpanded.
- `ArrayXd` `predictDerivative` (const `ArrayXd` &_X, int i) const
Predicts gradient of the ith (expanded) output state, _X must be unexpanded.
- `ArrayXd` `predictValues` (const `ArrayXd` &_X) const

- Predicts values of all output states, `_X` must be unexpanded.*
- ArrayXXd [predictDerivatives](#) (const ArrayXd &`_X`) const
Predicts gradients of all (expanded) output states, `_X` must be unexpanded.
- ArrayXd [exportExpectedCurve](#) (const ArrayXd &`_x`, int `_i`, int `_o`) const
Exports `_o` output vs. `_i` input curve, while other inputs are set to their expected values.
- ArrayXd [exportIntersectionCurve](#) (int `_s`, const ArrayXd &`_x`, int `_i`, int `_o`) const
Exports `_o` output vs. `_i` input curve, while other inputs are set to values in `_x`.
- void [setProgressSlot](#) (boost::function< void()> `_progressSlot`)
Sets external progress slot.

Public Attributes

- boost::function< void()> [progressSlot](#)
External progress slot, later connected to signal for optimization progress.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- [CA analyticalModel](#) ()

Private Attributes

- [svmtypes](#) `etype`
Holds chosen analytical model type.
- [CInputParameters](#) * `pIP`
Points to input parameters.
- [COutputStates](#) * `pOS`
Points to output states.
- boost::ptr_vector< [CSingleAM](#) > `sam`
Holds pointers to single output state analytical models.
- std::vector< ArrayXd > `X`
- std::vector< ArrayXd > `Y`

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.1.1 Detailed Description

Class for analytical representation of the model.

6.1.2 Member Enumeration Documentation

6.1.2.1 enum GoSUM::CA analyticalModel::svmtypes

Enumerator:

epsilonsvr
nusvr

6.1.3 Constructor & Destructor Documentation

6.1.3.1 `GoSUM::CA analyticalModel::CA analyticalModel () [inline], [private]`

6.1.3.2 `GoSUM::CA analyticalModel::CA analyticalModel (CInputParameters * _pIP, COutputStates * _pOS) [inline]`

6.1.3.3 `virtual GoSUM::CA analyticalModel::~~CA analyticalModel () [inline], [virtual]`

6.1.4 Member Function Documentation

6.1.4.1 `void GoSUM::CA analyticalModel::clear () [inline]`

Clears data.

6.1.4.2 `bool GoSUM::CA analyticalModel::empty () const [inline]`

Returns true if object is empty, true otherwise.

6.1.4.3 `ArrayXd GoSUM::CA analyticalModel::exportExpectedCurve (const ArrayXd & _x, int _i, int _o) const`

Exports `_o` output vs. `_i` input curve, while other inputs are set to their expected values.

6.1.4.4 `ArrayXd GoSUM::CA analyticalModel::exportIntersectionCurve (int _s, const ArrayXd & _x, int _i, int _o) const`

Exports `_o` output vs. `_i` input curve, while other inputs are set to values in `_x`.

6.1.4.5 `const std::vector<ArrayXd>& GoSUM::CA analyticalModel::inputSamples () [inline]`

Returns internal input parameter samples.

6.1.4.6 `void GoSUM::CA analyticalModel::learn (CMADS & _mads, std::ostream & _out = std::cout)`

Learns SVM models for all output states.

6.1.4.7 `const std::vector<ArrayXd>& GoSUM::CA analyticalModel::outputSamples () [inline]`

Returns internal output state samples.

6.1.4.8 `void GoSUM::CA analyticalModel::predict ()`

Computes predicted values on internal input/output samples.

6.1.4.9 `ArrayXd GoSUM::CA analyticalModel::predictDerivative (const ArrayXd & _X, int i) const`

Predicts gradient of the `i`th (expanded) output state, `_X` must be unexpanded.

6.1.4.10 `ArrayXXd GoSUM::CA analyticalModel::predictDerivatives (const ArrayXd & _X) const`

Predicts gradients of all (expanded) output states, `_X` must be unexpanded.

6.1.4.11 `double GoSUM::CA analyticalModel::predictValue (const ArrayXd & _X, int i) const`

Predicts value of the *i*th (expanded) output state, *_X* must be unexpanded.

6.1.4.12 `ArrayXd GoSUM::CA analyticalModel::predictValues (const ArrayXd & _X) const`

Predicts values of all output states, *_X* must be unexpanded.

6.1.4.13 `template<class Archive > void GoSUM::CA analyticalModel::serialize (Archive & ar, const unsigned int version)
[private]`

6.1.4.14 `void GoSUM::CA analyticalModel::setInputSamples (const std::vector< ArrayXd > & _X) [inline]`

Sets internal input parameter samples.

6.1.4.15 `void GoSUM::CA analyticalModel::setProgressSlot (boost::function< void()> _progressSlot) [inline]`

Sets external progress slot.

6.1.4.16 `void GoSUM::CA analyticalModel::setSvmType (svmtype _etype) [inline]`

Sets chosen svm type.

6.1.4.17 `svmtype GoSUM::CA analyticalModel::svmType () const [inline]`

Returns chosen svm type.

6.1.5 Friends And Related Function Documentation

6.1.5.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.1.6 Member Data Documentation

6.1.6.1 `svmtype GoSUM::CA analyticalModel::etype [private]`

Holds chosen analytical model type.

6.1.6.2 `CInputParameters* GoSUM::CA analyticalModel::pIP [private]`

Points to input parameters.

6.1.6.3 `COutputStates* GoSUM::CA analyticalModel::pOS [private]`

Points to output states.

6.1.6.4 `boost::function<void()> GoSUM::CA analyticalModel::progressSlot`

External progress slot, later connected to signal for optimization progress.

6.1.6.5 `boost::ptr_vector<CSingleAM> GoSUM::CA analyticalModel::sam` [private]

Holds pointers to single output state analytical models.

6.1.6.6 `std::vector<ArrayXd> GoSUM::CA analyticalModel::X` [private]

6.1.6.7 `std::vector<ArrayXd> GoSUM::CA analyticalModel::Y` [private]

Input parameter and output state samples for prediction.

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

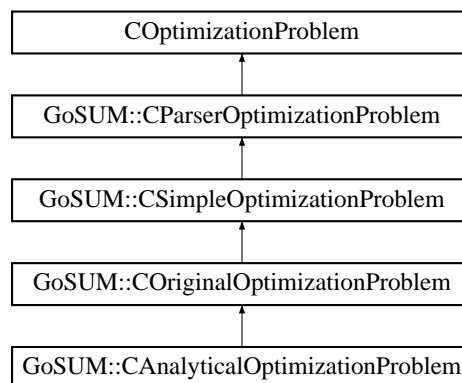
- C:/Development/core/[AnalyticalModel.h](#)
- C:/Development/core/[AnalyticalModel.cpp](#)

6.2 GoSUM::CA analyticalOptimizationProblem Class Reference

Class for the optimization problem based on [GoSUM](#) analytical model.

```
#include <ParserOptimizationProblem.h>
```

Inheritance diagram for GoSUM::CA analyticalOptimizationProblem:



Public Member Functions

- [CA analyticalOptimizationProblem](#) ([CInputParameters](#) *_pIP, [COutputStates](#) *_pOS, [CA analyticalModel](#) *_pAM)
- virtual [~CA analyticalOptimizationProblem](#) ()
- [ArrayXd inputPoint2ModelPoint](#) (const [ArrayXd](#) &_ip)
Converts input parameter point to model point.

Protected Member Functions

- `template<class Archive >`
void [serialize](#) (Archive &ar, const unsigned int version)
- [CA analyticalOptimizationProblem](#) ()

Protected Attributes

- [CA analyticalModel](#) * pAM

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.2.1 Detailed Description

Class for the optimization problem based on [GoSUM](#) analytical model.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 [GoSUM::CA analyticalOptimizationProblem::CA analyticalOptimizationProblem \(\)](#) `[inline], [protected]`

6.2.2.2 [GoSUM::CA analyticalOptimizationProblem::CA analyticalOptimizationProblem \(CInputParameters * _pIP, COutputStates * _pOS, CA analyticalModel * _pAM \)](#) `[inline]`

6.2.2.3 [virtual GoSUM::CA analyticalOptimizationProblem::~~CA analyticalOptimizationProblem \(\)](#) `[inline], [virtual]`

6.2.3 Member Function Documentation

6.2.3.1 [ArrayXd GoSUM::CA analyticalOptimizationProblem::inputPoint2ModelPoint \(const ArrayXd & _ip \)](#)

Converts input parameter point to model point.

Reimplemented from [GoSUM::COriginalOptimizationProblem](#).

6.2.3.2 [template<class Archive > void GoSUM::CA analyticalOptimizationProblem::serialize \(Archive & ar, const unsigned int version \)](#) `[protected]`

Reimplemented from [GoSUM::COriginalOptimizationProblem](#).

6.2.4 Friends And Related Function Documentation

6.2.4.1 [friend class boost::serialization::access](#) `[friend]`

Boost serialization.

6.2.5 Member Data Documentation

6.2.5.1 [CA analyticalModel* GoSUM::CA analyticalOptimizationProblem::pAM](#) `[protected]`

Points to analytical model.

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

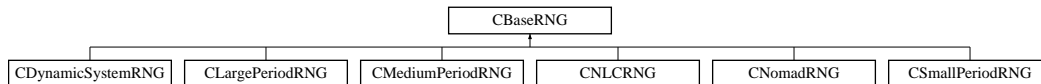
- C:/Development/core/[ParserOptimizationProblem.h](#)
- C:/Development/core/[ParserOptimizationProblem.cpp](#)

6.3 CBaseRNG Class Reference

Abstract base class for all uniform random number generators (RNG).

```
#include <RandomGenerators.h>
```

Inheritance diagram for CBaseRNG:



Public Member Functions

- virtual void [setSeed](#) (unsigned int s)=0
Sets seed of the RNG.
- virtual unsigned int [rndi](#) ()=0
Returns randomly generated unsigned int.
- virtual double [rnd](#) ()=0
Returns randomly generated double between 0 and 1.
- virtual double [rnd](#) (double a, double b)
Returns randomly generated double between a and b.

6.3.1 Detailed Description

Abstract base class for all uniform random number generators (RNG).

6.3.2 Member Function Documentation

6.3.2.1 virtual double CBaseRNG::rnd () [pure virtual]

Returns randomly generated double between 0 and 1.

Implemented in [CNomadRNG](#), [CNLCRNG](#), [CMediumPeriodRNG](#), [CSmallPeriodRNG](#), [CDynamicSystemRNG](#), and [CLargePeriodRNG](#).

6.3.2.2 virtual double CBaseRNG::rnd (double a, double b) [inline],[virtual]

Returns randomly generated double between a and b.

6.3.2.3 virtual unsigned int CBaseRNG::rndi () [pure virtual]

Returns randomly generated unsigned int.

Implemented in [CNomadRNG](#), [CNLCRNG](#), [CMediumPeriodRNG](#), [CSmallPeriodRNG](#), [CDynamicSystemRNG](#), and [CLargePeriodRNG](#).

6.3.2.4 virtual void CBaseRNG::setSeed (unsigned int s) [pure virtual]

Sets seed of the RNG.

Implemented in [CNomadRNG](#), [CNLCRNG](#), [CMediumPeriodRNG](#), [CSmallPeriodRNG](#), [CDynamicSystemRNG](#), and [CLargePeriodRNG](#).

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

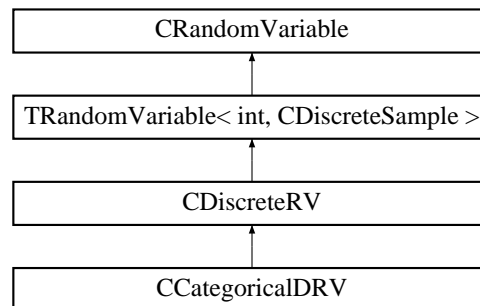
- C:/Development/core/[RandomGenerators.h](#)

6.4 CategoricalDRV Class Reference

Class for categorical discrete random variables derived from discrete random variables.

```
#include <RandomVariable.h>
```

Inheritance diagram for CategoricalDRV:



Public Member Functions

- [CategoricalDRV](#) ()
- virtual [~CategoricalDRV](#) ()
- [CategoricalDRV](#) (const [CategoricalDRV](#) &O)
- virtual void [setDistribution](#) (const [CDiscreteSample](#) &_aS)
Computes PMF and CDF from sample histogram.
- virtual double [probability](#) (int _k) const
Function that returns PMF.
- virtual double [cumulative](#) (int _k) const
Function that returns CDF.
- virtual int [expandedSize](#) () const
- virtual double [minValue](#) () const
After expansion, number of new variables is equal to the number of categories.
- virtual double [maxValue](#) () const
Returns maximal value of the random variable.
- virtual double [expectedValue](#) () const
Returns expected value of the random variable.
- virtual [distributiontype](#) [distributionType](#) () const
Returns enum type of the random variable distribution.
- virtual std::string [distributionName](#) () const
Returns name of the random variable distribution.
- virtual ArrayXi [exportDomain](#) () const
Exports domain of the random variable.
- virtual bool [isDistributionDefined](#) () const
Returns true if distribution is defined, false otherwise.
- virtual double [variance](#) () const
Returns variance of the random variable.

Protected Member Functions

- virtual double [doQuantile](#) (double _p) const
Quantile, formula implementation.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Private Attributes

- ArrayXd [p](#)
- ArrayXd [cdf](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.4.1 Detailed Description

Class for categorical discrete random variables derived from discrete random variables.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 `CategoricalDRV::CategoricalDRV ()` `[inline]`

6.4.2.2 `virtual CategoricalDRV::~~CategoricalDRV ()` `[inline], [virtual]`

6.4.2.3 `CategoricalDRV::CategoricalDRV (const CategoricalDRV & O)` `[inline]`

6.4.3 Member Function Documentation

6.4.3.1 `virtual double CategoricalDRV::cumulative (int _k) const` `[inline], [virtual]`

Function that returns CDF.

6.4.3.2 `virtual std::string CategoricalDRV::distributionName () const` `[inline], [virtual]`

Returns name of the random variable distribution.

Implements [CRandomVariable](#).

6.4.3.3 `virtual distributiontype CategoricalDRV::distributionType () const` `[inline], [virtual]`

Returns enum type of the random variable distribution.

Implements [CRandomVariable](#).

6.4.3.4 `double CategoricalDRV::doQuantile (double _p) const` `[protected], [virtual]`

Quantile, formula implementation.

Implements [CRandomVariable](#).

6.4.3.5 `virtual int CategoricalDRV::expandedSize () const` `[inline], [virtual]`

Reimplemented from [CRandomVariable](#).

6.4.3.6 `virtual double CategoricalDRV::expectedValue () const` `[inline], [virtual]`

Returns expected value of the random variable.

Implements [CRandomVariable](#).

6.4.3.7 `ArrayXi CategoricalDRV::exportDomain () const` `[virtual]`

Exports domain of the random variable.

Implements [TRandomVariable< t, T >](#).

6.4.3.8 `virtual bool CategoricalDRV::isDistributionDefined () const` `[inline], [virtual]`

Returns true if distribution is defined, false otherwise.

Reimplemented from [CRandomVariable](#).

6.4.3.9 `virtual double CategoricalDRV::maxValue () const` `[inline], [virtual]`

Returns maximal value of the random variable.

Implements [CRandomVariable](#).

6.4.3.10 `virtual double CategoricalDRV::minValue () const` `[inline], [virtual]`

After expansion, number of new variables is equal to the number of categories.

Returns minimal value of the random variable.

Implements [CRandomVariable](#).

6.4.3.11 `virtual double CategoricalDRV::probability (int _k) const` `[inline], [virtual]`

Function that returns PMF.

6.4.3.12 `template<class Archive > void CategoricalDRV::serialize (Archive & ar, const unsigned int version)`
`[private]`

6.4.3.13 `void CategoricalDRV::setDistribution (const CDiscreteSample & _aS)` `[virtual]`

Computes PMF and CDF from sample histogram.

6.4.3.14 `virtual double CategoricalDRV::variance () const` `[inline],[virtual]`

Returns variance of the random variable.

Implements [CRandomVariable](#).

6.4.4 Friends And Related Function Documentation

6.4.4.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.4.5 Member Data Documentation

6.4.5.1 `ArrayXd CategoricalDRV::cdf` `[private]`

PMF (p) and CDF (cdf) data.

6.4.5.2 `ArrayXd CategoricalDRV::p` `[private]`

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

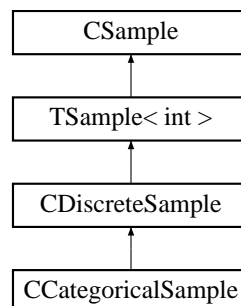
- C:/Development/core/[RandomVariable.h](#)
- C:/Development/core/[RandomVariable.cpp](#)

6.5 CategoricalSample Class Reference

Abstract class for categorical discrete samples.

```
#include <Sample.h>
```

Inheritance diagram for CategoricalSample:



Public Member Functions

- [CategoricalSample](#) ()
- [CategoricalSample](#) (const [CategoricalSample](#) &O)
- virtual [~CategoricalSample](#) ()
- virtual void [readSampleValue](#) (std::ifstream &_ifs, int _at)
Reads particular sample data from input file stream.
- virtual void [writeSampleValue](#) (std::ofstream &_ofs, int _at) const
< Writes particular sample data to output file stream.

- virtual void [setSampleSize](#) (int _n)
Sets particular sample data value.
- int [minValue](#) () const
Returns minimal sample data value.
- int [maxValue](#) () const
Returns maximal sample data value.
- virtual double [variance](#) () const
Returns sample variance (i.e. empirical).

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Private Attributes

- std::vector< std::string > [categories](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.5.1 Detailed Description

Abstract class for categorical discrete samples.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 `CCategoricalSample::CCategoricalSample ()` `[inline]`

6.5.2.2 `CCategoricalSample::CCategoricalSample (const CCategoricalSample & O)` `[inline]`

6.5.2.3 `virtual CCategoricalSample::~~CCategoricalSample ()` `[inline]`, `[virtual]`

6.5.3 Member Function Documentation

6.5.3.1 `int CCategoricalSample::maxValue () const` `[inline]`

Returns maximal sample data value.

Reimplemented from [TSample< t >](#).

6.5.3.2 `int CCategoricalSample::minValue () const` `[inline]`

Returns minimal sample data value.

Reimplemented from [TSample< t >](#).

6.5.3.3 `void CategoricalSample::readSampleValue (std::ifstream & _ifs, int _at)` [virtual]

Reads particular sample data from input file stream.

Reimplemented from [TSample< t >](#).

6.5.3.4 `template<class Archive > void CategoricalSample::serialize (Archive & ar, const unsigned int version)`
[private]

6.5.3.5 `virtual void CategoricalSample::setSampleSize (int _n)` [inline],[virtual]

Sets particular sample data value.

Reimplemented from [TSample< t >](#).

6.5.3.6 `virtual double CategoricalSample::variance () const` [inline],[virtual]

Returns sample variance (i.e. empirical).

Reimplemented from [CDiscreteSample](#).

6.5.3.7 `virtual void CategoricalSample::writeSampleValue (std::ofstream & _ofs, int _at) const` [inline],[virtual]

< Writes particular sample data to output file stream.

Reimplemented from [TSample< t >](#).

6.5.4 Friends And Related Function Documentation

6.5.4.1 `friend class boost::serialization::access` [friend]

Boost serialization.

6.5.5 Member Data Documentation

6.5.5.1 `std::vector<std::string> CategoricalSample::categories` [private]

Holds all category values.

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

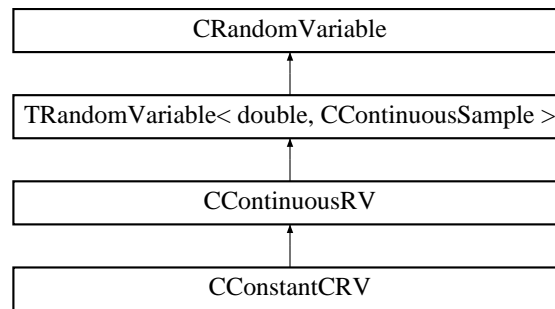
- C:/Development/core/[Sample.h](#)
- C:/Development/core/[Sample.cpp](#)

6.6 CConstantCRV Class Reference

Class for constant continuous random variables derived from continuous random variables.

`#include <RandomVariable.h>`

Inheritance diagram for CConstantCRV:



Public Member Functions

- `CConstantCRV ()`
- virtual `~CConstantCRV ()`
- `CConstantCRV (const CConstantCRV &O)`
- virtual void `setDistribution (double _xc, double _dp2=0)`
Set distribution parameter.
- virtual void `setDistribution (const CContinuousSample &_aS)`
Set distribution parameters from sample empirical parameters.
- virtual double `probability (double _x) const`
Function that returns PMF.
- virtual double `cumulative (double _x) const`
Function that returns CDF.
- virtual double `minValue () const`
Returns minimal value of the random variable.
- virtual double `maxValue () const`
Returns maximal value of the random variable.
- virtual double `expectedValue () const`
Returns expected value of the random variable.
- virtual `distributiontype distributionType () const`
Returns enum type of the random variable distribution.
- virtual `std::string distributionName () const`
Returns name of the random variable distribution.
- virtual `ArrayXd exportDomain () const`
Exports domain of the random variable.
- double `constantValue () const`
Returns value of constant random variable.
- void `setConstantValue (double _xc)`
Sets value of constant random variable.
- virtual double `variance () const`
Returns variance of the random variable.

Protected Member Functions

- virtual double `doQuantile (double _p) const`
Quantile, formula implementation.

Private Member Functions

- `template<class Archive >`
void `serialize (Archive &ar, const unsigned int version)`

Private Attributes

- double [xc](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.6.1 Detailed Description

Class for constant continuous random variables derived from continuous random variables.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 `CConstantCRV::CConstantCRV ()` [\[inline\]](#)

6.6.2.2 `virtual CConstantCRV::~~CConstantCRV ()` [\[inline\]](#), [\[virtual\]](#)

6.6.2.3 `CConstantCRV::CConstantCRV (const CConstantCRV & O)` [\[inline\]](#)

6.6.3 Member Function Documentation

6.6.3.1 `double CConstantCRV::constantValue () const` [\[inline\]](#)

Returns value of constant random variable.

6.6.3.2 `virtual double CConstantCRV::cumulative (double _x) const` [\[inline\]](#), [\[virtual\]](#)

Function that returns CDF.

6.6.3.3 `virtual std::string CConstantCRV::distributionName () const` [\[inline\]](#), [\[virtual\]](#)

Returns name of the random variable distribution.

Implements [CRandomVariable](#).

6.6.3.4 `virtual distributiontype CConstantCRV::distributionType () const` [\[inline\]](#), [\[virtual\]](#)

Returns enum type of the random variable distribution.

Implements [CRandomVariable](#).

6.6.3.5 `virtual double CConstantCRV::doQuantile (double _p) const` [\[inline\]](#), [\[protected\]](#), [\[virtual\]](#)

Quantile, formula implementation.

Implements [CRandomVariable](#).

6.6.3.6 virtual double CConstantCRV::expectedValue () const [inline],[virtual]

Returns expected value of the random variable.

Implements [CRandomVariable](#).

6.6.3.7 ArrayXd CConstantCRV::exportDomain () const [virtual]

Exports domain of the random variable.

Implements [TRandomVariable< t, T >](#).

6.6.3.8 virtual double CConstantCRV::maxValue () const [inline],[virtual]

Returns maximal value of the random variable.

Implements [CRandomVariable](#).

6.6.3.9 virtual double CConstantCRV::minValue () const [inline],[virtual]

Returns minimal value of the random variable.

Implements [CRandomVariable](#).

6.6.3.10 virtual double CConstantCRV::probability (double _x) const [inline],[virtual]

Function that returns PMF.

6.6.3.11 template<class Archive > void CConstantCRV::serialize (Archive & ar, const unsigned int version) [private]

6.6.3.12 void CConstantCRV::setConstantValue (double _xc) [inline]

Sets value of constant random variable.

6.6.3.13 void CConstantCRV::setDistribution (double _xc, double _dp2 = 0) [virtual]

Set distribution parameter.

6.6.3.14 virtual void CConstantCRV::setDistribution (const CContinuousSample & _aS) [inline],[virtual]

Set distribution parameters from sample empirical parameters.

6.6.3.15 virtual double CConstantCRV::variance () const [inline],[virtual]

Returns variance of the random variable.

Implements [CRandomVariable](#).

6.6.4 Friends And Related Function Documentation

6.6.4.1 friend class boost::serialization::access [friend]

Boost serialization.

6.6.5 Member Data Documentation

6.6.5.1 double CConstantCRV::xc [private]

Distribution parameter: constant integer value of the random variable.

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

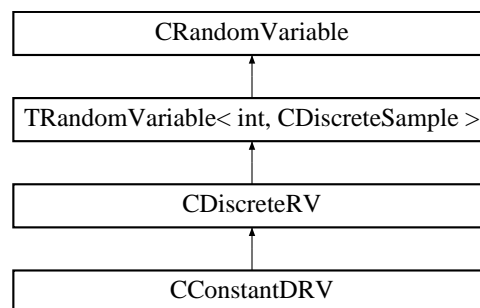
- C:/Development/core/[RandomVariable.h](#)
- C:/Development/core/[RandomVariable.cpp](#)

6.7 CConstantDRV Class Reference

Class for constant discrete random variables derived from discrete random variables.

```
#include <RandomVariable.h>
```

Inheritance diagram for CConstantDRV:



Public Member Functions

- [CConstantDRV](#) ()
- virtual [~CConstantDRV](#) ()
- [CConstantDRV](#) (const [CConstantDRV](#) &O)
- virtual void [setDistribution](#) (int _c, int _dp2=0)
Set distribution parameter.
- virtual void [setDistribution](#) (const [CDiscreteSample](#) &_aS)
Computes PMF and CDF from sample histogram.
- virtual double [probability](#) (int _k) const
Function that returns PMF.
- virtual double [cumulative](#) (int _k) const
Function that returns CDF.
- virtual double [minValue](#) () const
Returns minimal value of the random variable.
- virtual double [maxValue](#) () const
Returns maximal value of the random variable.
- virtual double [expectedValue](#) () const
Returns expected value of the random variable.
- virtual [distributiontype](#) [distributionType](#) () const
Returns enum type of the random variable distribution.
- virtual std::string [distributionName](#) () const
Returns name of the random variable distribution.
- virtual ArrayXi [exportDomain](#) () const

- Exports domain of the random variable.
- int `constantValue` () const
Returns value of constant random variable.
- void `setConstantValue` (int _c)
Sets value of constant random variable.
- virtual double `variance` () const
Returns variance of the random variable.

Protected Member Functions

- virtual double `doQuantile` (double _p) const
Quantile, formula implementation.

Private Member Functions

- template<class Archive >
void `serialize` (Archive &ar, const unsigned int version)

Private Attributes

- int `c`

Friends

- class `boost::serialization::access`
Boost serialization.

Additional Inherited Members

6.7.1 Detailed Description

Class for constant discrete random variables derived from discrete random variables.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 `CConstantDRV::CConstantDRV ()` [inline]

6.7.2.2 `virtual CConstantDRV::~~CConstantDRV ()` [inline], [virtual]

6.7.2.3 `CConstantDRV::CConstantDRV (const CConstantDRV & O)` [inline]

6.7.3 Member Function Documentation

6.7.3.1 `int CConstantDRV::constantValue () const` [inline]

Returns value of constant random variable.

6.7.3.2 `virtual double CConstantDRV::cumulative (int _k) const` [inline], [virtual]

Function that returns CDF.

6.7.3.3 `virtual std::string CConstantDRV::distributionName () const [inline], [virtual]`

Returns name of the random variable distribution.

Implements [CRandomVariable](#).

6.7.3.4 `virtual distributiontype CConstantDRV::distributionType () const [inline], [virtual]`

Returns enum type of the random variable distribution.

Implements [CRandomVariable](#).

6.7.3.5 `virtual double CConstantDRV::doQuantile (double _p) const [inline], [protected], [virtual]`

Quantile, formula implementation.

Implements [CRandomVariable](#).

6.7.3.6 `virtual double CConstantDRV::expectedValue () const [inline], [virtual]`

Returns expected value of the random variable.

Implements [CRandomVariable](#).

6.7.3.7 `ArrayXi CConstantDRV::exportDomain () const [virtual]`

Exports domain of the random variable.

Implements [TRandomVariable< t, T >](#).

6.7.3.8 `virtual double CConstantDRV::maxValue () const [inline], [virtual]`

Returns maximal value of the random variable.

Implements [CRandomVariable](#).

6.7.3.9 `virtual double CConstantDRV::minValue () const [inline], [virtual]`

Returns minimal value of the random variable.

Implements [CRandomVariable](#).

6.7.3.10 `virtual double CConstantDRV::probability (int _k) const [inline], [virtual]`

Function that returns PMF.

6.7.3.11 `template<class Archive > void CConstantDRV::serialize (Archive & ar, const unsigned int version) [private]`

6.7.3.12 `void CConstantDRV::setConstantValue (int _c) [inline]`

Sets value of constant random variable.

6.7.3.13 `void CConstantDRV::setDistribution (int _c, int _dp2 = 0) [virtual]`

Set distribution parameter.

6.7.3.14 `virtual void CConstantDRV::setDistribution (const CDiscreteSample & _aS) [inline],[virtual]`

Computes PMF and CDF from sample histogram.

6.7.3.15 `virtual double CConstantDRV::variance () const [inline],[virtual]`

Returns variance of the random variable.

Implements [CRandomVariable](#).

6.7.4 Friends And Related Function Documentation

6.7.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.7.5 Member Data Documentation

6.7.5.1 `int CConstantDRV::c [private]`

Distribution parameter: constant integer value of the random variable.

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

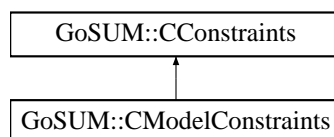
- C:/Development/core/[RandomVariable.h](#)
- C:/Development/core/[RandomVariable.cpp](#)

6.8 GoSUM::CConstraints Class Reference

Class for the constraints with parser functions.

```
#include <Constraints.h>
```

Inheritance diagram for GoSUM::CConstraints:



Public Member Functions

- [CConstraints](#) ()
- virtual [~CConstraints](#) ()
- virtual void [clear](#) ()
Clears object.
- int [size](#) () const
Returns size of the constraints.
- void [addExpression](#) (const std::string &_gexpr)
Adds constraint expression.
- void [eraseExpression](#) (int _at)
Erases particular constraint expression.

- void [setExpression](#) (const std::string &_gexpr, int _at)
Sets particular cosntraint expression.
- std::string [expression](#) (int _at) const
Returns particular constraint expression.
- std::string [roundoffEquality](#) (std::string _expr)
If _expr is an equality expression left=right it returns expression $abs(left-right) \leq TINY$.
- bool [validateExpressions](#) ()
Validates all expressions.
- void [parseExpressions](#) ()
Parses all expressions.
- double [evaluate](#) (const ArrayXd &_x, int _at)
Evalautes particular constraint value from variables values.
- bool [constraintsSatisfied](#) (const ArrayXd &_x)
Returns true if _x satisfies constraints, false otherwise.
- bool [findInExpressions](#) (const std::string &_name) const
Returns true if _name is found in constraint expressions, false otherwise.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- virtual void [setVariableNames](#) ()=0
Sets variable names for the parser.

Protected Attributes

- std::string [names](#)
Holds all names that are permitted in objective and constraint expressions.
- std::vector< std::string > [gexpr](#)
Holds expressions for constraints.
- FunctionParser [f](#)
Holds base function parser.
- std::vector< FunctionParser > [g](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.8.1 Detailed Description

Class for the constraints with parser functions.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 `GoSUM::CConstraints::CConstraints ()` `[inline]`

6.8.2.2 `virtual GoSUM::CConstraints::~~CConstraints ()` `[inline],[virtual]`

6.8.3 Member Function Documentation

6.8.3.1 `void GoSUM::CConstraints::addExpression (const std::string & _expr)` `[inline]`

Adds constraint expression.

6.8.3.2 `virtual void GoSUM::CConstraints::clear ()` `[inline],[virtual]`

Clears object.

Reimplemented in [GoSUM::CModelConstraints](#).

6.8.3.3 `bool GoSUM::CConstraints::constraintsSatisfied (const ArrayXd & _x)`

Returns true if *_x* satisfies constraints, false otherwise.

6.8.3.4 `void GoSUM::CConstraints::eraseExpression (int _at)`

Erases particular constraint expression.

6.8.3.5 `double GoSUM::CConstraints::evaluate (const ArrayXd & _x, int _at)` `[inline]`

Evaluate particular constraint value from variables values.

6.8.3.6 `std::string GoSUM::CConstraints::expression (int _at) const` `[inline]`

Returns particular constraint expression.

6.8.3.7 `bool GoSUM::CConstraints::findInExpressions (const std::string & _name) const`

Returns true if *_name* is found in constraint expressions, false otherwise.

6.8.3.8 `void GoSUM::CConstraints::parseExpressions ()`

Parses all expressions.

6.8.3.9 `std::string GoSUM::CConstraints::roundoffEquality (std::string _expr)`

If *_expr* is an equality expression left=right it returns expression $\text{abs}(\text{left-right}) \leq \text{TINY}$.

6.8.3.10 `template<class Archive > void GoSUM::CConstraints::serialize (Archive & ar, const unsigned int version)`
`[protected]`

Reimplemented in [GoSUM::CModelConstraints](#).

6.8.3.11 `void GoSUM::CConstraints::setExpression (const std::string & _expr, int _at)`

Sets particular cosntraint expression.

6.8.3.12 `virtual void GoSUM::CConstraints::setVariableNames () [protected], [pure virtual]`

Sets variable names for the parser.

Implemented in [GoSUM::CModelConstraints](#).

6.8.3.13 `int GoSUM::CConstraints::size () const [inline]`

Returns size of the constraints.

6.8.3.14 `bool GoSUM::CConstraints::validateExpressions ()`

Validates all expressions.

6.8.4 Friends And Related Function Documentation

6.8.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.8.5 Member Data Documentation

6.8.5.1 `FunctionParser GoSUM::CConstraints::f [protected]`

Holds base function parser.

6.8.5.2 `std::vector<FunctionParser> GoSUM::CConstraints::g [protected]`

Holds function parsers for constraints.

6.8.5.3 `std::vector<std::string> GoSUM::CConstraints::gexpr [protected]`

Holds expressions for constraints.

6.8.5.4 `std::string GoSUM::CConstraints::names [protected]`

Holds all names that are permitted in objective and constraint expressions.

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

- C:/Development/core/[Constraints.h](#)
- C:/Development/core/[Constraints.cpp](#)

6.9 GoSUM::CContainer Class Reference

Class for the [GoSUM](#) main project.

```
#include <Container.h>
```

Public Types

- enum [projecttype](#) {
 [samplegeneration](#), [modelanalysis](#), [dataanalysis](#), [simpleoptimization](#),
 [modeloptimization](#), [learnedmodeloptimization](#), [learneddataoptimization](#) }
- enum [optimizationmethodtype](#) { [mads](#), [ga](#) }

Public Member Functions

- [CContainer](#) ()
- virtual [~CContainer](#) ()
- void [addDefaultVariable](#) ([CModelVariables](#) *pMVs, [CRandomVariable::distributiontype](#) _dtype)
 Adds one model variable, input and output, with default distribution parameter values.
- void [addVariable](#) ([CModelVariables](#) *pMVs, [CRandomVariable::distributiontype](#) _dtype, double _a=0., double _b=0.)
 Adds one model variable, input and output.
- void [addVariable](#) ([CModelVariables](#) *pMVs, const std::string &_name, [CRandomVariable::distributiontype](#) _dtype, double _a=0., double _b=0.)
 Adds one model variable, input and output.
- void [eraseVariable](#) ([CModelVariables](#) *pMVs, std::string _name)
 Erases model variable with _name.
- void [cloneVariable](#) ([CModelVariables](#) *pMVs, int _at)
 Clones particular model variable.
- void [addVariables](#) ([CModelVariables](#) *pMVs, int _N, [CRandomVariable::distributiontype](#) _dtype, double _a=0., double _b=0.)
 Adds multiple model variables, input and output.
- void [addVariables](#) ([CModelVariables](#) *pMVs, const std::string &_name, int _N, [CRandomVariable::distributiontype](#) _dtype, double _a=0., double _b=0.)
 Adds multiple model variables, input and output.
- void [addInput](#) (const std::string &_name, [CRandomVariable::distributiontype](#) _type, double _a=0., double _b=0.)
 Adds one input parameter.
- void [addOutput](#) (const std::string &_name, [CRandomVariable::distributiontype](#) _type, double _a=0., double _b=0.)
 Adds one output state.
- void [addInputs](#) (int _N, const std::string &_name, [CRandomVariable::distributiontype](#) _type, double _a=0., double _b=0.)
 Adds multiple input parameters.
- void [addOutputs](#) (int _N, const std::string &_name, [CRandomVariable::distributiontype](#) _type, double _a=0., double _b=0.)
 Adds multiple output states.
- void [save](#) ()
 Saves project to binary format.
- void [load](#) ()
 Loads project from binary format.
- void [saveXml](#) ()
 Saves project to xml format.
- void [loadXml](#) ()
 Loads project from xml format.
- void [saveTxt](#) ()
 Saves project to txt format.
- void [loadTxt](#) ()

- Loads project from txt format.*

 - bool [containsTheoreticalVariables](#) (const std::string &_fname)
Returns true if file _fname contains theoretical variables, false otherwise.
 - bool [containsNamedTheoreticalVariables](#) (const std::string &_fname)
Returns true if file _fname contains named theoretical variables, false otherwise.
 - bool [containsEmpiricalVariables](#) (const std::string &_fname, int &SSize, std::vector< [CRandomVariable::distributiontype](#) > &dtypes)
Returns true if file _fname contains empirical variables, false otherwise.
 - bool [containsNamedEmpiricalVariables](#) (const std::string &_fname, int &SSize, std::vector< [CRandomVariable::distributiontype](#) > &dtypes)
Returns true if file _fname contains named empirical variables, false otherwise.
 - bool [containsDeclaredEmpiricalVariables](#) (const std::string &_fname, int &SSize, std::vector< [CRandomVariable::distributiontype](#) > &dtypes)
Returns true if file _fname contains explicitly typed empirical variables, false otherwise.
 - bool [containsNamedDeclaredEmpiricalVariables](#) (const std::string &_fname, int &SSize, std::vector< [CRandomVariable::distributiontype](#) > &dtypes)
Returns true if file _fname contains named explicitly typed empirical variables, false otherwise.
 - bool [containsPredictionSamples](#) (const std::string &_fname, int &Ncols, int &Nrows)
Returns true if file _fname contains prediction samples, false otherwise.
 - void [importTheoreticalVariables](#) ([GoSUM::CModelVariables](#) *pMVs, const std::string &_fname)
Imports multiple theoretical model variables from a file.
 - void [importNamedTheoreticalVariables](#) ([GoSUM::CModelVariables](#) *pMVs, const std::string &_fname)
Imports multiple theoretical model variables from a file.
 - void [importEmpiricalVariables](#) ([GoSUM::CModelVariables](#) *pMVs, const std::string &_fname, int &SSize, std::vector< [CRandomVariable::distributiontype](#) > &dtypes)
Imports multiple empirical model variables from a file.
 - void [importNamedEmpiricalVariables](#) ([GoSUM::CModelVariables](#) *pMVs, const std::string &_fname, int &SSize, std::vector< [CRandomVariable::distributiontype](#) > &dtypes)
Imports multiple empirical model variables from a file.
 - void [importDeclaredEmpiricalVariables](#) ([GoSUM::CModelVariables](#) *pMVs, const std::string &_fname, int &SSize, std::vector< [CRandomVariable::distributiontype](#) > &dtypes)
Imports multiple empirical model variables from a file.
 - void [importNamedDeclaredEmpiricalVariables](#) ([GoSUM::CModelVariables](#) *pMVs, const std::string &_fname, int &SSize, std::vector< [CRandomVariable::distributiontype](#) > &dtypes)
Imports multiple empirical model variables from a file.
 - void [importVariables](#) ([GoSUM::CModelVariables](#) *pMVs, const std::string &_fname)
Imports multiple model variables from a file, input and output.
 - void [exportSamples](#) ([CModelVariables](#) *pMVs, const std::string &_fname)
Exports samples for model variables, input and output.
 - void [importInputs](#) (const std::string &_fname)
Imports multiple input parameters.
 - void [importOutputs](#) (const std::string &_fname)
Imports multiple output states.
 - void [exportInputSamples](#) (const std::string &_fname)
Imports input samples from file.
 - void [exportOutputSamples](#) (const std::string &_fname)
Imports output samples from file.
 - bool [importPredictionInputSamples](#) (const std::string &_fname)
Imports prediction input samples.
 - void [exportPredictionOutputSamples](#) (const std::string &_fname)
Exports prediction output samples.
 - void [exportDerivativeSensitivity](#) (const std::string &_fname)

- Exports derivative sensitivity.*
 - void [exportAverageDerivative](#) (const std::string &_fname)
- Exports average derivative sensitivity.*
 - void [exportAbsoluteAverageDerivative](#) (const std::string &_fname)
- Exports absolute average derivative sensitivity.*
 - void [exportVarianceSensitivity](#) (const std::string &_fname)
- Exports variance sensitivity.*
 - void [exportFirstOrderANOVA](#) (const std::string &_fname)
- Exports first order ANOVA sensitivity.*
 - void [exportOptimizationMethod](#) (const std::string &_fname)
- Exports optimization method.*
 - void [exportOptimizationHistory](#) (const std::string &_fname)
- Exports optimization history.*
 - void [importModelConstraints](#) (const std::string &_fname)
- Imports model constraint expressions.*
 - void [importOptimizationConstraints](#) (const std::string &_fname)
- Imports optimization constraint expressions.*
 - void [importLowerBound](#) (const std::string &_fname)
- Imports lower bounds for optimization variables.*
 - void [importUpperBound](#) (const std::string &_fname)
- Imports upper bounds for optimization variables.*
 - void [importInitialValue](#) (const std::string &_fname)
- Imports initial values for optimization variables.*
 - void [resampleInputs](#) ()
- Resamples input parameters.*
 - void [evaluateOutputs](#) ()
- Evaluates output states using external executable.*
 - void [learnModel](#) ()
- Learns model.*
 - void [predict](#) ()
- Predicts (using analytical model).*
 - void [predictMean](#) (ArrayXd &ymu, ArrayXd &yvar)
- Predicts mean and variance using learned model.*
 - void [computeSensitivities](#) ()
- Computes sensitivities.*
 - void [optimize](#) ()
- Optimizes.*
 - void [reduce](#) ()
- Reduces model.*
 - void [minimize](#) ()
- Minimizes.*
 - void [maximize](#) ()
- Maximizes.*
 - void [clear](#) ()
- Clears all project content.*
 - void [clearResults](#) ()
- Clears all project results.*
 - void [clearForReducing](#) ()
- Clears all results except reducer.*
 - void [clearSamplingResults](#) (CModelVariables *_pMVs)
- Clears project results starting from particular sampling.*

- void `clearLearningResults` ()
Clears project results, starting from learning.
- void `clearSensitivityResults` ()
Clears project results, starting from sensitivity.
- void `clearOptimizationResults` ()
Clears optimization results.
- bool `emptyResults` () const
Returns true if results are empty, false otherwise.
- bool `emptySamplingResults` (CModelVariables *_pMVs) const
Returns true if results starting from the _pMVs sampling are empty, false otherwise.
- bool `emptyLearningResults` () const
Returns true if learning results are empty, false otherwise.
- bool `emptySensitivityResults` () const
Returns true if sensitivity results are empty, false otherwise.
- bool `emptyOptimizationResults` () const
Returns true if optimization results are empty, false otherwise.
- bool `emptySelectedSamples` () const
Returns true if selected samples are empty, false otherwise.
- void `setProjectPath` (const std::string &_prjPath)
Sets project path.
- std::string `projectPath` ()
Returns project path.
- void `setProjectName` (const std::string &_prjName)
Sets project name.
- std::string `projectName` ()
Returns project name.
- std::string `longProjectName` ()
Returns long project name.
- void `setProjectType` (projecttype _prjType)
Sets chosen project type.
- projecttype `projectType` ()
Returns chosen project type.
- void `setOptimizationMethod` (optimizationmethodtype _omType)
Sets chosen optimization method type.
- optimizationmethodtype `optimizationMethod` ()
Returns chosen optimization method type.
- int `learnEvaluationSize` ()
Returns maximal number of learn optimization evaluations.
- void `setThreadSize` (int _trdN)
Sets chosen thread size.
- void `setMatLabPath` (const std::string &_matlabPath)
Sets MatLab path.
- void `setRNG` (CRandomGenerator::rngtype _type)
Sets chosen rng type.
- void `setResampleType` (CHypercube::hctype _type)
Sets resample type.
- void `setResampleSize` (int _N)
Sets resample size.
- void `setVoronoiOptions` (int _maxiter, int _q, double _alpha2, double _beta2)
Sets Voronoi options.
- void `setModelEvaluator` (GoSUM::CModelEvaluator::evaluator_type _me, const std::string &_filename)

- Sets model evaluator.*

 - void [setSensitivityOptions](#) (int _N, double _eps1=0.005, double _eps2=0.01, double _eps3=0.01)
- Sets sensitivity parameters.*

 - void [setReductionType](#) (GoSUM::CReduction::reductiontype _rtype)
- Sets model reduction type.*

 - void [setReductionOutputs](#) (const std::vector< std::string > &_selOS)
- Sets reduction outputs.*

 - void [setReductionCutoffSize](#) (int _cutip)
- Sets reduction inputs cutoff size.*

 - void [setReductionCutoffValue](#) (double _cutval)
- Sets reduction inputs cut value.*

 - void [resetOptimizationVariable](#) (CModelVariables *pMVs, int _at)
- Resets values in the particular optimization variable.*

 - void [setObjective](#) (const std::string &_fexpr)
- Sets objective of the optimization problem.*

 - void [addOptimizationConstraint](#) (const std::string &_gexpr)
- Adds constraint of the optimization problem.*

 - void [setLowerBound](#) (const ArrayXd &_xL)
- Sets lower bound of the optimization variables.*

 - void [setUpperBound](#) (const ArrayXd &_xU)
- Sets upper bound of the optimization variables.*

 - void [setInitialValue](#) (const ArrayXd &_x0)
- Sets initial value of the optimization variables.*

 - void [setMadsMaxEvaluation](#) (int _maxeval)
- Sets MADS maximal number of evaluations.*

 - void [setMadsLHSearch](#) (int _lh0, int _lhi)
- Sets MADS lh search parameters.*

 - void [setMadsInitMeshSize](#) (double _ims)
- Sets MADS initial mesh size.*

 - void [setMadsMinPollSize](#) (double _mps)
- Sets MADS final poll size.*

 - bool [isSampleSelected](#) (int _i) const
- Returns true if particular sample value is selected, false otherwise.*

 - void [selectSamples](#) (CModelVariable *pmv, double _left, double _right)
- Selects sample values of variable pmv that are contained in [_left, _right] interval.*

 - void [separateBySelection](#) (const ArrayXd &X, std::vector< double > &x, std::vector< double > &selx) const
- Separates array X into two vectors based on the index (not contained/contained in selected samples).*

 - void [eraseSelectedSamples](#) ()
- Erases selected sample values in all model variables.*

 - int [inputsSize](#) ()
- Returns input parameters size.*

 - int [outputsSize](#) ()
- Returns output states size.*

 - CInputParameters & [inputParameters](#) ()
- Returns input parameters.*

 - CModelConstraints & [modelConstraints](#) ()
- Returns model constraints.*

 - CHypercube & [hyperCube](#) ()
- Returns hypercube.*

 - CModelVariable & [inputParameter](#) (int _at)
- Returns particular input parameter.*

- [CModelVariable](#) & [outputState](#) (int _at)
Returns particular output state.
- [COutputStates](#) & [outputStates](#) ()
Returns output states.
- [CEvaluator](#) & [modelEvaluator](#) ()
Returns evaluator.
- [CA analyticalModel](#) & [analyticalModel](#) ()
Returns analytical model.
- [CSensitivityAnalysis](#) & [sensitivityAnalysis](#) ()
Returns sensitivity analysis.
- [CReduction](#) & [reducer](#) ()
Returns reduction tool.
- [CSimpleOptimizationProblem](#) * [optimizationProblem](#) ()
Returns optimization problem.
- [CMADS](#) & [madsOptimizer](#) ()
Returns MADS.
- [CGAModelOptimization](#) & [gaOptimizer](#) ()
Returns MADS.
- `std::vector< int >` & [selectedSamples](#) ()
- `bool` [readyToSampleInputs](#) ()
Returns true if project is ready to sample inputs, false otherwise.
- `bool` [readyToEvaluateOutputs](#) ()
Returns true if project is ready to evaluate outputs, false otherwise.
- `bool` [readyToLearnModel](#) ()
Returns true if project is ready to learn model, false otherwise.
- `bool` [readyToComputeSensitivities](#) ()
Returns true if project is ready to compute sensitivities, false otherwise.
- `bool` [readyToReduceModel](#) ()
Returns true if project is ready to reduce model, false otherwise.
- `bool` [readyToSetOptimization](#) ()
Returns true if project is ready to set optimization, false otherwise.
- `bool` [readyToOptimize](#) ()
Returns true if project is ready to optimize, false otherwise.
- `void` [startResampling](#) ()
Starts sampling inputs.
- `void` [startEvaluating](#) ()
Starts evaluating outputs.
- `void` [startLearning](#) ()
Starts learning model.
- `void` [startSensitivityComputing](#) ()
Starts sensitivity computing.
- `void` [startOptimizing](#) ()
Starts optimizing.
- `void` [join](#) ()
Joins thread for computation.
- `int` [optimizationStepsSize](#) () `const`
Returns maximal number of optimization evaluations.
- `int` [learningStepsSize](#) () `const`
Returns learning steps size.

Static Public Member Functions

- static [projecttype](#) [ProjectType](#) (const std::string &_stype)
Converts project type name (string) to projecttype enumerator.
- static [optimizationmethodtype](#) [OptimizationMethodType](#) (const std::string &_stype)
Converts optimization method name (string) to optimizationmethodtype enumerator.

Public Attributes

- boost::thread [thrd](#)
Thread for computations.
- boost::signal< void()> [resamplingFinished](#)
Signal for sampling inputs finshed.
- boost::signal< void()> [evaluatingFinished](#)
Signal for evaluating outputs finshed.
- boost::signal< void()> [learningFinished](#)
Signal for learning model finshed.
- boost::signal< void()> [sensitivityComputingFinished](#)
Signal for sensitivity computing finshed.
- boost::signal< void()> [optimizingFinished](#)
Signal for optimizing finshed.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Private Attributes

- std::string [prjPath](#)
- std::string [prjName](#)
Project path and name.
- [projecttype](#) [prjType](#)
Project type.
- [optimizationmethodtype](#) [omType](#)
Optimization method type.
- [CHypercube](#) [hycube](#)
Basic hypercube, used for resampling.
- [CModelConstraints](#) [inconsts](#)
Model constraints, i.e. constraints on input paramters.
- [CInputParameters](#) [inputs](#)
Input parameters of the project.
- [COutputStates](#) [outputs](#)
Output states of the project.
- [CEvaluator](#) [evaluator](#)
Basic evalautor of the project, used only for model based projects.
- [CAnalyticalModel](#) [analytical](#)
Analytic model of the project.
- [CSensitivityAnalysis](#) [sensitivity](#)
Sensistivity analysis of the analytical model.

- [CReduction reduced](#)
Reduction operator.
- [CSimpleOptimizationProblem * pOP](#)
Defined optimization problem.
- [CMADS MADS](#)
Mesh Adaptive Direct Search optimization method.
- [CGAModelOptimization GAMO](#)
Genetic Algorithms optimization method.
- `std::vector< int >` [selectedsamples](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.9.1 Detailed Description

Class for the [GoSUM](#) main project.

6.9.2 Member Enumeration Documentation

6.9.2.1 enum GoSUM::CContainer::optimizationmethodtype

Enumerator:

mads
ga

6.9.2.2 enum GoSUM::CContainer::projecttype

Enumerator:

samplegeneration
modelanalysis
dataanalysis
simpleoptimization
modeloptimization
learnedmodeloptimization
learneddataoptimization

6.9.3 Constructor & Destructor Documentation

6.9.3.1 GoSUM::CContainer::CContainer ()

6.9.3.2 GoSUM::CContainer::~~CContainer () [virtual]

6.9.4 Member Function Documentation

6.9.4.1 void GoSUM::CContainer::addDefaultVariable (GoSUM::CModelVariables * pMVs, CRandomVariable::distributiontype _dtype)

Adds one model variable, input and output, with default distribution parameter values.

6.9.4.2 `void GoSUM::CContainer::addInput (const std::string & _name, CRandomVariable::distributiontype _type, double _a = 0., double _b = 0.) [inline]`

Adds one input parameter.

6.9.4.3 `void GoSUM::CContainer::addInputs (int _N, const std::string & _name, CRandomVariable::distributiontype _type, double _a = 0., double _b = 0.) [inline]`

Adds multiple input parameters.

6.9.4.4 `void GoSUM::CContainer::addOptimizationConstraint (const std::string & _gexpr)`

Adds constraint of the optimization problem.

6.9.4.5 `void GoSUM::CContainer::addOutput (const std::string & _name, CRandomVariable::distributiontype _type, double _a = 0., double _b = 0.) [inline]`

Adds one output state.

6.9.4.6 `void GoSUM::CContainer::addOutputs (int _N, const std::string & _name, CRandomVariable::distributiontype _type, double _a = 0., double _b = 0.) [inline]`

Adds multiple output states.

6.9.4.7 `void GoSUM::CContainer::addVariable (GoSUM::CModelVariables * pMVs, CRandomVariable::distributiontype _dtype, double _a = 0., double _b = 0.)`

Adds one model variable, input and output.

6.9.4.8 `void GoSUM::CContainer::addVariable (GoSUM::CModelVariables * pMVs, const std::string & _name, CRandomVariable::distributiontype _dtype, double _a = 0., double _b = 0.)`

Adds one model variable, input and output.

6.9.4.9 `void GoSUM::CContainer::addVariables (GoSUM::CModelVariables * pMVs, int _N, CRandomVariable::distributiontype _dtype, double _a = 0., double _b = 0.)`

Adds multiple model variables, input and output.

6.9.4.10 `void GoSUM::CContainer::addVariables (GoSUM::CModelVariables * pMVs, const std::string & _name, int _N, CRandomVariable::distributiontype _dtype, double _a = 0., double _b = 0.)`

Adds multiple model variables, input and output.

6.9.4.11 `CA analyticalModel& GoSUM::CContainer::analyticalModel () [inline]`

Returns analytical model.

6.9.4.12 `void GoSUM::CContainer::clear ()`

Clears all project content.

6.9.4.13 void GoSUM::CContainer::clearForReducing ()

Clears all results except reducer.

6.9.4.14 void GoSUM::CContainer::clearLearningResults ()

Clears project results, starting from learning.

6.9.4.15 void GoSUM::CContainer::clearOptimizationResults ()

Clears optimization results.

6.9.4.16 void GoSUM::CContainer::clearResults ()

Clears all project results.

6.9.4.17 void GoSUM::CContainer::clearSamplingResults (CModelVariables * *_pMVs*)

Clears project results starting from particular sampling.

6.9.4.18 void GoSUM::CContainer::clearSensitivityResults ()

Clears project results, starting from sensitivity.

6.9.4.19 void GoSUM::CContainer::cloneVariable (CModelVariables * *pMVs*, int *_at*)

Clones particular model variable.

6.9.4.20 void GoSUM::CContainer::computeSensitivities ()

Computes sensitivities.

6.9.4.21 bool GoSUM::CContainer::containsDeclaredEmpiricalVariables (const std::string & *_fname*, int & *SSize*, std::vector< CRandomVariable::distributiontype > & *dtypes*)

Returns true if file *_fname* contains explicitly typed empirical variables, false otherwise.

6.9.4.22 bool GoSUM::CContainer::containsEmpiricalVariables (const std::string & *_fname*, int & *SSize*, std::vector< CRandomVariable::distributiontype > & *dtypes*)

Returns true if file *_fname* contains empirical variables, false otherwise.

6.9.4.23 bool GoSUM::CContainer::containsNamedDeclaredEmpiricalVariables (const std::string & *_fname*, int & *SSize*, std::vector< CRandomVariable::distributiontype > & *dtypes*)

Returns true if file *_fname* contains named explicitly typed empirical variables, false otherwise.

6.9.4.24 `bool GoSUM::CContainer::containsNamedEmpiricalViarables (const std::string & _fname, int & SSize, std::vector< CRandomVariable::distributiontype > & dtypes)`

Returns true if file _fname contains named empirical variables, false otherwise.

6.9.4.25 `bool GoSUM::CContainer::containsNamedTheoreticalViarables (const std::string & _fname)`

Returns true if file _fname contains named theoretical variables, false otherwise.

6.9.4.26 `bool GoSUM::CContainer::containsPredictionSamples (const std::string & _fname, int & Ncols, int & Nrows)`

Returns true if file _fname contains prediction samples, false otherwise.

6.9.4.27 `bool GoSUM::CContainer::containsTheoreticalViarables (const std::string & _fname)`

Returns true if file _fname contains theoretical variables, false otherwise.

6.9.4.28 `bool GoSUM::CContainer::emptyLearningResults () const`

Returns true if learning results are empty, false otherwise.

6.9.4.29 `bool GoSUM::CContainer::emptyOptimizationResults () const`

Returns true if optimization results are empty, false otherwise.

6.9.4.30 `bool GoSUM::CContainer::emptyResults () const`

Returns true if results are empty, false otherwise.

6.9.4.31 `bool GoSUM::CContainer::emptySamplingResults (CModelVariables * _pMVs) const`

Returns true if results starting from the _pMVs sampling are empty, false otherwise.

6.9.4.32 `bool GoSUM::CContainer::emptySelectedSamples () const` `[inline]`

Returns true if selected samples are empty, false otherwise.

6.9.4.33 `bool GoSUM::CContainer::emptySensitivityResults () const`

Returns true if sensitivity results are empty, false otherwise.

6.9.4.34 `void GoSUM::CContainer::eraseSelectedSamples ()`

Erases selected sample values in all model variables.

6.9.4.35 `void GoSUM::CContainer::eraseVariable (CModelVariables * pMVs, std::string _name)`

Erases model variable with _name.

6.9.4.36 void GoSUM::CContainer::evaluateOutputs ()

Evaluates output states using external executable.

6.9.4.37 void GoSUM::CContainer::exportAbsoluteAverageDerivative (const std::string & *_fname*)

Exports absolute average derivative sensitivity.

6.9.4.38 void GoSUM::CContainer::exportAverageDerivative (const std::string & *_fname*)

Exports average derivative sensitivity.

6.9.4.39 void GoSUM::CContainer::exportDerivativeSensitivity (const std::string & *_fname*)

Exports derivative sensitivity.

6.9.4.40 void GoSUM::CContainer::exportFirstOrderANOVA (const std::string & *_fname*)

Exports first order ANOVA sensitivity.

6.9.4.41 void GoSUM::CContainer::exportInputSamples (const std::string & *_fname*) [inline]

Imports input samples from file.

6.9.4.42 void GoSUM::CContainer::exportOptimizationHistory (const std::string & *_fname*)

Exports optimization history.

6.9.4.43 void GoSUM::CContainer::exportOptimizationMethod (const std::string & *_fname*)

Exports optimization method.

6.9.4.44 void GoSUM::CContainer::exportOutputSamples (const std::string & *_fname*) [inline]

Imports output samples from file.

6.9.4.45 void GoSUM::CContainer::exportPredictionOutputSamples (const std::string & *_fname*)

Exports prediction output samples.

6.9.4.46 void GoSUM::CContainer::exportSamples (CModelVariables * *pmVs*, const std::string & *_fname*)

Exports samples for model variables, input and output.

6.9.4.47 void GoSUM::CContainer::exportVarianceSensitivity (const std::string & *_fname*)

Exports variance sensitivity.

6.9.4.48 CGAModelOptimization& GoSUM::CContainer::gaOptimizer () `[inline]`

Returns MADS.

6.9.4.49 CHypercube& GoSUM::CContainer::hyperCube () `[inline]`

Returns hypercube.

6.9.4.50 void GoSUM::CContainer::importDeclaredEmpiricalVariables (GoSUM::CModelVariables * *pmVs*, const std::string & *_fname*, int & *SSize*, std::vector< CRandomVariable::distributiontype > & *dtypes*)

Imports multiple empirical model variables from a file.

6.9.4.51 void GoSUM::CContainer::importEmpiricalVariables (GoSUM::CModelVariables * *pmVs*, const std::string & *_fname*, int & *SSize*, std::vector< CRandomVariable::distributiontype > & *dtypes*)

Imports multiple empirical model variables from a file.

6.9.4.52 void GoSUM::CContainer::importInitialValue (const std::string & *_fname*)

Imports initial values for optimization variables.

6.9.4.53 void GoSUM::CContainer::importInputs (const std::string & *_fname*) `[inline]`

Imports multiple input parameters.

6.9.4.54 void GoSUM::CContainer::importLowerBound (const std::string & *_fname*)

Imports lower bounds for optimization variables.

6.9.4.55 void GoSUM::CContainer::importModelConstraints (const std::string & *_fname*)

Imports model constraint expressions.

6.9.4.56 void GoSUM::CContainer::importNamedDeclaredEmpiricalVariables (GoSUM::CModelVariables * *pmVs*, const std::string & *_fname*, int & *SSize*, std::vector< CRandomVariable::distributiontype > & *dtypes*)

Imports multiple empirical model variables from a file.

6.9.4.57 void GoSUM::CContainer::importNamedEmpiricalVariables (GoSUM::CModelVariables * *pmVs*, const std::string & *_fname*, int & *SSize*, std::vector< CRandomVariable::distributiontype > & *dtypes*)

Imports multiple empirical model variables from a file.

6.9.4.58 void GoSUM::CContainer::importNamedTheoreticalVariables (GoSUM::CModelVariables * *pmVs*, const std::string & *_fname*)

Imports multiple theoretical model variables from a file.

6.9.4.59 `void GoSUM::CContainer::importOptimizationConstraints (const std::string & _fname)`

Imports optimization constraint expressions.

6.9.4.60 `void GoSUM::CContainer::importOutputs (const std::string & _fname)` `[inline]`

Imports multiple output states.

6.9.4.61 `bool GoSUM::CContainer::importPredictionInputSamples (const std::string & _fname)`

Imports prediction input samples.

6.9.4.62 `void GoSUM::CContainer::importTheoreticalVariables (GoSUM::CModelVariables * pMVs, const std::string & _fname)`

Imports multiple theoretical model variables from a file.

6.9.4.63 `void GoSUM::CContainer::importUpperBound (const std::string & _fname)`

Imports upper bounds for optimization variables.

6.9.4.64 `void GoSUM::CContainer::importVariables (GoSUM::CModelVariables * pMVs, const std::string & _fname)`

Imports multiple model variables from a file, input and output.

6.9.4.65 `CModelVariable& GoSUM::CContainer::inputParameter (int _at)` `[inline]`

Returns particular input parameter.

6.9.4.66 `CInputParameters& GoSUM::CContainer::inputParameters ()` `[inline]`

Returns input parameters.

6.9.4.67 `int GoSUM::CContainer::inputsSize ()` `[inline]`

Returns input parameters size.

6.9.4.68 `bool GoSUM::CContainer::isSampleSelected (int _i) const`

Returns true if particular sample value is selected, false otherwise.

6.9.4.69 `void GoSUM::CContainer::join ()`

Joins thread for computation.

6.9.4.70 `int GoSUM::CContainer::learnEvaluationSize ()` `[inline]`

Returns maximal number of learn optimization evaluations.

6.9.4.71 `int GoSUM::CContainer::learningStepsSize () const` `[inline]`

Returns learning steps size.

6.9.4.72 `void GoSUM::CContainer::learnModel ()`

Learns model.

6.9.4.73 `void GoSUM::CContainer::load ()`

Loads project from binary format.

6.9.4.74 `void GoSUM::CContainer::loadTxt ()`

Loads project from txt format.

6.9.4.75 `void GoSUM::CContainer::loadXml ()`

Loads project from xml format.

6.9.4.76 `std::string GoSUM::CContainer::longProjectName ()`

Returns long project name.

6.9.4.77 `CMADS& GoSUM::CContainer::madsOptimizer ()` `[inline]`

Returns MADS.

6.9.4.78 `void GoSUM::CContainer::maximize ()`

Maximizes.

6.9.4.79 `void GoSUM::CContainer::minimize ()`

Minimizes.

6.9.4.80 `CModelConstraints& GoSUM::CContainer::modelConstraints ()` `[inline]`

Returns model constraints.

6.9.4.81 `CEvaluator& GoSUM::CContainer::modelEvaluator ()` `[inline]`

Returns evaluator.

6.9.4.82 `optimizationmethodtype GoSUM::CContainer::optimizationMethod ()` `[inline]`

Returns chosen optimization method type.

6.9.4.83 `GoSUM::CContainer::optimizationmethodtype GoSUM::CContainer::OptimizationMethodType (const std::string & _stype) [static]`

Converts optimization method name (string) to optimizationmethodtype enumerator.

6.9.4.84 `CSimpleOptimizationProblem* GoSUM::CContainer::optimizationProblem () [inline]`

Returns optimization problem.

6.9.4.85 `int GoSUM::CContainer::optimizationStepsSize () const [inline]`

Returns maximal number of optimization evaluations.

6.9.4.86 `void GoSUM::CContainer::optimize ()`

Optimizes.

6.9.4.87 `int GoSUM::CContainer::outputsSize () [inline]`

Returns output states size.

6.9.4.88 `CModelVariable& GoSUM::CContainer::outputState (int _at) [inline]`

Returns particular output state.

6.9.4.89 `COutputStates& GoSUM::CContainer::outputStates () [inline]`

Returns output states.

6.9.4.90 `void GoSUM::CContainer::predict () [inline]`

Predicts (using analytical model).

6.9.4.91 `void GoSUM::CContainer::predictMean (ArrayXd & ymu, ArrayXd & yvar)`

Predicts mean and variance using learned model.

6.9.4.92 `std::string GoSUM::CContainer::projectName () [inline]`

Returns project name.

6.9.4.93 `std::string GoSUM::CContainer::projectPath () [inline]`

Returns project path.

6.9.4.94 `GoSUM::CContainer::projecttype GoSUM::CContainer::ProjectType (const std::string & _stype) [static]`

Converts project type name (string) to projecttype enumerator-.

6.9.4.95 `projecttype GoSUM::CContainer::projectType () [inline]`

Returns chosen project type.

6.9.4.96 `bool GoSUM::CContainer::readyToComputeSensitivities ()`

Returns true if project is ready to compute sensitivities, false otherwise.

6.9.4.97 `bool GoSUM::CContainer::readyToEvaluateOutputs ()`

Returns true if project is ready to evaluate outputs, false otherwise.

`outputs.empty() &&`

6.9.4.98 `bool GoSUM::CContainer::readyToLearnModel ()`

Returns true if project is ready to learn model, false otherwise.

6.9.4.99 `bool GoSUM::CContainer::readyToOptimize ()`

Returns true if project is ready to optimize, false otherwise.

6.9.4.100 `bool GoSUM::CContainer::readyToReduceModel ()`

Returns true if project is ready to reduce model, false otherwise.

6.9.4.101 `bool GoSUM::CContainer::readyToSampleInputs ()`

Returns true if project is ready to sample inputs, false otherwise.

6.9.4.102 `bool GoSUM::CContainer::readyToSetOptimization ()`

Returns true if project is ready to set optimization, false otherwise.

6.9.4.103 `void GoSUM::CContainer::reduce ()`

Reduces model.

6.9.4.104 `CReduction& GoSUM::CContainer::reducer () [inline]`

Returns reduction tool.

6.9.4.105 `void GoSUM::CContainer::resampleInputs ()`

Resamples input parameters.

6.9.4.106 `void GoSUM::CContainer::resetOptimizationVariable (CModelVariables * pmVs, int _at)`

Resets values in the particular optimization variable.

6.9.4.107 void GoSUM::CContainer::save ()

Saves project to binary format.

6.9.4.108 void GoSUM::CContainer::saveTxt ()

Saves project to txt format.

6.9.4.109 void GoSUM::CContainer::saveXml ()

Saves project to xml format.

6.9.4.110 std::vector<int>& GoSUM::CContainer::selectedSamples () [inline]

6.9.4.111 void GoSUM::CContainer::selectSamples (CModelVariable * pmv, double _left, double _right)

Selects sample values of variable pmv that are contained in [_left,_right] interval.

6.9.4.112 CSensitivityAnalysis& GoSUM::CContainer::sensitivityAnalysis () [inline]

Returns sensitivity analysis.

6.9.4.113 void GoSUM::CContainer::separateBySelection (const ArrayXd & X, std::vector< double > & x, std::vector< double > & selx) const

Separates array X into two vectors based on the index (not contained/contained in selected samples).

6.9.4.114 template<class Archive > void GoSUM::CContainer::serialize (Archive & ar, const unsigned int version)
[private]

6.9.4.115 void GoSUM::CContainer::setInitialValue (const ArrayXd & _x0)

Sets initial value of the optimization variables.

6.9.4.116 void GoSUM::CContainer::setLowerBound (const ArrayXd & _xL)

Sets lower bound of the optimization variables.

6.9.4.117 void GoSUM::CContainer::setMadsInitMeshSize (double _ims) [inline]

Sets MADS initial mesh size.

6.9.4.118 void GoSUM::CContainer::setMadsLHSearch (int _lh0, int _lhi) [inline]

Sets MADS lh search parameters.

6.9.4.119 void GoSUM::CContainer::setMadsMaxEvaluation (int _maxeval) [inline]

Sets MADS maximal number of evaluations.

6.9.4.120 void GoSUM::CContainer::setMadsMinPollSize (double *_mps*) [inline]

Sets MADS final poll size.

6.9.4.121 void GoSUM::CContainer::setMatLabPath (const std::string & *_matlabPath*) [inline]

Sets MatLab path.

6.9.4.122 void GoSUM::CContainer::setModelEvaluator (GoSUM::CModelEvaluator::evaluator_type *_me*, const std::string & *_filename*)

Sets model evaluator.

6.9.4.123 void GoSUM::CContainer::setObjective (const std::string & *_fexpr*)

Sets objective of the optimization problem.

6.9.4.124 void GoSUM::CContainer::setOptimizationMethod (optimizationmethod_type *_omType*) [inline]

Sets chosen optimization method type.

6.9.4.125 void GoSUM::CContainer::setProjectName (const std::string & *_prjName*) [inline]

Sets project name.

6.9.4.126 void GoSUM::CContainer::setProjectPath (const std::string & *_prjPath*) [inline]

Sets project path.

6.9.4.127 void GoSUM::CContainer::setProjectType (project_type *_prjType*)

Sets chosen project type.

6.9.4.128 void GoSUM::CContainer::setReductionCutoffSize (int *_cutip*) [inline]

Sets reduction inputs cutoff size.

6.9.4.129 void GoSUM::CContainer::setReductionCutoffValue (double *_cutval*) [inline]

Sets reduction inputs cut value.

6.9.4.130 void GoSUM::CContainer::setReductionOutputs (const std::vector< std::string > & *_seIOS*) [inline]

Sets reduction outputs.

6.9.4.131 void GoSUM::CContainer::setReductionType (GoSUM::CReduction::reduction_type *_rtype*) [inline]

Sets model reduction type.

6.9.4.132 void GoSUM::CContainer::setResampleSize (int *_N*) [inline]

Sets resample size.

6.9.4.133 void GoSUM::CContainer::setResampleType (CHypercube::hctype *_type*) [inline]

Sets resample type.

6.9.4.134 void GoSUM::CContainer::setRNG (CRandomGenerator::rngtype *_type*) [inline]

Sets chosen rng type.

6.9.4.135 void GoSUM::CContainer::setSensitivityOptions (int *_N*, double *_eps1* = 0.005, double *_eps2* = 0.01, double *_eps3* = 0.01)

Sets sensitivity parameters.

6.9.4.136 void GoSUM::CContainer::setThreadSize (int *_trdN*) [inline]

Sets chosen thread size.

6.9.4.137 void GoSUM::CContainer::setUpperBound (const ArrayXd & *_xU*)

Sets upper bound of the optimization variables.

6.9.4.138 void GoSUM::CContainer::setVoronoiOptions (int *_maxiter*, int *_q*, double *_alpha2*, double *_beta2*) [inline]

Sets Voronoi options.

6.9.4.139 void GoSUM::CContainer::startEvaluating ()

Starts evaluating outputs.

6.9.4.140 void GoSUM::CContainer::startLearning ()

Starts learning model.

6.9.4.141 void GoSUM::CContainer::startOptimizing ()

Starts optimizing.

6.9.4.142 void GoSUM::CContainer::startResampling ()

Starts sampling inputs.

6.9.4.143 void GoSUM::CContainer::startSensitivityComputing ()

Starts sensitivity computing.

6.9.5 Friends And Related Function Documentation

6.9.5.1 friend class boost::serialization::access [friend]

Boost serialization.

6.9.6 Member Data Documentation

6.9.6.1 CAnalyticalModel GoSUM::CContainer::analytical [private]

Analytic model of the project.

6.9.6.2 boost::signal<void()> GoSUM::CContainer::evaluatingFinished

Signal for evaluating outputs finished.

6.9.6.3 CEvaluator GoSUM::CContainer::evaluator [private]

Basic evaluator of the project, used only for model based projects.

6.9.6.4 CGAModelOptimization GoSUM::CContainer::GAMO [private]

Genetic Algorithms optimization method.

6.9.6.5 CHypercube GoSUM::CContainer::hycube [private]

Basic hypercube, used for resampling.

6.9.6.6 CModelConstraints GoSUM::CContainer::inconsts [private]

Model constraints, i.e. constraints on input parameters.

6.9.6.7 CInputParameters GoSUM::CContainer::inputs [private]

Input parameters of the project.

6.9.6.8 boost::signal<void()> GoSUM::CContainer::learningFinished

Signal for learning model finished.

6.9.6.9 CMADS GoSUM::CContainer::MADS [private]

Mesh Adaptive Direct Search optimization method.

6.9.6.10 optimizationmethodtype GoSUM::CContainer::omType [private]

Optimization method type.

6.9.6.11 boost::signal<void()> GoSUM::CContainer::optimizingFinished

Signal for optimizing finished.

6.9.6.12 COutputStates GoSUM::CContainer::outputs [private]

Output states of the project.

6.9.6.13 CSimpleOptimizationProblem* GoSUM::CContainer::pOP [private]

Defined optimization problem.

6.9.6.14 std::string GoSUM::CContainer::prjName [private]

Project path and name.

6.9.6.15 std::string GoSUM::CContainer::prjPath [private]**6.9.6.16 projecttype GoSUM::CContainer::prjType [private]**

Project type.

6.9.6.17 CReduction GoSUM::CContainer::reduced [private]

Reduction operator.

6.9.6.18 boost::signal<void()> GoSUM::CContainer::resamplingFinished

Signal for sampling inputs finished.

6.9.6.19 std::vector<int> GoSUM::CContainer::selectedsamples [private]

Vector of selected sample values indices.

6.9.6.20 CSensitivityAnalysis GoSUM::CContainer::sensitivity [private]

Sensistivity analysis of the analytical model.

6.9.6.21 boost::signal<void()> GoSUM::CContainer::sensitivityComputingFinished

Signal for sensitivity computing finished.

6.9.6.22 boost::thread GoSUM::CContainer::thrd

Thread for computations.

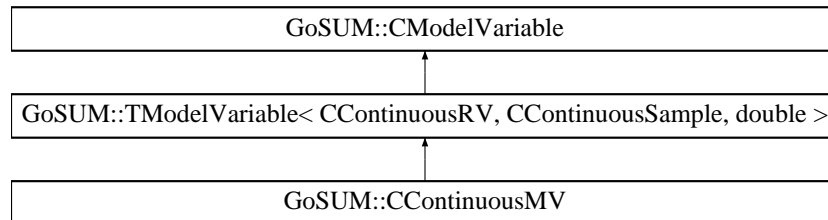
The documentation for this class was generated from the following files:

- C:/Development/core/[Container.h](#)
- C:/Development/core/[Container.cpp](#)

6.10 GoSUM::CContinuousMV Class Reference

```
#include <ModelVariable.h>
```

Inheritance diagram for GoSUM::CContinuousMV:



Public Member Functions

- [CContinuousMV](#) (const std::string &_uname, [CRandomVariable::distributiontype](#) _type)
Constructs new model variable by name and type.
- [CContinuousMV](#) (const [CContinuousMV](#) &O)
- virtual void [setTheoreticalDistribution](#) ()
Tests goodness-of-fit for theoretical random variables and replaces with the best that satisfies Kolomogorov-Smirnov.
- virtual void [setEmpiricalDistribution](#) ()
Compute distribution of the actual random variable from empirical sample data.
- virtual bool [isContinuous](#) () const
Returns true if model variable is continuous, false otherwise.
- virtual bool [isDiscrete](#) () const
Returns true if model variable is discrete, false otherwise.
- bool [hasDiscreteData](#) ()

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- [CContinuousMV](#) ()
Private constructor.

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.10.1 Constructor & Destructor Documentation

6.10.1.1 GoSUM::CContinuousMV::CContinuousMV () [inline], [private]

Private constructor.

6.10.1.2 `GoSUM::CContinuousMV::CContinuousMV (const std::string & _uname, CRandomVariable::distributiontype _type)`

Constructs new model variable by name and type.

6.10.1.3 `GoSUM::CContinuousMV::CContinuousMV (const CContinuousMV & O)`

6.10.2 Member Function Documentation

6.10.2.1 `bool GoSUM::CContinuousMV::hasDiscreteData () [inline]`

6.10.2.2 `virtual bool GoSUM::CContinuousMV::isContinuous () const [inline],[virtual]`

Returns true if model variable is continuous, false otherwise.

Implements [GoSUM::CModelVariable](#).

6.10.2.3 `virtual bool GoSUM::CContinuousMV::isDiscrete () const [inline],[virtual]`

Returns true if model variable is discrete, false otherwise.

Implements [GoSUM::CModelVariable](#).

6.10.2.4 `template<class Archive > void GoSUM::CContinuousMV::serialize (Archive & ar, const unsigned int version) [inline],[private]`

6.10.2.5 `virtual void GoSUM::CContinuousMV::setEmpiricalDistribution () [inline],[virtual]`

Compute distribution of the actual random variable from empirical sample data.

Implements [GoSUM::TModelVariable< R, S, t >](#).

6.10.2.6 `void GoSUM::CContinuousMV::setTheoreticalDistribution () [virtual]`

Tests goodness-of-fit for theoretical random variables and replaces with the best that satisfies Kolomogorov--Smirnov.

Implements [GoSUM::TModelVariable< R, S, t >](#).

6.10.3 Friends And Related Function Documentation

6.10.3.1 `friend class boost::serialization::access [friend]`

Boost serialization.

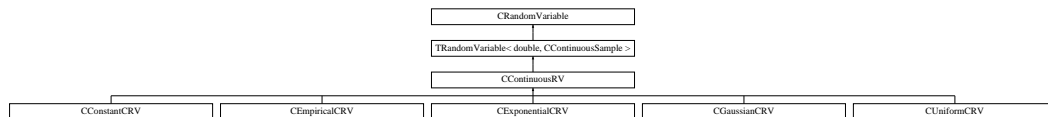
The documentation for this class was generated from the following files:

- C:/Development/core/[ModelVariable.h](#)
- C:/Development/core/[ModelVariable.cpp](#)

6.11 CContinuousRV Class Reference

```
#include <RandomVariable.h>
```

Inheritance diagram for CContinuousRV:



Public Member Functions

- [CContinuousRV](#) ()
- [CContinuousRV](#) (const [CContinuousRV](#) &O)
- double [probability](#) (double _x1, double _x2) const
Returns probablilty that random variable has value between _x1 and _x2.
- virtual double [quantile](#) (double _p) const
Quantile, only argument checking.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.11.1 Constructor & Destructor Documentation

6.11.1.1 [CContinuousRV::CContinuousRV \(\)](#) [[inline](#)]

6.11.1.2 [CContinuousRV::CContinuousRV \(const CContinuousRV & O \)](#) [[inline](#)]

6.11.2 Member Function Documentation

6.11.2.1 [double CContinuousRV::probability \(double _x1, double _x2 \) const](#) [[inline](#)]

Returns probablilty that random variable has value between _x1 and _x2.

6.11.2.2 [virtual double CContinuousRV::quantile \(double _p \) const](#) [[inline](#)], [[virtual](#)]

Quantile, only argument checking.

Implements [CRandomVariable](#).

6.11.2.3 [template<class Archive > void CContinuousRV::serialize \(Archive &ar, const unsigned int version \)](#) [[inline](#)], [[private](#)]

6.11.3 Friends And Related Function Documentation

6.11.3.1 [friend class boost::serialization::access](#) [[friend](#)]

Boost serialization.

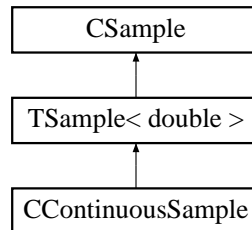
The documentation for this class was generated from the following file:

- C:/Development/core/[RandomVariable.h](#)

6.12 CContinuousSample Class Reference

```
#include <Sample.h>
```

Inheritance diagram for CContinuousSample:



Public Member Functions

- [CContinuousSample](#) ()
- virtual [~CContinuousSample](#) ()
- [CContinuousSample](#) (const [CContinuousSample](#) &O)
- virtual void [computeStatistics](#) (int _n)
Computes sample statistics, i.e. normalized histogram etc.
- virtual double [mean](#) () const
Returns empirical mean.
- virtual double [standardDeviation](#) () const
Returns empirical standard deviation.
- virtual double [exportHistogram](#) (ArrayXd &_x, ArrayXd &_H) const
Exports histogram (x,H).
- virtual void [exportSubHistogram](#) (ArrayXd &_subH, const std::vector< int > &subset) const
Exports histogram relative to a subset of sample data.
- virtual double [variance](#) () const
Returns sample variance (i.e. empirical).
- bool [hasDiscreteData](#) ()

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Protected Attributes

- double [mu](#)
- double [stdv](#)

Private Member Functions

- virtual void [computeNormalizedHistogram](#) (int *_n*)
Computes normalized histogram from sample data.
- virtual void [computeNormalizedSubHistogram](#) (ArrayXd &*_subH*, const std::vector< int > &*subset*) const
Computes normalized histogram from a subset of sample data.
- virtual void [computeMeanAndStandardDeviation](#) ()
Computes empirical mean and standard deviation.

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.12.1 Constructor & Destructor Documentation

- 6.12.1.1 `CContinuousSample::CContinuousSample ()` `[inline]`
- 6.12.1.2 `virtual CContinuousSample::~~CContinuousSample ()` `[inline],[virtual]`
- 6.12.1.3 `CContinuousSample::CContinuousSample (const CContinuousSample & O)` `[inline]`

6.12.2 Member Function Documentation

- 6.12.2.1 `void CContinuousSample::computeMeanAndStandardDeviation ()` `[private],[virtual]`

Computes empirical mean and standard deviation.

- 6.12.2.2 `void CContinuousSample::computeNormalizedHistogram (int _n)` `[private],[virtual]`

Computes normalized histogram from sample data.

- 6.12.2.3 `void CContinuousSample::computeNormalizedSubHistogram (ArrayXd & _subH, const std::vector< int > & subset) const` `[private],[virtual]`

Computes normalized histogram from a subset of sample data.

- 6.12.2.4 `virtual void CContinuousSample::computeStatistics (int _n)` `[inline],[virtual]`

Computes sample statistics, i.e. normalized histogram etc.

Implements [CSample](#).

- 6.12.2.5 `double CContinuousSample::exportHistogram (ArrayXd & _x, ArrayXd & _H) const` `[virtual]`

Exports histogram (*x*,*H*).

6.12.2.6 `void CContinuousSample::exportSubHistogram (ArrayXd & _subH, const std::vector< int > & subset) const`
`[virtual]`

Exports histogram relative to a subset of sample data.

Implements [TSample< t >](#).

6.12.2.7 `bool CContinuousSample::hasDiscreteData ()`

6.12.2.8 `virtual double CContinuousSample::mean () const` `[inline],[virtual]`

Returns empirical mean.

6.12.2.9 `template<class Archive > void CContinuousSample::serialize (Archive & ar, const unsigned int version)`
`[protected]`

Reimplemented from [TSample< t >](#).

6.12.2.10 `virtual double CContinuousSample::standardDeviation () const` `[inline],[virtual]`

Returns empirical standard deviation.

6.12.2.11 `virtual double CContinuousSample::variance () const` `[inline],[virtual]`

Returns sample variance (i.e.empirical).

Implements [CSample](#).

6.12.3 Friends And Related Function Documentation

6.12.3.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.12.4 Member Data Documentation

6.12.4.1 `double CContinuousSample::mu` `[protected]`

6.12.4.2 `double CContinuousSample::stdv` `[protected]`

Empirical mean (mu) and standard deviation (std).

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

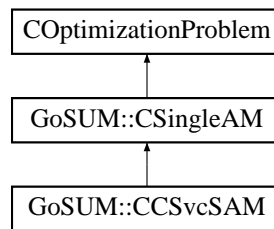
- [C:/Development/core/Sample.h](#)
- [C:/Development/core/Sample.cpp](#)

6.13 GoSUM::CCSvcSAM Class Reference

Class for the analytical model for single output state, SVC type.

`#include <AnalyticalModel.h>`

Inheritance diagram for GoSUM::CCSvcSAM:



Public Member Functions

- [CCSvcSAM](#) ()
- virtual [~CCSvcSAM](#) ()

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.13.1 Detailed Description

Class for the analytical model for single output state, SVC type.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 [GoSUM::CCSvcSAM::CCSvcSAM \(\)](#) `[inline]`

6.13.2.2 [virtual GoSUM::CCSvcSAM::~~CCSvcSAM \(\)](#) `[inline],[virtual]`

6.13.3 Member Function Documentation

6.13.3.1 [template<class Archive > void GoSUM::CCSvcSAM::serialize \(Archive & ar, const unsigned int version \)](#)
`[protected]`

Reimplemented from [GoSUM::CSingleAM](#).

6.13.4 Friends And Related Function Documentation

6.13.4.1 [friend class boost::serialization::access](#) `[friend]`

Boost serialization.

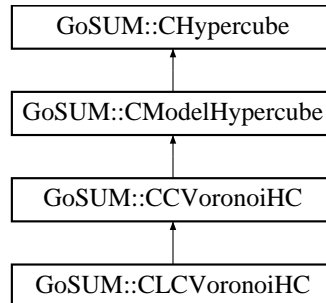
The documentation for this class was generated from the following files:

- C:/Development/core/[AnalyticalModel.h](#)
- C:/Development/core/[AnalyticalModel.cpp](#)

6.14 GoSUM::CCVoronoiHC Class Reference

```
#include <Hypercube.h>
```

Inheritance diagram for GoSUM::CCVoronoiHC:



Public Member Functions

- [CCVoronoiHC](#) ([CInputParameters](#) *[_pIP](#))
- virtual [~CCVoronoiHC](#) ()

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- virtual void [doGenerate](#) (int [_rssize](#), int [_dim](#), std::vector< [ArrayXd](#) > &[_samples](#))
Core of the generation.
- void [centralize](#) (std::vector< [ArrayXd](#) > &[_samples](#))
Centralizes model points [_samples](#).
- [CCVoronoiHC](#) ()

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.14.1 Constructor & Destructor Documentation

6.14.1.1 [GoSUM::CCVoronoiHC::CCVoronoiHC](#) () [\[inline\]](#), [\[protected\]](#)

6.14.1.2 [GoSUM::CCVoronoiHC::CCVoronoiHC](#) ([CInputParameters](#) *[_pIP](#)) [\[inline\]](#)

6.14.1.3 virtual [GoSUM::CCVoronoiHC::~~CCVoronoiHC](#) () [\[inline\]](#), [\[virtual\]](#)

6.14.2 Member Function Documentation

6.14.2.1 void [GoSUM::CCVoronoiHC::centralize](#) (std::vector< [ArrayXd](#) > & [_samples](#)) [\[protected\]](#)

Centralizes model points [_samples](#).

6.14.2.2 `void GoSUM::CCVoronoiHC::doGenerate (int _rssize, int _dim, std::vector< ArrayXd > & _samples)`
`[protected], [virtual]`

Core of the generation.

Implements [GoSUM::CModelHypercube](#).

Reimplemented in [GoSUM::CLCVoronoiHC](#).

6.14.2.3 `template<class Archive > void GoSUM::CCVoronoiHC::serialize (Archive & ar, const unsigned int version)`
`[protected]`

Reimplemented from [GoSUM::CModelHypercube](#).

Reimplemented in [GoSUM::CLCVoronoiHC](#).

6.14.3 Friends And Related Function Documentation

6.14.3.1 `friend class boost::serialization::access [friend]`

Boost serialization.

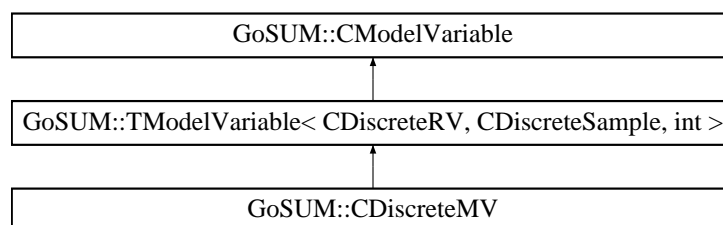
The documentation for this class was generated from the following files:

- C:/Development/core/[Hypercube.h](#)
- C:/Development/core/[Hypercube.cpp](#)

6.15 GoSUM::CDiscreteMV Class Reference

```
#include <ModelVariable.h>
```

Inheritance diagram for GoSUM::CDiscreteMV:



Public Member Functions

- [CDiscreteMV](#) (const std::string &_uname, [CRandomVariable::distributiontype](#) _type)
Constructs new model variable by name and type.
- [CDiscreteMV](#) (const [CDiscreteMV](#) &O)
- [CDiscreteMV](#) (const [CContinuousMV](#) &O)
- virtual void [setTheoreticalDistribution](#) ()
Detects constant sample and turns random variable to appropriate type.
- virtual void [setEmpiricalDistribution](#) ()
Compute distribution of the actual random variable from empirical sample data.
- virtual bool [isContinuous](#) () const
Returns true if model variable is continuous, false otherwise.
- virtual bool [isDiscrete](#) () const
Returns true if model variable is discrete, false otherwise.

Private Member Functions

- `template<class Archive >`
`void serialize (Archive &ar, const unsigned int version)`
- `CDiscreteMV ()`

Friends

- class `boost::serialization::access`
Boost serialization.

Additional Inherited Members

6.15.1 Constructor & Destructor Documentation

6.15.1.1 `GoSUM::CDiscreteMV::CDiscreteMV ()` `[inline], [private]`

6.15.1.2 `GoSUM::CDiscreteMV::CDiscreteMV (const std::string & _uname, CRandomVariable::distributiontype _type)`

Constructs new model variable by name and type.

6.15.1.3 `GoSUM::CDiscreteMV::CDiscreteMV (const CDiscreteMV & O)`

6.15.1.4 `GoSUM::CDiscreteMV::CDiscreteMV (const CContinuousMV & O)`

6.15.2 Member Function Documentation

6.15.2.1 `virtual bool GoSUM::CDiscreteMV::isContinuous () const` `[inline], [virtual]`

Returns true if model variable is continuous, false otherwise.

Implements [GoSUM::CModelVariable](#).

6.15.2.2 `virtual bool GoSUM::CDiscreteMV::isDiscrete () const` `[inline], [virtual]`

Returns true if model variable is discrete, false otherwise.

Implements [GoSUM::CModelVariable](#).

6.15.2.3 `template<class Archive > void GoSUM::CDiscreteMV::serialize (Archive & ar, const unsigned int version)`
`[inline], [private]`

6.15.2.4 `virtual void GoSUM::CDiscreteMV::setEmpiricalDistribution ()` `[inline], [virtual]`

Compute distribution of the actual random variable from empirical sample data.

Implements [GoSUM::TModelVariable< R, S, t >](#).

6.15.2.5 `void GoSUM::CDiscreteMV::setTheoreticalDistribution ()` `[virtual]`

Detects constant sample and turns random variable to appropriate type.

Implements [GoSUM::TModelVariable< R, S, t >](#).

6.15.3 Friends And Related Function Documentation

6.15.3.1 friend class `boost::serialization::access` [friend]

Boost serialization.

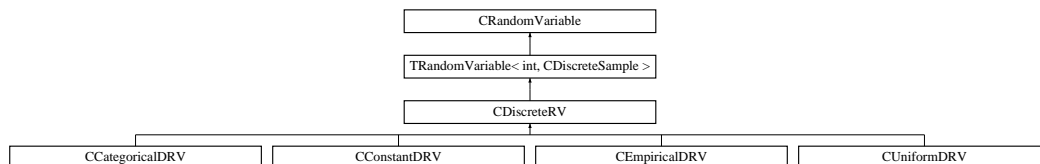
The documentation for this class was generated from the following files:

- [C:/Development/core/ModelVariable.h](#)
- [C:/Development/core/ModelVariable.cpp](#)

6.16 CDiscreteRV Class Reference

```
#include <RandomVariable.h>
```

Inheritance diagram for CDiscreteRV:



Public Member Functions

- [CDiscreteRV](#) ()
- [CDiscreteRV](#) (const [CDiscreteRV](#) &O)
- virtual double [quantile](#) (double _p) const
Quantile, only argument checking.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.16.1 Constructor & Destructor Documentation

6.16.1.1 CDiscreteRV::CDiscreteRV () [inline]

6.16.1.2 CDiscreteRV::CDiscreteRV (const CDiscreteRV & O) [inline]

6.16.2 Member Function Documentation

6.16.2.1 `virtual double CDiscreteRV::quantile (double p) const` `[inline],[virtual]`

Quantile, only argument checking.

Implements [CRandomVariable](#).

6.16.2.2 `template<class Archive > void CDiscreteRV::serialize (Archive & ar, const unsigned int version)` `[inline],[private]`

6.16.3 Friends And Related Function Documentation

6.16.3.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

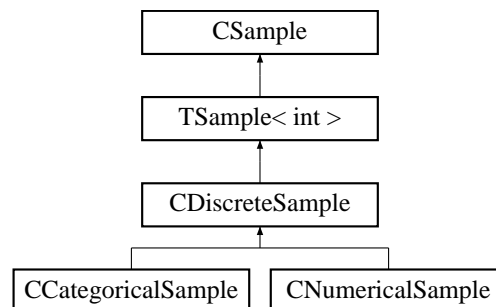
The documentation for this class was generated from the following file:

- C:/Development/core/[RandomVariable.h](#)

6.17 CDiscreteSample Class Reference

```
#include <Sample.h>
```

Inheritance diagram for CDiscreteSample:



Public Member Functions

- [CDiscreteSample](#) ()
- `virtual ~CDiscreteSample ()`
- [CDiscreteSample](#) (const [CDiscreteSample](#) &O)
- `virtual void computeStatistics (int _n)`
Computes sample statistics, i.e. normalized histogram etc.
- `virtual int exportHistogram (ArrayXi &_x, ArrayXd &_H) const`
*Exports histogram (*x*,*H*).*
- `virtual void exportSubHistogram (ArrayXd &_subH, const std::vector< int > &subset) const`
Exports histogram relative to a subset of sample data.
- `virtual double variance () const`
Returns sample variance (i.e. empirical).

Protected Member Functions

- `template<class Archive >`
`void serialize (Archive &ar, const unsigned int version)`

Private Member Functions

- virtual void [computeNormalizedHistogram](#) (int _n)
C.omputes normalized histogram from sample data.
- virtual void [computeNormalizedSubHistogram](#) (ArrayXd &_subH, const std::vector< int > &subset) const
Computes normalized histogram from a subset of sample data.

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.17.1 Constructor & Destructor Documentation

- 6.17.1.1 `CDiscreteSample::CDiscreteSample () [inline]`
- 6.17.1.2 `virtual CDiscreteSample::~CDiscreteSample () [inline],[virtual]`
- 6.17.1.3 `CDiscreteSample::CDiscreteSample (const CDiscreteSample & O) [inline]`

6.17.2 Member Function Documentation

- 6.17.2.1 `void CDiscreteSample::computeNormalizedHistogram (int _n) [private],[virtual]`

C.omputes normalized histogram from sample data.

- 6.17.2.2 `void CDiscreteSample::computeNormalizedSubHistogram (ArrayXd & _subH, const std::vector< int > & subset) const [private],[virtual]`

Computes normalized histogram from a subset of sample data.

- 6.17.2.3 `virtual void CDiscreteSample::computeStatistics (int _n) [inline],[virtual]`

Computes sample statistics, i.e. normalized histogram etc.

Implements [CSample](#).

- 6.17.2.4 `int CDiscreteSample::exportHistogram (ArrayXi & _x, ArrayXd & _H) const [virtual]`

Exports histogram (x,H).

- 6.17.2.5 `void CDiscreteSample::exportSubHistogram (ArrayXd & _subH, const std::vector< int > & subset) const [virtual]`

Exports histogram relative to a subset of sample data.

Implements [TSample< t >](#).

6.17.2.6 `template<class Archive > void CDiscreteSample::serialize (Archive & ar, const unsigned int version)`
`[protected]`

Reimplemented from [TSample< t >](#).

6.17.2.7 `virtual double CDiscreteSample::variance () const` `[inline],[virtual]`

Returns sample variance (i.e. empirical).

Implements [CSample](#).

Reimplemented in [CCategoricalSample](#), and [CNumericalSample](#).

6.17.3 Friends And Related Function Documentation

6.17.3.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

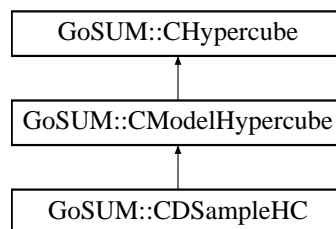
The documentation for this class was generated from the following files:

- [C:/Development/core/Sample.h](#)
- [C:/Development/core/Sample.cpp](#)

6.18 GoSUM::CDSampleHC Class Reference

`#include <Hypercube.h>`

Inheritance diagram for GoSUM::CDSampleHC:



Public Member Functions

- [CDSampleHC \(CInputParameters * _pIP\)](#)
- `virtual ~CDSampleHC ()`

Protected Member Functions

- `virtual void doGenerate (int _rssize, int _dim, std::vector< ArrayXd > & _samples)`
Core of the generation.
- [CDSampleHC \(\)](#)

Private Member Functions

- `template<class Archive >`
`void serialize (Archive &ar, const unsigned int version)`

Friends

- class [boost::serialization::access](#)

Boost serialization.

Additional Inherited Members

6.18.1 Constructor & Destructor Documentation

6.18.1.1 `GoSUM::CDSampleHC::CDSampleHC ()` `[inline]`, `[protected]`

6.18.1.2 `GoSUM::CDSampleHC::CDSampleHC (CInputParameters * _pIP)` `[inline]`

6.18.1.3 `virtual GoSUM::CDSampleHC::~~CDSampleHC ()` `[inline]`, `[virtual]`

6.18.2 Member Function Documentation

6.18.2.1 `void GoSUM::CDSampleHC::doGenerate (int _rssize, int _dim, std::vector< ArrayXd > & _samples)`
`[protected]`, `[virtual]`

Core of the generation.

Implements [GoSUM::CModelHypercube](#).

6.18.2.2 `template<class Archive > void GoSUM::CDSampleHC::serialize (Archive & ar, const unsigned int version)`
`[private]`

6.18.3 Friends And Related Function Documentation

6.18.3.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

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

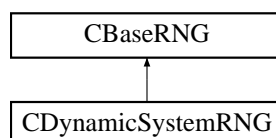
- C:/Development/core/[Hypercube.h](#)
- C:/Development/core/[Hypercube.cpp](#)

6.19 CDynamicSystemRNG Class Reference

Class for dynamic system uniform RNG.

```
#include <RandomGenerators.h>
```

Inheritance diagram for CDynamicSystemRNG:



Public Member Functions

- [CDynamicSystemRNG](#) ()
- [CDynamicSystemRNG](#) (unsigned int s)
- virtual [~CDynamicSystemRNG](#) ()
- virtual void [setSeed](#) (unsigned int s)
Sets seed of the RNG.
- virtual unsigned int [rndi](#) ()
Returns randomly generated unsigned int.
- virtual double [rnd](#) ()
Returns randomly generated double between 0 and 1.

Private Types

- typedef long long int [Long](#)

Private Member Functions

- Array< long double, Dynamic, 1 > [prepare](#) (unsigned int M)
Private function.

Private Attributes

- unsigned int [k](#)
Paramters of the DS RNG.
- unsigned int [i](#)
- unsigned int [j](#)
Paramters of the DS RNG.
- [Long](#) [n](#)
Paramters of the DS RNG.
- Array< long double, Dynamic, 1 > [x](#)
Paramters of the DS RNG.
- Array< long double, Dynamic, 1 > [w](#)
Paramters of the DS RNG.

6.19.1 Detailed Description

Class for dynamic system uniform RNG.

6.19.2 Member Typedef Documentation

6.19.2.1 typedef long long int [CDynamicSystemRNG::Long](#) `[private]`

6.19.3 Constructor & Destructor Documentation

6.19.3.1 [CDynamicSystemRNG::CDynamicSystemRNG](#) () `[inline]`

6.19.3.2 [CDynamicSystemRNG::CDynamicSystemRNG](#) (unsigned int s) `[inline]`

6.19.3.3 virtual [CDynamicSystemRNG::~~CDynamicSystemRNG](#) () `[inline],[virtual]`

6.19.4 Member Function Documentation

6.19.4.1 `Array< long double, Dynamic, 1 > CDynamicSystemRNG::prepare (unsigned int M)` `[private]`

Private function.

6.19.4.2 `virtual double CDynamicSystemRNG::rnd ()` `[inline],[virtual]`

Returns randomly generated double between 0 and 1.

Implements [CBaseRNG](#).

6.19.4.3 `virtual unsigned int CDynamicSystemRNG::rndi ()` `[inline],[virtual]`

Returns randomly generated unsigned int.

Implements [CBaseRNG](#).

6.19.4.4 `virtual void CDynamicSystemRNG::setSeed (unsigned int s)` `[inline],[virtual]`

Sets seed of the RNG.

Parameters

<i>s</i>	Sets seed of the RNG.
----------	-----------------------

Implements [CBaseRNG](#).

6.19.5 Member Data Documentation

6.19.5.1 `unsigned int CDynamicSystemRNG::i` `[private]`

6.19.5.2 `unsigned int CDynamicSystemRNG::j` `[private]`

Parameters of the DS RNG.

6.19.5.3 `unsigned int CDynamicSystemRNG::k` `[private]`

Parameters of the DS RNG.

6.19.5.4 `Long CDynamicSystemRNG::n` `[private]`

Parameters of the DS RNG.

6.19.5.5 `Array<long double,Dynamic,1> CDynamicSystemRNG::w` `[private]`

Parameters of the DS RNG.

6.19.5.6 `Array<long double,Dynamic,1> CDynamicSystemRNG::x` `[private]`

Parameters of the DS RNG.

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

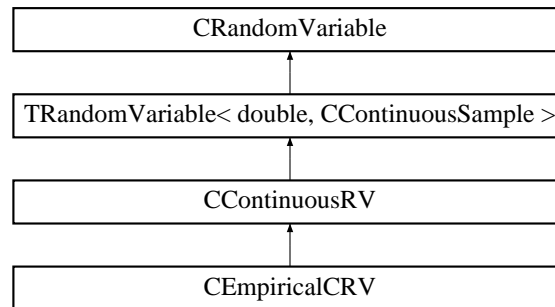
- C:/Development/core/[RandomGenerators.h](#)
- C:/Development/core/[RandomGenerators.cpp](#)

6.20 CEmpiricalCRV Class Reference

Class for empirical continuous random variables derived from continuous random variables.

```
#include <RandomVariable.h>
```

Inheritance diagram for CEmpiricalCRV:



Public Member Functions

- [CEmpiricalCRV](#) ()
- virtual [~CEmpiricalCRV](#) ()
- [CEmpiricalCRV](#) (const [CEmpiricalCRV](#) &O)
- virtual void [setDistribution](#) (const [CContinuousSample](#) &_aS)
Computes PMF and CDF from sample histogram, application of kernel density estimation algorithm.
- virtual double [probability](#) (double _x) const
Function that returns PDF.
- virtual double [cumulative](#) (double _x) const
Function that returns CDF.
- virtual double [minValue](#) () const
Returns minimal value of the random variable.
- virtual double [maxValue](#) () const
Returns maximal value of the random variable.
- virtual double [expectedValue](#) () const
Returns expected value of the random variable.
- virtual [distributiontype](#) [distributionType](#) () const
Returns enum type of the random variable distribution.
- virtual std::string [distributionName](#) () const
Returns name of the random variable distribution.
- virtual ArrayXd [exportDomain](#) () const
Exports domain of the random variable.
- virtual bool [isDistributionDefined](#) () const
Returns true if distribution is defined, false otherwise.
- double [mean](#) () const
Returns mean of empirical random variable.
- double [standardDeviation](#) () const
Returns standard deviation of empirical random variable.
- virtual double [variance](#) () const
Returns variance of the random variable.

Protected Member Functions

- virtual double [doQuantile](#) (double _p) const
Quantile, formula implementation.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Private Attributes

- ArrayXd [x](#)
- ArrayXd [p](#)
- ArrayXd [cdf](#)
PDF (x,p) and CDF (x,cdf) data.
- double [pbandwidth](#)
- double [qbandwidth](#)
Bandwidths of PDF and CDF, as computed in kernel density estimator.
- double [mu](#)
- double [stdv](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.20.1 Detailed Description

Class for empirical continuous random variables derived from continuous random variables.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 `CEmpiricalCRV::CEmpiricalCRV ()` `[inline]`

6.20.2.2 `virtual CEmpiricalCRV::~~CEmpiricalCRV ()` `[inline],[virtual]`

6.20.2.3 `CEmpiricalCRV::CEmpiricalCRV (const CEmpiricalCRV & O)` `[inline]`

6.20.3 Member Function Documentation

6.20.3.1 `virtual double CEmpiricalCRV::cumulative (double x) const` `[inline],[virtual]`

Function that returns CDF.

6.20.3.2 `virtual std::string CEmpiricalCRV::distributionName () const` `[inline],[virtual]`

Returns name of the random variable distribution.

Implements [CRandomVariable](#).

6.20.3.3 `virtual distributiontype CEmpiricalCRV::distributionType () const` `[inline],[virtual]`

Returns enum type of the random variable distribution.

Implements [CRandomVariable](#).

6.20.3.4 `double CEmpiricalCRV::doQuantile (double _p) const` `[protected],[virtual]`

Quantile, formula implementation.

Implements [CRandomVariable](#).

6.20.3.5 `virtual double CEmpiricalCRV::expectedValue () const` `[inline],[virtual]`

Returns expected value of the random variable.

Implements [CRandomVariable](#).

6.20.3.6 `ArrayXd CEmpiricalCRV::exportDomain () const` `[virtual]`

Exports domain of the random variable.

Implements [TRandomVariable< t, T >](#).

6.20.3.7 `virtual bool CEmpiricalCRV::isDistributionDefined () const` `[inline],[virtual]`

Returns true if distribution is defined, false otherwise.

Reimplemented from [CRandomVariable](#).

6.20.3.8 `virtual double CEmpiricalCRV::maxValue () const` `[inline],[virtual]`

Returns maximal value of the random variable.

Implements [CRandomVariable](#).

6.20.3.9 `double CEmpiricalCRV::mean () const` `[inline]`

Returns mean of empirical random variable.

6.20.3.10 `virtual double CEmpiricalCRV::minValue () const` `[inline],[virtual]`

Returns minimal value of the random variable.

Implements [CRandomVariable](#).

6.20.3.11 `virtual double CEmpiricalCRV::probability (double _x) const` `[inline],[virtual]`

Function that returns PDF.

6.20.3.12 `template<class Archive > void CEmpiricalCRV::serialize (Archive & ar, const unsigned int version)`
`[private]`

6.20.3.13 `void CEmpiricalCRV::setDistribution (const CContinuousSample & _aS) [virtual]`

Computes PMF and CDF from sample histogram, application of kernel density estimation algorithm.

6.20.3.14 `double CEmpiricalCRV::standardDeviation () const [inline]`

Returns standard deviation of empirical random variable.

6.20.3.15 `virtual double CEmpiricalCRV::variance () const [inline],[virtual]`

Returns variance of the random variable.

Implements [CRandomVariable](#).

6.20.4 Friends And Related Function Documentation

6.20.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.20.5 Member Data Documentation

6.20.5.1 `ArrayXd CEmpiricalCRV::cdf [private]`

PDF (x,p) and CDF (x,cdf) data.

6.20.5.2 `double CEmpiricalCRV::mu [private]`

6.20.5.3 `ArrayXd CEmpiricalCRV::p [private]`

6.20.5.4 `double CEmpiricalCRV::pbandwidth [private]`

6.20.5.5 `double CEmpiricalCRV::qbandwidth [private]`

Bandwidths of PDF and CDF, as computed in kernel density estimator.

6.20.5.6 `double CEmpiricalCRV::stdv [private]`

6.20.5.7 `ArrayXd CEmpiricalCRV::x [private]`

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

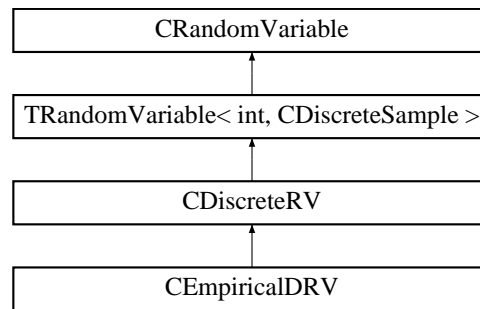
- C:/Development/core/[RandomVariable.h](#)
- C:/Development/core/[RandomVariable.cpp](#)

6.21 CEmpiricalDRV Class Reference

Class for categorical discrete random variables derived from discrete random variables.

`#include <RandomVariable.h>`

Inheritance diagram for CEmpiricalDRV:



Public Member Functions

- [CEmpiricalDRV](#) ()
- virtual [~CEmpiricalDRV](#) ()
- [CEmpiricalDRV](#) (const [CEmpiricalDRV](#) &O)
- virtual void [setDistribution](#) (const [CDiscreteSample](#) &_aS)
Computes PMF and CDF from sample histogram.
- virtual double [probability](#) (int _k) const
Function that returns PMF.
- virtual double [cumulative](#) (int _k) const
Function that returns CDF.
- virtual int [expandedSize](#) () const
- virtual double [minValue](#) () const
After expansion, number of new variables is equal to the number of categories.
- virtual double [maxValue](#) () const
Returns maximal value of the random variable.
- virtual double [expectedValue](#) () const
Returns expected value of the random variable.
- virtual [distributiontype](#) [distributionType](#) () const
Returns enum type of the random variable distribution.
- virtual std::string [distributionName](#) () const
Returns name of the random variable distribution.
- virtual ArrayXi [exportDomain](#) () const
Exports domain of the random variable.
- virtual bool [isDistributionDefined](#) () const
Returns true if distribution is defined, false otherwise.
- virtual double [variance](#) () const
Returns variance of the random variable.

Protected Member Functions

- virtual double [doQuantile](#) (double _p) const
Quantile, formula implementation.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Private Attributes

- ArrayXd [p](#)
- ArrayXd [cdf](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.21.1 Detailed Description

Class for categorical discrete random variables derived from discrete random variables.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 `CEmpiricalDRV::CEmpiricalDRV ()` `[inline]`

6.21.2.2 `virtual CEmpricalDRV::~~CEmpiricalDRV ()` `[inline],[virtual]`

6.21.2.3 `CEmpiricalDRV::CEmpiricalDRV (const CEmpricalDRV & O)` `[inline]`

6.21.3 Member Function Documentation

6.21.3.1 `virtual double CEmpricalDRV::cumulative (int _k) const` `[inline],[virtual]`

Function that returns CDF.

6.21.3.2 `virtual std::string CEmpricalDRV::distributionName () const` `[inline],[virtual]`

Returns name of the random variable distribution.

Implements [CRandomVariable](#).

6.21.3.3 `virtual distributiontype CEmpricalDRV::distributionType () const` `[inline],[virtual]`

Returns enum type of the random variable distribution.

Implements [CRandomVariable](#).

6.21.3.4 `double CEmpricalDRV::doQuantile (double _p) const` `[protected],[virtual]`

Quantile, formula implementation.

Implements [CRandomVariable](#).

6.21.3.5 `virtual int CEmpricalDRV::expandedSize () const` `[inline],[virtual]`

Reimplemented from [CRandomVariable](#).

6.21.3.6 virtual double CEmpiricalDRV::expectedValue () const [inline],[virtual]

Returns expected value of the random variable.

Implements [CRandomVariable](#).

6.21.3.7 ArrayXi CEmpiricalDRV::exportDomain () const [virtual]

Exports domain of the random variable.

Implements [TRandomVariable< t, T >](#).

6.21.3.8 virtual bool CEmpiricalDRV::isDistributionDefined () const [inline],[virtual]

Returns true if distribution is defined, false otherwise.

Reimplemented from [CRandomVariable](#).

6.21.3.9 virtual double CEmpiricalDRV::maxValue () const [inline],[virtual]

Returns maximal value of the random variable.

Implements [CRandomVariable](#).

6.21.3.10 virtual double CEmpiricalDRV::minValue () const [inline],[virtual]

After expansion, number of new variables is equal to the number of categories.

Returns minimal value of the random variable.

Implements [CRandomVariable](#).

6.21.3.11 virtual double CEmpiricalDRV::probability (int *k*) const [inline],[virtual]

Function that returns PMF.

6.21.3.12 template<class Archive > void CEmpiricalDRV::serialize (Archive & *ar*, const unsigned int *version*)
[private]

6.21.3.13 void CEmpiricalDRV::setDistribution (const CDiscreteSample & *aS*) [virtual]

Computes PMF and CDF from sample histogram.

6.21.3.14 virtual double CEmpiricalDRV::variance () const [inline],[virtual]

Returns variance of the random variable.

Implements [CRandomVariable](#).

6.21.4 Friends And Related Function Documentation

6.21.4.1 friend class boost::serialization::access [friend]

Boost serialization.

6.21.5 Member Data Documentation

6.21.5.1 ArrayXd CEmpiricalDRV::cdf [private]

PMF (p) and CDF (cdf) data.

6.21.5.2 ArrayXd CEmpiricalDRV::p [private]

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

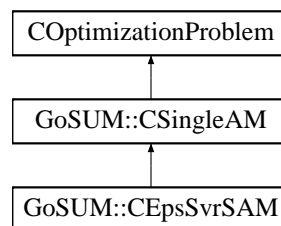
- C:/Development/core/[RandomVariable.h](#)
- C:/Development/core/[RandomVariable.cpp](#)

6.22 GoSUM::CEpsSvrSAM Class Reference

Class for the analytical model for single output state, epsilon-SVR type.

```
#include <AnalyticalModel.h>
```

Inheritance diagram for GoSUM::CEpsSvrSAM:



Public Member Functions

- [CEpsSvrSAM](#) ()
- virtual [~CEpsSvrSAM](#) ()
- virtual void [openOptimization](#) ()
Opens optimization.
- virtual void [optimizationPoint2SVMParam](#) (const ArrayXd &ov)
Converts optimization point to SVM parameters.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.22.1 Detailed Description

Class for the analytical model for single output state, epsilon-SVR type.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 `GoSUM::CEpsSvrSAM::CEpsSvrSAM ()` `[inline]`

6.22.2.2 `virtual GoSUM::CEpsSvrSAM::~~CEpsSvrSAM ()` `[inline], [virtual]`

6.22.3 Member Function Documentation

6.22.3.1 `void GoSUM::CEpsSvrSAM::openOptimization ()` `[virtual]`

Opens optimization.

Reimplemented from [GoSUM::CSingleAM](#).

6.22.3.2 `void GoSUM::CEpsSvrSAM::optimizationPoint2SVMParam (const ArrayXd & ov)` `[virtual]`

Converts optimization point to SVM parameters.

Reimplemented from [GoSUM::CSingleAM](#).

6.22.3.3 `template<class Archive > void GoSUM::CEpsSvrSAM::serialize (Archive & ar, const unsigned int version)`
`[protected]`

Reimplemented from [GoSUM::CSingleAM](#).

6.22.4 Friends And Related Function Documentation

6.22.4.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

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

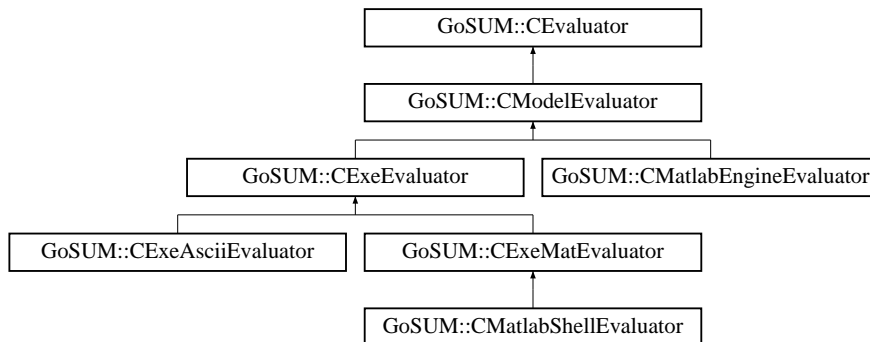
- C:/Development/core/[AnalyticalModel.h](#)
- C:/Development/core/[AnalyticalModel.cpp](#)

6.23 GoSUM::CEvaluator Class Reference

Class for [GoSUM](#) model evaluator.

```
#include <OriginalModel.h>
```

Inheritance diagram for GoSUM::CEvaluator:



Public Types

- enum [evaluator_type](#) { [exeascii](#), [exemat](#), [matlabshell](#), [matlabengine](#) }

Public Member Functions

- [CEvaluator](#) ()
- virtual [~CEvaluator](#) ()

Static Public Member Functions

- static void [clear](#) ()
Clears data.
- static int [ThreadSize](#) ()
Returns thread size.
- static void [SetThreadSize](#) (int _trdN)
Sets thread size.
- static void [SetType](#) (GoSUM::CEvaluator::evaluator_type _type)
Sets evaluator type.
- static void [SetExternalEvaluator](#) (const std::string &_filename)
Sets external evaluator.
- static [evaluator_type](#) [Type](#) (const std::string &_stpe)
Converts evaluator type name to evaluator type enumerator.
- static [evaluator_type](#) [Type](#) ()
Returns evaluator type.
- static [GoSUM::CModelEvaluator](#) * [New](#) (const [CInputParameters](#) *_pIP, [COutputStates](#) *_pOS, int _trdI=0)
Returns new model evaluator.
- static std::string [ExternalEvaluator](#) ()
Returns full path of the external evaluator.
- static std::string [ExternalEvaluatorFolder](#) ()
Returns folder of the external evaluator.

Public Attributes

- boost::signal< void()> [evaluatingProgressed](#)
Signal for evaluation progress, emitted after single evaluation.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Static Protected Attributes

- static int [trdN](#) = 1
Total number of threads and id of the particular thread this class runs in.
- static enum [evaluator_type](#)
Holds type of the evaluator.
- static std::string [path](#)
- static std::string [exe](#)
- static std::string [in](#)
- static std::string [out](#)
- static std::string [ext](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.23.1 Detailed Description

Class for [GoSUM](#) model evaluator.

6.23.2 Member Enumeration Documentation

6.23.2.1 enum GoSUM::CEvaluator::evaluator_type

Enumerator:

exeascii
exemat
matlabshell
matlabengine

6.23.3 Constructor & Destructor Documentation

6.23.3.1 GoSUM::CEvaluator::CEvaluator () [inline]

6.23.3.2 virtual GoSUM::CEvaluator::~~CEvaluator () [inline],[virtual]

6.23.4 Member Function Documentation

6.23.4.1 static void GoSUM::CEvaluator::clear () [inline],[static]

Clears data.

6.23.4.2 std::string GoSUM::CEvaluator::ExternalEvaluator () [static]

Returns full path of the external evaluator.

6.23.4.3 `static std::string GoSUM::CEvaluator::ExternalEvaluatorFolder () [inline],[static]`

Returns folder of the external evaluator.

6.23.4.4 `GoSUM::CModelEvaluator * GoSUM::CEvaluator::New (const CInputParameters * _pIP, COutputStates * _pOS, int _trdI = 0) [static]`

Returns new model evaluator.

6.23.4.5 `template<class Archive > void GoSUM::CEvaluator::serialize (Archive & ar, const unsigned int version) [protected]`

Reimplemented in [GoSUM::CMatlabEngineEvaluator](#), [GoSUM::CMatlabShellEvaluator](#), [GoSUM::CExeMatEvaluator](#), [GoSUM::CExeAsciiEvaluator](#), [GoSUM::CExeEvaluator](#), and [GoSUM::CModelEvaluator](#).

6.23.4.6 `void GoSUM::CEvaluator::SetExternalEvaluator (const std::string & _filename) [static]`

Sets external evaluator.

6.23.4.7 `static void GoSUM::CEvaluator::SetThreadSize (int _trdN) [inline],[static]`

Sets thread size.

6.23.4.8 `void GoSUM::CEvaluator::SetType (GoSUM::CEvaluator::evaluatortype _type) [static]`

Sets evaluator type.

6.23.4.9 `static int GoSUM::CEvaluator::ThreadSize () [inline],[static]`

Returns thread size.

6.23.4.10 `GoSUM::CEvaluator::evaluatortype GoSUM::CEvaluator::Type (const std::string & _stype) [static]`

Converts evaluator type name to evaluator type enumerator.

6.23.4.11 `static evaluatortype GoSUM::CEvaluator::Type () [inline],[static]`

Returns evaluator type.

6.23.5 Friends And Related Function Documentation

6.23.5.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.23.6 Member Data Documentation

6.23.6.1 `boost::signal<void()> GoSUM::CEvaluator::evaluatingProgressed`

Signal for evaluation progress, emitted after single evaluation.

6.23.6.2 `std::string GoSUM::CEvaluator::exe` `[static]`, `[protected]`

6.23.6.3 `std::string GoSUM::CEvaluator::ext` `[static]`, `[protected]`

Names of the external executable, and its input and output files.

6.23.6.4 `std::string GoSUM::CEvaluator::in` `[static]`, `[protected]`

6.23.6.5 `std::string GoSUM::CEvaluator::out` `[static]`, `[protected]`

6.23.6.6 `std::string GoSUM::CEvaluator::path` `[static]`, `[protected]`

6.23.6.7 `int GoSUM::CEvaluator::trdN = 1` `[static]`, `[protected]`

Total number of threads and id of the particular thread this class runs in.

6.23.6.8 `enum evaluatortype GoSUM::CEvaluator::type` `[static]`, `[protected]`

Holds type of the evaluator.

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

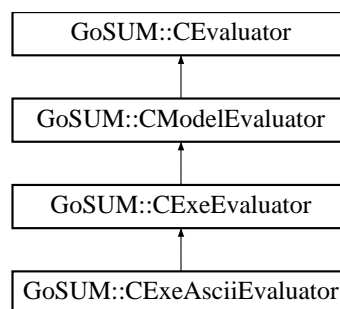
- C:/Development/core/[OriginalModel.h](#)
- C:/Development/core/[OriginalModel.cpp](#)

6.24 GoSUM::CExeAsciiEvaluator Class Reference

Class for the evaluator with ascii i/o and .exe.

```
#include <OriginalModel.h>
```

Inheritance diagram for GoSUM::CExeAsciiEvaluator:



Public Member Functions

- [CExeAsciiEvaluator](#) ()
- [CExeAsciiEvaluator](#) (const [CInputParameters](#) *_pIP, [COutputStates](#) *_pOS, int _trdI)
- virtual [~CExeAsciiEvaluator](#) ()

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

- virtual void [writeTo](#) (std::string _fname, const ArrayXd &_X, std::string _Xname)
- virtual void [readFrom](#) (std::string _fname, ArrayXd &_Y, std::string _Yname)
- virtual int [readSizeFrom](#) (std::string _fname, std::string _Yname)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.24.1 Detailed Description

Class for the evaluator with ascii i/o and .exe.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 `GoSUM::CExeAsciiEvaluator () [inline]`

6.24.2.2 `GoSUM::CExeAsciiEvaluator (const CInputParameters * _pIP, COutputStates * _pOS, int _trdl) [inline]`

6.24.2.3 `virtual GoSUM::CExeAsciiEvaluator::~~CExeAsciiEvaluator () [inline],[virtual]`

6.24.3 Member Function Documentation

6.24.3.1 `void GoSUM::CExeAsciiEvaluator::readFrom (std::string _fname, ArrayXd & _Y, std::string _Yname) [protected],[virtual]`

Implements [GoSUM::CExeEvaluator](#).

6.24.3.2 `int GoSUM::CExeAsciiEvaluator::readSizeFrom (std::string _fname, std::string _Yname) [protected],[virtual]`

Implements [GoSUM::CExeEvaluator](#).

6.24.3.3 `template<class Archive > void GoSUM::CExeAsciiEvaluator::serialize (Archive & ar, const unsigned int version) [protected]`

Reimplemented from [GoSUM::CExeEvaluator](#).

6.24.3.4 `void GoSUM::CExeAsciiEvaluator::writeTo (std::string _fname, const ArrayXd & _X, std::string _Xname) [protected],[virtual]`

Implements [GoSUM::CExeEvaluator](#).

6.24.4 Friends And Related Function Documentation

6.24.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

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

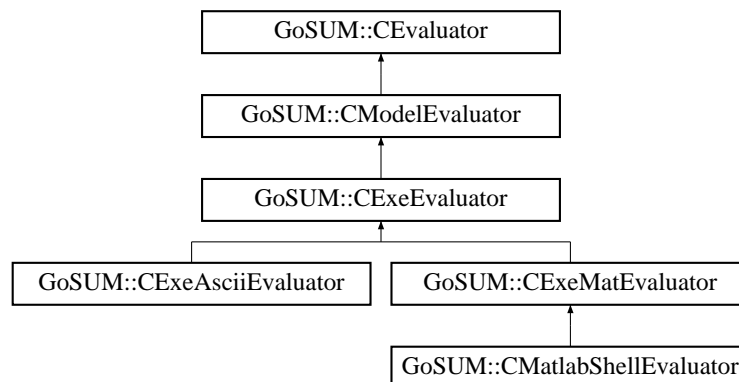
- C:/Development/core/OriginalModel.h
- C:/Development/core/OriginalModel.cpp

6.25 GoSUM::CExeEvaluator Class Reference

Template for the evaluator with some file i/o and .exe derived from [CModelEvaluator](#).

```
#include <OriginalModel.h>
```

Inheritance diagram for GoSUM::CExeEvaluator:



Public Member Functions

- [CExeEvaluator](#) ()
- [CExeEvaluator](#) (const [CInputParameters](#) * _pIP, [COutputStates](#) * _pOS, int _trdI)
- virtual [~CExeEvaluator](#) ()
- virtual void [openEvaluation](#) ()
Opens, i.e. prepares evaluation process.
- virtual [ArrayXd](#) [operator\(\)](#) (const [ArrayXd](#) &X)
Returns output state evaluated on for a single input parameter n-tuple.
- virtual int [evaluateOutputsSize](#) ()
Evaluates outputs size by running model evaluator.
- virtual void [closeEvaluation](#) ()
Closes evaluation process.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- virtual void [systemCommand](#) (std::ostream &cmd)
- virtual void [writeTo](#) (std::string _fname, const [ArrayXd](#) &_X, std::string _Xname)=0
- virtual void [readFrom](#) (std::string _fname, [ArrayXd](#) &_Y, std::string _Yname)=0
- virtual int [readSizeFrom](#) (std::string _fname, std::string _Yname)=0

Protected Attributes

- [QProcess](#) * [pP](#)
Points to the qprocess.

Friends

- class [boost::serialization::access](#)

Boost serialization.

Additional Inherited Members

6.25.1 Detailed Description

Template for the evaluator with some file i/o and .exe derived from [CModelEvaluator](#).

6.25.2 Constructor & Destructor Documentation

6.25.2.1 `GoSUM::CExeEvaluator::CExeEvaluator () [inline]`

6.25.2.2 `GoSUM::CExeEvaluator::CExeEvaluator (const CInputParameters * _pIP, COutputStates * _pOS, int _trdl) [inline]`

6.25.2.3 `virtual GoSUM::CExeEvaluator::~~CExeEvaluator () [inline],[virtual]`

6.25.3 Member Function Documentation

6.25.3.1 `void GoSUM::CExeEvaluator::closeEvaluation () [virtual]`

Closes evaluation process.

Implements [GoSUM::CModelEvaluator](#).

6.25.3.2 `int GoSUM::CExeEvaluator::evaluateOutputsSize () [virtual]`

Evaluates outputs size by running model evaluator.

Reimplemented from [GoSUM::CModelEvaluator](#).

6.25.3.3 `void GoSUM::CExeEvaluator::openEvaluation () [virtual]`

Opens, i.e. prepares evaluation process.

Implements [GoSUM::CModelEvaluator](#).

6.25.3.4 `ArrayXd GoSUM::CExeEvaluator::operator() (const ArrayXd & X) [virtual]`

Returns output state evaluated on for a single input parameter n-tuple.

Implements [GoSUM::CModelEvaluator](#).

6.25.3.5 `virtual void GoSUM::CExeEvaluator::readFrom (std::string _fname, ArrayXd & _Y, std::string _Yname) [protected],[pure virtual]`

Implemented in [GoSUM::CExeMatEvaluator](#), and [GoSUM::CExeAsciiEvaluator](#).

6.25.3.6 `virtual int GoSUM::CExeEvaluator::readSizeFrom (std::string _fname, std::string _Yname)` [protected], [pure virtual]

Implemented in [GoSUM::CExeMatEvaluator](#), and [GoSUM::CExeAsciiEvaluator](#).

6.25.3.7 `template<class Archive > void GoSUM::CExeEvaluator::serialize (Archive & ar, const unsigned int version)` [protected]

Reimplemented from [GoSUM::CModelEvaluator](#).

Reimplemented in [GoSUM::CMatlabShellEvaluator](#), [GoSUM::CExeMatEvaluator](#), and [GoSUM::CExeAsciiEvaluator](#).

6.25.3.8 `virtual void GoSUM::CExeEvaluator::systemCommand (std::ostream & cmd)` [inline], [protected], [virtual]

Reimplemented in [GoSUM::CMatlabShellEvaluator](#).

6.25.3.9 `virtual void GoSUM::CExeEvaluator::writeTo (std::string _fname, const ArrayXd & _X, std::string _Xname)` [protected], [pure virtual]

Implemented in [GoSUM::CExeMatEvaluator](#), and [GoSUM::CExeAsciiEvaluator](#).

6.25.4 Friends And Related Function Documentation

6.25.4.1 `friend class boost::serialization::access` [friend]

Boost serialization.

6.25.5 Member Data Documentation

6.25.5.1 `QProcess* GoSUM::CExeEvaluator::pP` [protected]

Points to the qprocess.

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

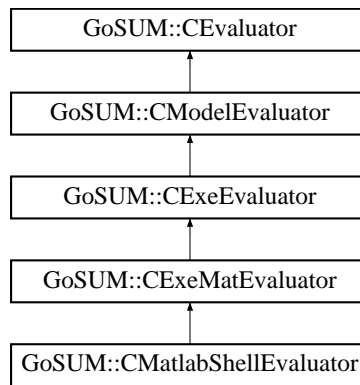
- C:/Development/core/[OriginalModel.h](#)
- C:/Development/core/[OriginalModel.cpp](#)

6.26 GoSUM::CExeMatEvaluator Class Reference

Class for the evaluator with Matlab .mat i/o and .exe.

```
#include <OriginalModel.h>
```

Inheritance diagram for GoSUM::CExeMatEvaluator:



Public Member Functions

- [CExeMatEvaluator](#) ()
- [CExeMatEvaluator](#) (const [CInputParameters](#) * _pIP, [COutputStates](#) * _pOS, int _trdl)
- virtual [~CExeMatEvaluator](#) ()

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- virtual void [writeTo](#) (std::string _fname, const ArrayXd &_X, std::string _Xname)
- virtual void [readFrom](#) (std::string _fname, ArrayXd &_Y, std::string _Yname)
- virtual int [readSizeFrom](#) (std::string _fname, std::string _Yname)

Protected Attributes

- [CMATLAB matlab](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.26.1 Detailed Description

Class for the evaluator with Matlab .mat i/o and .exe.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 [GoSUM::CExeMatEvaluator::CExeMatEvaluator \(\)](#) [[inline](#)]

6.26.2.2 [GoSUM::CExeMatEvaluator::CExeMatEvaluator \(const \[CInputParameters\]\(#\) * _pIP, \[COutputStates\]\(#\) * _pOS, int _trdl \)](#) [[inline](#)]

6.26.2.3 [virtual GoSUM::CExeMatEvaluator::~~CExeMatEvaluator \(\)](#) [[inline](#)], [[virtual](#)]

6.26.3 Member Function Documentation

6.26.3.1 `void GoSUM::CExeMatEvaluator::readFrom (std::string _fname, ArrayXd & _Y, std::string _Yname)`
`[protected], [virtual]`

Implements [GoSUM::CExeEvaluator](#).

6.26.3.2 `int GoSUM::CExeMatEvaluator::readSizeFrom (std::string _fname, std::string _Yname)` `[protected],`
`[virtual]`

Implements [GoSUM::CExeEvaluator](#).

6.26.3.3 `template<class Archive > void GoSUM::CExeMatEvaluator::serialize (Archive & ar, const unsigned int version)`
`[protected]`

Reimplemented from [GoSUM::CExeEvaluator](#).

Reimplemented in [GoSUM::CMatlabShellEvaluator](#).

6.26.3.4 `void GoSUM::CExeMatEvaluator::writeTo (std::string _fname, const ArrayXd & _X, std::string _Xname)`
`[protected], [virtual]`

Implements [GoSUM::CExeEvaluator](#).

6.26.4 Friends And Related Function Documentation

6.26.4.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.26.5 Member Data Documentation

6.26.5.1 **CMATLAB** `GoSUM::CExeMatEvaluator::matlab` `[protected]`

Object for matlab library functions.

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

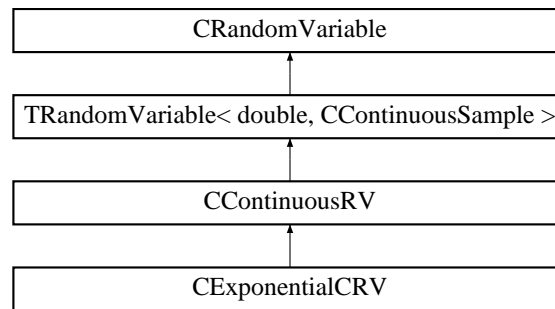
- C:/Development/core/[OriginalModel.h](#)
- C:/Development/core/[OriginalModel.cpp](#)

6.27 CExponentialCRV Class Reference

Class for exponential continuous random variables derived from continuous random variables.

```
#include <RandomVariable.h>
```

Inheritance diagram for CExponentialCRV:



Public Member Functions

- `CExponentialCRV ()`
- virtual `~CExponentialCRV ()`
- `CExponentialCRV (const CExponentialCRV &O)`
- virtual void `setDistribution (double _lambda, double _dp2=0)`
Set distribution parameters.
- virtual void `setDistribution (const CContinuousSample &_aS)`
Set distribution parameters from sample empirical parameters.
- virtual double `probability (double _x) const`
Function that returns PDF.
- virtual double `cumulative (double _x) const`
Function that returns CDF.
- virtual double `minValue () const`
Returns minimal value of the random variable.
- virtual double `maxValue () const`
Returns maximal value of the random variable.
- virtual double `expectedValue () const`
Returns expected value of the random variable.
- virtual `distributiontype distributionType () const`
Returns enum type of the random variable distribution.
- virtual `std::string distributionName () const`
Returns name of the random variable distribution.
- virtual `ArrayXd exportDomain () const`
Exports domain of the random variable.
- double `rateParameter () const`
Returns rate parameter of exponential random variable.
- void `setRateParameter (double _lambda)`
Sets rate parameter of exponential random variable.
- virtual double `variance () const`
Returns variance of the random variable.

Protected Member Functions

- virtual double `doQuantile (double _p) const`
Quantile, formula implementation.

Private Member Functions

- `template<class Archive >`
void `serialize (Archive &ar, const unsigned int version)`

Private Attributes

- double [lambda](#)
- double [beta](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.27.1 Detailed Description

Class for exponential continuous random variables derived from continuous random variables.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 `CExponentialCRV::CExponentialCRV ()` [\[inline\]](#)

6.27.2.2 `virtual CExponentialCRV::~~CExponentialCRV ()` [\[inline\]](#), [\[virtual\]](#)

6.27.2.3 `CExponentialCRV::CExponentialCRV (const CExponentialCRV & O)` [\[inline\]](#)

6.27.3 Member Function Documentation

6.27.3.1 `virtual double CExponentialCRV::cumulative (double _x) const` [\[inline\]](#), [\[virtual\]](#)

Function that returns CDF.

6.27.3.2 `virtual std::string CExponentialCRV::distributionName () const` [\[inline\]](#), [\[virtual\]](#)

Returns name of the random variable distribution.

Implements [CRandomVariable](#).

6.27.3.3 `virtual distributiontype CExponentialCRV::distributionType () const` [\[inline\]](#), [\[virtual\]](#)

Returns enum type of the random variable distribution.

Implements [CRandomVariable](#).

6.27.3.4 `virtual double CExponentialCRV::doQuantile (double _p) const` [\[inline\]](#), [\[protected\]](#), [\[virtual\]](#)

Quantile, formula implementation.

Implements [CRandomVariable](#).

6.27.3.5 `virtual double CExponentialCRV::expectedValue () const` [\[inline\]](#), [\[virtual\]](#)

Returns expected value of the random variable.

Implements [CRandomVariable](#).

6.27.3.6 `ArrayXd CExponentialCRV::exportDomain () const` `[virtual]`

Exports domain of the random variable.

Implements [TRandomVariable< t, T >](#).

6.27.3.7 `virtual double CExponentialCRV::maxValue () const` `[inline], [virtual]`

Returns maximal value of the random variable.

Implements [CRandomVariable](#).

6.27.3.8 `virtual double CExponentialCRV::minValue () const` `[inline], [virtual]`

Returns minimal value of the random variable.

Implements [CRandomVariable](#).

6.27.3.9 `virtual double CExponentialCRV::probability (double _x) const` `[inline], [virtual]`

Function that returns PDF.

6.27.3.10 `double CExponentialCRV::rateParameter () const` `[inline]`

Returns rate parameter of exponential random variable.

6.27.3.11 `template<class Archive > void CExponentialCRV::serialize (Archive & ar, const unsigned int version)`
`[private]`

6.27.3.12 `void CExponentialCRV::setDistribution (double _lambda, double _dp2 = 0)` `[virtual]`

Set distribution parameters.

6.27.3.13 `virtual void CExponentialCRV::setDistribution (const CContinuousSample & _aS)` `[inline], [virtual]`

Set distribution parameters from sample empirical parameters.

6.27.3.14 `void CExponentialCRV::setRateParameter (double _lambda)` `[inline]`

Sets rate parameter of exponential random variable.

6.27.3.15 `virtual double CExponentialCRV::variance () const` `[inline], [virtual]`

Returns variance of the random variable.

Implements [CRandomVariable](#).

6.27.4 Friends And Related Function Documentation

6.27.4.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.27.5 Member Data Documentation

6.27.5.1 double CExponentialCRV::beta [private]

Distribution parameters: rate parameter $\beta = 1/\lambda$.

6.27.5.2 double CExponentialCRV::lambda [private]

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

- C:/Development/core/[RandomVariable.h](#)
- C:/Development/core/[RandomVariable.cpp](#)

6.28 CFFTW Class Reference

Class interface for FFTW's libfftw library.

```
#include <FFTWLibrary.h>
```

Public Member Functions

- [CFFTW](#) ()
- virtual [~CFFTW](#) ()
- ArrayXd [discreteCosineTransform](#) (const ArrayXd &in)
Computes discrete cosine transform using fftw.
- ArrayXd [inverseDiscreteCosineTransform](#) (const ArrayXd &in)
Computes inverse discrete cosine transform using fftw.

6.28.1 Detailed Description

Class interface for FFTW's libfftw library.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 CFFTW::CFFTW () [inline]

6.28.2.2 virtual CFFTW::~~CFFTW () [inline], [virtual]

6.28.3 Member Function Documentation

6.28.3.1 ArrayXd CFFTW::discreteCosineTransform (const ArrayXd &in)

Computes discrete cosine transform using fftw.

6.28.3.2 ArrayXd CFFTW::inverseDiscreteCosineTransform (const ArrayXd &in)

Computes inverse discrete cosine transform using fftw.

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

- C:/Development/core/[FFTWLibrary.h](#)
- C:/Development/core/[FFTWLibrary.cpp](#)

6.29 CGAModelOptimization Class Reference

Class for the genetic algorithm, i.e. interface for GALIB.

```
#include <GAOptimization.h>
```

Public Member Functions

- [CGAModelOptimization](#) ()
- virtual [~CGAModelOptimization](#) ()
- [ArrayXd](#) [optimize](#) ([COptimizationProblem](#) * _pSOP, [std::ostream](#) & _out=[std::cout](#))
Solves the optimizations problem.
- [int](#) [progressStepsSize](#) () const
Returns progress steps size.
- [int](#) [populationSize](#) () const
Returns population size.
- [void](#) [setPopulationSize](#) ([int](#) _popsize)
Sets population size.

Static Public Member Functions

- [static](#) [ArrayXd](#) [GAGenome2ArrayXd](#) ([const](#) [GAGenome](#) &g)
Converts GAGenome -> hpyercube point.
- [static](#) [float](#) [Fitness](#) ([GAGenome](#) &g)
Evaluates fitness of an individual.

Protected Member Functions

- [template](#)<[class](#) [Archive](#) >
[void](#) [serialize](#) ([Archive](#) &ar, [const](#) [unsigned](#) [int](#) version)

Protected Attributes

- [int](#) [popsize](#)
- [ArrayXd](#) [Xbest](#)
- [double](#) [ybest](#)

Static Protected Attributes

- [static](#) [COptimizationProblem](#) * [pOP](#) = [NULL](#)
Points to the optimization problem.
- [static](#) [double](#) [C](#) = 1.e+2
Constant added to get positive fitness.

Friends

- [class](#) [boost::serialization::access](#)
Boost serialization.

6.29.1 Detailed Description

Class for the genetic algorithm, i.e. interface for GALIB.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 `CGAModelOptimization::CGAModelOptimization ()` `[inline]`

6.29.2.2 `virtual CGAModelOptimization::~~CGAModelOptimization ()` `[inline]`, `[virtual]`

6.29.3 Member Function Documentation

6.29.3.1 `float CGAModelOptimization::Fitness (GAGenome & g)` `[static]`

Evaluates fitness of an individual.

6.29.3.2 `ArrayXd CGAModelOptimization::GAGenome2ArrayXd (const GAGenome & g)` `[static]`

Converts GAGenome -> hypercube point.

6.29.3.3 `ArrayXd CGAModelOptimization::optimize (COptimizationProblem * _pSOP, std::ostream & _out = std::cout)`

Solves the optimizations problem.

6.29.3.4 `int CGAModelOptimization::populationSize () const` `[inline]`

Returns population size.

6.29.3.5 `int CGAModelOptimization::progressStepsSize () const` `[inline]`

Returns progress steps size.

6.29.3.6 `template<class Archive > void CGAModelOptimization::serialize (Archive & ar, const unsigned int version)` `[protected]`

6.29.3.7 `void CGAModelOptimization::setPopulationSize (int _popsize)` `[inline]`

Sets population size.

6.29.4 Friends And Related Function Documentation

6.29.4.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.29.5 Member Data Documentation

6.29.5.1 `double CGAModelOptimization::C = 1.e+2` `[static]`, `[protected]`

Constant added to get positive fitness.

6.29.5.2 **COptimizationProblem** * **CGAModelOptimization::pOP = NULL** [static], [protected]

Points to the optimization problem.

6.29.5.3 **int** **CGAModelOptimization::popsize** [protected]

6.29.5.4 **ArrayXd** **CGAModelOptimization::Xbest** [protected]

6.29.5.5 **double** **CGAModelOptimization::ybest** [protected]

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

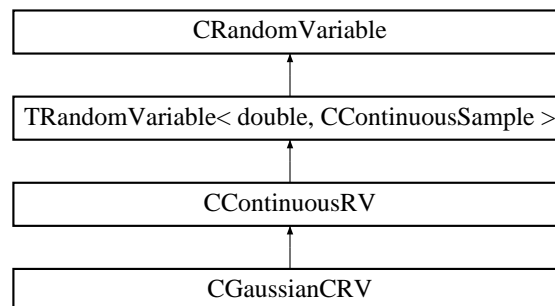
- C:/Development/core/[GAOptimization.h](#)
- C:/Development/core/[GAOptimization.cpp](#)

6.30 CGaussianCRV Class Reference

Class for Gaussian continuous random variables derived from continuous random variables.

```
#include <RandomVariable.h>
```

Inheritance diagram for CGaussianCRV:



Public Member Functions

- [CGaussianCRV](#) ()
- virtual [~CGaussianCRV](#) ()
- [CGaussianCRV](#) (const [CGaussianCRV](#) &O)
- virtual void [setDistribution](#) (double _mu, double _sigma)
Set distribution parameters.
- virtual void [setDistribution](#) (const [CContinuousSample](#) &_aS)
Set distribution parameters from sample empirical parameters.
- virtual double [probability](#) (double _x) const
Function that returns PDF.
- virtual double [cumulative](#) (double _x) const
Function that returns CDF.
- virtual double [minValue](#) () const
Returns minimal value of the random variable.
- virtual double [maxValue](#) () const
Returns maximal value of the random variable.
- virtual double [expectedValue](#) () const
Returns expected value of the random variable.

- virtual [distributiontype distributionType](#) () const
Returns enum type of the random variable distribution.
- virtual std::string [distributionName](#) () const
Returns name of the random variable distribution.
- virtual ArrayXd [exportDomain](#) () const
Exports domain of the random variable.
- double [mean](#) () const
Returns mean of Gaussian random variable.
- double [standardDeviation](#) () const
Returns standard deviation of Gaussian random variable.
- void [setMean](#) (double _mu)
Sets mean of Gaussian random variable.
- void [setStandardDeviation](#) (double _sigma)
Sets standard deviation of Gaussian random variable.
- virtual double [variance](#) () const
Returns variance of the random variable.

Protected Member Functions

- virtual double [doQuantile](#) (double _p) const
Quantile, formula implementation.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Private Attributes

- double [mu](#)
- double [sigma](#)
- double [sigma2](#)
- double [sigmasqrt2](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.30.1 Detailed Description

Class for Gaussian continuous random variables derived from continuous random variables.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 `CGaussianCRV::CGaussianCRV ()` `[inline]`

6.30.2.2 `virtual CGaussianCRV::~~CGaussianCRV ()` `[inline],[virtual]`

6.30.2.3 `CGaussianCRV::CGaussianCRV (const CGaussianCRV & O)` `[inline]`

6.30.3 Member Function Documentation

6.30.3.1 `virtual double CGaussianCRV::cumulative (double _x) const` `[inline],[virtual]`

Function that returns CDF.

6.30.3.2 `virtual std::string CGaussianCRV::distributionName () const` `[inline],[virtual]`

Returns name of the random variable distribution.

Implements [CRandomVariable](#).

6.30.3.3 `virtual distributiontype CGaussianCRV::distributionType () const` `[inline],[virtual]`

Returns enum type of the random variable distribution.

Implements [CRandomVariable](#).

6.30.3.4 `virtual double CGaussianCRV::doQuantile (double _p) const` `[inline],[protected],[virtual]`

Quantile, formula implementation.

Implements [CRandomVariable](#).

6.30.3.5 `virtual double CGaussianCRV::expectedValue () const` `[inline],[virtual]`

Returns expected value of the random variable.

Implements [CRandomVariable](#).

6.30.3.6 `ArrayXd CGaussianCRV::exportDomain () const` `[virtual]`

Exports domain of the random variable.

Implements [TRandomVariable< t, T >](#).

6.30.3.7 `virtual double CGaussianCRV::maxValue () const` `[inline],[virtual]`

Returns maximal value of the random variable.

Implements [CRandomVariable](#).

6.30.3.8 `double CGaussianCRV::mean () const` `[inline]`

Returns mean of Gaussian random variable.

6.30.3.9 `virtual double CGaussianCRV::minValue () const [inline],[virtual]`

Returns minimal value of the random variable.

Implements [CRandomVariable](#).

6.30.3.10 `virtual double CGaussianCRV::probability (double _x) const [inline],[virtual]`

Function that returns PDF.

6.30.3.11 `template<class Archive > void CGaussianCRV::serialize (Archive & ar, const unsigned int version) [private]`

6.30.3.12 `void CGaussianCRV::setDistribution (double _mu, double _sigma) [virtual]`

Set distribution parameters.

6.30.3.13 `virtual void CGaussianCRV::setDistribution (const CContinuousSample & _aS) [inline],[virtual]`

Set distribution parameters from sample empirical parameters.

6.30.3.14 `void CGaussianCRV::setMean (double _mu) [inline]`

Sets mean of Gaussian random variable.

6.30.3.15 `void CGaussianCRV::setStandardDeviation (double _sigma) [inline]`

Sets standard deviation of Gaussian random variable.

6.30.3.16 `double CGaussianCRV::standardDeviation () const [inline]`

Returns standard deviation of Gaussian random variable.

6.30.3.17 `virtual double CGaussianCRV::variance () const [inline],[virtual]`

Returns variance of the random variable.

Implements [CRandomVariable](#).

6.30.4 Friends And Related Function Documentation

6.30.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.30.5 Member Data Documentation

6.30.5.1 `double CGaussianCRV::mu [private]`

6.30.5.2 `double CGaussianCRV::sigma [private]`

6.30.5.3 `double CGaussianCRV::sigma2` [private]

6.30.5.4 `double CGaussianCRV::sigmasqrt2` [private]

Distribution parameters: mean (μ) and standard deviation (σ).

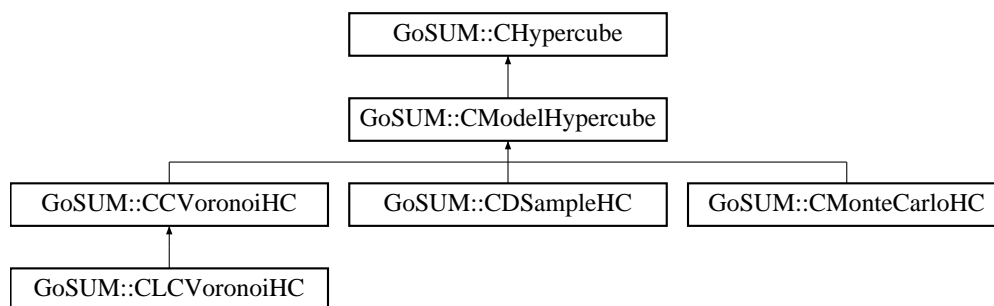
The documentation for this class was generated from the following files:

- C:/Development/core/[RandomVariable.h](#)
- C:/Development/core/[RandomVariable.cpp](#)

6.31 GoSUM::CHypercube Class Reference

```
#include <Hypercube.h>
```

Inheritance diagram for GoSUM::CHypercube:



Public Types

- enum `hctype` { `dsample`, `montecarlo`, `cvoronoi`, `lcvoronoi` }

Public Member Functions

- `CHypercube()`
- virtual `~CHypercube()`

Static Public Member Functions

- static `hctype Type` (const std::string &_stype)
Returns hypercube type enumerator from hypercube type name.
- static `hctype Type` ()
Returns hypercube type.
- static void `SetType` (hctype _etype)
Sets hypercube type.
- static void `VoronoiOptions` (int &_maxiter, int &_q, double &_alpha2, double &_beta2)
- static void `SetVoronoiOptions` (int _maxiter, int _q, double _alpha2, double _beta2)
- static int `CvtIterationSize` ()
Returns CVT iteration size.
- static void `SetCvtIterationSize` (int _maxiter)
Sets CVT iteration size.
- static int `CvtPointsSize` ()

- Returns CVT points size.*
- static void [SetCvtPointsSize](#) (int _q)
Sets CVT points size.
- static double [CvtOldCenterCoefficient](#) ()
Returns CVT parameter #1.
- static void [SetCvtOldCenterCoefficient](#) (double _alpha2)
Sets CVT paramter #1.
- static double [CvtNewCenterCoefficient](#) ()
Returns CVT paramter #2.
- static void [SetCvtNewCenterCoefficient](#) (double _beta2)
Sets CVT paramter #2.
- static [GoSUM::CModelHypercube](#) * [New](#) ([CInputParameters](#) * _pIP)
Returns new model hypercube.
- static int [ProgressSize](#) (int _rssize, int _dim)
Returns progress steps size.

Public Attributes

- boost::signal< void()> [generatingProgressed](#)
Signal for centralize progress.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Static Protected Attributes

- static [hctype](#) [etype](#) = [GoSUM::CHypercube::dsample](#)
Holds hypercube type.
- static int [maxiter](#) = 10000
- static int [q](#) = 1
Holds CVT parameters.
- static double [alpha2](#) = 0.
- static double [beta2](#) = 1.

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.31.1 Member Enumeration Documentation

6.31.1.1 enum [GoSUM::CHypercube::hctype](#)

Enumerator:

dsample
montecarlo
cvoronoi
lcvoronoi

6.31.2 Constructor & Destructor Documentation

6.31.2.1 `GoSUM::CHypercube::CHypercube () [inline]`

6.31.2.2 `virtual GoSUM::CHypercube::~~CHypercube () [inline],[virtual]`

6.31.3 Member Function Documentation

6.31.3.1 `static int GoSUM::CHypercube::CvtIterationSize () [inline],[static]`

Returns CVT iteration size.

6.31.3.2 `static double GoSUM::CHypercube::CvtNewCenterCoefficient () [inline],[static]`

Returns CVT paramter #2.

6.31.3.3 `static double GoSUM::CHypercube::CvtOldCenterCoefficient () [inline],[static]`

Returns CVT parameter #1.

6.31.3.4 `static int GoSUM::CHypercube::CvtPointsSize () [inline],[static]`

Returns CVT points size.

6.31.3.5 `GoSUM::CModelHypercube * GoSUM::CHypercube::New (CInputParameters * _pIP) [static]`

Returns new model hypercube.

6.31.3.6 `int GoSUM::CHypercube::ProgressSize (int _rssize, int _dim) [static]`

Returns progress steps size.

6.31.3.7 `template<class Archive > void GoSUM::CHypercube::serialize (Archive & ar, const unsigned int version) [protected]`

Reimplemented in [GoSUM::CLCVoronoiHC](#), [GoSUM::CCVoronoiHC](#), [GoSUM::CMonteCarloHC](#), and [GoSUM::C-ModelHypercube](#).

6.31.3.8 `static void GoSUM::CHypercube::SetCvtIterationSize (int _maxiter) [inline],[static]`

Sets CVT iteration size.

6.31.3.9 `static void GoSUM::CHypercube::SetCvtNewCenterCoefficient (double _beta2) [inline],[static]`

Sets CVT paramter #2.

6.31.3.10 `static void GoSUM::CHypercube::SetCvtOldCenterCoefficient (double _alpha2) [inline],[static]`

Sets CVT paramter #1.

6.31.3.11 `static void GoSUM::CHypercube::SetCvtPointsSize (int _q)` `[inline], [static]`

Sets CVT points size.

6.31.3.12 `void GoSUM::CHypercube::SetType (hctype _etype)` `[static]`

Sets hypercube type.

6.31.3.13 `static void GoSUM::CHypercube::SetVoronoiOptions (int _maxiter, int _q, double _alpha2, double _beta2)`
`[inline], [static]`

Parameters

<i>_beta2</i>	Sets Voronoi options.
---------------	-----------------------

6.31.3.14 `GoSUM::CHypercube::hctype GoSUM::CHypercube::Type (const std::string & _stype)` `[static]`

Returns hypercube type enumerator from hypercube type name.

6.31.3.15 `static hctype GoSUM::CHypercube::Type ()` `[inline], [static]`

Returns hypercube type.

6.31.3.16 `static void GoSUM::CHypercube::VoronoiOptions (int & _maxiter, int & _q, double & _alpha2, double & _beta2)`
`[inline], [static]`

Parameters

<i>_beta2</i>	Returns Voronoi options.
---------------	--------------------------

6.31.4 Friends And Related Function Documentation

6.31.4.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.31.5 Member Data Documentation

6.31.5.1 `double GoSUM::CHypercube::alpha2 = 0.` `[static], [protected]`

6.31.5.2 `double GoSUM::CHypercube::beta2 = 1.` `[static], [protected]`

Holds CVT parameters.

6.31.5.3 `GoSUM::CHypercube::hctype GoSUM::CHypercube::etype = GoSUM::CHypercube::dsample`
`[static], [protected]`

Holds hypercube type.

6.31.5.4 `boost::signal<void()> GoSUM::CHypercube::generatingProgressed`

Signal for centralize progress.

6.31.5.5 `int GoSUM::CHypercube::maxiter = 10000` `[static], [protected]`

6.31.5.6 `int GoSUM::CHypercube::q = 1` `[static], [protected]`

Holds CVT parameters.

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

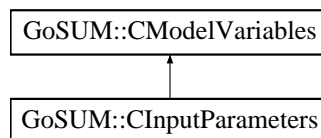
- `C:/Development/core/Hypercube.h`
- `C:/Development/core/Hypercube.cpp`

6.32 GoSUM::CInputParameters Class Reference

Class for `GoSUM` input parameters.

```
#include <Model.h>
```

Inheritance diagram for `GoSUM::CInputParameters`:



Public Member Functions

- `CInputParameters (CModelConstraints *_pIC)`
- `virtual ~CInputParameters ()`
- `int resampleSize () const`
Returns resampling size.
- `void setResampleSize (int _RSsize)`
Sets resampling size.
- `void generateSamples (std::vector< ArrayXd > &_samples)`
Generates samples on external vector of samples.
- `void generateSamples ()`
Generates samples on internal model variable samples.
- `int resampleStepsSize () const`
Returns resample steps size.
- `CModelConstraints * constraints () const`
Returns pointer to model constraints.
- `void setProgressSlot (boost::function< void()> _progressSlot)`
Sets external progress slot.

Public Attributes

- `boost::function< void()> progressSlot`
External progress slot, later connected to signal for evalaution progress.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- [CInputParameters](#) ()

Private Attributes

- [CModelConstraints](#) * [pIC](#)
Points to model constraints.
- int [RSSize](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.32.1 Detailed Description

Class for [GoSUM](#) input parameters.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 `GoSUM::CInputParameters::CInputParameters () [inline], [private]`

6.32.2.2 `GoSUM::CInputParameters::CInputParameters (CModelConstraints * pIC) [inline]`

6.32.2.3 `virtual GoSUM::CInputParameters::~~CInputParameters () [inline], [virtual]`

6.32.3 Member Function Documentation

6.32.3.1 `CModelConstraints* GoSUM::CInputParameters::constraints () const [inline]`

Returns pointer to model constraints.

6.32.3.2 `void GoSUM::CInputParameters::generateSamples (std::vector< ArrayXd > & _samples)`

Generates samples on external vector of samples.

6.32.3.3 `void GoSUM::CInputParameters::generateSamples ()`

Generates samples on internal model variable samples.

6.32.3.4 `int GoSUM::CInputParameters::resampleSize () const [inline]`

Returns resampling size.

6.32.3.5 `int GoSUM::CInputParameters::resampleStepsSize () const`

Returns resample steps size.

6.32.3.6 `template<class Archive > void GoSUM::CInputParameters::serialize (Archive & ar, const unsigned int version)
[private]`

6.32.3.7 `void GoSUM::CInputParameters::setProgressSlot (boost::function< void()> _progressSlot) [inline]`

Sets external progress slot.

6.32.3.8 `void GoSUM::CInputParameters::setResampleSize (int _Rssize) [inline]`

Sets resampling size.

6.32.4 Friends And Related Function Documentation

6.32.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.32.5 Member Data Documentation

6.32.5.1 `CModelConstraints* GoSUM::CInputParameters::pIC [private]`

Points to model constraints.

6.32.5.2 `boost::function<void()> GoSUM::CInputParameters::progressSlot`

External progress slot, later connected to signal for evaluation progress.

6.32.5.3 `int GoSUM::CInputParameters::Rssize [private]`

Holds resample size.

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

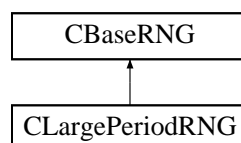
- C:/Development/core/Model.h
- C:/Development/core/Model.cpp

6.33 CLargePeriodRNG Class Reference

Class for uniform RNG with large period = $3.138 \cdot 10^{57}$.

`#include <RandomGenerators.h>`

Inheritance diagram for CLargePeriodRNG:



Public Member Functions

- [CLargePeriodRNG](#) ()
- [CLargePeriodRNG](#) (unsigned int s)
- virtual [~CLargePeriodRNG](#) ()
- virtual void [setSeed](#) (unsigned int s)
Sets seed of the RNG.
- virtual unsigned int [rndi](#) ()
Returns randomly generated unsigned int.
- virtual double [rnd](#) ()
Returns randomly generated double between 0 and 1.

Private Types

- typedef unsigned long long int [Ullong](#)

Private Member Functions

- [Ullong int64](#) ()

Private Attributes

- [Ullong u](#)
- [Ullong v](#)
- [Ullong w](#)
Parameters of the large period RNG.

6.33.1 Detailed Description

Class for uniform RNG with large period = 3.138×10^{57} .

6.33.2 Member Typedef Documentation

6.33.2.1 typedef unsigned long long int [CLargePeriodRNG::Ullong](#) [private]

6.33.3 Constructor & Destructor Documentation

6.33.3.1 [CLargePeriodRNG::CLargePeriodRNG](#) () [inline]

6.33.3.2 [CLargePeriodRNG::CLargePeriodRNG](#) (unsigned int s) [inline]

6.33.3.3 virtual [CLargePeriodRNG::~~CLargePeriodRNG](#) () [inline],[virtual]

6.33.4 Member Function Documentation

6.33.4.1 [Ullong CLargePeriodRNG::int64](#) () [inline],[private]

6.33.4.2 virtual double [CLargePeriodRNG::rnd](#) () [inline],[virtual]

Returns randomly generated double between 0 and 1.

Implements [CBaseRNG](#).

6.33.4.3 `virtual unsigned int CLargePeriodRNG::rndi () [inline],[virtual]`

Returns randomly generated unsigned int.

Implements [CBaseRNG](#).

6.33.4.4 `virtual void CLargePeriodRNG::setSeed (unsigned int s) [inline],[virtual]`

Sets seed of the RNG.

Parameters

<code>s</code>	Sets seed of the RNG.
----------------	-----------------------

Implements [CBaseRNG](#).

6.33.5 Member Data Documentation

6.33.5.1 `Ullong CLargePeriodRNG::u [private]`

6.33.5.2 `Ullong CLargePeriodRNG::v [private]`

6.33.5.3 `Ullong CLargePeriodRNG::w [private]`

Parameters of the large period RNG.

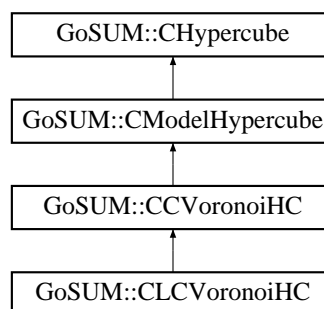
The documentation for this class was generated from the following file:

- `C:/Development/core/RandomGenerators.h`

6.34 GoSUM::CLCVoronoiHC Class Reference

```
#include <Hypercube.h>
```

Inheritance diagram for GoSUM::CLCVoronoiHC:



Public Member Functions

- [CLCVoronoiHC](#) ([CInputParameters](#) *_pIP)
- `virtual ~CLCVoronoiHC ()`

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- virtual void [doGenerate](#) (int _rssize, int _dim, std::vector< ArrayXd > &_samples)
Core of the generation.
- void [latinize](#) (std::vector< ArrayXd > &_samples)
Latinizes model points _samples.
- [CLCVoronoiHC](#) ()

Static Protected Member Functions

- static bool [compcoo](#) (ArrayXd &a, ArrayXd &b)

Static Protected Attributes

- static int [coo](#)
Used in compcoo.

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.34.1 Constructor & Destructor Documentation

6.34.1.1 GoSUM::CLCVoronoiHC::CLCVoronoiHC () [inline],[protected]

6.34.1.2 GoSUM::CLCVoronoiHC::CLCVoronoiHC (CInputParameters * _pIP) [inline]

6.34.1.3 virtual GoSUM::CLCVoronoiHC::~~CLCVoronoiHC () [inline],[virtual]

6.34.2 Member Function Documentation

6.34.2.1 static bool GoSUM::CLCVoronoiHC::compcoo (ArrayXd &a, ArrayXd &b) [inline],[static],[protected]

Compare points by coo coordinate value.

6.34.2.2 void GoSUM::CLCVoronoiHC::doGenerate (int _rssize, int _dim, std::vector< ArrayXd > &_samples) [protected],[virtual]

Core of the generation.

Reimplemented from [GoSUM::CCVoronoiHC](#).

6.34.2.3 void GoSUM::CLCVoronoiHC::latinize (std::vector< ArrayXd > &_samples) [protected]

Latinizes model points _samples.

6.34.2.4 `template<class Archive > void GoSUM::CLCVoronoiHC::serialize (Archive & ar, const unsigned int version)`
`[protected]`

Reimplemented from [GoSUM::CCVoronoiHC](#).

6.34.3 Friends And Related Function Documentation

6.34.3.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.34.4 Member Data Documentation

6.34.4.1 `int GoSUM::CLCVoronoiHC::coo` `[static], [protected]`

Used in compcoo.

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

- [C:/Development/core/Hypercube.h](#)
- [C:/Development/core/Hypercube.cpp](#)

6.35 CMADS Class Reference

Class for the mesh addaptive direct serach, i.e. interface for NOMAD.

```
#include <MADSOptimization.h>
```

Public Member Functions

- [CMADS](#) ()
- virtual [~CMADS](#) ()
- `ArrayXd optimize (COptimizationProblem *_pOP, std::ostream & _out=std::cout)`
Solves the optimizations problem.
- `int evaluationSize () const`
Returns maximal number of evaluations.
- `void setEvaluationSize (int _maxeval)`
Sets maximal number of evaluations.
- `int initialLHSearch () const`
Returns initial value for LH search.
- `void setInitialLHSearch (int _lh0)`
Sets initial value for LH search.
- `int iterationLHSearch () const`
Returns iteration value for LH search.
- `void setIterationLHSearch (int _lhi)`
Sets iteration value for LH search.
- `double initalMeshSize () const`
Returns initial mesh size factor.
- `void setInitialMeshSize (double _ims)`
Sets initial mesh size factor.
- `double minimalPollSize () const`

- Returns minimal poll size factor.*
- void [setMinimalPollSize](#) (double _mps)
Sets minimal poll size factor.
- int [progressStepsSize](#) () const
Returns progress steps size.

Static Public Member Functions

- static NOMAD::Point [ArrayXd2NOMADPoint](#) (const ArrayXd &x)
Utility: converts ArrayXd to NOMAD::Point.
- static ArrayXd [NOMADPoint2ArrayXd](#) (const NOMAD::Point &p)
Utility: converts NOMAD::Point to ArrayXd.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Protected Attributes

- [COptimizationProblem](#) * [pOP](#)
Points to the optimization problem.
- int [maxeval](#)
Holds maximal number of objective & constraints evaluations during optimization.
- int [lh0](#)
- int [lhi](#)
- double [ims](#)
- double [mps](#)
- ArrayXd [Xbest](#)
- double [ybest](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.35.1 Detailed Description

Class for the mesh addaptive direct serach, i.e. interface for NOMAD.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 [CMADS](#) () [inline]

6.35.2.2 virtual [CMADS::~~CMADS](#) () [inline],[virtual]

6.35.3 Member Function Documentation

6.35.3.1 [NOMAD::Point CMADS::ArrayXd2NOMADPoint](#) (const ArrayXd & x) [static]

Utility: converts ArrayXd to NOMAD::Point.

6.35.3.2 `int CMADS::evaluationSize () const [inline]`

Returns maximal number of evaluations.

6.35.3.3 `double CMADS::initaMeshSize () const [inline]`

Returns initial mesh size factor.

6.35.3.4 `int CMADS::initialLHSearch () const [inline]`

Returns initial value for LH search.

6.35.3.5 `int CMADS::iterationLHSearch () const [inline]`

Returns iteration value for LH search.

6.35.3.6 `double CMADS::minimalPollSize () const [inline]`

Returns minimal poll size factor.

6.35.3.7 `ArrayXd CMADS::NOMADPoint2ArrayXd (const NOMAD::Point & p) [static]`

Utility: converts NOMAD::Point to ArrayXd.

6.35.3.8 `ArrayXd CMADS::optimize (COptimizationProblem * _pOP, std::ostream & _out = std::cout)`

Solves the optimizations problem.

6.35.3.9 `int CMADS::progressStepsSize () const [inline]`

Returns progress steps size.

6.35.3.10 `template<class Archive > void CMADS::serialize (Archive & ar, const unsigned int version) [protected]`

6.35.3.11 `void CMADS::setEvaluationSize (int _maxeval) [inline]`

Sets maximal number of evaluations.

6.35.3.12 `void CMADS::setInitialLHSearch (int _lh0) [inline]`

Sets initial value for LH search.

6.35.3.13 `void CMADS::setInitialMeshSize (double _ims) [inline]`

Sets initial mesh size factor.

6.35.3.14 `void CMADS::setIterationLHSearch (int _lhi) [inline]`

Sets iteration value for LH search.

6.35.3.15 void CMADS::setMinimalPollSize (double *_mps*) [inline]

Sets minimal poll size factor.

6.35.4 Friends And Related Function Documentation

6.35.4.1 friend class boost::serialization::access [friend]

Boost serialization.

6.35.5 Member Data Documentation

6.35.5.1 double CMADS::ims [protected]

6.35.5.2 int CMADS::lh0 [protected]

6.35.5.3 int CMADS::lhi [protected]

6.35.5.4 int CMADS::maxeval [protected]

Holds maximal number of objective & constraints evaluations during optimization.

6.35.5.5 double CMADS::mps [protected]

6.35.5.6 COptimizationProblem* CMADS::pOP [protected]

Points to the optimization problem.

6.35.5.7 ArrayXd CMADS::Xbest [protected]

6.35.5.8 double CMADS::ybest [protected]

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

- C:/Development/core/MADSOptimization.h
- C:/Development/core/MADSOptimization.cpp

6.36 CMADSEvaluator Class Reference

Subclass of the NOMAD::Evaluator class.

```
#include <MADSOptimization.h>
```

Public Member Functions

- [CMADSEvaluator](#) (const NOMAD::Parameters &p, [COptimizationProblem](#) *_pOP)
- virtual [~CMADSEvaluator](#) ()
- virtual bool [eval_x](#) (NOMAD::Eval_Point &x, const NOMAD::Double &h_max, bool &count_eval) const
Overload of the NOMAD::Evaluator evaluation member function.

Protected Attributes

- [COptimizationProblem](#) * [pOP](#)
Points to the optimization problem.

6.36.1 Detailed Description

Subclass of the `NOMAD::Evaluator` class.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 `CMADSEvaluator::CMADSEvaluator (const NOMAD::Parameters & p, COptimizationProblem * _pOP)`
[inline]

6.36.2.2 `virtual CMADSEvaluator::~CMADSEvaluator ()` [inline],[virtual]

6.36.3 Member Function Documentation

6.36.3.1 `bool CMADSEvaluator::eval_x (NOMAD::Eval.Point & x, const NOMAD::Double & h_max, bool & count_eval) const`
[virtual]

Overload of the `NOMAD::Evaluator` evaluation member function.

6.36.4 Member Data Documentation

6.36.4.1 `COptimizationProblem* CMADSEvaluator::pOP` [protected]

Points to the optimization problem.

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

- `C:/Development/core/MADSOptimization.h`
- `C:/Development/core/MADSOptimization.cpp`

6.37 CMATLAB Class Reference

Class interface for Matlab's libmat and libmx dynamic libraries.

```
#include <MatlabLibrary.h>
```

Public Member Functions

- [CMATLAB](#) ()
- virtual [~CMATLAB](#) ()
- `MATFile * matOpen (const string &filename, const string &mode)`
Opens mat file using libmat.
- `bool matClose (MATFile *pmat)`
Closes mat file using libmat.
- `bool matPutVariable (MATFile *pmat, const string &name, const mxArray *pa)`
Puts variable in mat file using libmat.
- `mxArray * matGetVariable (MATFile *pmat, const string &name)`
Gets variable from mat file using libmat.

- `mxArray * mxCreateDoubleMatrix` (int N, int M)
Creates double matrix using libmx.
- `void mxDestroyArray` (mxArray *pa)
Destroys array using libmx.
- `int mxGetNumberOfElements` (mxArray *pa)
Returns number of elements in mxArray using libmx.
- `double * mxGetPr` (mxArray *pa)
Sets data in double matrix using libmx.
- `Engine * engOpenSingleUse` (const char *startcmd)
Opens engine for single use.
- `Engine * engOpen` (const string &startcmd)
Opens engine.
- `int engClose` (Engine *ep)
Closes engine.
- `int engGetVisible` (Engine *ep, bool *bVal)
Gets if engine is (in)visible.
- `int engSetVisible` (Engine *ep, bool newVal)
Sets engine to (in)visible.
- `int engEvalString` (Engine *ep, const string &str)
Gives to engine a string to evaluate.
- `mxArray * engGetVariable` (Engine *ep, const string &name)
Gets variable from engine.
- `int engPutVariable` (Engine *ep, const string &var_name, const mxArray *ap)
Puts variable into engine.
- `int engOutputBuffer` (Engine *ep, char *buffer, int buflen)
Sets engines output buffer.
- `void matPut` (string filename, const ArrayXd &X, string Xname)
Puts 1 array to .mat file.
- `void matGet` (string filename, ArrayXd &X, string Xname)
gets 1 array from .mat file.

Static Public Member Functions

- `static void SetPath` (const std::string &_path)
Sets path to the matlab lib.
- `static std::string & Path` ()
Returns path to the matlab lib.

Static Public Attributes

- `static std::string path` = "C:\\Program Files\\MATLAB\\R2010b\\bin\\win64"
Path to the matlab lib.

Private Attributes

- `QLibrary libmat`
QLibrary for libmat.
- `QLibrary libmx`
QLibrary for libmx.
- `QLibrary libeng`

- QLibrary for libeng.*
- [matftype1 pMatOpen](#)
 - Reference to matOpen function from libmat.*
- [matftype2 pMatClose](#)
 - Reference to matClose function from libmat.*
- [matftype3 pMatPutVariable](#)
 - Reference to matPutVariable function from libmat.*
- [matftype4 pMatGetVariable](#)
 - Reference to matGetVariable function from libmat.*
- [mxftype1 pMxCreateDoubleMatrix](#)
 - Reference to mxCreateDoubleMatrix function from libmx.*
- [mxftype2 pMxDestroyArray](#)
 - Reference to mxDestroyArray function from libmx.*
- [mxftype3 pMxGetPr](#)
 - Reference to mxGetPr function from libmx.*
- [mxftype4 pMxSetPr](#)
 - Reference to mxSetPr function from libmx.*
- [mxftype5 pMxGetNumberOfElements](#)
 - Reference to mxSetPr function from libmx.*
- [engftype1 pEngOpenSingleUse](#)
 - Reference to engOpenSingleUse function from libeng.*
- [engftype2 pEngOpen](#)
 - Reference to engOpen function from libeng.*
- [engftype3 pEngClose](#)
 - Reference to engClose function from libeng.*
- [engftype4 pEngGetVisible](#)
 - Reference to engGetVisible function from libeng.*
- [engftype5 pEngSetVisible](#)
 - Reference to engSetVisible function from libeng.*
- [engftype6 pEngEvalString](#)
 - Reference to engEvalString function from libeng.*
- [engftype7 pEngGetVariable](#)
 - Reference to engGetVariable function from libeng.*
- [engftype8 pEngPutVariable](#)
 - Reference to engPutVariable function from libeng.*
- [engftype9 pEngOutputBuffer](#)
 - Reference to engOutputBuffer function from libeng.*

6.37.1 Detailed Description

Class interface for Matlab's libmat and libmx dynamic libraries.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 `CMATLAB::CMATLAB ()`

6.37.2.2 `virtual CMATLAB::~~CMATLAB () [inline],[virtual]`

6.37.3 Member Function Documentation

6.37.3.1 `int CMATLAB::engClose (Engine * ep) [inline]`

Closes engine.

6.37.3.2 `int CMATLAB::engEvalString (Engine * ep, const string & str)` `[inline]`

Gives to engine a string to evaluate.

6.37.3.3 `mxArray* CMATLAB::engGetVariable (Engine * ep, const string & name)` `[inline]`

Gets variable from engine.

6.37.3.4 `int CMATLAB::engGetVisible (Engine * ep, bool * bVal)` `[inline]`

Gets if engine is (in)visible.

6.37.3.5 `Engine* CMATLAB::engOpen (const string & startcmd)` `[inline]`

Opens engine.

6.37.3.6 `Engine* CMATLAB::engOpenSingleUse (const char * startcmd)` `[inline]`

Opens engine for single use.

6.37.3.7 `int CMATLAB::engOutputBuffer (Engine * ep, char * buffer, int buflen)` `[inline]`

Sets engines output buffer.

6.37.3.8 `int CMATLAB::engPutVariable (Engine * ep, const string & var_name, const mxArray * ap)` `[inline]`

Puts variable into engine.

6.37.3.9 `int CMATLAB::engSetVisible (Engine * ep, bool newVal)` `[inline]`

Sets engine to (in)visible.

6.37.3.10 `bool CMATLAB::matClose (MATFile * pmat)` `[inline]`

Closes mat file using libmat.

6.37.3.11 `void CMATLAB::matGet (string filename, ArrayXd & X, string Xname)`

gets 1 array from .mat file.

6.37.3.12 `mxArray* CMATLAB::matGetVariable (MATFile * pmat, const string & name)` `[inline]`

Gets variable from mat file using libmat.

6.37.3.13 `MATFile* CMATLAB::matOpen (const string & filename, const string & mode)` `[inline]`

Opens mat file using libmat.

6.37.3.14 void CMATLAB::matPut (string *filename*, const ArrayXd & *X*, string *Xname*)

Puts 1 array to .mat file.

6.37.3.15 bool CMATLAB::matPutVariable (MATFile * *pmat*, const string & *name*, const mxArray * *pa*) [inline]

Puts variable in mat file using libmat.

6.37.3.16 mxArray* CMATLAB::mxCreateDoubleMatrix (int *N*, int *M*) [inline]

Creates double matrix using libmx.

6.37.3.17 void CMATLAB::mxDestroyArray (mxArray * *pa*) [inline]

Destroys array using libmx.

6.37.3.18 int CMATLAB::mxGetNumberOfElements (mxArray * *pa*) [inline]

Returns number of elements in mxArray using libmx.

6.37.3.19 double* CMATLAB::mxGetPr (mxArray * *pa*) [inline]

Sets data in double matrix using libmx.

6.37.3.20 std::string & CMATLAB::Path () [static]

Returns path to the matlab lib.

6.37.3.21 void CMATLAB::SetPath (const std::string & *_path*) [static]

Sets path to the matlab lib.

6.37.4 Member Data Documentation

6.37.4.1 QLibrary CMATLAB::libeng [private]

QLibrary for libeng.

6.37.4.2 QLibrary CMATLAB::libmat [private]

QLibrary for libmat.

6.37.4.3 QLibrary CMATLAB::libmx [private]

QLibrary for libmx.

6.37.4.4 std::string CMATLAB::path = "C:\\Program Files\\MATLAB\\R2010b\\bin\\win64" [static]

Path to the matlab lib.

6.37.4.5 engftype3 CMATLAB::pEngClose [private]

Reference to engClose function from libeng.

6.37.4.6 engftype6 CMATLAB::pEngEvalString [private]

Reference to engEvalString function from libeng.

6.37.4.7 engftype7 CMATLAB::pEngGetVariable [private]

Reference to engGetVariable function from libeng.

6.37.4.8 engftype4 CMATLAB::pEngGetVisible [private]

Reference to engGetVisible function from libeng.

6.37.4.9 engftype2 CMATLAB::pEngOpen [private]

Reference to engOpen function from libeng.

6.37.4.10 engftype1 CMATLAB::pEngOpenSingleUse [private]

Reference to engOpenSingleUse function from libeng.

6.37.4.11 engftype9 CMATLAB::pEngOutputBuffer [private]

Reference to engOutputBuffer function from libeng.

6.37.4.12 engftype8 CMATLAB::pEngPutVariable [private]

Reference to engPutVariable function from libeng.

6.37.4.13 engftype5 CMATLAB::pEngSetVisible [private]

Reference to engSetVisible function from libeng.

6.37.4.14 matftype2 CMATLAB::pMatClose [private]

Reference to matClose function from libmat.

6.37.4.15 matftype4 CMATLAB::pMatGetVariable [private]

Reference to matGetVariable function from libmat.

6.37.4.16 matftype1 CMATLAB::pMatOpen [private]

Reference to matOpen function from libmat.

6.37.4.17 `matftype3` `CMATLAB::pMatPutVariable` [private]

Reference to `matPutVariable` function from `libmat`.

6.37.4.18 `mxftype1` `CMATLAB::pMxCreateDoubleMatrix` [private]

Reference to `mxCreateDoubleMatrix` function from `libmx`.

6.37.4.19 `mxftype2` `CMATLAB::pMxDestroyArray` [private]

Reference to `mxDestroyArray` function from `libmx`.

6.37.4.20 `mxftype5` `CMATLAB::pMxGetNumberOfElements` [private]

Reference to `mxSetPr` function from `libmx`.

6.37.4.21 `mxftype3` `CMATLAB::pMxGetPr` [private]

Reference to `mxGetPr` function from `libmx`.

6.37.4.22 `mxftype4` `CMATLAB::pMxSetPr` [private]

Reference to `mxSetPr` function from `libmx`.

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

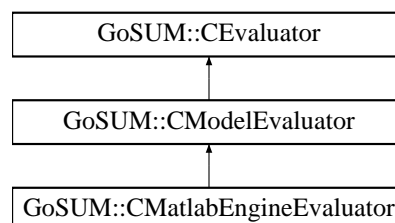
- [C:/Development/core/MatlabLibrary.h](#)
- [C:/Development/core/MatlabLibrary.cpp](#)

6.38 `GoSUM::CMatlabEngineEvaluator` Class Reference

Class for the evaluator through Matlab engine derived from [CModelEvaluator](#).

```
#include <OriginalModel.h>
```

Inheritance diagram for `GoSUM::CMatlabEngineEvaluator`:



Public Member Functions

- [CMatlabEngineEvaluator](#) ()
- [CMatlabEngineEvaluator](#) (const [CInputParameters](#) *_pIP, [COutputStates](#) *_pOS, int _trdI)
- virtual [~CMatlabEngineEvaluator](#) ()
- virtual void [openEvaluation](#) ()

Opens, i.e. prepares evaluation process.

- virtual ArrayXd [operator\(\)](#) (const ArrayXd &X)
Returns output state evaluated on a single input parameter n-tuple.
- virtual int [evaluateOutputsSize](#) ()
Evaluates outputs size by running model evaluator.
- virtual void [closeEvaluation](#) ()
Closes evaluation process.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Protected Attributes

- Engine * [ep](#)
Points to the matlab engine.
- [CMATLAB matlab](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.38.1 Detailed Description

Class for the evaluator through Matlab engine derived from [CModelEvaluator](#).

6.38.2 Constructor & Destructor Documentation

6.38.2.1 `GoSUM::CMatlabEngineEvaluator::CMatlabEngineEvaluator ()` `[inline]`

6.38.2.2 `GoSUM::CMatlabEngineEvaluator::CMatlabEngineEvaluator (const InputParameters * _pIP, COutputStates * _pOS, int _trdl)` `[inline]`

6.38.2.3 `virtual GoSUM::CMatlabEngineEvaluator::~~CMatlabEngineEvaluator ()` `[inline]`, `[virtual]`

6.38.3 Member Function Documentation

6.38.3.1 `void GoSUM::CMatlabEngineEvaluator::closeEvaluation ()` `[virtual]`

Closes evaluation process.

Implements [GoSUM::CModelEvaluator](#).

6.38.3.2 `int GoSUM::CMatlabEngineEvaluator::evaluateOutputsSize ()` `[virtual]`

Evaluates outputs size by running model evaluator.

Reimplemented from [GoSUM::CModelEvaluator](#).

6.38.3.3 `void GoSUM::CMatlabEngineEvaluator::openEvaluation () [virtual]`

Opens, i.e. prepares evaluation process.

Implements [GoSUM::CModelEvaluator](#).

6.38.3.4 `ArrayXd GoSUM::CMatlabEngineEvaluator::operator() (const ArrayXd & X) [virtual]`

Returns output state evaluated on a single input parameter n-tuple.

Implements [GoSUM::CModelEvaluator](#).

6.38.3.5 `template<class Archive > void GoSUM::CMatlabEngineEvaluator::serialize (Archive & ar, const unsigned int version) [protected]`

Reimplemented from [GoSUM::CModelEvaluator](#).

6.38.4 Friends And Related Function Documentation

6.38.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.38.5 Member Data Documentation

6.38.5.1 `Engine* GoSUM::CMatlabEngineEvaluator::ep [protected]`

Points to the matlab engine.

6.38.5.2 `CMATLAB GoSUM::CMatlabEngineEvaluator::matlab [protected]`

Object for matlab library functions.

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

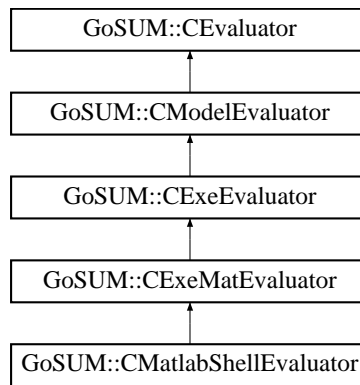
- C:/Development/core/[OriginalModel.h](#)
- C:/Development/core/[OriginalModel.cpp](#)

6.39 GoSUM::CMatlabShellEvaluator Class Reference

Class for the evaluator through Matlab engine and .mat i/o.

`#include <OriginalModel.h>`

Inheritance diagram for GoSUM::CMatlabShellEvaluator:



Public Member Functions

- [CMatlabShellEvaluator](#) ()
- [CMatlabShellEvaluator](#) (const [CInputParameters](#) * _pIP, [COutputStates](#) * _pOS, int _trdl)
- virtual [~CMatlabShellEvaluator](#) ()

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- virtual void [systemCommand](#) (std::ostream &cmd)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.39.1 Detailed Description

Class for the evaluator through Matlab engine and .mat i/o.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 [GoSUM::CMatlabShellEvaluator::CMatlabShellEvaluator \(\)](#) [[inline](#)]

6.39.2.2 [GoSUM::CMatlabShellEvaluator::CMatlabShellEvaluator \(const \[CInputParameters\]\(#\) * _pIP, \[COutputStates\]\(#\) * _pOS, int _trdl \)](#) [[inline](#)]

6.39.2.3 [virtual GoSUM::CMatlabShellEvaluator::~~CMatlabShellEvaluator \(\)](#) [[inline](#)], [[virtual](#)]

6.39.3 Member Function Documentation

6.39.3.1 [template<class Archive > void GoSUM::CMatlabShellEvaluator::serialize \(Archive & ar, const unsigned int version \)](#) [[protected](#)]

Reimplemented from [GoSUM::CExeMatEvaluator](#).

6.39.3.2 `virtual void GoSUM::CMatlabShellEvaluator::systemCommand (std::ostream & cmd) [inline], [protected], [virtual]`

Reimplemented from [GoSUM::CExeEvaluator](#).

6.39.4 Friends And Related Function Documentation

6.39.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

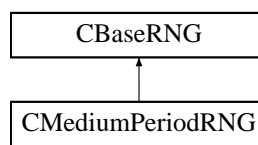
The documentation for this class was generated from the following files:

- C:/Development/core/[OriginalModel.h](#)
- C:/Development/core/[OriginalModel.cpp](#)

6.40 CMediumPeriodRNG Class Reference

```
#include <RandomGenerators.h>
```

Inheritance diagram for CMediumPeriodRNG:



Public Member Functions

- [CMediumPeriodRNG](#) ()
- [CMediumPeriodRNG](#) (unsigned int s)
- virtual [~CMediumPeriodRNG](#) ()
- virtual void [setSeed](#) (unsigned int s)
Sets seed of the RNG.
- virtual unsigned int [rndi](#) ()
Returns randomly generated unsigned int.
- virtual double [rnd](#) ()
Returns randomly generated double between 0 and 1.

Static Private Attributes

- static long [idum2](#) = 123456789
Parameters of the medium period RNG.
- static long [iy2](#) = 0
Parameters of the medium period RNG.
- static long [iv2](#) [NTAB]
Parameters of the medium period RNG.
- static long [idum](#) = 0
Parameters of the medium period RNG.

6.40.1 Constructor & Destructor Documentation

6.40.1.1 CMediumPeriodRNG::CMediumPeriodRNG () [inline]

6.40.1.2 CMediumPeriodRNG::CMediumPeriodRNG (unsigned int s) [inline]

6.40.1.3 virtual CMediumPeriodRNG::~~CMediumPeriodRNG () [inline],[virtual]

6.40.2 Member Function Documentation

6.40.2.1 double CMediumPeriodRNG::rnd () [virtual]

Returns randomly generated double between 0 and 1.

Implements [CBaseRNG](#).

6.40.2.2 virtual unsigned int CMediumPeriodRNG::rmdi () [inline],[virtual]

Returns randomly generated unsigned int.

Implements [CBaseRNG](#).

6.40.2.3 void CMediumPeriodRNG::setSeed (unsigned int s) [virtual]

Sets seed of the RNG.

Implements [CBaseRNG](#).

6.40.3 Member Data Documentation

6.40.3.1 long CMediumPeriodRNG::idum = 0 [static],[private]

Paramters of the medium period RNG.

6.40.3.2 long CMediumPeriodRNG::idum2 = 123456789 [static],[private]

Paramters of the medium period RNG.

6.40.3.3 long CMediumPeriodRNG::iv2 [static],[private]

Paramters of the medium period RNG.

6.40.3.4 long CMediumPeriodRNG::iy2 = 0 [static],[private]

Paramters of the medium period RNG.

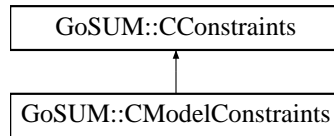
The documentation for this class was generated from the following files:

- C:/Development/core/[RandomGenerators.h](#)
- C:/Development/core/[RandomGenerators.cpp](#)

6.41 GoSUM::CModelConstraints Class Reference

```
#include <Constraints.h>
```

Inheritance diagram for GoSUM::CModelConstraints:



Public Member Functions

- [CModelConstraints](#) ()
- virtual [~CModelConstraints](#) ()
- virtual void [clear](#) ()
Clears object.
- void [set](#) ([CInputParameters](#) *_pIP)
Sets input parameters pointer.
- bool [isConstrained](#) (int _at) const
Returns status (constrained or not) for particular input parameter.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- virtual void [setVariableNames](#) ()
Sets variable names for the parser.

Protected Attributes

- [CInputParameters](#) * [pIP](#)
Points to input parameters.
- std::vector< bool > [status](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.41.1 Constructor & Destructor Documentation

6.41.1.1 [GoSUM::CModelConstraints::CModelConstraints \(\)](#) [inline]

6.41.1.2 virtual [GoSUM::CModelConstraints::~~CModelConstraints \(\)](#) [inline],[virtual]

6.41.2 Member Function Documentation

6.41.2.1 virtual void [GoSUM::CModelConstraints::clear \(\)](#) [inline],[virtual]

Clears object.

Reimplemented from [GoSUM::CConstraints](#).

6.41.2.2 `bool GoSUM::CModelConstraints::isConstrained (int _at) const` `[inline]`

Returns status (cosntrained or not) for particular input parameter.

6.41.2.3 `template<class Archive > void GoSUM::CModelConstraints::serialize (Archive & ar, const unsigned int version)`
`[protected]`

Reimplemented from [GoSUM::CConstraints](#).

6.41.2.4 `void GoSUM::CModelConstraints::set (CInputParameters * _pIP)`

Sets input parameters pointer.

6.41.2.5 `void GoSUM::CModelConstraints::setVariableNames ()` `[protected]`, `[virtual]`

Sets variable names for the parser.

Implements [GoSUM::CConstraints](#).

6.41.3 Friends And Related Function Documentation

6.41.3.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.41.4 Member Data Documentation

6.41.4.1 `CInputParameters* GoSUM::CModelConstraints::pIP` `[protected]`

Points to input parameters.

6.41.4.2 `std::vector<bool> GoSUM::CModelConstraints::status` `[protected]`

Status (constrained or not) for each input parameter.

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

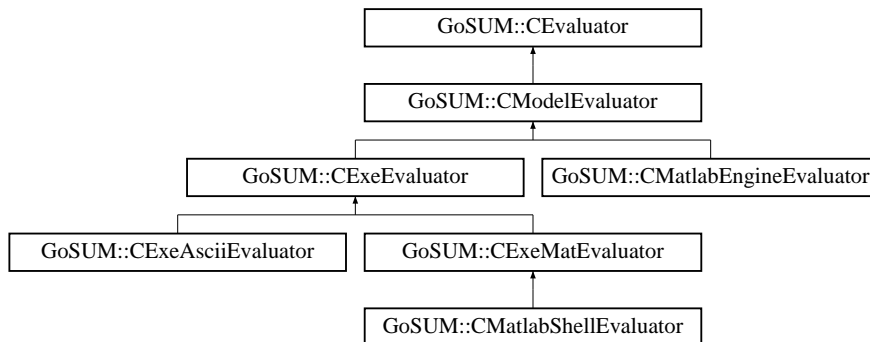
- C:/Development/core/[Constraints.h](#)
- C:/Development/core/[Constraints.cpp](#)

6.42 GoSUM::CModelEvaluator Class Reference

Class for [GoSUM](#) model evaluator.

```
#include <OriginalModel.h>
```

Inheritance diagram for GoSUM::CModelEvaluator:



Public Member Functions

- [CModelEvaluator](#) ()
- [CModelEvaluator](#) (const [CInputParameters](#) * _pIP, [COutputStates](#) * _pOS, int _trdl)
- virtual [~CModelEvaluator](#) ()
- virtual void [openEvaluation](#) ()=0
Opens, i.e. prepares evaluation process.
- virtual [ArrayXd operator\(\)](#) (const [ArrayXd](#) &X)=0
Returns output state evaluated on a single input parameter n-tuple.
- virtual void [operator\(\)](#) ()
Thread calls this operator which then evaluates output states from every trdl-th input parameter n-tuple.
- virtual int [evaluateOutputsSize](#) ()
Evaluates outputs size by running model evaluator.
- virtual void [closeEvaluation](#) ()=0
Closes evaluation process.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Protected Attributes

- const [CInputParameters](#) * pIP
Pointer to [GoSUM](#) input parameters.
- [COutputStates](#) * pOS
Pointer to [GoSUM](#) output states.
- int trdl
- int I

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.42.1 Detailed Description

Class for [GoSUM](#) model evaluator.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 GoSUM::CModelEvaluator::CModelEvaluator () [inline]

6.42.2.2 GoSUM::CModelEvaluator::CModelEvaluator (const CInputParameters * *_pIP*, COutputStates * *_pOS*, int *_trdI*) [inline]

6.42.2.3 virtual GoSUM::CModelEvaluator::~~CModelEvaluator () [inline],[virtual]

6.42.3 Member Function Documentation

6.42.3.1 virtual void GoSUM::CModelEvaluator::closeEvaluation () [pure virtual]

Closes evaluation process.

Implemented in [GoSUM::CMatlabEngineEvaluator](#), and [GoSUM::CExeEvaluator](#).

6.42.3.2 virtual int GoSUM::CModelEvaluator::evaluateOutputsSize () [inline],[virtual]

Evaluates outputs size by running model evaluator.

Reimplemented in [GoSUM::CMatlabEngineEvaluator](#), and [GoSUM::CExeEvaluator](#).

6.42.3.3 virtual void GoSUM::CModelEvaluator::openEvaluation () [pure virtual]

Opens, i.e. prepares evaluation process.

Implemented in [GoSUM::CMatlabEngineEvaluator](#), and [GoSUM::CExeEvaluator](#).

6.42.3.4 virtual ArrayXd GoSUM::CModelEvaluator::operator() (const ArrayXd & *X*) [pure virtual]

Returns output state evaluated on a single input parameter n-tuple.

Implemented in [GoSUM::CMatlabEngineEvaluator](#), and [GoSUM::CExeEvaluator](#).

6.42.3.5 void GoSUM::CModelEvaluator::operator() () [virtual]

Thread calls this operator which then evaluates output states from every *trdI*-th input parameter n-tuple.

6.42.3.6 template<class Archive > void GoSUM::CModelEvaluator::serialize (Archive & *ar*, const unsigned int *version*) [protected]

Reimplemented from [GoSUM::CEvaluator](#).

Reimplemented in [GoSUM::CMatlabEngineEvaluator](#), [GoSUM::CMatlabShellEvaluator](#), [GoSUM::CExeMatEvaluator](#), [GoSUM::CExeAsciiEvaluator](#), and [GoSUM::CExeEvaluator](#).

6.42.4 Friends And Related Function Documentation

6.42.4.1 friend class boost::serialization::access [friend]

Boost serialization.

6.42.5 Member Data Documentation

6.42.5.1 `int GoSUM::CModelEvaluator::l` `[protected]`

Total number of threads and id of the particular thread this class runs in.

6.42.5.2 `const CInputParameters* GoSUM::CModelEvaluator::pIP` `[protected]`

Pointer to [GoSUM](#) input parameters.

6.42.5.3 `COutputStates* GoSUM::CModelEvaluator::pOS` `[protected]`

Pointer to [GoSUM](#) output states.

6.42.5.4 `int GoSUM::CModelEvaluator::trdl` `[protected]`

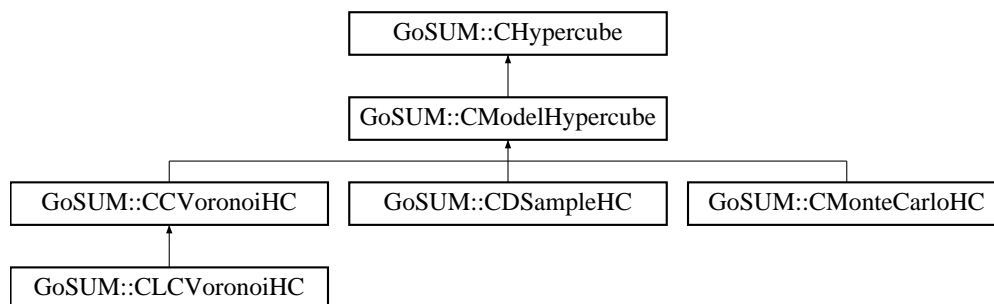
The documentation for this class was generated from the following files:

- C:/Development/core/[OriginalModel.h](#)
- C:/Development/core/[OriginalModel.cpp](#)

6.43 GoSUM::CModelHypercube Class Reference

```
#include <Hypercube.h>
```

Inheritance diagram for GoSUM::CModelHypercube:



Public Member Functions

- [CModelHypercube](#) ([CInputParameters](#) *_pIP)
- virtual [~CModelHypercube](#) ()
- virtual void [generate](#) (int _rssize, int _dim, std::vector< [ArrayXd](#) > &_samples)
Generates _samples: _rssize model points (satisfying constraints) of dimension _dim.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- [ArrayXd](#) & [generateHCPPoint](#) ([ArrayXd](#) &x)
- void [generateHCPPoints](#) (int N, int dim, std::vector< [ArrayXd](#) > &y)
Generates N hypercube points.

- void [generateModelPoints](#) (int N, int dim, std::vector< ArrayXd > &y)
Generates N model points.
- void [checkHCPoints](#) (std::vector< ArrayXd > &y)
Checks if hypercube points when mapped to model points satisfy constraints.
- void [hcPoints2ModelPoints](#) (std::vector< ArrayXd > &y)
Maps hypercube points to model points.
- virtual void [doGenerate](#) (int _rssize, int _dim, std::vector< ArrayXd > &_samples)=0
Core of the generation.
- [CModelHypercube](#) ()

Protected Attributes

- [CInputParameters](#) * [pIP](#)
Holds pointer to input parameters.
- [CModelConstraints](#) * [pIC](#)
Holds pointer to input parameter constraints.
- int [maxtries](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.43.1 Constructor & Destructor Documentation

- 6.43.1.1 [GoSUM::CModelHypercube::CModelHypercube \(\)](#) `[inline]`, `[protected]`
- 6.43.1.2 [GoSUM::CModelHypercube::CModelHypercube \(CInputParameters * _pIP \)](#) `[inline]`
- 6.43.1.3 [virtual GoSUM::CModelHypercube::~CModelHypercube \(\)](#) `[inline]`, `[virtual]`

6.43.2 Member Function Documentation

- 6.43.2.1 [void GoSUM::CModelHypercube::checkHCPoints \(std::vector< ArrayXd > & y \)](#) `[protected]`

Checks if hypercube points when mapped to model points satisfy constraints.

- 6.43.2.2 [virtual void GoSUM::CModelHypercube::doGenerate \(int _rssize, int _dim, std::vector< ArrayXd > & _samples \)](#)
`[protected]`, `[pure virtual]`

Core of the generation.

Implemented in [GoSUM::CLCVoronoiHC](#), [GoSUM::CCVoronoiHC](#), [GoSUM::CMonteCarloHC](#), and [GoSUM::CD-SampleHC](#).

- 6.43.2.3 [void GoSUM::CModelHypercube::generate \(int _rssize, int _dim, std::vector< ArrayXd > & _samples \)](#)
`[virtual]`

Generates _samples: _rssize model points (satisfying constraints) of dimension _dim.

6.43.2.4 `ArrayXd& GoSUM::CModelHypercube::generateHCPPoint (ArrayXd & x)` `[inline]`, `[protected]`

Parameters

<code>x</code>	Generates single hypercube point.
----------------	-----------------------------------

6.43.2.5 `void GoSUM::CModelHypercube::generateHCPoints (int N, int dim, std::vector< ArrayXd > & y)`
`[protected]`

Generates N hypercube points.

6.43.2.6 `void GoSUM::CModelHypercube::generateModelPoints (int N, int dim, std::vector< ArrayXd > & y)`
`[protected]`

Generates N model points.

6.43.2.7 `void GoSUM::CModelHypercube::hcPoints2ModelPoints (std::vector< ArrayXd > & y)` `[protected]`

Maps hypercube points to model points.

6.43.2.8 `template<class Archive > void GoSUM::CModelHypercube::serialize (Archive & ar, const unsigned int version)`
`[protected]`

Reimplemented from [GoSUM::CHypercube](#).

Reimplemented in [GoSUM::CLCVoronoiHC](#), [GoSUM::CCVoronoiHC](#), and [GoSUM::CMonteCarloHC](#).

6.43.3 Friends And Related Function Documentation

6.43.3.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.43.4 Member Data Documentation

6.43.4.1 `int GoSUM::CModelHypercube::maxtries` `[protected]`

Holds maximal number of tries to satisfy model constraints.

6.43.4.2 `CModelConstraints* GoSUM::CModelHypercube::pIC` `[protected]`

Holds pointer to input parameter constraints.

6.43.4.3 `CInputParameters* GoSUM::CModelHypercube::pIP` `[protected]`

Holds pointer to input parameters.

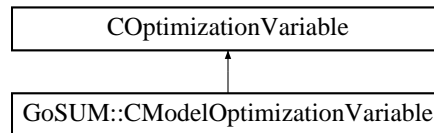
The documentation for this class was generated from the following files:

- [C:/Development/core/Hypercube.h](#)
- [C:/Development/core/Hypercube.cpp](#)

6.44 GoSUM::CModelOptimizationVariable Class Reference

```
#include <ParserOptimizationProblem.h>
```

Inheritance diagram for GoSUM::CModelOptimizationVariable:



Public Member Functions

- [CModelOptimizationVariable](#) ([CModelVariable](#) * _pip)
- [CModelOptimizationVariable](#) ([CModelOptimizationVariable](#) & _O)
- virtual [~CModelOptimizationVariable](#) ()
- [CModelVariable](#) * [inputParameter](#) () const
Returns pointer to the input parameter to which model optimization variable is connected.
- void [reset](#) ()
Resets lower bound, upper bound and initial value.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- [CModelOptimizationVariable](#) ()

Protected Attributes

- [CModelVariable](#) * [pip](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.44.1 Constructor & Destructor Documentation

- 6.44.1.1 [GoSUM::CModelOptimizationVariable::CModelOptimizationVariable \(\)](#) [inline], [protected]
- 6.44.1.2 [GoSUM::CModelOptimizationVariable::CModelOptimizationVariable \(CModelVariable * _pip \)](#) [inline]
- 6.44.1.3 [GoSUM::CModelOptimizationVariable::CModelOptimizationVariable \(CModelOptimizationVariable & _O \)](#) [inline]
- 6.44.1.4 [virtual GoSUM::CModelOptimizationVariable::~~CModelOptimizationVariable \(\)](#) [inline], [virtual]

6.44.2 Member Function Documentation

- 6.44.2.1 [CModelVariable*](#) [GoSUM::CModelOptimizationVariable::inputParameter \(\)](#) const [inline]

Returns pointer to the input parameter to which model optimization variable is connected.

6.44.2.2 `void GoSUM::CModelOptimizationVariable::reset () [inline]`

Resets lower bound, upper bound and initial value.

6.44.2.3 `template<class Archive > void GoSUM::CModelOptimizationVariable::serialize (Archive & ar, const unsigned int version) [protected]`

Reimplemented from [COptimizationVariable](#).

6.44.3 Friends And Related Function Documentation

6.44.3.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.44.4 Member Data Documentation

6.44.4.1 `CModelVariable* GoSUM::CModelOptimizationVariable::pip [protected]`

Holds pointer to the input parameter to which model optimization variable is connected.

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

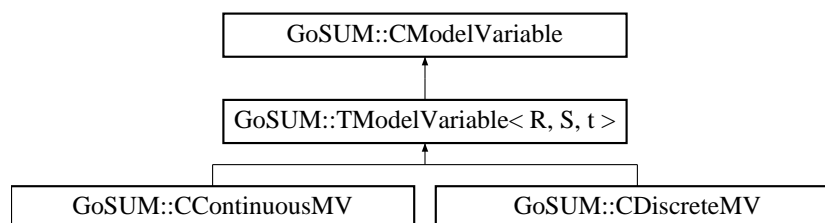
- C:/Development/core/[ParserOptimizationProblem.h](#)
- C:/Development/core/[ParserOptimizationProblem.cpp](#)

6.45 GoSUM::CModelVariable Class Reference

Class for any variable in the [GoSUM](#) model (input or output).

```
#include <ModelVariable.h>
```

Inheritance diagram for GoSUM::CModelVariable:



Public Member Functions

- [CModelVariable](#) ()
- virtual [~CModelVariable](#) ()
- [CModelVariable](#) * [clone](#) (std::string _uname)
Clones itself to a new model variable with different name.
- void [setName](#) (const std::string &_uname)
Set name of the model variable.
- std::string [name](#) () const
Returns name of the model variable.

- virtual void `clearSample` ()=0
Clears variable's sample.
- virtual int `expandedSize` () const =0
- virtual void `setSampleSize` (int _n)=0
Returns number of variables after expansion for analytical model.
- virtual int `sampleSize` () const =0
Returns sample size.
- virtual void `setSampleValue` (double _val, int _at)=0
Sets particular sample value.
- virtual double `sampleValue` (int _at) const =0
Returns particular sample value.
- virtual void `readSampleValue` (std::ifstream &_ifs, int _at)=0
Reads particular sample value from input file stream.
- virtual void `writeSampleValue` (std::ofstream &_ofs, int _at) const =0
Writes particular sample value to output file stream.
- virtual void `generateSampleValue` (double _ranval, int _at)=0
Generates particular sample value using actual random variable model.
- virtual double `generateSampleValue` (double _p)=0
Generates sample value using actual random variable model.
- virtual void `computeStatistics` (int _n)=0
Computes statistics from sample (histogram, empirical mean and standard deviation etc.).
- virtual double `minValue` () const =0
Returns minimal variable value.
- virtual double `maxValue` () const =0
Returns maximal variable value.
- virtual double `expectedValue` () const =0
Returns expected variable value.
- virtual void `setEmpiricalDistribution` ()
Computes distribution of the actual random variable from empirical sample data.
- virtual void `setTheoreticalDistribution` ()
Tests goodness-of-fit for theoretical random variables and replaces with the best that satisfies Kolomogorov-Smirnov.
- virtual void `setDistribution` ()
Computes distribution of the actual random variable from available data.
- virtual
`CRandomVariable::distributiontype distributionType` () const =0
Returns type of the random variable distribution.
- virtual std::string `distributionName` () const =0
Returns name of the random variable distribution.
- virtual bool `isContinuous` () const =0
Returns true if model variable is continuous, false otherwise.
- virtual bool `isDiscrete` () const =0
Returns true if model variable is discrete, false otherwise.
- virtual bool `isDistributionDefined` () const =0
Returns true if distribution is defined, false otherwise.
- `CContinuousMV * cast2ContinuousMV` ()
Safe cast of model variable to CContinuousMV type.
- `CDiscreteMV * cast2DiscreteMV` ()
Safe cast of model variable to CDiscreteMV type.
- virtual `CRandomVariable * randomVariable` () const =0
Returns random variable.
- virtual `CSample * sample` () const =0
Returns sample.
- `ArrayXd castExportSample` ()
Exports model variables sample and casts it to array of doubles.

Protected Member Functions

- `template<class Archive >`
void `serialize` (Archive &ar, const unsigned int version)

Protected Attributes

- `std::string` `uname`

Friends

- class `boost::serialization::access`
Boost serialization.

6.45.1 Detailed Description

Class for any variable in the `GoSUM` model (input or output).

6.45.2 Constructor & Destructor Documentation

6.45.2.1 `GoSUM::CModelVariable::CModelVariable ()` [`inline`]

6.45.2.2 `virtual GoSUM::CModelVariable::~~CModelVariable ()` [`inline`], [`virtual`]

6.45.3 Member Function Documentation

6.45.3.1 `GoSUM::CContinuousMV * GoSUM::CModelVariable::cast2ContinuousMV ()`

Safe cast of model variable to `CContinuousMV` type.

6.45.3.2 `GoSUM::CDiscreteMV * GoSUM::CModelVariable::cast2DiscreteMV ()`

Safe cast of model variable to `CDiscreteMV` type.

6.45.3.3 `ArrayXd GoSUM::CModelVariable::castExportSample ()`

Exports model variables sample and casts it to array of doubles.

6.45.3.4 `virtual void GoSUM::CModelVariable::clearSample ()` [`pure virtual`]

Clears variable's sample.

Implemented in `GoSUM::TModelVariable< R, S, t >`.

6.45.3.5 `GoSUM::CModelVariable * GoSUM::CModelVariable::clone (std::string _uname)`

Clones itself to a new model variable with different name.

6.45.3.6 `virtual void GoSUM::CModelVariable::computeStatistics (int _n) [pure virtual]`

Computes statistics from sample (histogram, empirical mean and standard deviation etc.).

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.7 `virtual std::string GoSUM::CModelVariable::distributionName () const [pure virtual]`

Returns name of the random variable distribution.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.8 `virtual CRandomVariable::distributiontype GoSUM::CModelVariable::distributionType () const [pure virtual]`

Returns type of the random variable distribution.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.9 `virtual int GoSUM::CModelVariable::expandedSize () const [pure virtual]`

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.10 `virtual double GoSUM::CModelVariable::expectedValue () const [pure virtual]`

Returns expected variable value.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.11 `virtual void GoSUM::CModelVariable::generateSampleValue (double _ranval, int _at) [pure virtual]`

Generates particular sample value using actual random variable model.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.12 `virtual double GoSUM::CModelVariable::generateSampleValue (double _p) [pure virtual]`

Generates sample value using actual random variable model.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.13 `virtual bool GoSUM::CModelVariable::isContinuous () const [pure virtual]`

Returns true if model variable is continuous, false otherwise.

Implemented in [GoSUM::CContinuousMV](#), and [GoSUM::CDiscreteMV](#).

6.45.3.14 `virtual bool GoSUM::CModelVariable::isDiscrete () const [pure virtual]`

Returns true if model variable is discrete, false otherwise.

Implemented in [GoSUM::CContinuousMV](#), and [GoSUM::CDiscreteMV](#).

6.45.3.15 `virtual bool GoSUM::CModelVariable::isDistributionDefined () const [pure virtual]`

Returns true if distribution is defined, false otherwise.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.16 `virtual double GoSUM::CModelVariable::maxValue () const [pure virtual]`

Returns maximal variable value.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.17 `virtual double GoSUM::CModelVariable::minValue () const [pure virtual]`

Returns minimal variable value.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.18 `std::string GoSUM::CModelVariable::name () const [inline]`

Returns name of the model variable.

6.45.3.19 `virtual CRandomVariable* GoSUM::CModelVariable::randomVariable () const [pure virtual]`

Returns random variable.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.20 `virtual void GoSUM::CModelVariable::readSampleValue (std::ifstream & _ifs, int _at) [pure virtual]`

Reads particular sample value from input file stream.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.21 `virtual CSample* GoSUM::CModelVariable::sample () const [pure virtual]`

Returns sample.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.22 `virtual int GoSUM::CModelVariable::sampleSize () const [pure virtual]`

Returns sample size.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.23 `virtual double GoSUM::CModelVariable::sampleValue (int _at) const [pure virtual]`

Returns particular sample value.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.24 `template<class Archive > void GoSUM::CModelVariable::serialize (Archive & ar, const unsigned int version)
[inline], [protected]`

Reimplemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.25 `virtual void GoSUM::CModelVariable::setDistribution () [inline],[virtual]`

Computes distribution of the actual random variable from available data.

6.45.3.26 `virtual void GoSUM::CModelVariable::setEmpiricalDistribution () [inline],[virtual]`

Computes distribution of the actual random variable from empirical sample data.

Reimplemented in [GoSUM::CContinuousMV](#), [GoSUM::CDiscreteMV](#), and [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.27 `void GoSUM::CModelVariable::setName (const std::string & _uname) [inline]`

Set name of the model variable.

6.45.3.28 `virtual void GoSUM::CModelVariable::setSampleSize (int _n) [pure virtual]`

Returns number of variables after expansion for analytical model.

Sets sample size.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.29 `virtual void GoSUM::CModelVariable::setSampleValue (double _val, int _at) [pure virtual]`

Sets particular sample value.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.30 `virtual void GoSUM::CModelVariable::setTheoreticalDistribution () [inline],[virtual]`

Tests goodness-of-fit for theoretical random variables and replaces with the best that satisfies Kolomogorov--Smirnov.

Reimplemented in [GoSUM::CContinuousMV](#), [GoSUM::CDiscreteMV](#), and [GoSUM::TModelVariable< R, S, t >](#).

6.45.3.31 `virtual void GoSUM::CModelVariable::writeSampleValue (std::ofstream & _ofs, int _at) const [pure virtual]`

Writes particular sample value to output file stream.

Implemented in [GoSUM::TModelVariable< R, S, t >](#).

6.45.4 Friends And Related Function Documentation

6.45.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.45.5 Member Data Documentation

6.45.5.1 `std::string GoSUM::CModelVariable::uname [protected]`

Unique name of the model variable.

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

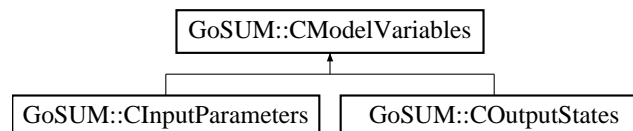
- [C:/Development/core/ModelVariable.h](#)
- [C:/Development/core/ModelVariable.cpp](#)

6.46 GoSUM::CModelVariables Class Reference

Class for the vector of model variables (inputs & outputs).

```
#include <Model.h>
```

Inheritance diagram for GoSUM::CModelVariables:



Public Member Functions

- [CModelVariables](#) ()
- virtual [~CModelVariables](#) ()
- [std::string baseName](#) () const
Returns base name.
- void [setBaseName](#) (std::string _basename)
Sets base name.
- [std::string nextName](#) () const
Returns next variable name.
- bool [empty](#) () const
Returns true if object contains no model variables, false otherwise.
- bool [emptySamples](#) () const
Returns true if object contains no samples, false otherwise.
- void [clear](#) ()
- void [clearSamples](#) ()
Clears data.
- const [CModelVariable & operator\(\)](#) (int _at) const
Clears samples.
- [CModelVariable & operator\(\)](#) (int _at)
Returns paticular model variable.
- int [find](#) (const std::string &_name) const
< Finds model variable by name, returns it's index.
- int [size](#) () const
Returns the size of model variables.
- int [expandedSize](#) () const
Returns expanded size of model variables (due to categorical).
- int [sampleSize](#) () const
Returns sample size.
- [CModelVariable * add](#) ([CModelVariable * _pMV](#))
Adds, i.e. pushes back model variable.
- [CModelVariable * insert](#) ([CModelVariable * _pMV](#), int _before)
- void [erase](#) (int _at)
- [CModelVariable * replace](#) (int _at, [CModelVariable * _pMV](#))
- void [rename](#) (int _at, const std::string &_name)

- `std::vector< ArrayXd > samples ()`
Returns all samples.
- `void setSamples (const std::vector< ArrayXd > &_samples)`
Sets samples of all model variables to values in _samples.
- `void computeStatistics (int _n)`
Computes statistics of all model variables.
- `void computeStatistics (const std::vector< int > &_selected, int _n)`
- `void setDistributions (const std::vector< int > &_selected)`
For selected model variables detects distribution from empirical sample data.
- `void setDistributions ()`
For all model variables detects distribution from empirical sample data.
- `void setNTuple (const ArrayXd &X, int _at)`
Sets _at sample values of all model variables.
- `ArrayXd nTuple (int _at) const`
Returns n-tuple containing _at sample values of all model variables.
- `ArrayXd expandedNTuple (int _at) const`
Returns expanded n-tuple containing _at sample values of all model variables.
- `ArrayXd expandNTuple (const ArrayXd &X) const`
Expands given n-tuple.
- `void setNormalization ()`
- `double normalize (double Vi, int i) const`
(Re)sets normalization of model variables using empirical data in samples.
- `ArrayXd normalize (const ArrayXd &V) const`
Normalizes values of all (expanded) model variables.
- `double denormalize (double Vi, int i) const`
Denormalizes value of the ith (expanded) model variable.
- `ArrayXd denormalize (const ArrayXd &V) const`
Denormalizes values of all (expanded) model variables.
- `double dNormalize (int i) const`
Returns derivative of the normalization function for ith (expanded) model variable.
- `ArrayXd dNormalize () const`
Returns derivatives of the normalization function for (expanded) model variables.
- `double dDenormalize (int i) const`
Returns derivative of the denormalization function for ith (expanded) model variable.
- `ArrayXd dDenormalize () const`
Returns derivatives of the denormalization function for (expanded) model variables.
- `ArrayXd hcPoint2ModelPoint (const ArrayXd &x)`
Maps hypercube point to model point.
- `void eraseSelectedSamples (const std::vector< int > &sel)`

Protected Member Functions

- `template<class Archive >`
`void serialize (Archive &ar, const unsigned int version)`

Static Protected Member Functions

- `static bool isModelVariableName (const CModelVariable &_aMV)`
Used in member function Find.

Protected Attributes

- `std::string` [basename](#)
Holds basename used for generating model variable unique names.
- `boost::ptr_vector< CModelVariable >` [mvs](#)
Holds pointers to model variables.
- `ArrayXd` [trn](#)
- `ArrayXd` [scl](#)

Static Protected Attributes

- `static std::string` [nameToFind](#) = ""
Used in member function Find.

Friends

- `class` [boost::serialization::access](#)
Boost serialization.

6.46.1 Detailed Description

Class for the vector of model variables (inputs & outputs).

6.46.2 Constructor & Destructor Documentation

6.46.2.1 `GoSUM::CModelVariables::CModelVariables ()` `[inline]`

6.46.2.2 `virtual GoSUM::CModelVariables::~~CModelVariables ()` `[inline]`, `[virtual]`

6.46.3 Member Function Documentation

6.46.3.1 `CModelVariable* GoSUM::CModelVariables::add (CModelVariable * _pMV)` `[inline]`

Adds, i.e. pushes back model variable.

6.46.3.2 `std::string GoSUM::CModelVariables::baseName () const` `[inline]`

Returns base name.

6.46.3.3 `void GoSUM::CModelVariables::clear ()` `[inline]`

6.46.3.4 `void GoSUM::CModelVariables::clearSamples ()` `[inline]`

Clears data.

6.46.3.5 `void GoSUM::CModelVariables::computeStatistics (int _n)` `[inline]`

Computes statistics of all model variables.

6.46.3.6 `void GoSUM::CModelVariables::computeStatistics (const std::vector< int > & _selected, int _n)` `[inline]`

Parameters

<code><i>_n</i></code>	Computes statistics of selected model variables.
------------------------	--

6.46.3.7 `double GoSUM::CModelVariables::dDenormalize (int i) const` `[inline]`

Returns derivative of the denormalization function for *ith* (expanded) model variable.

6.46.3.8 `ArrayXd GoSUM::CModelVariables::dDenormalize () const` `[inline]`

Returns derivatives of the denormalization function for (expanded) model variables.

6.46.3.9 `double GoSUM::CModelVariables::denormalize (double Vi, int i) const` `[inline]`

Denoramlizes value of the *ith* (expanded) model variable.

6.46.3.10 `ArrayXd GoSUM::CModelVariables::denormalize (const ArrayXd & V) const` `[inline]`

Denormalizes values of all (expanded) model variables.

6.46.3.11 `double GoSUM::CModelVariables::dNormalize (int i) const` `[inline]`

Returns derivative of the normalization function for *ith* (expanded) model variable.

6.46.3.12 `ArrayXd GoSUM::CModelVariables::dNormalize () const` `[inline]`

Returns derivatives of the normalization function for (expanded) model variables.

6.46.3.13 `bool GoSUM::CModelVariables::empty () const` `[inline]`

Returns true if object contains no model variables, false otherwise.

6.46.3.14 `bool GoSUM::CModelVariables::emptySamples () const` `[inline]`

Returns true if object contains no samples, false otherwise.

6.46.3.15 `void GoSUM::CModelVariables::erase (int _at)` `[inline]`

Parameters

<code><i>_at</i></code>	Erases particular model variable.
-------------------------	-----------------------------------

6.46.3.16 `void GoSUM::CModelVariables::eraseSelectedSamples (const std::vector< int > & sel)` `[inline]`

Parameters

<code><i>sel</i></code>	Erases selected sample values.
-------------------------	--------------------------------

6.46.3.17 `ArrayXd GoSUM::CModelVariables::expandedNTuple (int _at) const`

Returns expanded n-tuple containing `_at` sample values of all model variables.

6.46.3.18 `int GoSUM::CModelVariables::expandedSize () const`

Returns expanded size of model variables (due to categorical).

6.46.3.19 `ArrayXd GoSUM::CModelVariables::expandNTuple (const ArrayXd & X) const`

Expands given n-tuple.

6.46.3.20 `int GoSUM::CModelVariables::find (const std::string & _name) const` `[inline]`

< Finds model variable by name, returns it's index.

6.46.3.21 `ArrayXd GoSUM::CModelVariables::hcPoint2ModelPoint (const ArrayXd & x)`

Maps hypercube point to model point.

6.46.3.22 `CModelVariable* GoSUM::CModelVariables::insert (CModelVariable * _pMV, int _before)` `[inline]`

Parameters

<code>_before</code>	Inserts model variable in position <code>_before</code> .
----------------------	---

6.46.3.23 `static bool GoSUM::CModelVariables::isModelVariableName (const CModelVariable & _aMV)` `[inline]`,
`[static]`, `[protected]`

Used in member function Find.

6.46.3.24 `std::string GoSUM::CModelVariables::nextName () const`

Returns next variable name.

6.46.3.25 `double GoSUM::CModelVariables::normalize (double Vi, int i) const` `[inline]`

(Re)sets normalization of model variables using empirical data in samples.

Normalizes value of the `i`th (expanded) model variable.

6.46.3.26 `ArrayXd GoSUM::CModelVariables::normalize (const ArrayXd & V) const` `[inline]`

Normalizes values of all (expanded) model variables.

6.46.3.27 `ArrayXd GoSUM::CModelVariables::nTuple (int _at) const`

Returns n-tuple containing `_at` sample values of all model variables.

6.46.3.28 `const CModelVariable& GoSUM::CModelVariables::operator()(int _at) const` `[inline]`

Clears samples.

Returns particular model variable.

6.46.3.29 `CModelVariable& GoSUM::CModelVariables::operator()(int _at)` `[inline]`

Returns particular model variable.

6.46.3.30 `void GoSUM::CModelVariables::rename (int _at, const std::string & _name)` `[inline]`

Parameters

<i>_name</i>	Renames particular model variable.
--------------	------------------------------------

6.46.3.31 `CModelVariable* GoSUM::CModelVariables::replace (int _at, CModelVariable * _pMV)` `[inline]`

Parameters

<i>_pMV</i>	Replaces model variable in position <i>_at</i> with model variable <i>_pMV</i> 2. The replaced variable is destroyed.
-------------	---

6.46.3.32 `std::vector< ArrayXd > GoSUM::CModelVariables::samples ()`

Returns all samples.

6.46.3.33 `int GoSUM::CModelVariables::sampleSize () const` `[inline]`

Returns sample size.

6.46.3.34 `template<class Archive > void GoSUM::CModelVariables::serialize (Archive & ar, const unsigned int version)`
`[protected]`

6.46.3.35 `void GoSUM::CModelVariables::setBaseName (std::string _basename)` `[inline]`

Sets base name.

6.46.3.36 `void GoSUM::CModelVariables::setDistributions (const std::vector< int > & _selected)`

For selected model variables detects distribution from empirical sample data.

6.46.3.37 `void GoSUM::CModelVariables::setDistributions ()`

For all model variables detects distribution from empirical sample data.

6.46.3.38 `void GoSUM::CModelVariables::setNormalization ()`

6.46.3.39 `void GoSUM::CModelVariables::setNTuple (const ArrayXd & X, int _at)`

Sets *_at* sample values of all model variables.

6.46.3.40 `void GoSUM::CModelVariables::setSamples (const std::vector< ArrayXd > & _samples)`

Sets samples of all model variables to values in `_samples`.

6.46.3.41 `int GoSUM::CModelVariables::size () const` `[inline]`

Returns the size of model variables.

6.46.4 Friends And Related Function Documentation

6.46.4.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.46.5 Member Data Documentation

6.46.5.1 `std::string GoSUM::CModelVariables::basename` `[protected]`

Holds basename used for generating model variable unique names.

6.46.5.2 `boost::ptr_vector<CModelVariable> GoSUM::CModelVariables::mvs` `[protected]`

Holds pointers to model variables.

6.46.5.3 `std::string GoSUM::CModelVariables::nameToFind = ""` `[static], [protected]`

Used in member function Find.

6.46.5.4 `ArrayXd GoSUM::CModelVariables::scl` `[protected]`

Holds translation and scaling values for normalization of model variables.

6.46.5.5 `ArrayXd GoSUM::CModelVariables::trn` `[protected]`

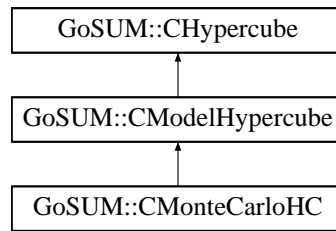
The documentation for this class was generated from the following files:

- [C:/Development/core/Model.h](#)
- [C:/Development/core/Model.cpp](#)

6.47 GoSUM::CMonteCarloHC Class Reference

```
#include <Hypercube.h>
```

Inheritance diagram for GoSUM::CMonteCarloHC:



Public Member Functions

- [CMonteCarloHC](#) ([CInputParameters](#) **pIP*)
- virtual [~CMonteCarloHC](#) ()

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- virtual void [doGenerate](#) (int *_rssize*, int *_dim*, std::vector< [ArrayXd](#) > &*_samples*)
Core of the generation.
- [CMonteCarloHC](#) ()

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.47.1 Constructor & Destructor Documentation

6.47.1.1 [GoSUM::CMonteCarloHC::CMonteCarloHC \(\)](#) [[inline](#)], [[protected](#)]

6.47.1.2 [GoSUM::CMonteCarloHC::CMonteCarloHC \(\[CInputParameters\]\(#\) * *pIP* \)](#) [[inline](#)]

6.47.1.3 virtual [GoSUM::CMonteCarloHC::~~CMonteCarloHC \(\)](#) [[inline](#)], [[virtual](#)]

6.47.2 Member Function Documentation

6.47.2.1 void [GoSUM::CMonteCarloHC::doGenerate \(int *_rssize*, int *_dim*, std::vector< \[ArrayXd\]\(#\) > & *_samples* \)](#)
[[protected](#)], [[virtual](#)]

Core of the generation.

Implements [GoSUM::CModelHypercube](#).

6.47.2.2 template<class Archive > void [GoSUM::CMonteCarloHC::serialize \(Archive & *ar*, const unsigned int *version* \)](#)
[[protected](#)]

Reimplemented from [GoSUM::CModelHypercube](#).

6.47.3 Friends And Related Function Documentation

6.47.3.1 friend class boost::serialization::access [friend]

Boost serialization.

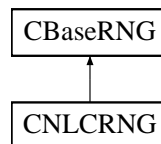
The documentation for this class was generated from the following files:

- C:/Development/core/[Hypercube.h](#)
- C:/Development/core/[Hypercube.cpp](#)

6.48 CNLCRNG Class Reference

```
#include <RandomGenerators.h>
```

Inheritance diagram for CNLCRNG:



Public Member Functions

- [CNLCRNG](#) ()
- [CNLCRNG](#) (unsigned int s)
- virtual [~CNLCRNG](#) ()
- virtual void [setSeed](#) (unsigned int s)
Sets seed of the RNG.
- virtual unsigned int [rndi](#) ()
Returns randomly generated unsigned int.
- virtual double [rnd](#) ()
Returns randomly generated double between 0 and 1.

Static Private Attributes

- static int [inext](#)
- static int [inextp](#)
Paramters of the nlc RNG.
- static long [ma](#) [56]
Paramters of the nlc RNG.

6.48.1 Constructor & Destructor Documentation

6.48.1.1 CNLCRNG::CNLCRNG () [inline]

6.48.1.2 CNLCRNG::CNLCRNG (unsigned int s) [inline]

6.48.1.3 virtual CNLCRNG::~~CNLCRNG () [inline],[virtual]

6.48.2 Member Function Documentation

6.48.2.1 `double CNLCRNG::rnd() [virtual]`

Returns randomly generated double between 0 and 1.

Implements [CBaseRNG](#).

6.48.2.2 `virtual unsigned int CNLCRNG::rndi() [inline],[virtual]`

Returns randomly generated unsigned int.

Implements [CBaseRNG](#).

6.48.2.3 `void CNLCRNG::setSeed(unsigned int s) [virtual]`

Sets seed of the RNG.

Implements [CBaseRNG](#).

6.48.3 Member Data Documentation

6.48.3.1 `int CNLCRNG::inext [static],[private]`

6.48.3.2 `int CNLCRNG::inextp [static],[private]`

Parameters of the nlc RNG.

6.48.3.3 `long CNLCRNG::ma [static],[private]`

Parameters of the nlc RNG.

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

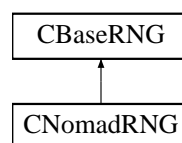
- C:/Development/core/[RandomGenerators.h](#)
- C:/Development/core/[RandomGenerators.cpp](#)

6.49 CNomadRNG Class Reference

Class for NOMAD RNG with period = $2^{96}-1$.

```
#include <RandomGenerators.h>
```

Inheritance diagram for CNomadRNG:



Public Member Functions

- [CNomadRNG](#) ()
- [CNomadRNG](#) (unsigned int s)
- virtual [~CNomadRNG](#) ()

- virtual void [setSeed](#) (unsigned int s)
Sets seed of the RNG.
- virtual unsigned int [rndi](#) ()
Returns randomly generated unsigned int.
- virtual double [rnd](#) ()
Returns randomly generated double between 0 and 1.

Static Private Attributes

- static unsigned int [x](#) = 123456789
- static unsigned int [y](#) = 362436069
- static unsigned int [z](#) = 521288629

6.49.1 Detailed Description

Class for NOMAD RNG with period = $2^{96}-1$.

6.49.2 Constructor & Destructor Documentation

6.49.2.1 `CNomadRNG::CNomadRNG ()` `[inline]`

6.49.2.2 `CNomadRNG::CNomadRNG (unsigned int s)` `[inline]`

6.49.2.3 `virtual CNomadRNG::~~CNomadRNG ()` `[inline]`, `[virtual]`

6.49.3 Member Function Documentation

6.49.3.1 `virtual double CNomadRNG::rnd ()` `[inline]`, `[virtual]`

Returns randomly generated double between 0 and 1.

Implements [CBaseRNG](#).

6.49.3.2 `unsigned int CNomadRNG::rndi ()` `[virtual]`

Returns randomly generated unsigned int.

Implements [CBaseRNG](#).

6.49.3.3 `virtual void CNomadRNG::setSeed (unsigned int s)` `[inline]`, `[virtual]`

Sets seed of the RNG.

Implements [CBaseRNG](#).

6.49.4 Member Data Documentation

6.49.4.1 `unsigned int CNomadRNG::x = 123456789` `[static]`, `[private]`

6.49.4.2 `unsigned int CNomadRNG::y = 362436069` `[static]`, `[private]`

6.49.4.3 `unsigned int CNomadRNG::z = 521288629` `[static]`, `[private]`

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

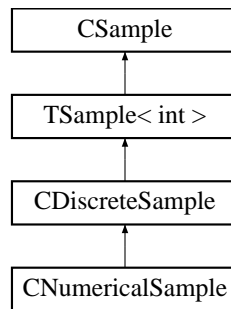
- C:/Development/core/[RandomGenerators.h](#)
- C:/Development/core/[RandomGenerators.cpp](#)

6.50 CNumericalSample Class Reference

Abstract class for numerical discrete samples.

```
#include <Sample.h>
```

Inheritance diagram for CNumericalSample:



Public Member Functions

- [CNumericalSample](#) ()
- [CNumericalSample](#) (const [CNumericalSample](#) &O)
- [CNumericalSample](#) (const [CContinuousSample](#) &O)
- virtual [~CNumericalSample](#) ()
- virtual void [readSampleValue](#) (std::ifstream &_ifs, int _at)
Reads particular sample data from input file stream.
- virtual void [writeSampleValue](#) (std::ofstream &_ofs, int _at) const
< Writes particular sample data to output file stream.
- virtual void [setSampleSize](#) (int _n)
Sets particular sample data value.
- int [minValue](#) () const
Returns minimal sample data value.
- int [maxValue](#) () const
Returns maximal sample data value.
- virtual double [variance](#) () const
Returns sample variance (i.e. empirical).

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Private Attributes

- std::vector< double > [values](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.50.1 Detailed Description

Abstract class for numerical discrete samples.

6.50.2 Constructor & Destructor Documentation

6.50.2.1 `CNumericalSample::CNumericalSample ()` `[inline]`

6.50.2.2 `CNumericalSample::CNumericalSample (const CNumericalSample & O)` `[inline]`

6.50.2.3 `CNumericalSample::CNumericalSample (const CContinuousSample & O)`

6.50.2.4 `virtual CNumericalSample::~~CNumericalSample ()` `[inline],[virtual]`

6.50.3 Member Function Documentation

6.50.3.1 `int CNumericalSample::maxValue () const` `[inline]`

Returns maximal sample data value.

Reimplemented from [TSample< t >](#).

6.50.3.2 `int CNumericalSample::minValue () const` `[inline]`

Returns minimal sample data value.

Reimplemented from [TSample< t >](#).

6.50.3.3 `void CNumericalSample::readSampleValue (std::ifstream & _ifs, int _at)` `[virtual]`

Reads particular sample data from input file stream.

Reimplemented from [TSample< t >](#).

6.50.3.4 `template<class Archive > void CNumericalSample::serialize (Archive & ar, const unsigned int version)`
`[private]`

6.50.3.5 `virtual void CNumericalSample::setSampleSize (int _n)` `[inline],[virtual]`

Sets particular sample data value.

Reimplemented from [TSample< t >](#).

6.50.3.6 `virtual double CNumericalSample::variance () const` `[inline],[virtual]`

Returns sample variance (i.e. empirical).

Reimplemented from [CDiscreteSample](#).

6.50.3.7 `virtual void CNumericalSample::writeSampleValue (std::ofstream & _ofs, int _at) const [inline], [virtual]`

< Writes particular sample data to output file stream.

Reimplemented from [TSample< t >](#).

6.50.4 Friends And Related Function Documentation

6.50.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.50.5 Member Data Documentation

6.50.5.1 `std::vector<double> CNumericalSample::values [private]`

Holds all numerical values.

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

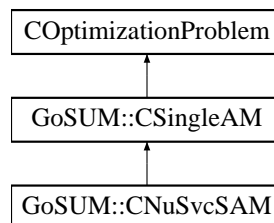
- C:/Development/core/[Sample.h](#)
- C:/Development/core/[Sample.cpp](#)

6.51 GoSUM::CNuSvcSAM Class Reference

Class for the analytical model for single output state, nu-SVC type.

```
#include <AnalyticalModel.h>
```

Inheritance diagram for GoSUM::CNuSvcSAM:



Public Member Functions

- [CNuSvcSAM \(\)](#)
- `virtual ~CNuSvcSAM ()`

Protected Member Functions

- `template<class Archive >`
void [serialize](#) (Archive &ar, const unsigned int version)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.51.1 Detailed Description

Class for the analytical model for single output state, nu-SVC type.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 `GoSUM::CNUSvcSAM::CNUSvcSAM ()` `[inline]`

6.51.2.2 `virtual GoSUM::CNUSvcSAM::~~CNUSvcSAM ()` `[inline],[virtual]`

6.51.3 Member Function Documentation

6.51.3.1 `template<class Archive > void GoSUM::CNUSvcSAM::serialize (Archive & ar, const unsigned int version)`
`[protected]`

Reimplemented from [GoSUM::CSingleAM](#).

6.51.4 Friends And Related Function Documentation

6.51.4.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

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

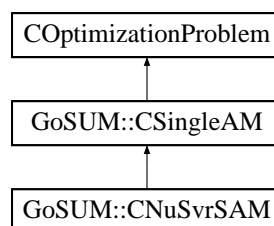
- C:/Development/core/[AnalyticalModel.h](#)
- C:/Development/core/[AnalyticalModel.cpp](#)

6.52 GoSUM::CNUSvrSAM Class Reference

Class for the analytical model for single output state, nu-SVR type.

```
#include <AnalyticalModel.h>
```

Inheritance diagram for GoSUM::CNUSvrSAM:



Public Member Functions

- [CNUSvrSAM \(\)](#)
- `virtual ~CNUSvrSAM ()`
- `virtual void openOptimization ()`
Opens optimization.
- `virtual void optimizationPoint2SVMParm (const ArrayXd &ov)`
Converts optimization point to SVM parameters.

Protected Member Functions

- `template<class Archive >`
`void serialize (Archive &ar, const unsigned int version)`

Friends

- `class boost::serialization::access`
Boost serialization.

Additional Inherited Members

6.52.1 Detailed Description

Class for the analytical model for single output state, nu-SVR type.

6.52.2 Constructor & Destructor Documentation

6.52.2.1 `GoSUM::CNuSvrSAM::CNuSvrSAM ()` `[inline]`

6.52.2.2 `virtual GoSUM::CNuSvrSAM::~~CNuSvrSAM ()` `[inline], [virtual]`

6.52.3 Member Function Documentation

6.52.3.1 `void GoSUM::CNuSvrSAM::openOptimization ()` `[virtual]`

Opens optimization.

Reimplemented from [GoSUM::CSingleAM](#).

6.52.3.2 `void GoSUM::CNuSvrSAM::optimizationPoint2SVMParam (const ArrayXd & ov)` `[virtual]`

Converts optimization point to SVM parameters.

Reimplemented from [GoSUM::CSingleAM](#).

6.52.3.3 `template<class Archive > void GoSUM::CNuSvrSAM::serialize (Archive & ar, const unsigned int version)`
`[protected]`

Reimplemented from [GoSUM::CSingleAM](#).

6.52.4 Friends And Related Function Documentation

6.52.4.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

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

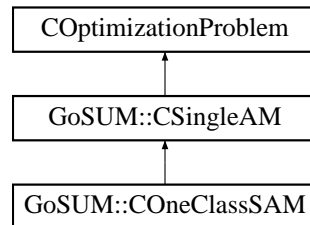
- `C:/Development/core/AnalyticalModel.h`
- `C:/Development/core/AnalyticalModel.cpp`

6.53 GoSUM::COneClassSAM Class Reference

Class for the analytical model for single output state, one class type.

```
#include <AnalyticalModel.h>
```

Inheritance diagram for GoSUM::COneClassSAM:



Public Member Functions

- [COneClassSAM](#) ()
- virtual [~COneClassSAM](#) ()

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.53.1 Detailed Description

Class for the analytical model for single output state, one class type.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 [GoSUM::COneClassSAM::COneClassSAM \(\)](#) [[inline](#)]

6.53.2.2 [virtual GoSUM::COneClassSAM::~~COneClassSAM \(\)](#) [[inline](#)], [[virtual](#)]

6.53.3 Member Function Documentation

6.53.3.1 [template<class Archive > void GoSUM::COneClassSAM::serialize \(Archive & ar, const unsigned int version \)](#)
[[protected](#)]

Reimplemented from [GoSUM::CSingleAM](#).

6.53.4 Friends And Related Function Documentation

6.53.4.1 friend class boost::serialization::access [friend]

Boost serialization.

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

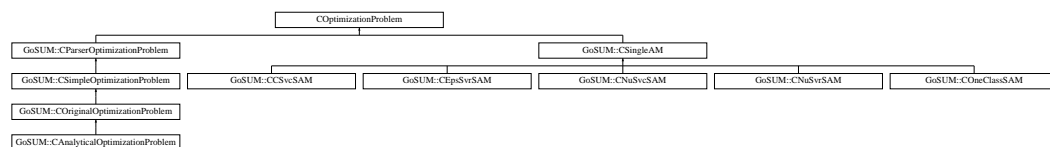
- C:/Development/core/[AnalyticalModel.h](#)
- C:/Development/core/[AnalyticalModel.cpp](#)

6.54 COptimizationProblem Class Reference

Class for the optimization problem.

```
#include <OptimizationProblem.h>
```

Inheritance diagram for COptimizationProblem:



Public Member Functions

- [COptimizationProblem](#) ()
- virtual [~COptimizationProblem](#) ()
- virtual void [clear](#) ()
Clears all.
- virtual void [clearHistory](#) ()
Clears results.
- bool [emptyHistory](#) () const
Returns true if results are empty. false otherwise.
- bool [isMinimization](#) ()
Returns true if it is a minimization problem, false otherwise.
- bool [isMaximization](#) ()
Returns true if it is a maximization problem, false otherwise.
- void [setMinimization](#) ()
Sets optimization problem to minimization.
- void [setMaximization](#) ()
Sets optimization problem to maximization.
- int [dimension](#) () const
Returns dimension of the problem.
- [COptimizationVariable](#) * [addVariable](#) ([COptimizationVariable](#) * _pOV)
Adds optimization variable.
- void [eraseVariable](#) (int _at)
Erases particular optimization variable.
- void [setLowerBound](#) (const [ArrayXd](#) & _xL)
Sets lower bound of the optimization variables.
- void [setUpperBound](#) (const [ArrayXd](#) & _xU)
Sets upper bound of the optimization variables.

- void [setInitialValue](#) (const ArrayXd &_x0)
Sets initial value of the optimization variables.
- ArrayXd [lowerBound](#) () const
Returns lower bound of the optimization variables.
- ArrayXd [upperBound](#) () const
returns upper bound of the optimization variables.
- ArrayXd [initialValue](#) () const
Returns initial value of the optimization variables.
- void [setLowerBound](#) (int _at, double _xL)
Sets lower bound of the optimization variables.
- void [setUpperBound](#) (int _at, double _xU)
Sets upper bound of the optimization variables.
- void [setInitialValue](#) (int _at, double _x0)
Sets initial value of the optimization variables.
- double [lowerBound](#) (int _at) const
Returns lower bound of the optimization variables.
- double [upperBound](#) (int _at) const
Returns upper bound of the optimization variables.
- double [initialValue](#) (int _at) const
Returns initial value of the optimization variables.
- virtual double [objective](#) (const ArrayXd &_ov)=0
Evaluates objective function value from optimization variables values.
- virtual bool [isFeasible](#) (const ArrayXd &_ov)
Returns true if optimization variable is feasible, false otherwise.
- virtual int [constraintsSize](#) () const =0
Returns size of the constraints.
- virtual double [constraint](#) (const ArrayXd &_ov, int _at)=0
Evalautes particular constraint value from optimization variables values.
- virtual bool [evaluate](#) (const ArrayXd &_hp, ArrayXd &_ep)
Evaluates objective and all constraints from optimization variables values and returns true if it is feasible, false otherwise.
- virtual void [openOptimization](#) ()
- virtual void [closeOptimization](#) ()
Opens, i.e. prepares optimization.
- void [writeHistory](#) (const ArrayXd &_ov, double _res)
Closes optimization, i.e. closes what was opened in openOptimization.
- int [historySize](#) () const
Returns history size.
- double [objectiveHistory](#) (int _i) const
Returns particular objective history.
- double [variableHistory](#) (int _i, int _j) const
Returns particular variable history.
- ArrayXd [exportObjectiveHistory](#) () const
Exports objective history.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Protected Attributes

- bool [minimize](#)
Indicates if it is a minimization problem.
- boost::ptr_vector
< [COptimizationVariable](#) > [ovs](#)
Holds all optimization variables.
- std::vector< std::pair
< ArrayXd, double > > [history](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.54.1 Detailed Description

Class for the optimization problem.

6.54.2 Constructor & Destructor Documentation

6.54.2.1 `COptimizationProblem::COptimizationProblem ()` `[inline]`

6.54.2.2 `virtual COptimizationProblem::~~COptimizationProblem ()` `[inline], [virtual]`

6.54.3 Member Function Documentation

6.54.3.1 `COptimizationVariable* COptimizationProblem::addVariable (COptimizationVariable * _pOV)`
`[inline]`

Adds optimization variable.

6.54.3.2 `virtual void COptimizationProblem::clear ()` `[inline], [virtual]`

Clears all.

Reimplemented in [GoSUM::CSimpleOptimizationProblem](#).

6.54.3.3 `virtual void COptimizationProblem::clearHistory ()` `[inline], [virtual]`

Clears results.

Reimplemented in [GoSUM::CSimpleOptimizationProblem](#).

6.54.3.4 `virtual void COptimizationProblem::closeOptimization ()` `[inline], [virtual]`

Opens, i.e. prepares optimization.

Reimplemented in [GoSUM::COriginalOptimizationProblem](#), [GoSUM::CSimpleOptimizationProblem](#), and [GoSUM::CParserOptimizationProblem](#).

6.54.3.5 `virtual double COptimizationProblem::constraint (const ArrayXd & _ov, int _at)` `[pure virtual]`

Evaluates particular constraint value from optimization variables values.

Implemented in [GoSUM::CSingleAM](#), and [GoSUM::CParserOptimizationProblem](#).

6.54.3.6 `virtual int COptimizationProblem::constraintsSize () const` `[pure virtual]`

Returns size of the constraints.

Implemented in [GoSUM::CSingleAM](#), and [GoSUM::CParserOptimizationProblem](#).

6.54.3.7 `int COptimizationProblem::dimension () const` `[inline]`

Returns dimension of the problem.

6.54.3.8 `bool COptimizationProblem::emptyHistory () const` `[inline]`

Returns true if results are empty. false otherwise.

6.54.3.9 `void COptimizationProblem::eraseVariable (int _at)`

Erases particular optimization variable.

6.54.3.10 `bool COptimizationProblem::evaluate (const ArrayXd & _hp, ArrayXd & _ep)` `[virtual]`

Evaluates objective and all constraints from optimization variables values and returns true if it is feasible, false otherwise.

Reimplemented in [GoSUM::CSimpleOptimizationProblem](#).

6.54.3.11 `ArrayXd COptimizationProblem::exportObjectiveHistory () const`

Exports objective history.

6.54.3.12 `int COptimizationProblem::historySize () const` `[inline]`

Returns history size.

6.54.3.13 `ArrayXd COptimizationProblem::initialValue () const`

Returns initial value of the optimization variables.

6.54.3.14 `double COptimizationProblem::initialValue (int _at) const` `[inline]`

Returns initial value of the optimization variables.

6.54.3.15 `virtual bool COptimizationProblem::isFeasible (const ArrayXd & _ov)` `[inline], [virtual]`

Returns true if optimization variable is feasible, false otherwise.

Reimplemented in [GoSUM::CSimpleOptimizationProblem](#).

6.54.3.16 `bool COptimizationProblem::isMaximization () [inline]`

Returns true if it is a maximization problem, false otherwise.

6.54.3.17 `bool COptimizationProblem::isMinimization () [inline]`

Returns true if it is a minimization problem, false otherwise.

6.54.3.18 `ArrayXd COptimizationProblem::lowerBound () const`

Returns lower bound of the optimization variables.

6.54.3.19 `double COptimizationProblem::lowerBound (int _at) const [inline]`

Returns lower bound of the optimization variables.

6.54.3.20 `virtual double COptimizationProblem::objective (const ArrayXd & _ov) [pure virtual]`

Evaluates objective function value from optimization variables values.

Implemented in [GoSUM::CSingleAM](#), and [GoSUM::CParserOptimizationProblem](#).

6.54.3.21 `double COptimizationProblem::objectiveHistory (int _i) const [inline]`

Returns particular objective history.

6.54.3.22 `virtual void COptimizationProblem::openOptimization () [inline],[virtual]`

Reimplemented in [GoSUM::CNuSvrSAM](#), [GoSUM::CEpsSvrSAM](#), [GoSUM::COriginalOptimizationProblem](#), [GoSUM::CSimpleOptimizationProblem](#), [GoSUM::CSingleAM](#), and [GoSUM::CParserOptimizationProblem](#).

6.54.3.23 `template<class Archive > void COptimizationProblem::serialize (Archive & ar, const unsigned int version) [inline],[protected]`

Reimplemented in [GoSUM::CNuSvrSAM](#), [GoSUM::CA analyticalOptimizationProblem](#), [GoSUM::CEpsSvrSAM](#), [GoSUM::COriginalOptimizationProblem](#), [GoSUM::COneClassSAM](#), [GoSUM::CNuSvcSAM](#), [GoSUM::CSimpleOptimizationProblem](#), [GoSUM::CCSvcSAM](#), [GoSUM::CSingleAM](#), and [GoSUM::CParserOptimizationProblem](#).

6.54.3.24 `void COptimizationProblem::setInitialValue (const ArrayXd & _x0)`

Sets initial value of the optimization variables.

6.54.3.25 `void COptimizationProblem::setInitialValue (int _at, double _x0) [inline]`

Sets initial value of the optimization variables.

6.54.3.26 `void COptimizationProblem::setLowerBound (const ArrayXd & _xL)`

Sets lower bound of the optimization variables.

6.54.3.27 void COptimizationProblem::setLowerBound (int *_at*, double *_xL*) [inline]

Sets lower bound of the optimization variables.

6.54.3.28 void COptimizationProblem::setMaximization () [inline]

Sets optimization problem to maximization.

6.54.3.29 void COptimizationProblem::setMinimization () [inline]

Sets optimization problem to minimization.

6.54.3.30 void COptimizationProblem::setUpperBound (const ArrayXd & *_xU*)

Sets upper bound of the optimization variables.

6.54.3.31 void COptimizationProblem::setUpperBound (int *_at*, double *_xU*) [inline]

Sets upper bound of the optimization variables.

6.54.3.32 ArrayXd COptimizationProblem::upperBound () const

returns upper bound of the optimization variables.

6.54.3.33 double COptimizationProblem::upperBound (int *_at*) const [inline]

Returns upper bound of the optimization variables.

6.54.3.34 double COptimizationProblem::variableHistory (int *_i*, int *_j*) const [inline]

Returns particular variable history.

6.54.3.35 void COptimizationProblem::writeHistory (const ArrayXd & *_ov*, double *_res*)

Closes optimization, i.e. closes what was opened in openOptimization.

6.54.4 Friends And Related Function Documentation

6.54.4.1 friend class boost::serialization::access [friend]

Boost serialization.

6.54.5 Member Data Documentation

6.54.5.1 std::vector< std::pair<ArrayXd,double> > COptimizationProblem::history [protected]

Holds optimization history, i.e. sequence of best results up to every evaluation.

6.54.5.2 bool COptimizationProblem::minimize [protected]

Indicates if it is a minimization problem.

6.54.5.3 boost::ptr_vector<COptimizationVariable> COptimizationProblem::ovs [protected]

Holds all optimization variables.

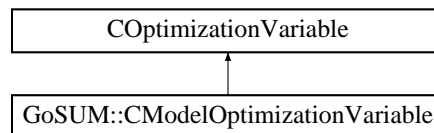
The documentation for this class was generated from the following files:

- C:/Development/core/OptimizationProblem.h
- C:/Development/core/OptimizationProblem.cpp

6.55 COptimizationVariable Class Reference

```
#include <OptimizationProblem.h>
```

Inheritance diagram for COptimizationVariable:



Public Member Functions

- COptimizationVariable ()
- COptimizationVariable (double _xL, double _xU, double _x0)
- COptimizationVariable (COptimizationVariable &_O)
- virtual ~COptimizationVariable ()
- double lowerBound () const
Returns lower bound.
- double upperBound () const
Returns upper bound.
- double initialValue () const
Returns initial value.
- void setLowerBound (double _xL)
Sets lower bound.
- void setUpperBound (double _xU)
Sets upper bound.
- void setInitialValue (double _x0)
Sets initial value.
- void set (double _xL, double _xU, double _x0)
Sets lower bound, upper bound, initial value.

Protected Member Functions

- template<class Archive >
void serialize (Archive &ar, const unsigned int version)

Protected Attributes

- double [xL](#)
- double [xU](#)
- double [x0](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.55.1 Constructor & Destructor Documentation

6.55.1.1 `COptimizationVariable::COptimizationVariable ()` [\[inline\]](#)

6.55.1.2 `COptimizationVariable::COptimizationVariable (double _xL, double _xU, double _x0)` [\[inline\]](#)

6.55.1.3 `COptimizationVariable::COptimizationVariable (COptimizationVariable & _O)` [\[inline\]](#)

6.55.1.4 `virtual COptimizationVariable::~~COptimizationVariable ()` [\[inline\]](#), [\[virtual\]](#)

6.55.2 Member Function Documentation

6.55.2.1 `double COptimizationVariable::initialValue () const` [\[inline\]](#)

Returns initial value.

6.55.2.2 `double COptimizationVariable::lowerBound () const` [\[inline\]](#)

Returns lower bound.

6.55.2.3 `template<class Archive > void COptimizationVariable::serialize (Archive & ar, const unsigned int version)`
[\[inline\]](#), [\[protected\]](#)

Reimplemented in [GoSUM::CModelOptimizationVariable](#).

6.55.2.4 `void COptimizationVariable::set (double _xL, double _xU, double _x0)`

Sets lower bound, upper bound, initial value.

6.55.2.5 `void COptimizationVariable::setInitialValue (double _x0)`

Sets initial value.

6.55.2.6 `void COptimizationVariable::setLowerBound (double _xL)`

Sets lower bound.

6.55.2.7 `void COptimizationVariable::setUpperBound (double _xU)`

Sets upper bound.

6.55.2.8 `double COptimizationVariable::upperBound () const` `[inline]`

Returns upper bound.

6.55.3 Friends And Related Function Documentation

6.55.3.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.55.4 Member Data Documentation

6.55.4.1 `double COptimizationVariable::x0` `[protected]`

Holds lower bound, upper bound and initial value of a optimization variable.

6.55.4.2 `double COptimizationVariable::xL` `[protected]`

6.55.4.3 `double COptimizationVariable::xU` `[protected]`

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

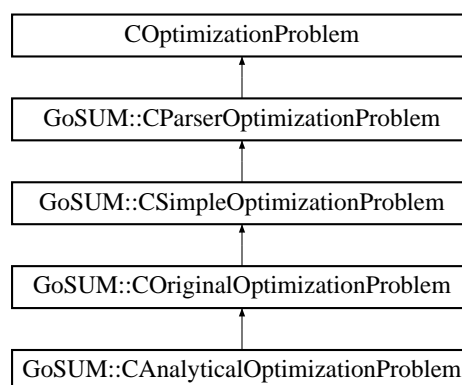
- [C:/Development/core/OptimizationProblem.h](#)
- [C:/Development/core/OptimizationProblem.cpp](#)

6.56 GoSUM::COriginalOptimizationProblem Class Reference

Class for the optimization problem based on [GoSUM](#) analytical model.

```
#include <ParserOptimizationProblem.h>
```

Inheritance diagram for GoSUM::COriginalOptimizationProblem:



Public Member Functions

- [COriginalOptimizationProblem](#) ([CInputParameters](#) *`_pIP`, [COutputStates](#) *`_pOS`)
- virtual [~COriginalOptimizationProblem](#) ()
- virtual void [openOptimization](#) ()
- virtual void [closeOptimization](#) ()

Opens, i.e. prepares optimization.

- ArrayXd [inputPoint2ModelPoint](#) (const ArrayXd &_ip)

Closes optimization, i.e. closes what was opened in openOptimization.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- [COriginalOptimizationProblem](#) ()
- virtual void [setVariableNames](#) ()

Sets variable names for the parser.

Protected Attributes

- [COutputStates](#) * [pOS](#)

Friends

- class [boost::serialization::access](#)

Boost serialization.

Additional Inherited Members

6.56.1 Detailed Description

Class for the optimization problem based on [GoSUM](#) analytical model.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 [GoSUM::COriginalOptimizationProblem::COriginalOptimizationProblem \(\)](#) `[inline], [protected]`

6.56.2.2 [GoSUM::COriginalOptimizationProblem::COriginalOptimizationProblem \(CInputParameters * _pIP, COutputStates * _pOS \)](#) `[inline]`

6.56.2.3 [virtual GoSUM::COriginalOptimizationProblem::~~COriginalOptimizationProblem \(\)](#) `[inline], [virtual]`

6.56.3 Member Function Documentation

6.56.3.1 [virtual void GoSUM::COriginalOptimizationProblem::closeOptimization \(\)](#) `[inline], [virtual]`

Opens, i.e. prepares optimization.

Reimplemented from [GoSUM::CSimpleOptimizationProblem](#).

6.56.3.2 [ArrayXd GoSUM::COriginalOptimizationProblem::inputPoint2ModelPoint \(const ArrayXd & _ip \)](#)

Closes optimization, i.e. closes what was opened in openOptimization.

Converts input parameter point to model point.

Reimplemented from [GoSUM::CSimpleOptimizationProblem](#).

Reimplemented in [GoSUM::CA analyticalOptimizationProblem](#).

6.56.3.3 `virtual void GoSUM::COriginalOptimizationProblem::openOptimization () [inline], [virtual]`

Reimplemented from [GoSUM::CSimpleOptimizationProblem](#).

6.56.3.4 `template<class Archive > void GoSUM::COriginalOptimizationProblem::serialize (Archive & ar, const unsigned int version) [protected]`

Reimplemented from [GoSUM::CSimpleOptimizationProblem](#).

Reimplemented in [GoSUM::CA analyticalOptimizationProblem](#).

6.56.3.5 `void GoSUM::COriginalOptimizationProblem::setVariableNames () [protected], [virtual]`

Sets variable names for the parser.

Reimplemented from [GoSUM::CSimpleOptimizationProblem](#).

6.56.4 Friends And Related Function Documentation

6.56.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.56.5 Member Data Documentation

6.56.5.1 `COutputStates* GoSUM::COriginalOptimizationProblem::pOS [protected]`

Points to output states.

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

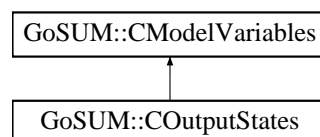
- C:/Development/core/[ParserOptimizationProblem.h](#)
- C:/Development/core/[ParserOptimizationProblem.cpp](#)

6.57 GoSUM::COutputStates Class Reference

Class for [GoSUM](#) output states.

```
#include <Model.h>
```

Inheritance diagram for GoSUM::COutputStates:



Public Member Functions

- [COutputStates](#) ()
- virtual [~COutputStates](#) ()
- bool [setEvaluatorCompatible](#) (const [CInputParameters](#) &inputs)
- void [evaluate](#) (const [CInputParameters](#) &inputs)

Starts end joins multiple threads for model function evaluation.

- void [openEvaluation](#) (const [CInputParameters](#) &inputs)
- ArrayXd [evaluate](#) (const ArrayXd &X)

Opens evaluation on single input parameter n-tuples.

- void [closeEvaluation](#) ()
- void [setProgressSlot](#) (boost::function< void()> _progressSlot)

Sets external progress slot.

Public Attributes

- boost::function< void()> [progressSlot](#)

Closes evaluation on single input parameter n-tuples.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Private Attributes

- [GoSUM::CModelEvaluator](#) * [pE](#)

Friends

- class [boost::serialization::access](#)
boost serialization

Additional Inherited Members

6.57.1 Detailed Description

Class for [GoSUM](#) output states.

6.57.2 Constructor & Destructor Documentation

6.57.2.1 [GoSUM::COutputStates::COutputStates \(\)](#) [[inline](#)]

6.57.2.2 [virtual GoSUM::COutputStates::~~COutputStates \(\)](#) [[inline](#)], [[virtual](#)]

6.57.3 Member Function Documentation

6.57.3.1 [void GoSUM::COutputStates::closeEvaluation \(\)](#)

6.57.3.2 [void GoSUM::COutputStates::evaluate \(const \[CInputParameters\]\(#\) & *inputs* \)](#)

Starts end joins multiple threads for model function evaluation.

6.57.3.3 [ArrayXd GoSUM::COutputStates::evaluate \(const ArrayXd & *X* \)](#)

Opens evaluation on single input parameter n-tuples.

Evaluates output state on single input parameter n-tuple.

6.57.3.4 void GoSUM::COutputStates::openEvaluation (const CInputParameters & *inputs*)

6.57.3.5 template<class Archive > void GoSUM::COutputStates::serialize (Archive & *ar*, const unsigned int *version*)
[private]

6.57.3.6 bool GoSUM::COutputStates::setEvaluatorCompatible (const CInputParameters & *inputs*)

6.57.3.7 void GoSUM::COutputStates::setProgressSlot (boost::function< void()> *progressSlot*) [inline]

Sets external progress slot.

6.57.4 Friends And Related Function Documentation

6.57.4.1 friend class boost::serialization::access [friend]

boost serialization

6.57.5 Member Data Documentation

6.57.5.1 GoSUM::CModelEvaluator* GoSUM::COutputStates::pE [private]

Holds pointer to model evaluator used for single evaluations.

6.57.5.2 boost::function<void()> GoSUM::COutputStates::progressSlot

Closes evaluation on single input parameter n-tuples.

External progress slot, later connected to signal for evaluation progress.

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

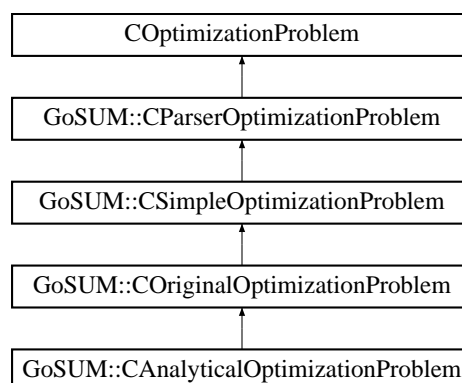
- C:/Development/core/[Model.h](#)
- C:/Development/core/[Model.cpp](#)

6.58 GoSUM::CParserOptimizationProblem Class Reference

Class for the optimization problem with parser functions.

```
#include <ParserOptimizationProblem.h>
```

Inheritance diagram for GoSUM::CParserOptimizationProblem:



Public Member Functions

- [CParserOptimizationProblem](#) ()
- virtual [~CParserOptimizationProblem](#) ()
- void [setObjectiveExpression](#) (const std::string &_fexpr)
Sets objective expression.
- std::string [objectiveExpression](#) ()
Returns objective expression.
- virtual int [constraintsSize](#) () const
Returns size of the constraints.
- void [clearConstraintExpressions](#) ()
Clears constraints expressions.
- void [addConstraintExpression](#) (const std::string &_gexpr)
Adds constraint expression.
- void [eraseConstraintExpression](#) (int _at)
Erases particular constraint expression.
- void [setConstraintExpression](#) (const std::string &_gexpr, int _at)
Sets particular constraint expression.
- std::string [constraintExpression](#) (int _at)
Returns particular constraint expression.
- std::string [roundoffEquality](#) (std::string _expr)
If _expr is an equality expression left=right it returns expression $\text{abs}(\text{left}-\text{right}) \leq \text{TINY}$.
- bool [validateExpressions](#) ()
Validates all expressions.
- void [parseExpressions](#) ()
Parses all expressions.
- virtual void [openOptimization](#) ()
- virtual void [closeOptimization](#) ()
Opens, i.e. prepares optimization.
- double [objective](#) (const ArrayXd &_mv)
Closes optimization, i.e. closes what was opened in openOptimization.
- double [constraint](#) (const ArrayXd &_mv, int _at)
Evaluates particular constraint value from model variables values.

Public Attributes

- boost::signal< void()> [optimizingProgressed](#)
Signal for optimizing progress, emitted on single objective evaluation.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- virtual void [setVariableNames](#) ()=0
Sets variable names for the parser.

Protected Attributes

- `std::string` [names](#)
Holds all names that are permitted in objective and constraint expressions.
- `std::string` [fexpr](#)
Holds objective function expression.
- `std::vector< std::string >` [gexpr](#)
Holds expressions for constraints.
- `FunctionParser` [f](#)
Holds function parser for objective function.
- `std::vector< FunctionParser >` [g](#)

Friends

- `class` [boost::serialization::access](#)
Boost serialization.

6.58.1 Detailed Description

Class for the optimization problem with parser functions.

6.58.2 Constructor & Destructor Documentation

6.58.2.1 `GoSUM::CParserOptimizationProblem::CParserOptimizationProblem ()` `[inline]`

6.58.2.2 `virtual GoSUM::CParserOptimizationProblem::~~CParserOptimizationProblem ()` `[inline]`, `[virtual]`

6.58.3 Member Function Documentation

6.58.3.1 `void GoSUM::CParserOptimizationProblem::addConstraintExpression (const std::string & _gexpr)` `[inline]`

Adds constraint expression.

6.58.3.2 `void GoSUM::CParserOptimizationProblem::clearConstraintExpressions ()` `[inline]`

Clears constraints expressions.

6.58.3.3 `void GoSUM::CParserOptimizationProblem::closeOptimization ()` `[virtual]`

Opens, i.e. prepares optimization.

Reimplemented from [COptimizationProblem](#).

Reimplemented in [GoSUM::COriginalOptimizationProblem](#), and [GoSUM::CSimpleOptimizationProblem](#).

6.58.3.4 `double GoSUM::CParserOptimizationProblem::constraint (const ArrayXd & _mv, int _at)` `[inline]`, `[virtual]`

Evaluate particular constraint value from model variables values.

Implements [COptimizationProblem](#).

6.58.3.5 `std::string GoSUM::CParserOptimizationProblem::constraintExpression (int _at)` `[inline]`

Returns particular constraint expression.

6.58.3.6 `virtual int GoSUM::CParserOptimizationProblem::constraintsSize () const` `[inline],[virtual]`

Returns size of the constraints.

Implements [COptimizationProblem](#).

6.58.3.7 `void GoSUM::CParserOptimizationProblem::eraseConstraintExpression (int _at)`

Erases particular constraint expression.

6.58.3.8 `double GoSUM::CParserOptimizationProblem::objective (const ArrayXd & _mv)` `[inline],[virtual]`

Closes optimization, i.e. closes what was opened in `openOptimization`.

Evaluates objective function value from model variables values.

Implements [COptimizationProblem](#).

6.58.3.9 `std::string GoSUM::CParserOptimizationProblem::objectiveExpression ()` `[inline]`

Returns objective expression.

6.58.3.10 `void GoSUM::CParserOptimizationProblem::openOptimization ()` `[virtual]`

Reimplemented from [COptimizationProblem](#).

Reimplemented in [GoSUM::COriginalOptimizationProblem](#), and [GoSUM::CSimpleOptimizationProblem](#).

6.58.3.11 `void GoSUM::CParserOptimizationProblem::parseExpressions ()`

Parses all expressions.

6.58.3.12 `std::string GoSUM::CParserOptimizationProblem::roundoffEquality (std::string _expr)`

If `_expr` is an equality expression `left=right` it returns expression `abs(left-right)<=TINY`.

6.58.3.13 `template<class Archive > void GoSUM::CParserOptimizationProblem::serialize (Archive & ar, const unsigned int version)` `[protected]`

Reimplemented from [COptimizationProblem](#).

Reimplemented in [GoSUM::CA analyticalOptimizationProblem](#), [GoSUM::COriginalOptimizationProblem](#), and [GoSUM::CSimpleOptimizationProblem](#).

6.58.3.14 `void GoSUM::CParserOptimizationProblem::setConstraintExpression (const std::string & _gexpr, int _at)`

Sets particular constraint expression.

6.58.3.15 void GoSUM::CParserOptimizationProblem::setObjectiveExpression (const std::string & *fexpr*) [inline]

Sets objective expression.

6.58.3.16 virtual void GoSUM::CParserOptimizationProblem::setVariableNames () [protected], [pure virtual]

Sets variable names for the parser.

Implemented in [GoSUM::COriginalOptimizationProblem](#), and [GoSUM::CSimpleOptimizationProblem](#).

6.58.3.17 bool GoSUM::CParserOptimizationProblem::validateExpressions ()

Validates all expressions.

6.58.4 Friends And Related Function Documentation

6.58.4.1 friend class boost::serialization::access [friend]

Boost serialization.

6.58.5 Member Data Documentation

6.58.5.1 FunctionParser GoSUM::CParserOptimizationProblem::f [protected]

Holds function parser for objective function.

6.58.5.2 std::string GoSUM::CParserOptimizationProblem::fexpr [protected]

Holds objective function expression.

6.58.5.3 std::vector<FunctionParser> GoSUM::CParserOptimizationProblem::g [protected]

Holds function parsers for constraints.

6.58.5.4 std::vector<std::string> GoSUM::CParserOptimizationProblem::gexpr [protected]

Holds expressions for constraints.

6.58.5.5 std::string GoSUM::CParserOptimizationProblem::names [protected]

Holds all names that are permitted in objective and constraint expressions.

6.58.5.6 boost::signal<void()> GoSUM::CParserOptimizationProblem::optimizingProgressed

Signal for optimizing progress, emitted on single objective evaluation.

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

- C:/Development/core/[ParserOptimizationProblem.h](#)
- C:/Development/core/[ParserOptimizationProblem.cpp](#)

6.59 CPlot2D Class Reference

```
#include <Plot2D.h>
```

Public Member Functions

- [CPlot2D](#) ()
- virtual [~CPlot2D](#) ()
- [QwtPlotCanvas](#) * [canvas](#) ()
Returns pointer to plot canvas.
- void [newPlot](#) ([QWidget](#) *pwidget, double xmin, double xmax, double ymin, double ymax, double lym, double lymax)
Creates new 2D plot with given min and max values on the axis.
- void [newColorBarPlot](#) ([QWidget](#) *pwidget, double xmin, double xmax, double ymin, double ymax, double zmin, double zmax)
Creates new 2D plot with given min and max values on the axis.
- void [setTitles](#) (string title, [Qt::GlobalColor](#) clr, string btitle, [Qt::GlobalColor](#) bclr, string ttitle, [Qt::GlobalColor](#) tclr, string ltitle, [Qt::GlobalColor](#) lclr, string rtitle, [Qt::GlobalColor](#) rclr)
- void [replot](#) ()
Sets titles (text and color) for all axis.
- void [clear](#) ()
Clears object.
- void [plotWithPen](#) (const [ArrayXd](#) &X, const [ArrayXd](#) &Y, string name, [Qt::GlobalColor](#) clr)
Plots data using pen.
- void [plotWithSymbol](#) (const [ArrayXd](#) &X, const [ArrayXd](#) &Y, string name, [Qt::GlobalColor](#) clr)
Plots data using symbols.
- void [plotWithSymbol](#) (const [std::vector](#)< double > &X, const [std::vector](#)< double > &Y, string name, [Qt::GlobalColor](#) clr)
Plots data using symbols.
- void [plotWithHistogram](#) (const [ArrayXi](#) &Xi, const [ArrayXd](#) &H, string name, [Qt::GlobalColor](#) clr)
Plots data using histogram.
- void [plotWithHistogram](#) (const [ArrayXd](#) &X, const [ArrayXd](#) &H, string name, [Qt::GlobalColor](#) clr)
Plots data using histogram.
- void [plotWithSpectrogram](#) (const [ArrayXXd](#) &A, const [std::vector](#)< [std::string](#) > &xlabels, const [std::vector](#)< [std::string](#) > &ylabels)
Plots data using spectrogram.
- [QwtPlot](#) * [getPlot](#) ()
Returns the QwtPlot object.

Private Member Functions

- [QwtLinearColorMap](#) * [newColorMap](#) ()

Private Attributes

- [QwtPlot](#) * [plot](#)
Holds pointer to the QwtPlot object.

6.59.1 Constructor & Destructor Documentation

6.59.1.1 `CPlot2D::CPlot2D ()` `[inline]`

6.59.1.2 `virtual CPlot2D::~~CPlot2D ()` `[inline]`, `[virtual]`

6.59.2 Member Function Documentation

6.59.2.1 `QwtPlotCanvas * CPlot2D::canvas ()`

Returns pointer to plot canvas.

6.59.2.2 `void CPlot2D::clear ()` `[inline]`

Clears object.

6.59.2.3 `QwtPlot* CPlot2D::getPlot ()` `[inline]`

Returns the QwtPlot object.

6.59.2.4 `void CPlot2D::newColorBarPlot (QWidget * pwidget, double xmin, double xmax, double ymin, double ymax, double zmin, double zmax)`

Creates new 2D plot with given min and max values on the axis.

6.59.2.5 `QwtLinearColorMap * CPlot2D::newColorMap ()` `[private]`

6.59.2.6 `void CPlot2D::newPlot (QWidget * pwidget, double xmin, double xmax, double ymin, double ymax, double lymin, double lymax)`

Creates new 2D plot with given min and max values on the axis.

6.59.2.7 `void CPlot2D::plotWithHistogram (const ArrayXi & Xi, const ArrayXd & H, string name, Qt::GlobalColor clr)`

Plots data using histogram.

6.59.2.8 `void CPlot2D::plotWithHistogram (const ArrayXd & X, const ArrayXd & H, string name, Qt::GlobalColor clr)`

Plots data using histogram.

6.59.2.9 `void CPlot2D::plotWithPen (const ArrayXd & X, const ArrayXd & Y, string name, Qt::GlobalColor clr)`

Plots data using pen.

6.59.2.10 `void CPlot2D::plotWithSpectrogram (const ArrayXXd & A, const std::vector< std::string > & xlabels, const std::vector< std::string > & ylabels)`

Plots data using spectrogram.

6.59.2.11 void CPlot2D::plotWithSymbol (const ArrayXd & X, const ArrayXd & Y, string name, Qt::GlobalColor clr)

Plots data using symbols.

6.59.2.12 void CPlot2D::plotWithSymbol (const std::vector< double > & X, const std::vector< double > & Y, string name, Qt::GlobalColor clr)

Plots data using symbols.

6.59.2.13 void CPlot2D::replot () [inline]

Sets titles (text and color) for all axis.

Replots the 2D plot.

6.59.2.14 void CPlot2D::setTitles (string title, Qt::GlobalColor clr, string btitle, Qt::GlobalColor bclr, string ttitle, Qt::GlobalColor tclr, string ltitle, Qt::GlobalColor lclr, string rtitle, Qt::GlobalColor rclr)

6.59.3 Member Data Documentation

6.59.3.1 QwtPlot* CPlot2D::plot [private]

Holds pointer to the QwtPlot object.

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

- C:/Development/core/Plot2D.h
- C:/Development/core/Plot2D.cpp

6.60 CRandomGenerator Class Reference

Class that holds all implemented uniform RNGs as static. and allows switching between them.

```
#include <RandomGenerators.h>
```

Public Types

- enum rngtype {
smallperiod, mediumperiod, nonlinearcongruential, largeperiod,
dynamicsystem, nomads }

Static Public Member Functions

- static rngtype Type (std::string _stype)
Converts rng type name (string) to rngtype enumerator.
- static void Set (rngtype _type, unsigned int s=111)
Sets RNG of rngtype _type as active.
- static void SetSmallPeriod (unsigned int s=111)
Sets small period RNG as active.
- static void SetMediumPeriod (unsigned int s=111)
Sets medium period RNG as active.
- static void SetNonlinearCongruential (unsigned int s=111)

- Sets non linearly congruential RNG as active.*
- static void [SetLargePeriod](#) (unsigned int s=111)
Sets large period RNG as active.
- static void [SetDynamicSystem](#) (unsigned int s)
Sets dynamic system RNG as active.
- static void [SetNomads](#) (unsigned int s=111)
Sets dynamic system RNG as active.
- static [rngtype](#) & [Type](#) ()
Returns type of the active RNG.
- static void [SetSeed](#) (unsigned int s)
Sets seed of the active RNG.
- static unsigned int [RndInt](#) ()
Returns an unsigned int randomly generated by the active RNG.
- static double [Rnd](#) ()
Returns a double between 0 and 1 randomly generated by the active RNG.
- static double [Rnd](#) (double a, double b)
Returns a double between a and b randomly generated by the active RNG.

Static Private Attributes

- static [CSmallPeriodRNG](#) sp
Small period RNG, static.
- static [CMediumPeriodRNG](#) mp
Medium period RNG, static.
- static [CNLCRNG](#) nlc
Non linearly congruential RNG, static.
- static [CLargePeriodRNG](#) lp
Large period RNG, static.
- static [CDynamicSystemRNG](#) ds
Dynamic system RNG, static.
- static [CNomadRNG](#) mad
Dynamic system RNG, static.
- static [CBaseRNG](#) * p = &[CRandomGenerator::ds](#)
Pointer to the active RNG, static.
- static enum [rngtype](#) type = [CRandomGenerator::dynamicsystem](#)
Type of the active RNG.

6.60.1 Detailed Description

Class that holds all implemented uniform RNGs as static. and allows switching between them.

6.60.2 Member Enumeration Documentation

6.60.2.1 enum [CRandomGenerator::rngtype](#)

Enumerator:

smallperiod
mediumperiod
nonlinearcongruential
largeperiod
dynamicsystem
nomads

6.60.3 Member Function Documentation

6.60.3.1 `static double CRandomGenerator::Rnd () [inline],[static]`

Returns a double between 0 and 1 randomly generated by the active RNG.

6.60.3.2 `static double CRandomGenerator::Rnd (double a, double b) [inline],[static]`

Returns a double between *a* and *b* randomly generated by the active RNG.

6.60.3.3 `static unsigned int CRandomGenerator::RndInt () [inline],[static]`

Returns an unsigned int randomly generated by the active RNG.

6.60.3.4 `void CRandomGenerator::Set (rngtype _type, unsigned int s = 111) [static]`

Sets RNG of *rngtype _type* as active.

6.60.3.5 `static void CRandomGenerator::SetDynamicSystem (unsigned int s) [inline],[static]`

Sets dynamic system RNG as active.

6.60.3.6 `static void CRandomGenerator::SetLargePeriod (unsigned int s = 111) [inline],[static]`

Sets large period RNG as active.

6.60.3.7 `static void CRandomGenerator::SetMediumPeriod (unsigned int s = 111) [inline],[static]`

Sets medium period RNG as active.

6.60.3.8 `static void CRandomGenerator::SetNomads (unsigned int s = 111) [inline],[static]`

Sets dynamic system RNG as active.

6.60.3.9 `static void CRandomGenerator::SetNonlinearCongruential (unsigned int s = 111) [inline],[static]`

Sets non linearly congruential RNG as active.

6.60.3.10 `static void CRandomGenerator::SetSeed (unsigned int s) [inline],[static]`

Sets seed of the active RNG.

6.60.3.11 `static void CRandomGenerator::SetSmallPeriod (unsigned int s = 111) [inline],[static]`

Sets small period RNG as active.

6.60.3.12 `CRandomGenerator::rngtype CRandomGenerator::Type (std::string _stype) [static]`

Converts rng type name (string) to rngtype enumerator.

6.60.3.13 **static rngtype& CRandomGenerator::Type ()** [inline],[static]

Returns type of the active RNG.

6.60.4 Member Data Documentation

6.60.4.1 **CDynamicSystemRNG CRandomGenerator::ds** [static],[private]

Dynamic system RNG, static.

6.60.4.2 **CLargePeriodRNG CRandomGenerator::lp** [static],[private]

Large period RNG, static.

6.60.4.3 **CNomadRNG CRandomGenerator::mad** [static],[private]

Dynamic system RNG, static.

6.60.4.4 **CMediumPeriodRNG CRandomGenerator::mp** [static],[private]

Medium period RNG, static.

6.60.4.5 **CNLCRNG CRandomGenerator::nlc** [static],[private]

Non linearly congruential RNG, static.

6.60.4.6 **CBaseRNG * CRandomGenerator::p = &CRandomGenerator::ds** [static],[private]

Pointer to the active RNG, static.

6.60.4.7 **CSmallPeriodRNG CRandomGenerator::sp** [static],[private]

Small period RNG, static.

6.60.4.8 **CRandomGenerator::rngtype CRandomGenerator::type = CRandomGenerator::dynamicsystem**
[static],[private]

Type of the active RNG.

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

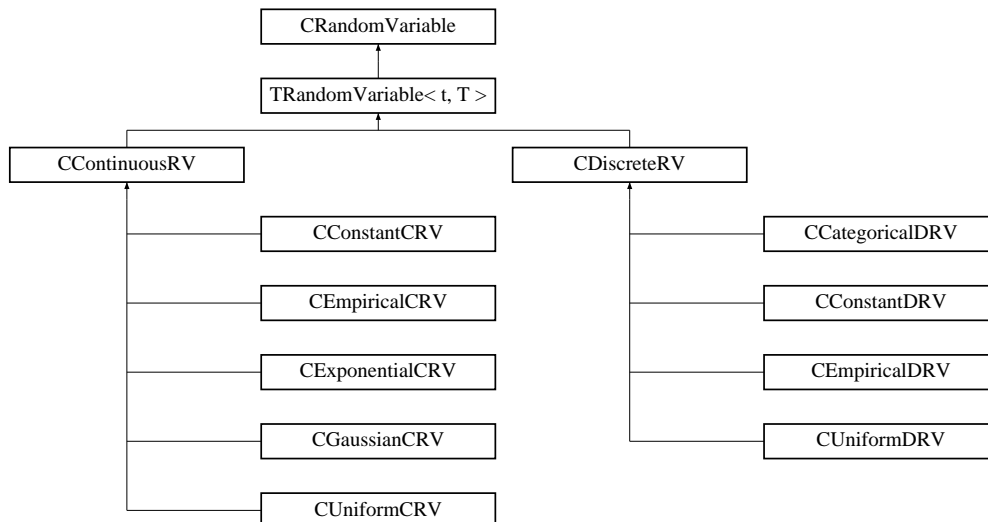
- C:/Development/core/[RandomGenerators.h](#)
- C:/Development/core/[RandomGenerators.cpp](#)

6.61 CRandomVariable Class Reference

Abstract base class for all random variables.

```
#include <RandomVariable.h>
```

Inheritance diagram for CRandomVariable:



Public Types

- enum `distributiontype` {
`constantdiscrete`, `uniformdiscrete`, `empiricaldiscrete`, `categorical`,
`constantcontinuous`, `uniformcontinuous`, `gaussian`, `exponential`,
`empiricalcontinuous` }

Public Member Functions

- `CRandomVariable` ()
- virtual `~CRandomVariable` ()
- `CRandomVariable` (const `CRandomVariable` &O)
- virtual double `quantile` (double `_p`) const =0
Returns quantile (inverse cumulative distribution function, probit function) of `_p` in [0, 1].
- virtual int `expandedSize` () const
- virtual double `minValue` () const =0
Returns number of variables after expansion for SVM and similar, 1 except in cateogrical variable.
- virtual double `maxValue` () const =0
Returns maximal value of the random variable.
- virtual double `expectedValue` () const =0
Returns expected value of the random variable.
- virtual `distributiontype` `distributionType` () const =0
Returns enum type of the random variable distribution.
- virtual std::string `distributionName` () const =0
Returns name of the random variable distribution.
- virtual bool `isDistributionDefined` () const
Returns true if distribution is defined, false otherwise.
- virtual double `variance` () const =0
Returns variance of the random variable.

Static Public Member Functions

- static `distributiontype` `Type` (std::string `_stype`)
Converts distribution name (string) to distributiontype enumerator.
- static bool `IsDistributionType` (std::string `_stype`)

Returns true if it is a distribution type, false otherwise.

- static bool [IsEmpirical](#) ([distributiontype](#) _type)
Returns true if distribution type is empirical, false otherwise.
- static bool [IsTheoretical](#) ([distributiontype](#) _type)
Returns true if distribution type is theoretical, false otherwise.
- static bool [IsDiscrete](#) ([distributiontype](#) _type)
Returns true if distribution type is discrete, false otherwise.
- static bool [IsContinuous](#) ([distributiontype](#) _type)
Returns true if distribution type is continuous, false otherwise.
- static int [DistributionParametersSize](#) ([distributiontype](#) _type)
Returns size (number of) distribution parameters.

Protected Member Functions

- virtual double [doQuantile](#) (double _p) const =0
Quantile, formula implementation without checking argument.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.61.1 Detailed Description

Abstract base class for all random variables.

6.61.2 Member Enumeration Documentation

6.61.2.1 enum CRandomVariable::distributiontype

Enumerator:

constantdiscrete
uniformdiscrete
empiricaldiscrete
categorical
constantcontinuous
uniformcontinuous
gaussian
exponential
empiricalcontinuous

6.61.3 Constructor & Destructor Documentation

6.61.3.1 `CRandomVariable::CRandomVariable ()` `[inline]`

6.61.3.2 `virtual CRandomVariable::~~CRandomVariable ()` `[inline],[virtual]`

6.61.3.3 `CRandomVariable::CRandomVariable (const CRandomVariable & O)` `[inline]`

6.61.4 Member Function Documentation

6.61.4.1 `virtual std::string CRandomVariable::distributionName () const` `[pure virtual]`

Returns name of the random variable distribution.

Implemented in [CEmpiricalCRV](#), [CExponentialCRV](#), [CGaussianCRV](#), [CUniformCRV](#), [CConstantCRV](#), [CCategoricalDRV](#), [CEmpiricalDRV](#), [CUniformDRV](#), and [CConstantDRV](#).

6.61.4.2 `int CRandomVariable::DistributionParametersSize (CRandomVariable::distributiontype _type)` `[static]`

Returns size (number of) distribution parameters.

6.61.4.3 `virtual distributiontype CRandomVariable::distributionType () const` `[pure virtual]`

Returns enum type of the random variable distribution.

Implemented in [CEmpiricalCRV](#), [CExponentialCRV](#), [CGaussianCRV](#), [CUniformCRV](#), [CConstantCRV](#), [CCategoricalDRV](#), [CEmpiricalDRV](#), [CUniformDRV](#), and [CConstantDRV](#).

6.61.4.4 `virtual double CRandomVariable::doQuantile (double _p) const` `[protected],[pure virtual]`

Quantile, formula implementation without checking argument.

Implemented in [CEmpiricalCRV](#), [CExponentialCRV](#), [CGaussianCRV](#), [CUniformCRV](#), [CConstantCRV](#), [CCategoricalDRV](#), [CEmpiricalDRV](#), [CUniformDRV](#), and [CConstantDRV](#).

6.61.4.5 `virtual int CRandomVariable::expandedSize () const` `[inline],[virtual]`

Reimplemented in [CCategoricalDRV](#), and [CEmpiricalDRV](#).

6.61.4.6 `virtual double CRandomVariable::expectedValue () const` `[pure virtual]`

Returns expected value of the random variable.

Implemented in [CEmpiricalCRV](#), [CExponentialCRV](#), [CGaussianCRV](#), [CUniformCRV](#), [CConstantCRV](#), [CCategoricalDRV](#), [CEmpiricalDRV](#), [CUniformDRV](#), and [CConstantDRV](#).

6.61.4.7 `bool CRandomVariable::IsContinuous (CRandomVariable::distributiontype _type)` `[static]`

Returns true if distribution type is continuous, false otherwise.

6.61.4.8 `bool CRandomVariable::IsDiscrete (CRandomVariable::distributiontype _type)` `[static]`

Returns true if distribution type is discrete, false otherwise.

6.61.4.9 `virtual bool CRandomVariable::isDistributionDefined () const [inline],[virtual]`

Returns true if distribution is defined, false otherwise.

Reimplemented in [CEmpiricalCRV](#), [CCategoricalDRV](#), and [CEmpiricalDRV](#).

6.61.4.10 `bool CRandomVariable::IsDistributionType (std::string _stype) [static]`

Returns true if it is a distribution type, false otherwise.

6.61.4.11 `bool CRandomVariable::IsEmpirical (CRandomVariable::distributiontype _type) [static]`

Returns true if distribution type is empirical, false otherwise.

6.61.4.12 `bool CRandomVariable::IsTheoretical (CRandomVariable::distributiontype _type) [static]`

Returns true if distribution type is theoretical, false otherwise.

6.61.4.13 `virtual double CRandomVariable::maxValue () const [pure virtual]`

Returns maximal value of the random variable.

Implemented in [CEmpiricalCRV](#), [CExponentialCRV](#), [CGaussianCRV](#), [CUniformCRV](#), [CConstantCRV](#), [C-CategoricalDRV](#), [CEmpiricalDRV](#), [CUniformDRV](#), and [CConstantDRV](#).

6.61.4.14 `virtual double CRandomVariable::minValue () const [pure virtual]`

Returns number of variables after expansion for SVM and similar, 1 except in cateogrical variable.

Returns minimal value of the random variable.

Implemented in [CEmpiricalCRV](#), [CExponentialCRV](#), [CGaussianCRV](#), [CUniformCRV](#), [CConstantCRV](#), [C-CategoricalDRV](#), [CEmpiricalDRV](#), [CUniformDRV](#), and [CConstantDRV](#).

6.61.4.15 `virtual double CRandomVariable::quantile (double _p) const [pure virtual]`

Returns quantile (inverse cumulative distribution function, probit function) of _p in [0,1].

Implemented in [CContinuousRV](#), and [CDiscreteRV](#).

6.61.4.16 `template<class Archive > void CRandomVariable::serialize (Archive & ar, const unsigned int version) [inline],[private]`

6.61.4.17 `CRandomVariable::distributiontype CRandomVariable::Type (std::string _stype) [static]`

Converts distribution name (string) to distributiontype enumerator.

6.61.4.18 `virtual double CRandomVariable::variance () const [pure virtual]`

Returns variance of the random variable.

Implemented in [CEmpiricalCRV](#), [CExponentialCRV](#), [CGaussianCRV](#), [CUniformCRV](#), [CConstantCRV](#), [C-CategoricalDRV](#), [CEmpiricalDRV](#), [CUniformDRV](#), and [CConstantDRV](#).

6.61.5 Friends And Related Function Documentation

6.61.5.1 friend class boost::serialization::access [friend]

Boost serialization.

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

- C:/Development/core/[RandomVariable.h](#)
- C:/Development/core/[RandomVariable.cpp](#)

6.62 GoSUM::CReduction Class Reference

Class for sensitivity analysis of the model.

```
#include <Reduction.h>
```

Public Types

- enum [reductiontype](#) {
 [derivative](#), [averagederivative](#), [absoluteaveragederivative](#), [variance](#),
 [anova](#) }
- enum [cutofftype](#) { [cutoffvalue](#), [cutoffsize](#) }

Public Member Functions

- [CReduction](#) ([CInputParameters](#) *_pIP, [COutputStates](#) *_pOS, [CA analyticalModel](#) *_pAM, [CSensitivityAnalysis](#) *_pSA)
- virtual [~CReduction](#) ()
- void [clear](#) ()
 Clears data.
- void [initialize](#) ()
 Initializes reducing.
- const [ArrayXXd](#) & [sensitivityIndex](#) ()
 Returns actual sensitivitiy index.
- void [setReductionType](#) (enum [reductiontype](#) _redtype)
 Sets reduction type.
- void [evaluateMaximalSensitivities](#) ()
 Evaluates maximal indices for all input parameters relative to selected output states.
- void [selectInputParameters](#) ()
 Cuts by actual cutoff criteria.
- int [cutoffSize](#) () const
 Returns cutoff size.
- double [cutoffValue](#) () const
 return cutoff value.
- void [setCutoffSize](#) (int _cutip)
 Sets cutoff size.
- void [setCutoffValue](#) (double _cutval)
 Sets cutoff value.
- void [selectOutputState](#) (int _o)
 Selects output state for reduction criteria.
- void [selectOutputs](#) (const std::vector< std::string > &_selOS)

- Selects output states named in the `_selOS`.*
- bool `isSelectedOutputState` (int `_o`) const
Returns true if particular output state is selected, false otherwise.
- bool `isSelectedInputParameter` (int `_i`) const
Returns true if particular input parameter is selected, false otherwise.
- double `maximalSensitivity` (int `_i`)
Returns maximal (relative to selected output states) sensitivity index for particular input parameter.
- double `maximalSensitivity` ()
Returns maximal (relative to selected output states) sensitivity index.
- void `eraseNonSelectedVariables` ()
Reduces the model, i.e. deletes all input parameters and output states that are not selected.

Static Public Member Functions

- static `reductiontype ReductionType` (const std::string &`_stype`)
Converts reduction type name (string) to reductiontype enumerator.
- static `reductiontype ReductionType` ()
Returns actual reduction type.
- static `cutofftype CutoffType` ()
Returns actual criteria type.
- static void `SetReductionType` (enum `reductiontype` `_redtype`)
Sets reduction type.
- static void `SetCutoffType` (enum `cutofftype` `_cuttype`)
Sets reduction type.

Static Public Attributes

- static enum `reductiontype redtype`
Holds actual reduction type.
- static enum `cutofftype cuttype`
Holds actual criteria type.

Private Member Functions

- template<class Archive >
 void `serialize` (Archive &`ar`, const unsigned int `version`)
- `CReduction` ()

Private Attributes

- `CInputParameters` * `pIP`
Points to input parameters.
- `COutputStates` * `pOS`
Points to output states.
- `CAnalyticalModel` * `pAM`
Points to the analytical model.
- `CSensitivityAnalysis` * `pSA`
Points to the sensitivity analysis.
- std::vector< std::pair< double,
 bool > > `rankIP`

Holds pairs of input parameter appropriate maximal sensitivity indices (relative to selected output states), it sorted in descending order and appropriate selection status.

- `std::vector< bool >` `rankOS`

Holds pairs of output state names and appropriate selection status.

- `double` `cutval`

Holds cutoff value.

- `int` `cutip`

Holds size of the selected inputs.

- `int` `cutos`

Friends

- class `boost::serialization::access`

Boost serialization.

6.62.1 Detailed Description

Class for sensitivity analysis of the model.

6.62.2 Member Enumeration Documentation

6.62.2.1 enum `GoSUM::CReduction::cutofftype`

Enumerator:

cutoffvalue

cutoffsize

6.62.2.2 enum `GoSUM::CReduction::reductiontype`

Enumerator:

derivative

averagederivative

absoluteaveragederivative

variance

anova

6.62.3 Constructor & Destructor Documentation

6.62.3.1 `GoSUM::CReduction::CReduction ()` `[inline]`, `[private]`

6.62.3.2 `GoSUM::CReduction::CReduction (CInputParameters * _pIP, COutputStates * _pOS, CAnalyticalModel * _pAM, CSensitivityAnalysis * _pSA)` `[inline]`

6.62.3.3 `virtual GoSUM::CReduction::~~CReduction ()` `[inline]`, `[virtual]`

6.62.4 Member Function Documentation

6.62.4.1 `void GoSUM::CReduction::clear ()` `[inline]`

Clears data.

6.62.4.2 `int GoSUM::CReduction::cutoffSize () const [inline]`

Returns cutoff size.

6.62.4.3 `static cutofftype GoSUM::CReduction::CutoffType () [inline],[static]`

Returns actual criteria type.

6.62.4.4 `double GoSUM::CReduction::cutoffValue () const [inline]`

return cutoff value.

6.62.4.5 `void GoSUM::CReduction::eraseNonSelectedVariables ()`

Reduces the model, i.e. deletes all input paramters and output states that are not selected.

6.62.4.6 `void GoSUM::CReduction::evaluateMaximalSensitivities ()`

Evaluates maximal indices for all input parameters relative to selected output states.

6.62.4.7 `void GoSUM::CReduction::initialize ()`

Initializes reducing.

6.62.4.8 `bool GoSUM::CReduction::isSelectedInputParameter (int _i) const`

Returns true if particular input parameter is selected, false otherwise.

6.62.4.9 `bool GoSUM::CReduction::isSelectedOutputState (int _o) const`

Returns true if particular output state is selected, false otherwise.

6.62.4.10 `double GoSUM::CReduction::maximalSensitivity (int _i)`

Returns maximal (relative to selected output states) sensitivity index for particular input parameter.

6.62.4.11 `double GoSUM::CReduction::maximalSensitivity ()`

Returns maximal (relative to selected output states) sensitivity index.

6.62.4.12 `GoSUM::CReduction::reductiontype GoSUM::CReduction::ReductionType (const std::string & _stype) [static]`

Converts reduction type name (string) to reductiontype enumerator.

6.62.4.13 `static reductiontype GoSUM::CReduction::ReductionType () [inline],[static]`

Returns actual reduction type.

6.62.4.14 `void GoSUM::CReduction::selectInputParameters ()`

Cuts by actual cutoff criteria.

6.62.4.15 `void GoSUM::CReduction::selectOutputs (const std::vector< std::string > & _selOS)`

Selects output states named in the `_selOS`.

6.62.4.16 `void GoSUM::CReduction::selectOutputState (int _o)`

Selects output state for reduction criteria.

6.62.4.17 `const ArrayXXd & GoSUM::CReduction::sensitivityIndex ()`

Returns actual sensitivitiy index.

6.62.4.18 `template<class Archive > void GoSUM::CReduction::serialize (Archive & ar, const unsigned int version)`
`[private]`

6.62.4.19 `void GoSUM::CReduction::setCutoffSize (int _cutip)`

Sets cutoff size.

6.62.4.20 `static void GoSUM::CReduction::SetCutoffType (enum cutofftype _cuttype)` `[inline], [static]`

Sets reduction type.

6.62.4.21 `void GoSUM::CReduction::setCutoffValue (double _cutval)`

Sets cutoff value.

6.62.4.22 `static void GoSUM::CReduction::SetReductionType (enum reductiontype _redtype)` `[inline], [static]`

Sets reduction type.

6.62.4.23 `void GoSUM::CReduction::setReductionType (enum reductiontype _redtype)`

Sets reduction type.

6.62.5 Friends And Related Function Documentation

6.62.5.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.62.6 Member Data Documentation

6.62.6.1 `int GoSUM::CReduction::cutip` `[private]`

Holds size of the selected inputs.

6.62.6.2 `int GoSUM::CReduction::cutos` `[private]`

Holds size of the selected outputs.

6.62.6.3 `enum cutofftype GoSUM::CReduction::cuttype` `[static]`

Holds actual criteria type.

6.62.6.4 `double GoSUM::CReduction::cutval` `[private]`

Holds cutoff value.

6.62.6.5 `CA analyticalModel* GoSUM::CReduction::pAM` `[private]`

Points to the analytical model.

6.62.6.6 `CInputParameters* GoSUM::CReduction::pIP` `[private]`

Points to input parameters.

6.62.6.7 `COutputStates* GoSUM::CReduction::pOS` `[private]`

Points to output states.

6.62.6.8 `CSensitivityAnalysis* GoSUM::CReduction::pSA` `[private]`

Points to the sensitivity analysis.

6.62.6.9 `std::vector< std::pair<double,bool> > GoSUM::CReduction::rankIP` `[private]`

Holds pairs of input parameter appropriate maximal sensitivity indices (relative to selected output states), it sorted in descending order and appropriate selection status.

6.62.6.10 `std::vector<bool> GoSUM::CReduction::rankOS` `[private]`

Holds pairs of output state names and appropriate selection status.

6.62.6.11 `enum reductiontype GoSUM::CReduction::redtype` `[static]`

Holds actual reduction type.

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

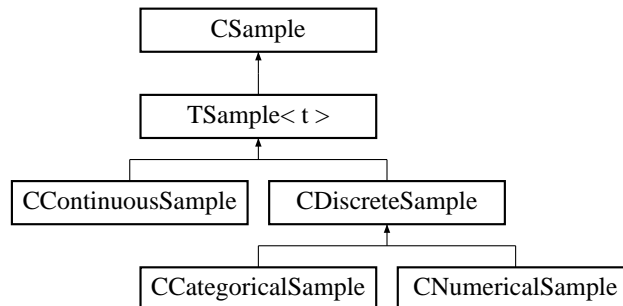
- [C:/Development/core/Reduction.h](#)
- [C:/Development/core/Reduction.cpp](#)

6.63 CSample Class Reference

Abstract class for all samples.

```
#include <Sample.h>
```

Inheritance diagram for CSample:



Public Member Functions

- [CSample](#) ()
- virtual [~CSample](#) ()
- [CSample](#) (const [CSample](#) &O)
- virtual void [clear](#) ()=0
Clears object.
- virtual void [setSampleSize](#) (int _n)=0
(Re)sizes the sample, all previous data is lost.
- virtual int [sampleSize](#) () const =0
Returns sample size (number of data in the sample).
- virtual void [setSampleValue](#) (double _val, int _at)=0
Sets particular sample data value.
- virtual void [readSampleValue](#) (std::ifstream &_ifs, int _at)=0
Reads particular sample data from input file stream.
- virtual void [writeSampleValue](#) (std::ofstream &_ofs, int _at) const =0
Writes particular sample data to output file stream.
- virtual void [computeStatistics](#) (int _n)=0
Computes sample statistics, i.e. normalized histogram etc.
- virtual double [variance](#) () const =0
Returns sample variance (i.e. empirical).
- virtual std::vector< int > [select](#) (double _left, double _right) const =0
Returns vector of indices of sample values selected if they are in the interval [_left,_right].
- virtual void [eraseSelected](#) (const std::vector< int > &sel)=0
Erases sample values on positions defiend by sel.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Static Protected Attributes

- static int [maxdiscretesize](#) = 11

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.63.1 Detailed Description

Abstract class for all samples.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 `CSample::CSample ()` `[inline]`

6.63.2.2 `virtual CSample::~~CSample ()` `[inline],[virtual]`

6.63.2.3 `CSample::CSample (const CSample & O)` `[inline]`

6.63.3 Member Function Documentation

6.63.3.1 `virtual void CSample::clear ()` `[pure virtual]`

Clears object.

Implemented in [TSample< t >](#).

6.63.3.2 `virtual void CSample::computeStatistics (int _n)` `[pure virtual]`

Computes sample statistics, i.e. normalized histogram etc.

Implemented in [CContinuousSample](#), and [CDiscreteSample](#).

6.63.3.3 `virtual void CSample::eraseSelected (const std::vector< int > & sel)` `[pure virtual]`

Erases sample values on positions defiend by sel.

Implemented in [TSample< t >](#).

6.63.3.4 `virtual void CSample::readSampleValue (std::ifstream & _ifs, int _at)` `[pure virtual]`

Reads particular sample data from input file stream.

Implemented in [CCategoricalSample](#), [CNumericalSample](#), and [TSample< t >](#).

6.63.3.5 `virtual int CSample::sampleSize () const` `[pure virtual]`

Returns sample size (number of data in the sample).

Implemented in [TSample< t >](#).

6.63.3.6 `virtual std::vector<int> CSample::select (double _left, double _right) const` `[pure virtual]`

Returns vector of indices of sample values selected if they are in the interval [_left,_right].

Implemented in [TSample< t >](#).

6.63.3.7 `template<class Archive > void CSample::serialize (Archive & ar, const unsigned int version)` `[inline]`,
`[protected]`

Reimplemented in [CContinuousSample](#), [CDiscreteSample](#), and [TSample< t >](#).

6.63.3.8 `virtual void CSample::setSampleSize (int n)` `[pure virtual]`

(Re)sizes the sample, all previous data is lost.

Implemented in [CCategoricalSample](#), [CNumericalSample](#), and [TSample< t >](#).

6.63.3.9 `virtual void CSample::setSampleValue (double val, int at)` `[pure virtual]`

Sets particular sample data value.

Implemented in [TSample< t >](#).

6.63.3.10 `virtual double CSample::variance () const` `[pure virtual]`

Returns sample variance (i.e.empirical).

Implemented in [CCategoricalSample](#), [CNumericalSample](#), [CContinuousSample](#), and [CDiscreteSample](#).

6.63.3.11 `virtual void CSample::writeSampleValue (std::ofstream & ofs, int at) const` `[pure virtual]`

Writes particular sample data to output file stream.

Implemented in [CCategoricalSample](#), [CNumericalSample](#), and [TSample< t >](#).

6.63.4 Friends And Related Function Documentation

6.63.4.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.63.5 Member Data Documentation

6.63.5.1 `int CSample::maxdiscretesize = 11` `[static]`, `[protected]`

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

- C:/Development/core/[Sample.h](#)
- C:/Development/core/[Sample.cpp](#)

6.64 GoSUM::CScript Class Reference

Class for the [GoSUM](#) script format.

```
#include <Script.h>
```

Public Member Functions

- [CScript](#) ()
- virtual [~CScript](#) ()
- void [read](#) ([GoSUM::CContainer](#) *_pContainer, const std::string &_fileName)
Reads script file and interprets all script commands.

Private Member Functions

- std::string [line2Path](#) (std::istream &_line)
Returns path from the input line.
- void [addVariable](#) (std::istream &iscl)
Tells project to add model variable.
- void [addVariables](#) (std::istream &iscl)
Tells project to add model variables.
- void [importVariables](#) (std::istream &iscl)
Tells project to import model variables.
- void [setProjectPath](#) (std::istream &iscl)
- void [setProjectName](#) (std::istream &iscl)
- void [setProjectType](#) (std::istream &iscl)
- void [setOptimizationMethod](#) (std::istream &iscl)
- void [setCoreSize](#) (std::istream &iscl)
- void [setMatLabPath](#) (std::istream &iscl)
- void [setRNG](#) (std::istream &iscl)
- void [addInput](#) (std::istream &iscl)
- void [addOutput](#) (std::istream &iscl)
- void [addInputs](#) (std::istream &iscl)
- void [addOutputs](#) (std::istream &iscl)
- void [importInputs](#) (std::istream &iscl)
- void [importOutputs](#) (std::istream &iscl)
- void [setResampleType](#) (std::istream &iscl)
- void [setResampleSize](#) (std::istream &iscl)
- void [setVoronoiOptions](#) (std::istream &iscl)
- void [resampleInputs](#) (std::istream &iscl)
- void [setModelEvaluator](#) (std::istream &iscl)
- void [evaluateOutputs](#) (std::istream &iscl)
- void [exportInputSamples](#) (std::istream &iscl)
- void [exportOutputSamples](#) (std::istream &iscl)
- void [learnModel](#) (std::istream &iscl)
- void [importPredictionInputSamples](#) (std::istream &iscl)
- void [exportPredictionOutputSamples](#) (std::istream &iscl)
- void [predict](#) (std::istream &iscl)
- void [setSensitivityOptions](#) (std::istream &iscl)
- void [computeSensitivities](#) (std::istream &iscl)
- void [setReductionType](#) (std::istream &iscl)
- void [setReductionOutputs](#) (std::istream &iscl)
- void [setReductionCutoffSize](#) (std::istream &iscl)
- void [setReductionCutoffValue](#) (std::istream &iscl)
- void [reduce](#) (std::istream &iscl)
- void [setObjective](#) (std::istream &iscl)
- void [addOptimizationConstraint](#) (std::istream &iscl)
- void [setConstLowerBound](#) (std::istream &iscl)
- void [setLowerBound](#) (std::istream &iscl)

- void [setConstUpperBound](#) (std::istream &iscl)
- void [setUpperBound](#) (std::istream &iscl)
- void [setConstInitialValue](#) (std::istream &iscl)
- void [setInitialValue](#) (std::istream &iscl)
- void [setMadsMaxEvaluation](#) (std::istream &iscl)
- void [setMadsLHSearch](#) (std::istream &iscl)
- void [setMadsInitMeshSize](#) (std::istream &iscl)
- void [setMadsMinPollSize](#) (std::istream &iscl)
- void [minimize](#) (std::istream &iscl)
- void [maximize](#) (std::istream &iscl)
- void [save](#) (std::istream &iscl)
- void [load](#) (std::istream &iscl)
- void [saveXml](#) (std::istream &iscl)
- void [loadXml](#) (std::istream &iscl)
- void [saveTxt](#) (std::istream &iscl)
- void [loadTxt](#) (std::istream &iscl)
- void [exportDerivativeSensitivity](#) (std::istream &iscl)
- void [exportAverageDerivative](#) (std::istream &iscl)
- void [exportAbsoluteAverageDerivative](#) (std::istream &iscl)
- void [exportVarianceSensitivity](#) (std::istream &iscl)
- void [exportANOVA1](#) (std::istream &iscl)
- void [exportOptimizationMethod](#) (std::istream &iscl)
- void [exportOptimizationHistory](#) (std::istream &iscl)

Private Attributes

- vector< pair< std::string,
[paction](#) > > [cmds](#)
Vector of pairs: first is string of the admissible script command, second is pointer to action function.
- int [noeqI](#)
- CContainer * [pContainer](#)
Pointer to [GoSUM](#) project container.
- void(CContainer::* [padd](#))(const std::string &, CRandomVariable::distributiontype, double, double)
Pointer to member function that tells project to add model variable.
- void(CContainer::* [padds](#))(int, const std::string &, CRandomVariable::distributiontype, double, double)
Pointer to member function that tells project to add model variables.
- void(CContainer::* [pimport](#))(const std::string &_fname)
Pointer to member function that tells project to import data from import file.

6.64.1 Detailed Description

Class for the [GoSUM](#) script format.

6.64.2 Constructor & Destructor Documentation

6.64.2.1 [GoSUM::CScript::CScript \(\)](#)

6.64.2.2 [virtual GoSUM::CScript::~~CScript \(\)](#) [[inline](#)], [[virtual](#)]

6.64.3 Member Function Documentation

6.64.3.1 void GoSUM::CScript::addInput (std::istream & *iscl*) [inline], [private]

Tells project to add one input parameter. Format of the related script command is:

add_input = nm tp a b

where **nm** is the input parameter name (must be unique), **tp** is the distribution type, and **a** and **b** are distribution parameters, for some types they must be omitted. More precisely, following distribution types, and related distribution parameters can be used:

- *constantdiscrete*, then **a** is the constant value of that random variable, **b** must be omitted,
- *uniformdiscrete*, then **a** and **b** are the lower and the upper bound of that random variable,
- *categorical*, then **a** and **b** must be omitted,
- *constantcontinuous*, then **a** is the constant value of that random variable, **b** must be omitted,
- *uniformcontinuous*, then **a** and **b** are the lower and the upper bound of that random variable,
- *gaussian*, then **a** is the mean and **b** is the standard deviation of the gaussian distribution,
- *exponential*, then **a** is the rate parameter of the exponential parameter, **b** must be omitted,
- *empiricalcontinuous*, then **a** and **b** must be omitted.

6.64.3.2 void GoSUM::CScript::addInputs (std::istream & *iscl*) [inline], [private]

Tells project to add multiple input parameters. Format of the related script command is:

add_inputs = sz nm tp a b

where **sz** is the number of input parameters to be added, **nm** is the common part for the input parameters name (must be unique), **tp** is the distribution type, and **a** and **b** are distribution parameters, for some types they must be omitted. Multiple adding of input parameters is possible only for input parameters that have the same distribution, and their names are generated by adding a suffix to the common part of the name. For more details on distribution types and parameters, see description of **add_input** script command.

6.64.3.3 void GoSUM::CScript::addOptimizationConstraint (std::istream & *iscl*) [private]

Tells project to add an optimization constraint. Format of the related script command is:

add_optimization_constraint = gexpr

where **gexpr** is the analytical expression of the constraint.

6.64.3.4 void GoSUM::CScript::addOutput (std::istream & *iscl*) [inline], [private]

Tells project to add one output state. Format of the related script command is:

add_output = nm tp a b

where **nm** is the output state name (must be unique), **tp** is the distribution type, and **a** and **b** are distribution parameters, for some types they must be omitted. For more details on distribution types and parameters, see description of **add_input** script command.

6.64.3.5 void GoSUM::CScript::addOutputs (std::istream & *iscl*) [inline], [private]

Tells project to add multiple output states. Format of the related script command is:

add_outputs = sz nm tp a b

where **sz** is the number of output states to be added, **nm** is the common part for the output state name (must be unique), **tp** is the distribution type, and **a** and **b** are distribution parameters, for some types they must be omitted. Multiple adding of output states is possible only for output states that have the same distribution, and their names are generated by adding a suffix to the common part of the name. For more details, see description of **add_input** script command.

6.64.3.6 void GoSUM::CScript::addVariable (std::istream & *iscl*) [private]

Tells project to add model variable.

6.64.3.7 void GoSUM::CScript::addVariables (std::istream & *iscl*) [private]

Tells project to add model variables.

6.64.3.8 void GoSUM::CScript::computeSensitivities (std::istream & *iscl*) [private]

Tells project to compute sensitivities. Format of the related script command is:

compute_sensitivities

6.64.3.9 void GoSUM::CScript::evaluateOutputs (std::istream & *iscl*) [private]

Tells project to evaluate output states. Format of the related script command is:

evaluate_outputs

6.64.3.10 void GoSUM::CScript::exportAbsoluteAverageDerivative (std::istream & *iscl*) [private]

Tells project to export absolute average derivative. Format of the related script command is:

export_absolute_average_derivative = fn

where **fn** is the name of the export file.

6.64.3.11 void GoSUM::CScript::exportANOVA1 (std::istream & *iscl*) [private]

Tells project to export first order ANOVA. Format of the related script command is:

export_ANOVA1 = fn

where **fn** is the name of the export file.

6.64.3.12 void GoSUM::CScript::exportAverageDerivative (std::istream & *iscl*) [private]

Tells project to export average derivative. Format of the related script command is:

export_average_derivative = fn

where **fn** is the name of the export file.

6.64.3.13 void GoSUM::CScript::exportDerivativeSensitivity (std::istream & *iscl*) [private]

Tells project to export derivative sensitivities. Format of the related script command is:

export_derivative_sensitivity = fn

where **fn** is the name of the export file.

6.64.3.14 void GoSUM::CScript::exportInputSamples (std::istream & *iscl*) [private]

Tells project to export input samples. Format of the related script command is:

export_input_samples = fn

where **fn** is the name of the export file.

6.64.3.15 void GoSUM::CScript::exportOptimizationHistory (std::istream & *iscl*) [private]

Tells project to export optimization history. Format of the related script command is:

export_optimization_history = fn

where **fn** is the name of the export file.

6.64.3.16 void GoSUM::CScript::exportOptimizationMethod (std::istream & *iscl*) [private]

Tells project to export optimization method. Format of the related script command is:

export_optimization_method = fn

where **fn** is the name of the export file.

6.64.3.17 void GoSUM::CScript::exportOutputSamples (std::istream & *iscl*) [private]

Tells project to export output samples. Format of the related script command is:

export_output_samples = fn

where **fn** is the name of the export file.

6.64.3.18 void GoSUM::CScript::exportPredictionOutputSamples (std::istream & *iscl*) [private]

Tells project to export prediction output samples. Format of the related script command is:

export_prediction_output_samples = fn

where **fn** is the name of the export file.

6.64.3.19 void GoSUM::CScript::exportVarianceSensitivity (std::istream & *iscl*) [private]

Tells project to export variance sensitivities. Format of the related script command is:

export_variance_sensitivity = fn

where **fn** is the name of the export file.

6.64.3.20 void GoSUM::CScript::importInputs (std::istream & *iscl*) [inline],[private]

Tells project to import input parameters. Format of the related script command is:

import_inputs = fn

where **fn** is the name of the import file. Import file must have one of the following four formats. (1) In the case of **named theoretical variables format** each line must contain variable name, variable distribution type, and distribution parameters. (2) In the case of **theoretical variables format** each line must contain variable distribution type and distribution parameters. (3) In the case of **named empirical variables format** first line must begin with a # sign and must contain names of all imported variables. The rest of the file must contain sample data, with columns as variables, and rows as samples. (4) In the case of **empirical variables format** file must contain sample data, with columns as variables, and rows as samples. For more details on distribution types and distribution parameters see

description of **add_input** script command. (5) In the case of **named declared empirical variables format** first line must begin with a # sign and must contain names of all imported variables, and second line must begin with a double # sign and must contain types of all imported variables. The rest of the file must contain sample data, with columns as variables, and rows as samples. (6) In the case of **declared empirical variables format** first line must begin with a double # sign and must contain types of all imported variables. The rest of the file must contain sample data, with columns as variables, and rows as samples. For more details on distribution types and distribution parameters see description of **add_input** script command.

6.64.3.21 void GoSUM::CScript::importOutputs (std::istream & *iscl*) [inline], [private]

Tells project to import output states. Format of the related script command is:

import_outputs = fn

where **fn** is the name of the import file. For the description of the import file format see cescription of **import_inputs** command, format cases (3), (4), (5), and (6).

6.64.3.22 void GoSUM::CScript::importPredictionInputSamples (std::istream & *iscl*) [private]

Tells project to import prediction input samples. Format of the related script command is:

import_prediction_input_samples = fn

where **fn** is the name of the import file.

6.64.3.23 void GoSUM::CScript::importVariables (std::istream & *iscl*) [private]

Tells project to import model variables.

6.64.3.24 void GoSUM::CScript::loadModel (std::istream & *iscl*) [private]

Tells project to learn analytical model from the original model. Format of the related script command is:

learn_model

6.64.3.25 std::string GoSUM::CScript::line2Path (std::istream & *line*) [private]

Returns path from the input line.

6.64.3.26 void GoSUM::CScript::load (std::istream & *iscl*) [private]

Tells project to load from binary format. Format of the related script command is:

load

6.64.3.27 void GoSUM::CScript::loadTxt (std::istream & *iscl*) [private]

Tells project to load from txt format. Format of the related script command is:

load_txt

6.64.3.28 void GoSUM::CScript::loadXml (std::istream & *iscl*) [private]

Tells project to load from xml format. Format of the related script command is:

load_xml

6.64.3.29 void GoSUM::CScript::maximize (std::istream & *iscl*) [private]

Tells project to find maximum of the previously defined optimization problem. Format of the related script command is:

maximize

6.64.3.30 void GoSUM::CScript::minimize (std::istream & *iscl*) [private]

Tells project to find minimum of the previously defined optimization problem. Format of the related script command is:

minimize

6.64.3.31 void GoSUM::CScript::predict (std::istream & *iscl*) [private]

Tells project to predict using analytical model. Format of the related script command is:

predict

6.64.3.32 void GoSUM::CScript::read (GoSUM::CContainer * *pContainer*, const std::string & *fileName*)

Reads script file and interprets all script commands.

6.64.3.33 void GoSUM::CScript::reduce (std::istream & *iscl*) [private]

Tells project to reduce model. Format of the related script command is:

reduce

6.64.3.34 void GoSUM::CScript::resampleInputs (std::istream & *iscl*) [private]

Tells project to resample input parameters. Format of the related script command is:

resample_inputs

6.64.3.35 void GoSUM::CScript::save (std::istream & *iscl*) [private]

Tells project to save to binary format. Format of the related script command is:

save

6.64.3.36 void GoSUM::CScript::saveTxt (std::istream & *iscl*) [private]

Tells project to save to txt format. Format of the related script command is:

save_txt

6.64.3.37 void GoSUM::CScript::saveXml (std::istream & *iscl*) [private]

Tells project to save to xml format. Format of the related script command is:

save_xml

6.64.3.38 void GoSUM::CScript::setConstInitialValue (std::istream & *iscl*) [private]

Tells project to set constant initial value. Format of the related script command is:

set_const_upper_bound = x0

where **x0** is the same initial value for all decision variables.

6.64.3.39 void GoSUM::CScript::setConstLowerBound (std::istream & *iscl*) [private]

Tells project to set constant lower bounds. Format of the related script command is:

set_const_lower_bound = x1

where **x1** is the same lower bound value for all decision variables.

6.64.3.40 void GoSUM::CScript::setConstUpperBound (std::istream & *iscl*) [private]

Tells project to set constant upper bound. Format of the related script command is:

set_const_upper_bound = xu

where **xu** is the same upper bound value for all decision variables.

6.64.3.41 void GoSUM::CScript::setCoreSize (std::istream & *iscl*) [private]

Tells project to set core size. Format of the related script command is:

set_core_size = cs

where **cs** is the core size, i.e. number of cores to be used when multithreading is used in computations. The optimal value is one less than the number of computer cores.

6.64.3.42 void GoSUM::CScript::setInitialValue (std::istream & *iscl*) [private]

Tells project to set initial values for all decision variables. Format of the related script command is:

set_initial_value = x1_0 x2_0 ...

where **x1_0** , **x2_0** , etc. are initial values for decision variables.

6.64.3.43 void GoSUM::CScript::setLowerBound (std::istream & *iscl*) [private]

Tells project to set lower bounds for all decision variables. Format of the related script command is:

set_lower_bound = x1_L x2_L ...

where **x1_L** , **x2_L** , etc. are lower bounds for decision variables.

6.64.3.44 void GoSUM::CScript::setMadsInitMeshSize (std::istream & *iscl*) [private]

Tells project to set initial mesh size in MADS. Format of the related script command is:

set_mads_init_mesh_size = ims

where **ims** is the initial mesh size.

6.64.3.45 void GoSUM::CScript::setMadsLHSearch (std::istream & *iscl*) [private]

Tells project to set LH search parameters in MADS. Format of the related script command is:

set_mads_lh_search = lh0 lhi

where **lh0** is the initial LH value, and **lhi** is the LH iteration.

6.64.3.46 `void GoSUM::CScript::setMadsMaxEvaluation (std::istream & iscl) [private]`

Tells project to set maximal number of evaluations in MADS. Format of the related script command is:

set_mads_max_evaluation = maxeval

where **maxeval** is the maximal number of evaluations.

6.64.3.47 `void GoSUM::CScript::setMadsMinPollSize (std::istream & iscl) [private]`

Tells project to set minimal poll size in MADS. Format of the related script command is:

set_mads_min_poll_size = mps

where **mps** is the minimal poll size.

6.64.3.48 `void GoSUM::CScript::setMatLabPath (std::istream & iscl) [private]`

Tells project to set MatLab path. Format of the related script command is:

set_matlab_path = mp

where **mp** is the MatLab path, i.e. the path of the MATLAB.exe.

6.64.3.49 `void GoSUM::CScript::setModelEvaluator (std::istream & iscl) [private]`

Tells project to set model evaluator. Format of the related script command is:

set_model_evaluator = me exe

where **me** is one of the following model evaluator types: *exeascii*, *exemat*, *matlabshell*, *matlabengine*; and **exe** is the name of the executable.

6.64.3.50 `void GoSUM::CScript::setObjective (std::istream & iscl) [private]`

Tells project to set objective. Format of the related script command is:

set_objective = fexpr

where **fexpr** is the analytical expression of the objective function.

6.64.3.51 `void GoSUM::CScript::setOptimizationMethod (std::istream & iscl) [private]`

Tells project to set optimization method. Format of the related script command is:

set_optimization_method = om

where **om** is one of the following optimization methods: *mads*, *ga*.

6.64.3.52 `void GoSUM::CScript::setProjectName (std::istream & iscl) [private]`

Tells project to set project name. Format of the related script command is:

set_project_name = pn

where **pn** is the name of the project.

6.64.3.53 `void GoSUM::CScript::setProjectPath (std::istream & iscl) [private]`

Tells project to set project path. Format of the related script command is:

set_project_path = pp

where **pp** is the path which contains all the project related files, must end with a backslash.

6.64.3.54 `void GoSUM::CScript::setProjectType (std::istream & iscl) [private]`

Tells project to set project type. Format of the related script command is:

set_project_type = pt

where **pt** is one of the following project types: *samplegeneration*, *modelanalysis*, *dataanalysis*, *simpleopt*, *modelopt*, *learnedmodelopt*, *learneddataopt*.

6.64.3.55 `void GoSUM::CScript::setReductionCutoffSize (std::istream & iscl) [private]`

Tells project to set reduction cut. Format of the related script command is:

set_reduction_cutoff_size = c

where **c** is size of chosen top inputs.

6.64.3.56 `void GoSUM::CScript::setReductionCutoffValue (std::istream & iscl) [private]`

Tells project to set reduction cut value. Format of the related script command is:

set_reduction_cutoff_value = c

where **c** is the cutoff value for choosing top inputs.

6.64.3.57 `void GoSUM::CScript::setReductionOutputs (std::istream & iscl) [private]`

Tells project to set/choose outputs for the model reduction. Format of the related script command is:

set_reduction_outputs = n o1 o2 ...

where **rn** is size of chosen outputs, and **o1**, **o2**, etc. are names of the chosen outputs.

6.64.3.58 `void GoSUM::CScript::setReductionType (std::istream & iscl) [private]`

Tells project to set model reduction type. Format of the related script command is:

set_reduction_type = rt

where **rt** is one of the following model reduction types: *derivative*, *averagederivative*, *absoluteaveragederivative*, *variance*.

6.64.3.59 `void GoSUM::CScript::setResampleSize (std::istream & iscl) [private]`

Tells project to set resample size. Format of the related script command is:

set_resample_size = sz

where **sz** is the resampling size, i.e. number of the samples to be generated.

6.64.3.60 void GoSUM::CScript::setResampleType (std::istream & *iscl*) [private]

Tells project to set resample type. Format of the related script command is:

set_resample_type = rg

where **rg** is one of the following sample generator types: *dsample*, *montecarlo*, *cvoronoi*, *lcvoronoi*.

6.64.3.61 void GoSUM::CScript::setRNG (std::istream & *iscl*) [private]

Tells project to set random number generator (RNG) type. Format of the related script command is:

set_rng = rng

where **rng** is one of the following RNG types: *smallperiod* (10^8), *mediumperiod* ($2 * 10^{18}$), *nonlinearcongruential*, *largeperiod* ($3.138 * 10^{57}$), *dsample* (default), *mads*.

6.64.3.62 void GoSUM::CScript::setSensitivityOptions (std::istream & *iscl*) [private]

Tells project to set sensitivity options. Format of the related script command is:

set_sensitivity_options = sz eps1 eps2 eps3

where **sz** is the resized sample size, **eps1** is the sensitivity error (default 0.005), **eps2** is the separability error (default 0.01), and **eps3** is the coupling error (default 0.01).

6.64.3.63 void GoSUM::CScript::setUpperBound (std::istream & *iscl*) [private]

Tells project to set upper bounds for all decision variables. Format of the related script command is:

set_upper_bound = x1_U x2_U ...

where **x1_U**, **x2_U**, etc. are upper bounds for decision variables.

6.64.3.64 void GoSUM::CScript::setVoronoiOptions (std::istream & *iscl*) [private]

Tells project to set Centralized Voronoi Tessellation (CVT) options. Format of the related script command is:

set_voronoi_options = maxiter q alpha2 beta2

where **maxiter** is maximal number of iterations in CVT, and **q**, **alpha2**, and **beta2** are CVT parameters.

6.64.4 Member Data Documentation

6.64.4.1 vector< pair<std::string,paction> > GoSUM::CScript::cmds [private]

Vector of pairs: first is string of the admissible script command, second is pointer to action function.

6.64.4.2 int GoSUM::CScript::noeq [private]

6.64.4.3 void(CContainer::* GoSUM::CScript::padd)(const std::string &, CRandomVariable::distributiontype,double,double) [private]

Pointer to member function that tells project to add model variable.

6.64.4.4 `void(CContainer::* GoSUM::CScript::padds)(int,const std::string &,CRandomVariable::distributiontype,double,double) [private]`

Pointer to member function that tells project to add model variables.

6.64.4.5 `CContainer* GoSUM::CScript::pContainer [private]`

Pointer to [GoSUM](#) project container.

6.64.4.6 `void(CContainer::* GoSUM::CScript::pimport)(const std::string &_fname) [private]`

Pointer to member function that tells project to import data from import file.

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

- C:/Development/core/[Script.h](#)
- C:/Development/core/[Script.cpp](#)

6.65 GoSUM::CSensitivityAnalysis Class Reference

Class for sensitivity analysis of the model.

```
#include <SensitivityAnalysis.h>
```

Public Member Functions

- [CSensitivityAnalysis](#) ([CInputParameters](#) *_pIP, [COutputStates](#) *_pOS, [CA analyticalModel](#) *_pAM)
- virtual [~CSensitivityAnalysis](#) ()
- void [clear](#) ()
- void [setSensitivityOptions](#) (int _RSsize, double _eps1, double _eps2, double _eps3)
- int [resampleSize](#) () const
Returns resample size.
- void [setResampleSize](#) (int _RSsize)
Sets resample size.
- double [sensitivityError](#) () const
Returns sensitivity error.
- void [setSensitivityError](#) (double _eps1)
Sets sensitivity error.
- double [separabilityError](#) () const
Returns separability error.
- void [setSeparabilityError](#) (double _eps2)
Sets separability error.
- double [couplingError](#) () const
Returns coupling error.
- void [setCouplingError](#) (double _eps3)
Sets coupling error.
- void [prepare](#) ()
Prepares sensitivity analysis computation.
- void [computeDerivativeSensitivities](#) ()
Computes derivative sensitivities.
- void [computeVarianceSensitivities](#) ()

- Computes variance sensitivities.*

 - void `computeVarianceSensitivities1` ()

Computes variance sensitivities (Homma & Saltelli approach).

 - void `computeSensitivities` ()

Computes sensitivities.

 - const ArrayXXd & `derivativeSensitivity` ()

Returns derivative sensitivity (L2) for all output states (rows) over all input parameters (columns).

 - const ArrayXXd & `averageDerivative` ()

Returns average derivative for all output states (rows) over all input parameters (columns).

 - const ArrayXXd & `absoluteAverageDerivative` ()

Returns absolute average derivative (L1) for all output states (rows) over all input parameters (columns).

 - const ArrayXXd & `varianceSensitivity` ()

Returns variance sensitivity (L2) for all output states (rows) over all input parameters (columns).

 - const ArrayXXd & `firstOrderANOVA` ()

Returns first order ANOVA for all output states (rows) over all input parameters (columns).

 - std::vector< std::pair< int, int > > `secondOrderANOVAPairs` ()

Returns second order ANOVA pairs.

 - ArrayXXd `secondOrderANOVA` (const std::vector< std::pair< int, int > > &a2L)
 - bool `emptyDerivativeSensitivity` () const

Returns second order ANOVA indices for given pairs.

 - bool `emptyAverageDerivativeSensitivity` () const

Returns true if average derivative is empty, false otherwise.

 - bool `emptyAbsoluteAverageDerivativeSensitivity` () const

Returns true if absolute average derivative (L1) is empty, false otherwise.

 - bool `emptyVarianceSensitivity` () const

Returns true if variance sensitivity (L2) is empty, false otherwise.

 - bool `emptyFirstOrderANOVA` () const

Returns true if 1st order ANOVA is empty, false otherwise.

 - bool `empty` () const

Returns true if all sensitivity indices are empty, false otherwise.

 - int `progressStepsSize` () const

Returns progress steps size.

Public Attributes

- boost::signal< void()> `computingProgressed`
- Signal for computing progress, emitted on each sensitivity index computed.*

Private Member Functions

- template<class Archive >
void `serialize` (Archive &ar, const unsigned int version)
- `CSensitivityAnalysis` ()

Private Attributes

- [CInputParameters](#) * [pIP](#)
Points to input parameters.
- [COutputStates](#) * [pOS](#)
Points to output states.
- [CA analyticalModel](#) * [pAM](#)
Points to the analytical model.
- int [RSize](#)
Resized sample size.
- double [eps1](#)
Sensitivity error.
- double [eps2](#)
Separability error.
- double [eps3](#)
Coupling error.
- std::vector< ArrayXd > [sample1](#)
- std::vector< ArrayXd > [sample2](#)
- ArrayXXd [y](#)
- ArrayXd [yvar](#)
- ArrayXd [ymu](#)
- ArrayXXd [si1der](#)
Holds derivative sensitivity (L2) for all output states (rows) over all input parameters (columns).
- ArrayXXd [ader](#)
Holds average derivative for all output states (rows) over all input parameters (columns).
- ArrayXXd [aader](#)
Holds absolute average derivative (L1) for all output states (rows) over all input parameters (columns).
- ArrayXXd [siT](#)
Holds variance sensitivity (L2) for all output states (rows) over all input parameters (columns).
- ArrayXXd [si1](#)
Holds first order ANOVA sensitivity for all output states (rows) over all input parameters (columns).
- std::vector< std::vector
< std::pair< std::pair< int,
int >, double > > > [si2](#)
Holds second order ANOVA sensitivity for all output states over all input parameter pairs.

Friends

- class [boost::serialization::access](#)

6.65.1 Detailed Description

Class for sensitivity analysis of the model.

6.65.2 Constructor & Destructor Documentation

6.65.2.1 `GoSUM::CSensitivityAnalysis::CSensitivityAnalysis () [inline], [private]`

6.65.2.2 `GoSUM::CSensitivityAnalysis::CSensitivityAnalysis (CInputParameters * _pIP, COutputStates * _pOS, CA analyticalModel * _pAM) [inline]`

6.65.2.3 `virtual GoSUM::CSensitivityAnalysis::~~CSensitivityAnalysis () [inline], [virtual]`

6.65.3 Member Function Documentation

6.65.3.1 `const ArrayXXd& GoSUM::CSensitivityAnalysis::absoluteAverageDerivative () [inline]`

Returns absolute average derivative (L1) for all output states (rows) over all input parameters (columns).

6.65.3.2 `const ArrayXXd& GoSUM::CSensitivityAnalysis::averageDerivative () [inline]`

Returns average derivative for all output states (rows) over all input parameters (columns).

6.65.3.3 `void GoSUM::CSensitivityAnalysis::clear () [inline]`

6.65.3.4 `void GoSUM::CSensitivityAnalysis::computeDerivativeSensitivities ()`

Computes derivative sensitivities.

6.65.3.5 `void GoSUM::CSensitivityAnalysis::computeSensitivities ()`

Computes sensitivities.

6.65.3.6 `void GoSUM::CSensitivityAnalysis::computeVarianceSensitivities ()`

Computes variance sensitivities.

6.65.3.7 `void GoSUM::CSensitivityAnalysis::computeVarianceSensitivities1 ()`

Computes variance sensitivities (Homma & Saltelli approach).

6.65.3.8 `double GoSUM::CSensitivityAnalysis::couplingError () const [inline]`

Returns coupling error.

6.65.3.9 `const ArrayXXd& GoSUM::CSensitivityAnalysis::derivativeSensitivity () [inline]`

Returns derivative sensitivity (L2) for all output states (rows) over all input parameters (columns).

6.65.3.10 `bool GoSUM::CSensitivityAnalysis::empty () const [inline]`

Returns true if all sentitivity indices are empty, false otherwise.

6.65.3.11 `bool GoSUM::CSensitivityAnalysis::emptyAbsoluteAverageDerivativeSensitivity () const [inline]`

Returns true if absolute average derivative (L1) is empty, false otherwise.

6.65.3.12 `bool GoSUM::CSensitivityAnalysis::emptyAverageDerivativeSensitivity () const [inline]`

Returns true if average derivative is empty, false otherwise.

6.65.3.13 `bool GoSUM::CSensitivityAnalysis::emptyDerivativeSensitivity () const` `[inline]`

Returns second order ANOVA indices for given pairs.

Returns true if derivative sensitivity (L2) is empty, false otherwise.

6.65.3.14 `bool GoSUM::CSensitivityAnalysis::emptyFirstOrderANOVA () const` `[inline]`

Returns true if 1st order ANOVA is empty, false otherwise.

6.65.3.15 `bool GoSUM::CSensitivityAnalysis::emptyVarianceSensitivity () const` `[inline]`

Returns true if variance sensitivity (L2) is empty, false otherwise.

6.65.3.16 `const ArrayXXd& GoSUM::CSensitivityAnalysis::firstOrderANOVA ()` `[inline]`

Returns first order ANOVA for all output states (rows) over all input parameters (columns).

6.65.3.17 `void GoSUM::CSensitivityAnalysis::prepare ()`

Prepares sensitivity analysis computation.

6.65.3.18 `int GoSUM::CSensitivityAnalysis::progressStepsSize () const` `[inline]`

Returns progress steps size.

6.65.3.19 `int GoSUM::CSensitivityAnalysis::resampleSize () const` `[inline]`

Returns resample size.

6.65.3.20 `ArrayXXd GoSUM::CSensitivityAnalysis::secondOrderANOVA (const std::vector< std::pair< int, int > > & a2L)`

6.65.3.21 `std::vector< std::pair< int, int > > GoSUM::CSensitivityAnalysis::secondOrderANOVAPairs ()`

Returns second order ANOVA pairs.

6.65.3.22 `double GoSUM::CSensitivityAnalysis::sensitivityError () const` `[inline]`

Returns sensitivity error.

6.65.3.23 `double GoSUM::CSensitivityAnalysis::separabilityError () const` `[inline]`

Returns separability error.

6.65.3.24 `template<class Archive > void GoSUM::CSensitivityAnalysis::serialize (Archive & ar, const unsigned int version)`
`[private]`

6.65.3.25 `void GoSUM::CSensitivityAnalysis::setCouplingError (double _eps3)` `[inline]`

Sets coupling error.

6.65.3.26 `void GoSUM::CSensitivityAnalysis::setResampleSize (int _Rssize) [inline]`

Sets resample size.

6.65.3.27 `void GoSUM::CSensitivityAnalysis::setSensitivityError (double _eps1) [inline]`

Sets sensitivity error.

6.65.3.28 `void GoSUM::CSensitivityAnalysis::setSensitivityOptions (int _Rssize, double _eps1, double _eps2, double _eps3) [inline]`

Parameters

<i>_eps3</i>	Sets options for sensitivity analysis.
--------------	--

6.65.3.29 `void GoSUM::CSensitivityAnalysis::setSeparabilityError (double _eps2) [inline]`

Sets separability error.

6.65.3.30 `const ArrayXXd& GoSUM::CSensitivityAnalysis::varianceSensitivity () [inline]`

Returns variance sensitivity (L2) for all output states (rows) over all input parameters (columns).

6.65.4 Friends And Related Function Documentation

6.65.4.1 `friend class boost::serialization::access [friend]`

6.65.5 Member Data Documentation

6.65.5.1 `ArrayXXd GoSUM::CSensitivityAnalysis::aader [private]`

Holds absolute average derivative (L1) for all output states (rows) over all input parameters (columns).

6.65.5.2 `ArrayXXd GoSUM::CSensitivityAnalysis::ader [private]`

Holds average derivative for all output states (rows) over all input parameters (columns).

6.65.5.3 `boost::signal<void()> GoSUM::CSensitivityAnalysis::computingProgressed`

Signal for computing progress, emitted on each sensitivity index computed.

6.65.5.4 `double GoSUM::CSensitivityAnalysis::eps1 [private]`

Sensitivity error.

6.65.5.5 `double GoSUM::CSensitivityAnalysis::eps2 [private]`

Separability error.

6.65.5.6 `double GoSUM::CSensitivityAnalysis::eps3` [private]

Coupling error.

6.65.5.7 `CA analyticalModel* GoSUM::CSensitivityAnalysis::pAM` [private]

Points to the analytical model.

6.65.5.8 `CInputParameters* GoSUM::CSensitivityAnalysis::pIP` [private]

Points to input parameters.

6.65.5.9 `COutputStates* GoSUM::CSensitivityAnalysis::pOS` [private]

Points to output states.

6.65.5.10 `int GoSUM::CSensitivityAnalysis::RSize` [private]

Resized sample size.

6.65.5.11 `std::vector<ArrayXd> GoSUM::CSensitivityAnalysis::sample1` [private]

6.65.5.12 `std::vector<ArrayXd> GoSUM::CSensitivityAnalysis::sample2` [private]

6.65.5.13 `ArrayXXd GoSUM::CSensitivityAnalysis::si1` [private]

Holds first order ANOVA sensitivity for all output states (rows) over all input parameters (columns).

6.65.5.14 `ArrayXXd GoSUM::CSensitivityAnalysis::si1der` [private]

Holds derivative sensitivity (L2) for all output states (rows) over all input parameters (columns).

6.65.5.15 `std::vector<std::vector<std::pair<std::pair<int,int>,double> > > GoSUM::CSensitivityAnalysis::si2`
[private]

Holds second order ANOVA sensitivity for all output states over all input parameter pairs.

6.65.5.16 `ArrayXXd GoSUM::CSensitivityAnalysis::siT` [private]

Holds variance sensitivity (L2) for all output states (rows) over all input parameters (columns).

6.65.5.17 `ArrayXXd GoSUM::CSensitivityAnalysis::y` [private]

6.65.5.18 `ArrayXd GoSUM::CSensitivityAnalysis::ymu` [private]

6.65.5.19 `ArrayXd GoSUM::CSensitivityAnalysis::yvar` [private]

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

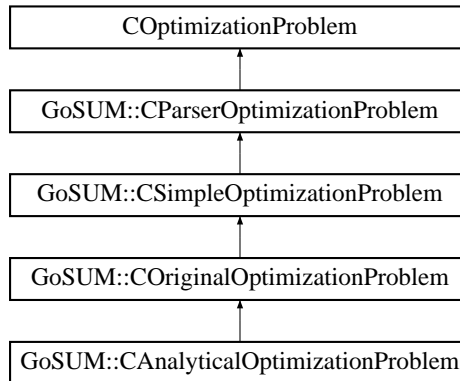
- C:/Development/core/[SensitivityAnalysis.h](#)
- C:/Development/core/[SensitivityAnalysis.cpp](#)

6.66 GoSUM::CSimpleOptimizationProblem Class Reference

Class for the optimization problem based only on [GoSUM](#) input parameters.

```
#include <ParserOptimizationProblem.h>
```

Inheritance diagram for GoSUM::CSimpleOptimizationProblem:



Public Member Functions

- [CSimpleOptimizationProblem](#) ([CInputParameters](#) * _pIP)
- virtual [~CSimpleOptimizationProblem](#) ()
- virtual void [clear](#) ()
Clears all.
- virtual void [clearHistory](#) ()
Clears results.
- int [findConnectedTo](#) ([CModelVariable](#) * _pip) const
Finds optimization variable connected to a particular input parameter.
- const [CModelOptimizationVariable](#) & [variable](#) (int _at) const
Returns particular optimization variable.
- [CModelOptimizationVariable](#) & [variable](#) (int _at)
Returns particular optimization variable.
- std::string [variableName](#) (int _at)
Returns name of the particular optimization variable.
- virtual bool [isFeasible](#) (const ArrayXd & _ip)
Returns true if model variables is admissible, false otherwise.
- virtual void [openOptimization](#) ()
- virtual void [closeOptimization](#) ()
Opens, i.e. prepares optimization.
- virtual bool [evaluate](#) (const ArrayXd & _hp, ArrayXd & _ep)
Closes optimization, i.e. closes what was opened in openOptimization.
- ArrayXd [inputPoint2ModelPoint](#) (const ArrayXd & _ip)
Converts input parameter point to model point.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- [CSimpleOptimizationProblem](#) ()
- virtual void [setVariableNames](#) ()
Sets variable names for the parser.

Static Protected Member Functions

- static bool [isModelVariable](#) (const [COptimizationVariable](#) &_aOV)
Used in member function Find.

Protected Attributes

- [CInputParameters](#) * [pIP](#)
Points to input parameters.
- std::vector< int > [ovind](#)
- std::vector< int > [uvind](#)
Vectors of indices of optimization variables (ovind) and uncertain variables (uvind) among input parameters.
- std::vector< ArrayXd > [samples](#)

Static Protected Attributes

- static [CModelVariable](#) * [pipToFind](#) = NULL
Used in member function Find.

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.66.1 Detailed Description

Class for the optimization problem based only on [GoSUM](#) input parameters.

6.66.2 Constructor & Destructor Documentation

6.66.2.1 [GoSUM::CSimpleOptimizationProblem::CSimpleOptimizationProblem \(\)](#) [inline],[protected]

6.66.2.2 [GoSUM::CSimpleOptimizationProblem::CSimpleOptimizationProblem \(CInputParameters * _pIP \)](#) [inline]

6.66.2.3 [virtual GoSUM::CSimpleOptimizationProblem::~~CSimpleOptimizationProblem \(\)](#) [inline],[virtual]

6.66.3 Member Function Documentation

6.66.3.1 [virtual void GoSUM::CSimpleOptimizationProblem::clear \(\)](#) [inline],[virtual]

Clears all.

Reimplemented from [COptimizationProblem](#).

6.66.3.2 [virtual void GoSUM::CSimpleOptimizationProblem::clearHistory \(\)](#) [inline],[virtual]

Clears results.

Reimplemented from [COptimizationProblem](#).

6.66.3.3 void GoSUM::CSimpleOptimizationProblem::closeOptimization () [virtual]

Opens, i.e. prepares optimization.

Reimplemented from [GoSUM::CParserOptimizationProblem](#).

Reimplemented in [GoSUM::COriginalOptimizationProblem](#).

6.66.3.4 bool GoSUM::CSimpleOptimizationProblem::evaluate (const ArrayXd & *_hp*, ArrayXd & *_ep*) [virtual]

Closes optimization, i.e. closes what was opened in openOptimization.

Evaluates objective and all constraints from optimization variables values and returns true if it is feasible, false otherwise.

Reimplemented from [COptimizationProblem](#).

6.66.3.5 int GoSUM::CSimpleOptimizationProblem::findConnectedTo (CModelVariable * *_pip*) const

Finds optimization variable connected to a particular input parameter.

6.66.3.6 ArrayXd GoSUM::CSimpleOptimizationProblem::inputPoint2ModelPoint (const ArrayXd & *_ip*) [inline]

Converts input parameter point to model point.

Reimplemented in [GoSUM::CA analyticalOptimizationProblem](#), and [GoSUM::COriginalOptimizationProblem](#).

6.66.3.7 virtual bool GoSUM::CSimpleOptimizationProblem::isFeasible (const ArrayXd & *_ip*) [inline],[virtual]

Returns true if model variables is admissible, false otherwise.

Reimplemented from [COptimizationProblem](#).

6.66.3.8 static bool GoSUM::CSimpleOptimizationProblem::isModelVariable (const COptimizationVariable & *_aOV*)
[inline],[static],[protected]

Used in member function Find.

6.66.3.9 void GoSUM::CSimpleOptimizationProblem::openOptimization () [virtual]

Reimplemented from [GoSUM::CParserOptimizationProblem](#).

Reimplemented in [GoSUM::COriginalOptimizationProblem](#).

6.66.3.10 template<class Archive > void GoSUM::CSimpleOptimizationProblem::serialize (Archive & *ar*, const unsigned int *version*) [protected]

Reimplemented from [GoSUM::CParserOptimizationProblem](#).

Reimplemented in [GoSUM::CA analyticalOptimizationProblem](#), and [GoSUM::COriginalOptimizationProblem](#).

6.66.3.11 void GoSUM::CSimpleOptimizationProblem::setVariableNames () [protected],[virtual]

Sets variable names for the parser.

Implements [GoSUM::CParserOptimizationProblem](#).

Reimplemented in [GoSUM::COriginalOptimizationProblem](#).

6.66.3.12 `const CModelOptimizationVariable& GoSUM::CSimpleOptimizationProblem::variable (int _at) const`
`[inline]`

Returns particular optimization variable.

6.66.3.13 `CModelOptimizationVariable& GoSUM::CSimpleOptimizationProblem::variable (int _at)` `[inline]`

Returns particular optimization variable.

6.66.3.14 `std::string GoSUM::CSimpleOptimizationProblem::variableName (int _at)` `[inline]`

Returns name of the particular optimization variable.

6.66.4 Friends And Related Function Documentation

6.66.4.1 `friend class boost::serialization::access` `[friend]`

Boost serialization.

6.66.5 Member Data Documentation

6.66.5.1 `std::vector<int> GoSUM::CSimpleOptimizationProblem::ovind` `[protected]`

6.66.5.2 `CInputParameters* GoSUM::CSimpleOptimizationProblem::pip` `[protected]`

Points to input parameters.

6.66.5.3 `GoSUM::CModelVariable * GoSUM::CSimpleOptimizationProblem::pipToFind = NULL` `[static]`,
`[protected]`

Used in member function Find.

6.66.5.4 `std::vector<ArrayXd> GoSUM::CSimpleOptimizationProblem::samples` `[protected]`

Samples for the uncertain variables.

6.66.5.5 `std::vector<int> GoSUM::CSimpleOptimizationProblem::uvid` `[protected]`

Vectors of indices of optimization variables (ovind) and uncertain variables (uvid) among input parameters.

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

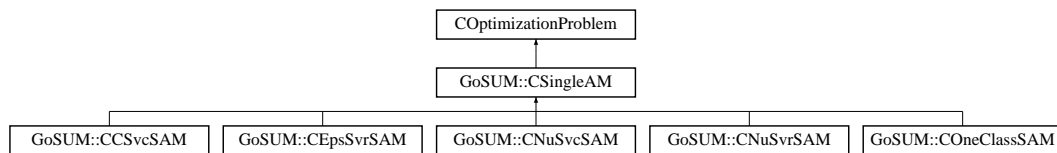
- C:/Development/core/[ParserOptimizationProblem.h](#)
- C:/Development/core/[ParserOptimizationProblem.cpp](#)

6.67 GoSUM::CSingleAM Class Reference

Base class for the analytical model for single output state, interface between analytical model and SVM.

`#include <AnalyticalModel.h>`

Inheritance diagram for GoSUM::CSingleAM:



Public Member Functions

- [CSingleAM](#) ()
- virtual [~CSingleAM](#) ()
- void [learn](#) (int _osi, [CMADS](#) &_mads, std::ostream &_out=std::cout)
Learns SVM models for all output states.
- virtual int [constraintsSize](#) () const
Returns size of the constraints.
- virtual double [constraint](#) (const ArrayXd &mv, int _at)
Evaluates particular constraint value from model variables values.
- virtual ArrayXd [modelVariables](#) (const ArrayXd &x) const
Returns model variables values from optimization variables values.
- virtual double [objective](#) (const ArrayXd &ov)
Evaluates objective function value from optimization variables values.
- virtual void [openOptimization](#) ()
Opens optimization.
- virtual void [optimizationPoint2SVMParam](#) (const ArrayXd &ov)
Converts optimization point to SVM parameters.
- double [f](#) (const ArrayXd &x) const
Returns value of the output state according to the single output state analytical model.
- ArrayXd [df](#) (const ArrayXd &x) const
Returns gradient of the output state according to the single output state the analytical model.

Static Public Member Functions

- static [GoSUM::CSingleAM * New](#) (const [CA analyticalModel](#) *_pAM)
Returns new SingleAM of type defined in _pAM.
- static void [OpenSVM](#) ([CInputParameters](#) *_pIP, [COutputStates](#) *_pOS)
Opens SVM learning (sets prob and xspace).
- static void [CloseSVM](#) ()
Closes SVM learning.
- static void [SetMaximalLearningSize](#) (int _maxlearnsz)
Sets maximal (sample) size used in learning.
- static int [MaximalLearningSize](#) ()
Returns maximal (sample) size used in learning.
- static void [SetLearningFraction](#) (int _learnfrac)
Sets (sample) fraction used in learning.
- static int [LearningFraction](#) ()
Returns (sample) fraction used in learning.

Public Attributes

- boost::signal< void()> [learningProgressed](#)
Signal for learning progress, emitted on every objective evaluation.

Protected Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
- double [rbfKernel](#) (const ArrayXd &x, int j) const
Returns RBF kernel for jth support vector.

Protected Attributes

- struct svm_parameter [param](#)
SVM format specific.
- struct svm_model * [model](#)
SVM format specific.
- int [osi](#)
Index of the related output state.
- double [gamma](#)
- double [C](#)
- double [p](#)
- double [nu](#)
SVM exponent constant in the RBF kernel case.
- double [coef0](#)
SVM free coefficient in the polynomial kernel case.
- int [degree](#)
SVM degree in the polynomial kernel case.
- int [Nsvm](#)
SVM number of support vectors.
- ArrayXd [coeff](#)
SVM coefficients.
- std::vector< ArrayXd > [Xsv](#)

Static Protected Attributes

- static [CInputParameters](#) * [pIP](#) = NULL
Points to input parameters.
- static [COutputStates](#) * [pOS](#) = NULL
Points to output states.
- static ArrayXi [perm](#)
Permutation of sample indices.
- static struct svm_problem [prob](#)
SVM format specific.
- static std::vector< struct
svm_node > [xspace](#)
SVM format specific.
- static int [maxlearnsize](#) = 1000
- static double [learnfrac](#) = 0.05

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.67.1 Detailed Description

Base class for the analytical model for single output state, interface between analytical model and SVM.

6.67.2 Constructor & Destructor Documentation

6.67.2.1 `GoSUM::CSingleAM::CSingleAM () [inline]`

6.67.2.2 `virtual GoSUM::CSingleAM::~~CSingleAM () [inline],[virtual]`

6.67.3 Member Function Documentation

6.67.3.1 `void GoSUM::CSingleAM::CloseSVM () [static]`

Closes SVM learning.

6.67.3.2 `virtual double GoSUM::CSingleAM::constraint (const ArrayXd & mv, int _at) [inline],[virtual]`

Evaluates particular constraint value from model variables values.

Implements [COptimizationProblem](#).

6.67.3.3 `virtual int GoSUM::CSingleAM::constraintsSize () const [inline],[virtual]`

Returns size of the constraints.

Implements [COptimizationProblem](#).

6.67.3.4 `ArrayXd GoSUM::CSingleAM::df (const ArrayXd & x) const`

Returns gradient of the output state according to the single output state the analytical model.

6.67.3.5 `double GoSUM::CSingleAM::f (const ArrayXd & x) const`

Returns value of the output state according to the single output state analytical model.

6.67.3.6 `void GoSUM::CSingleAM::learn (int _osi, CMADS & _mads, std::ostream & _out = std::cout)`

Learns SVM models for all output states.

6.67.3.7 `static int GoSUM::CSingleAM::LearningFraction () [inline],[static]`

Returns (sample) fraction used in learning.

6.67.3.8 `static int GoSUM::CSingleAM::MaximalLearningSize () [inline],[static]`

Returns maximal (sample) size used in learning.

6.67.3.9 `virtual ArrayXd GoSUM::CSingleAM::modelVariables (const ArrayXd & x) const [inline],[virtual]`

Returns model variables values from optimization variables values.

6.67.3.10 `GoSUM::CSingleAM * GoSUM::CSingleAM::New (const CAntalyticalModel * _pAM) [static]`

Returns new SingleAM of type defined in _pAM.

6.67.3.11 `double GoSUM::CSingleAM::objective (const ArrayXd & ov) [virtual]`

Evaluates objective function value from optimization variables values.

Implements [COptimizationProblem](#).

6.67.3.12 `void GoSUM::CSingleAM::openOptimization () [virtual]`

Opens optmization.

Reimplemented from [COptimizationProblem](#).

Reimplemented in [GoSUM::CNuSvrSAM](#), and [GoSUM::CEpsSvrSAM](#).

6.67.3.13 `void GoSUM::CSingleAM::OpenSVM (CInputParameters * _pIP, COutputStates * _pOS) [static]`

Opens SVM learning (sets prob and xspace).

6.67.3.14 `virtual void GoSUM::CSingleAM::optimizationPoint2SVMParam (const ArrayXd & ov) [inline],
[virtual]`

Converts optimization point to SVM parameters.

Reimplemented in [GoSUM::CNuSvrSAM](#), and [GoSUM::CEpsSvrSAM](#).

6.67.3.15 `double GoSUM::CSingleAM::rbfKernel (const ArrayXd & x, int j) const [inline], [protected]`

Returns RBF kernel for jth support vector.

6.67.3.16 `template<class Archive > void GoSUM::CSingleAM::serialize (Archive & ar, const unsigned int version)
[protected]`

Reimplemented from [COptimizationProblem](#).

Reimplemented in [GoSUM::CNuSvrSAM](#), [GoSUM::CEpsSvrSAM](#), [GoSUM::COneClassSAM](#), [GoSUM::CNuSvcSAM](#), and [GoSUM::CCSvcSAM](#).

6.67.3.17 `static void GoSUM::CSingleAM::SetLearningFraction (int _learnfrac) [inline], [static]`

Sets (sample) fraction used in learning.

6.67.3.18 `static void GoSUM::CSingleAM::SetMaximalLearningSize (int _maxlearnsz) [inline], [static]`

Sets maximal (sample) size used in learning.

6.67.4 Friends And Related Function Documentation

6.67.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.67.5 Member Data Documentation

6.67.5.1 `double GoSUM::CSingleAM::C` [protected]

6.67.5.2 `double GoSUM::CSingleAM::coef0` [protected]

SVM free coefficient in the polynomial kernel case.

6.67.5.3 `ArrayXd GoSUM::CSingleAM::coeff` [protected]

SVM coefficients.

6.67.5.4 `int GoSUM::CSingleAM::degree` [protected]

SVM degree in the polynomial kernel case.

6.67.5.5 `double GoSUM::CSingleAM::gamma` [protected]

6.67.5.6 `double GoSUM::CSingleAM::learnfrac = 0.05` [static], [protected]

6.67.5.7 `boost::signal<void()> GoSUM::CSingleAM::learningProgressed`

Signal for learning progress, emitted on every objective evaluation.

6.67.5.8 `int GoSUM::CSingleAM::maxlearnsize = 1000` [static], [protected]

6.67.5.9 `struct svm_model* GoSUM::CSingleAM::model` [protected]

SVM format specific.

6.67.5.10 `int GoSUM::CSingleAM::Nsvm` [protected]

SVM number of support vectors.

6.67.5.11 `double GoSUM::CSingleAM::nu` [protected]

SVM exponent constant in the RBF kernel case.

6.67.5.12 `int GoSUM::CSingleAM::osi` [protected]

Index of the related output state.

6.67.5.13 `double GoSUM::CSingleAM::p` [protected]

6.67.5.14 `struct svm_parameter GoSUM::CSingleAM::param` [protected]

SVM format specific.

6.67.5.15 `ArrayXi GoSUM::CSingleAM::perm` [static], [protected]

Permutation of sample indices.

6.67.5.16 `GoSUM::CInputParameters * GoSUM::CSingleAM::pIP = NULL` `[static], [protected]`

Points to input parameters.

6.67.5.17 `GoSUM::COutputStates * GoSUM::CSingleAM::pOS = NULL` `[static], [protected]`

Points to output states.

6.67.5.18 `struct svm_problem GoSUM::CSingleAM::prob` `[static], [protected]`

SVM format specific.

6.67.5.19 `std::vector< struct svm_node > GoSUM::CSingleAM::xspace` `[static], [protected]`

SVM format specific.

6.67.5.20 `std::vector<ArrayXd> GoSUM::CSingleAM::Xsv` `[protected]`

SVM support vectors.

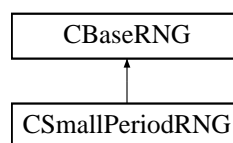
The documentation for this class was generated from the following files:

- C:/Development/core/[AnalyticalModel.h](#)
- C:/Development/core/[AnalyticalModel.cpp](#)

6.68 CSmallPeriodRNG Class Reference

```
#include <RandomGenerators.h>
```

Inheritance diagram for CSmallPeriodRNG:



Public Member Functions

- [CSmallPeriodRNG](#) ()
- [CSmallPeriodRNG](#) (unsigned int s)
- virtual [~CSmallPeriodRNG](#) ()
- virtual void [setSeed](#) (unsigned int s)
Sets seed of the RNG.
- virtual unsigned int [rndi](#) ()
Returns randomly generated unsigned int.
- virtual double [rnd](#) ()
Returns randomly generated double between 0 and 1.

Static Private Attributes

- static long `iy1` = 0
Paramters of the small period RNG.
- static long `iv1` [NTAB]
Paramters of the small period RNG.
- static long `idum1` = 0
Paramters of the small period RNG.

6.68.1 Constructor & Destructor Documentation

6.68.1.1 `CSmallPeriodRNG::CSmallPeriodRNG ()` [inline]

6.68.1.2 `CSmallPeriodRNG::CSmallPeriodRNG (unsigned int s)` [inline]

6.68.1.3 `virtual CSmallPeriodRNG::~~CSmallPeriodRNG ()` [inline], [virtual]

6.68.2 Member Function Documentation

6.68.2.1 `double CSmallPeriodRNG::rnd ()` [virtual]

Returns randomly generated double between 0 and 1.

Implements [CBaseRNG](#).

6.68.2.2 `virtual unsigned int CSmallPeriodRNG::rmdi ()` [inline], [virtual]

Returns randomly generated unsigned int.

Implements [CBaseRNG](#).

6.68.2.3 `void CSmallPeriodRNG::setSeed (unsigned int s)` [virtual]

Sets seed of the RNG.

Implements [CBaseRNG](#).

6.68.3 Member Data Documentation

6.68.3.1 `long CSmallPeriodRNG::idum1 = 0` [static], [private]

Paramters of the small period RNG.

6.68.3.2 `long CSmallPeriodRNG::iv1` [static], [private]

Paramters of the small period RNG.

6.68.3.3 `long CSmallPeriodRNG::iy1 = 0` [static], [private]

Paramters of the small period RNG.

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

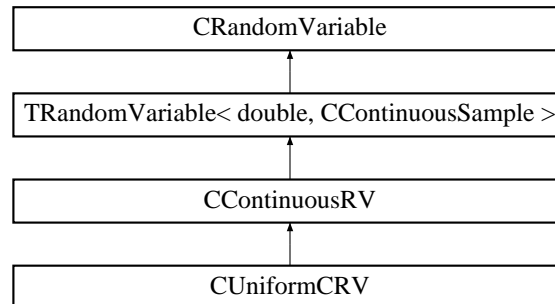
- `C:/Development/core/RandomGenerators.h`
- `C:/Development/core/RandomGenerators.cpp`

6.69 CUniformCRV Class Reference

Class for uniform continuous random variables derived from continuous random variables.

```
#include <RandomVariable.h>
```

Inheritance diagram for CUniformCRV:



Public Member Functions

- [CUniformCRV](#) ()
- virtual [~CUniformCRV](#) ()
- [CUniformCRV](#) (const [CUniformCRV](#) &O)
- virtual void [setDistribution](#) (double _a, double _b)
Set distribution parameters.
- virtual void [setDistribution](#) (const [CContinuousSample](#) &_aS)
Set distribution parameters from sample empirical parameters.
- virtual double [probability](#) (double _x) const
Function that returns PDF.
- virtual double [cumulative](#) (double _x) const
Function that returns CDF.
- virtual double [minValue](#) () const
Returns minimal value of the random variable.
- virtual double [maxValue](#) () const
Returns maximal value of the random variable.
- virtual double [expectedValue](#) () const
Returns expected value of the random variable.
- virtual [distributiontype](#) [distributionType](#) () const
Returns enum type of the random variable distribution.
- virtual std::string [distributionName](#) () const
Returns name of the random variable distribution.
- virtual ArrayXd [exportDomain](#) () const
Exports domain of the random variable.
- double [lowerBound](#) () const
Returns lower bound of the uniform random variable.
- double [upperBound](#) () const
Returns upper bound of the uniform random variable.
- void [setLowerBound](#) (double _a)
Sets lower bound of the uniform random variable.
- void [setUpperBound](#) (double _b)
Sets upper bound of the uniform random variable.
- virtual double [variance](#) () const
Returns variance of the random variable.

Protected Member Functions

- virtual double [doQuantile](#) (double _p) const
Quantile, formula implementation.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Private Attributes

- double [a](#)
- double [b](#)
- double [d](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.69.1 Detailed Description

Class for uniform continuous random variables derived from continuous random variables.

6.69.2 Constructor & Destructor Documentation

6.69.2.1 CUniformCRV::CUniformCRV () `[inline]`

6.69.2.2 virtual CUniformCRV::~~CUniformCRV () `[inline], [virtual]`

6.69.2.3 CUniformCRV::CUniformCRV (const CUniformCRV & O) `[inline]`

6.69.3 Member Function Documentation

6.69.3.1 virtual double CUniformCRV::cumulative (double _x) const `[inline], [virtual]`

Function that returns CDF.

6.69.3.2 virtual std::string CUniformCRV::distributionName () const `[inline], [virtual]`

Returns name of the random variable distribution.

Implements [CRandomVariable](#).

6.69.3.3 virtual distributiontype CUniformCRV::distributionType () const `[inline], [virtual]`

Returns enum type of the random variable distribution.

Implements [CRandomVariable](#).

6.69.3.4 `virtual double CUniformCRV::doQuantile (double _p) const` `[inline], [protected], [virtual]`

Quantile, formula implementation.

Implements [CRandomVariable](#).

6.69.3.5 `virtual double CUniformCRV::expectedValue () const` `[inline], [virtual]`

Returns expected value of the random variable.

Implements [CRandomVariable](#).

6.69.3.6 `ArrayXd CUniformCRV::exportDomain () const` `[virtual]`

Exports domain of the random variable.

Implements [TRandomVariable< t, T >](#).

6.69.3.7 `double CUniformCRV::lowerBound () const` `[inline]`

Returns lower bound of the uniform random variable.

6.69.3.8 `virtual double CUniformCRV::maxValue () const` `[inline], [virtual]`

Returns maximal value of the random variable.

Implements [CRandomVariable](#).

6.69.3.9 `virtual double CUniformCRV::minValue () const` `[inline], [virtual]`

Returns minimal value of the random variable.

Implements [CRandomVariable](#).

6.69.3.10 `virtual double CUniformCRV::probability (double _x) const` `[inline], [virtual]`

Function that returns PDF.

6.69.3.11 `template<class Archive > void CUniformCRV::serialize (Archive & ar, const unsigned int version)` `[private]`

6.69.3.12 `void CUniformCRV::setDistribution (double _a, double _b)` `[virtual]`

Set distribution parameters.

6.69.3.13 `virtual void CUniformCRV::setDistribution (const CContinuousSample & _aS)` `[inline], [virtual]`

Set distribution parameters from sample empirical parameters.

6.69.3.14 `void CUniformCRV::setLowerBound (double _a)` `[inline]`

Sets lower bound of the uniform random variable.

6.69.3.15 `void CUniformCRV::setUpperBound (double _b) [inline]`

Sets upper bound of the uniform radnom variable.

6.69.3.16 `double CUniformCRV::upperBound () const [inline]`

Returns upper bound of the uniform radnom variable.

6.69.3.17 `virtual double CUniformCRV::variance () const [inline],[virtual]`

Returns variance of the random variable.

Implements [CRandomVariable](#).

6.69.4 Friends And Related Function Documentation

6.69.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.69.5 Member Data Documentation

6.69.5.1 `double CUniformCRV::a [private]`

6.69.5.2 `double CUniformCRV::b [private]`

6.69.5.3 `double CUniformCRV::d [private]`

Distribution parameters: random variable can have any value between a and b, ($d=b-a$).

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

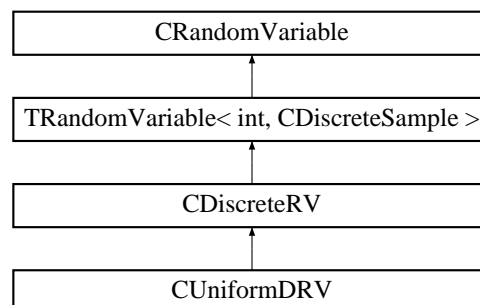
- C:/Development/core/[RandomVariable.h](#)
- C:/Development/core/[RandomVariable.cpp](#)

6.70 CUniformDRV Class Reference

Class for uniform discrete random variables derived from discrete random variables.

```
#include <RandomVariable.h>
```

Inheritance diagram for CUniformDRV:



Public Member Functions

- [CUniformDRV](#) ()
- virtual [~CUniformDRV](#) ()
- [CUniformDRV](#) (const [CUniformDRV](#) &O)
- virtual void [setDistribution](#) (int _a, int _b)
Set distribution parameters.
- virtual double [probability](#) (int _k) const
Function that returns PMF.
- virtual double [cumulative](#) (int _k) const
Function that returns CDF.
- virtual double [minValue](#) () const
Returns minimal value of the random variable.
- virtual double [maxValue](#) () const
Returns maximal value of the random variable.
- virtual double [expectedValue](#) () const
Returns expected value of the random variable.
- virtual [distributiontype](#) [distributionType](#) () const
Returns enum type of the random variable distribution.
- virtual std::string [distributionName](#) () const
Returns name of the random variable distribution.
- virtual ArrayXi [exportDomain](#) () const
Exports domain of the random variable.
- int [lowerBound](#) () const
Returns lower bound of the uniform radnom variable.
- int [upperBound](#) () const
Returns upper bound of the uniform radnom variable.
- void [setLowerBound](#) (int _a)
Sets lower bound of the uniform radnom variable.
- void [setUpperBound](#) (int _b)
Sets upper bound of the uniform radnom variable.
- virtual double [variance](#) () const
Returns variance of the random variable.

Protected Member Functions

- virtual double [doQuantile](#) (double _p) const
Quantile, formula implementation.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Private Attributes

- int [a](#)
- int [b](#)
- int [n](#)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.70.1 Detailed Description

Class for uniform discrete random variables derived from discrete random variables.

6.70.2 Constructor & Destructor Documentation

6.70.2.1 `CUniformDRV::CUniformDRV ()` `[inline]`

6.70.2.2 `virtual CUniformDRV::~~CUniformDRV ()` `[inline], [virtual]`

6.70.2.3 `CUniformDRV::CUniformDRV (const CUniformDRV & O)` `[inline]`

6.70.3 Member Function Documentation

6.70.3.1 `virtual double CUniformDRV::cumulative (int _k) const` `[inline], [virtual]`

Function that returns CDF.

6.70.3.2 `virtual std::string CUniformDRV::distributionName () const` `[inline], [virtual]`

Returns name of the random variable distribution.

Implements [CRandomVariable](#).

6.70.3.3 `virtual distributiontype CUniformDRV::distributionType () const` `[inline], [virtual]`

Returns enum type of the random variable distribution.

Implements [CRandomVariable](#).

6.70.3.4 `virtual double CUniformDRV::doQuantile (double _p) const` `[inline], [protected], [virtual]`

Quantile, formula implementation.

Implements [CRandomVariable](#).

6.70.3.5 `virtual double CUniformDRV::expectedValue () const` `[inline], [virtual]`

Returns expected value of the random variable.

Implements [CRandomVariable](#).

6.70.3.6 `ArrayXi CUniformDRV::exportDomain () const` `[virtual]`

Exports domain of the random variable.

Implements [TRandomVariable< t, T >](#).

6.70.3.7 `int CUniformDRV::lowerBound () const [inline]`

Returns lower bound of the uniform radnom variable.

6.70.3.8 `virtual double CUniformDRV::maxValue () const [inline],[virtual]`

Returns maximal value of the random variable.

Implements [CRandomVariable](#).

6.70.3.9 `virtual double CUniformDRV::minValue () const [inline],[virtual]`

Returns minimal value of the random variable.

Implements [CRandomVariable](#).

6.70.3.10 `virtual double CUniformDRV::probability (int _k) const [inline],[virtual]`

Function that returns PMF.

6.70.3.11 `template<class Archive > void CUniformDRV::serialize (Archive & ar, const unsigned int version) [private]`

6.70.3.12 `void CUniformDRV::setDistribution (int _a, int _b) [virtual]`

Set distribution parameters.

6.70.3.13 `void CUniformDRV::setLowerBound (int _a) [inline]`

Sets lower bound of the uniform radnom variable.

6.70.3.14 `void CUniformDRV::setUpperBound (int _b) [inline]`

Sets upper bound of the uniform radnom variable.

6.70.3.15 `int CUniformDRV::upperBound () const [inline]`

Returns upper bound of the uniform radnom variable.

6.70.3.16 `virtual double CUniformDRV::variance () const [inline],[virtual]`

Returns variance of the random variable.

Implements [CRandomVariable](#).

6.70.4 Friends And Related Function Documentation

6.70.4.1 `friend class boost::serialization::access [friend]`

Boost serialization.

6.70.5 Member Data Documentation

6.70.5.1 `int CUniformDRV::a` `[private]`

6.70.5.2 `int CUniformDRV::b` `[private]`

6.70.5.3 `int CUniformDRV::n` `[private]`

Distribution parameters: random variable can have one of the $n=b-a+1$ integer values between a and b .

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

- [C:/Development/core/RandomVariable.h](#)
- [C:/Development/core/RandomVariable.cpp](#)

6.71 Cx_ZetaGammamax Class Reference

Class for the function $f(x)=x\text{-zeta}\cdot\text{gamma}^{[7]}(x)$.

```
#include <VariousMath.h>
```

Public Member Functions

- [Cx_ZetaGammamax](#) (const int `_N`, const ArrayXd &`_i2`, const ArrayXd &`_a2`)
- double [evaluate](#) (double `_x`) const

Private Member Functions

- [Cx_ZetaGammamax](#) ()

Private Attributes

- int `N`
- ArrayXd `i2`
- ArrayXd `a2`

6.71.1 Detailed Description

Class for the function $f(x)=x\text{-zeta}\cdot\text{gamma}^{[7]}(x)$.

6.71.2 Constructor & Destructor Documentation

6.71.2.1 `Cx_ZetaGammamax::Cx_ZetaGammamax ()` `[inline]`, `[private]`

6.71.2.2 `Cx_ZetaGammamax::Cx_ZetaGammamax (const int _N, const ArrayXd &_i2, const ArrayXd &_a2)` `[inline]`

6.71.3 Member Function Documentation

6.71.3.1 `double Cx_ZetaGammamax::evaluate (double _x)` const

6.71.4 Member Data Documentation

6.71.4.1 `ArrayXd Cx.ZetaGammax::a2` `[private]`

6.71.4.2 `ArrayXd Cx.ZetaGammax::i2` `[private]`

6.71.4.3 `int Cx.ZetaGammax::N` `[private]`

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

- `C:/Development/core/VariousMath.h`
- `C:/Development/core/VariousMath.cpp`

6.72 DS_n Class Reference

```
#include <Ds.h>
```

Public Member Functions

- `DSn` (unsigned int *var*)
constructor
- `~DSn` ()
destructor
- long double `doub` ()
generator

Private Types

- typedef long long int `Long`

Private Attributes

- unsigned int `k`
- unsigned int `i`
- unsigned int `j`
- `Long n`
- `Array< long double, Dynamic, 1 > x`
- `Array< long double, Dynamic, 1 > w`

6.72.1 Member Typedef Documentation

6.72.1.1 `typedef long long int DSn::Long` `[private]`

6.72.2 Constructor & Destructor Documentation

6.72.2.1 `DSn::DSn (unsigned int var)` `[inline]`

constructor

6.72.2.2 `DSn::~DSn ()` `[inline]`

destructor

6.72.3 Member Function Documentation

6.72.3.1 `long double DSn::doub () [inline]`

generator

6.72.4 Member Data Documentation

6.72.4.1 `unsigned int DSn::i [private]`

6.72.4.2 `unsigned int DSn::j [private]`

6.72.4.3 `unsigned int DSn::k [private]`

6.72.4.4 `Long DSn::n [private]`

6.72.4.5 `Array<long double,Dynamic,1> DSn::w [private]`

6.72.4.6 `Array<long double,Dynamic,1> DSn::x [private]`

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

- `C:/Development/core/Ds.h`

6.73 GoSUM Struct Reference

Classes

- class [CA analyticalModel](#)
Class for analytical representation of the model.
- class [CA analyticalOptimizationProblem](#)
Class for the optimization problem based on [GoSUM](#) analytical model.
- class [CConstraints](#)
Class for the constraints with parser functions.
- class [CContainer](#)
Class for the [GoSUM](#) main project.
- class [CContinuousMV](#)
- class [CCSvcSAM](#)
Class for the analytical model for single output state, SVC type.
- class [CCVoronoiHC](#)
- class [CDiscreteMV](#)
- class [CDSampleHC](#)
- class [CEpsSvrSAM](#)
Class for the analytical model for single output state, epsilon-SVR type.
- class [CEvaluator](#)
Class for [GoSUM](#) model evaluator.
- class [CExeAsciiEvaluator](#)
Class for the evaluator with ascii i/o and .exe.
- class [CExeEvaluator](#)
Template for the evaluator with some file i/o and .exe derived from [CModelEvaluator](#).
- class [CExeMatEvaluator](#)
Class for the evaluator with Matlab .mat i/o and .exe.

- class [CHypercube](#)
- class [CInputParameters](#)
Class for [GoSUM](#) input parameters.
- class [CLCVoronoiHC](#)
- class [CMatlabEngineEvaluator](#)
Class for the evaluator through Matlab engine derived from [CModelEvaluator](#).
- class [CMatlabShellEvaluator](#)
Class for the evaluator through Matlab engine and .mat i/o.
- class [CModelConstraints](#)
- class [CModelEvaluator](#)
Class for [GoSUM](#) model evaluator.
- class [CModelHypercube](#)
- class [CModelOptimizationVariable](#)
- class [CModelVariable](#)
Class for any variable in the [GoSUM](#) model (input or output).
- class [CModelVariables](#)
Class for the vector of model variables (inputs & outputs).
- class [CMonteCarloHC](#)
- class [CNuSvcSAM](#)
Class for the analytical model for single output state, nu-SVC type.
- class [CNuSvrSAM](#)
Class for the analytical model for single output state, nu-SVR type.
- class [COneClassSAM](#)
Class for the analytical model for single output state, one class type.
- class [COriginalOptimizationProblem](#)
Class for the optimization problem based on [GoSUM](#) analytical model.
- class [COutputStates](#)
Class for [GoSUM](#) output states.
- class [CParserOptimizationProblem](#)
Class for the optimization problem with parser functions.
- class [CReduction](#)
Class for sensitivity analysis of the model.
- class [CScript](#)
Class for the [GoSUM](#) script format.
- class [CSensitivityAnalysis](#)
Class for sensitivity analysis of the model.
- class [CSimpleOptimizationProblem](#)
Class for the optimization problem based only on [GoSUM](#) input parameters.
- class [CSingleAM](#)
Base class for the analytical model for single output state, interface between analytical model and SVM.
- class [TModelVariable](#)

Public Types

- typedef [TModelVariable](#)
`< CDiscreteRV, CDiscreteSample,
int > CTDiscreteMV`
- typedef [TModelVariable](#)
`< CContinuousRV,
CContinuousSample, double > CTContinuousMV`
- typedef void([CScript](#)::* [paction](#))(std::istream &)
Typdef for pointer to member function of [CScript](#) that is the response to specific script commands.

Public Attributes

- [NULL](#)

6.73.1 Member Typedef Documentation

6.73.1.1 `typedef TModelVariable<CContinuousRV,CContinuousSample,double> GoSUM::CTContinuousMV`

6.73.1.2 `typedef TModelVariable<CDiscreteRV,CDiscreteSample,int> GoSUM::CTDiscreteMV`

6.73.1.3 `typedef void(CScript::* GoSUM::paction)(std::istream &)`

Typdef for pointer to member function of [CScript](#) that is the response to specific script commands.

6.73.2 Member Data Documentation

6.73.2.1 `GoSUM::NULL`

The documentation for this struct was generated from the following files:

- C:/Development/core/[AnalyticalModel.cpp](#)
- C:/Development/core/[ModelVariable.h](#)
- C:/Development/core/[Script.h](#)

6.74 Ran Class Reference

```
#include <ran.h>
```

Public Member Functions

- [Ran](#) ([Ullong](#) j)
- [Ullong int64](#) ()
- double [doub](#) ()
- unsigned int [int32](#) ()

Private Attributes

- [Ullong](#) u
- [Ullong](#) v
- [Ullong](#) w

6.74.1 Constructor & Destructor Documentation

6.74.1.1 `Ran::Ran(Ullong j) [inline]`

6.74.2 Member Function Documentation

6.74.2.1 `double Ran::doub() [inline]`

6.74.2.2 `unsigned int Ran::int32() [inline]`

6.74.2.3 **Ullong Ran::int64 ()** [inline]

6.74.3 Member Data Documentation

6.74.3.1 **Ullong Ran::u** [private]

6.74.3.2 **Ullong Ran::v** [private]

6.74.3.3 **Ullong Ran::w** [private]

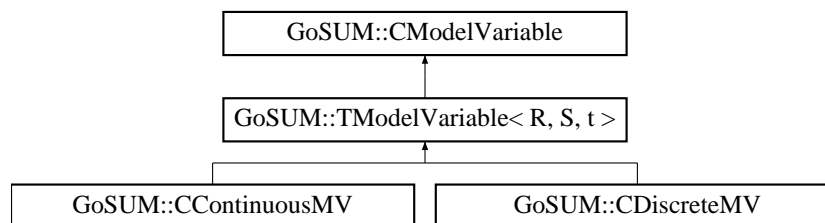
The documentation for this class was generated from the following file:

- C:/Development/core/[ran.h](#)

6.75 GoSUM::TModelVariable< R, S, t > Class Template Reference

```
#include <ModelVariable.h>
```

Inheritance diagram for GoSUM::TModelVariable< R, S, t >:



Public Member Functions

- [TModelVariable](#) ()
- virtual [~TModelVariable](#) ()
- virtual void [clearSample](#) ()
Clears variable's sample.
- virtual int [expandedSize](#) () const
- virtual void [setSampleSize](#) (int _n)
Returns number of variables after expansion for analytical model.
- virtual int [sampleSize](#) () const
Returns sample size.
- virtual void [setSampleValue](#) (double _val, int _at)
Sets particular sample value.
- virtual double [sampleValue](#) (int _at) const
Returns particular sample value.
- virtual void [readSampleValue](#) (std::ifstream &_ifs, int _at)
Reads particular sample value from input file stream.
- virtual void [writeSampleValue](#) (std::ofstream &_ofs, int _at) const
Writes particular sample value to output file stream.
- virtual void [generateSampleValue](#) (double _p, int _at)
generates particular sample value using actual random variable model.
- virtual double [generateSampleValue](#) (double _p)
Generates sample value using actual random variable model.
- virtual void [computeStatistics](#) (int _n)

- Computes histogram and statistical indicators.*
- virtual double `minValue` () const
Returns minimal variable value.
- virtual double `maxValue` () const
Returns maximal variable value.
- virtual double `expectedValue` () const
Returns expected variable value.
- virtual `CRandomVariable * randomVariable` () const
Returns random variable.
- virtual `CSample * sample` () const
Returns sample.
- void `setDistributionParameters` (t_dp1, t_dp2=0)
Set parameters of the theoretical distribution.
- virtual void `setTheoreticalDistribution` ()=0
Detects distribution and turns random variable to appropriate type.
- virtual void `setEmpiricalDistribution` ()=0
Compute distribution of the actual random variable from empirical sample data.
- Array< t, Dynamic, 1 > `exportDomain` () const
Exports probability values for given _x values.
- ArrayXd `exportProbability` (const Array< t, Dynamic, 1 > &_x) const
Exports probability values for given _x values.
- ArrayXd `exportCumulative` (const Array< t, Dynamic, 1 > &_x) const
Exports cumulative values for given _x values.
- ArrayXd `exportQuantile` (const ArrayXd &_p) const
Exports quantile values for given _x values.
- Array< t, Dynamic, 1 > `exportSample` () const
Exports sample values.
- virtual
`CRandomVariable::distributiontype distributionType` () const
Returns type of the random variable distribution.
- virtual std::string `distributionName` () const
Returns name of the random variable distribution.
- t `normalizedHistogram` (Array< t, Dynamic, 1 > &_x, ArrayXd &_H) const
Returns normalized histogram (x,H).
- t `exportHistogram` (Array< t, Dynamic, 1 > &_x, ArrayXd &_H) const
Returns histogram (x,H).
- void `exportSubHistogram` (ArrayXd &_subH, const std::vector< int > &subset) const
Returns subhistogram.
- virtual bool `isDistributionDefined` () const
Returns true if distribution is defined, false otherwise.

Protected Member Functions

- template<class Archive >
void `serialize` (Archive &ar, const unsigned int version)

Protected Attributes

- R * `pRV`
Pointer to a random variable that models behaviour of model variable.
- S * `pS`

Friends

- class [boost::serialization::access](#)
Boost serialization.

6.75.1 Constructor & Destructor Documentation

6.75.1.1 `template<class R , class S , typename t > GoSUM::TModelVariable< R, S, t >::TModelVariable ()`
`[inline]`

6.75.1.2 `template<class R , class S , typename t > virtual GoSUM::TModelVariable< R, S, t >::~~TModelVariable ()`
`[inline], [virtual]`

6.75.2 Member Function Documentation

6.75.2.1 `template<class R , class S , typename t > virtual void GoSUM::TModelVariable< R, S, t >::clearSample ()`
`[inline], [virtual]`

Clears variable's sample.

Implements [GoSUM::CModelVariable](#).

6.75.2.2 `template<class R , class S , typename t > virtual void GoSUM::TModelVariable< R, S, t >::computeStatistics (`
`int _n) [inline], [virtual]`

Computes histogram and statistical indicators.

Implements [GoSUM::CModelVariable](#).

6.75.2.3 `template<class R , class S , typename t > virtual std::string GoSUM::TModelVariable< R, S, t`
`>::distributionName () const [inline], [virtual]`

Returns name of the random variable distribution.

Implements [GoSUM::CModelVariable](#).

6.75.2.4 `template<class R , class S , typename t > virtual CRandomVariable::distributiontype`
`GoSUM::TModelVariable< R, S, t >::distributionType () const [inline], [virtual]`

Returns type of the random variable distribution.

Implements [GoSUM::CModelVariable](#).

6.75.2.5 `template<class R , class S , typename t > virtual int GoSUM::TModelVariable< R, S, t >::expandedSize ()`
`const [inline], [virtual]`

Implements [GoSUM::CModelVariable](#).

6.75.2.6 `template<class R , class S , typename t > virtual double GoSUM::TModelVariable< R, S, t >::expectedValue ()`
`const [inline], [virtual]`

Returns expected variable value.

Implements [GoSUM::CModelVariable](#).

6.75.2.7 `template<class R , class S , typename t > ArrayXd GoSUM::TModelVariable< R, S, t >::exportCumulative (const Array< t, Dynamic, 1 > & _x) const [inline]`

Exports cumulative values for given `_x` values.

6.75.2.8 `template<class R , class S , typename t > Array<t,Dynamic,1> GoSUM::TModelVariable< R, S, t >::exportDomain () const [inline]`

Exports probability values for given `_x` values.

6.75.2.9 `template<class R , class S , typename t > t GoSUM::TModelVariable< R, S, t >::exportHistogram (Array< t, Dynamic, 1 > & _x, ArrayXd & _H) const [inline]`

Returns histogram (x,H).

6.75.2.10 `template<class R , class S , typename t > ArrayXd GoSUM::TModelVariable< R, S, t >::exportProbability (const Array< t, Dynamic, 1 > & _x) const [inline]`

Exports probability values for given `_x` values.

6.75.2.11 `template<class R , class S , typename t > ArrayXd GoSUM::TModelVariable< R, S, t >::exportQuantile (const ArrayXd & _p) const [inline]`

Exports quantile values for given `_x` values.

6.75.2.12 `template<class R , class S , typename t > Array<t,Dynamic,1> GoSUM::TModelVariable< R, S, t >::exportSample () const [inline]`

Exports sample values.

6.75.2.13 `template<class R , class S , typename t > void GoSUM::TModelVariable< R, S, t >::exportSubHistogram (ArrayXd & _subH, const std::vector< int > & subset) const [inline]`

Returns subhistogram.

6.75.2.14 `template<class R , class S , typename t > virtual void GoSUM::TModelVariable< R, S, t >::generateSampleValue (double _p, int _at) [inline],[virtual]`

generates particular sample value using actual random variable model.

Implements [GoSUM::CModelVariable](#).

6.75.2.15 `template<class R , class S , typename t > virtual double GoSUM::TModelVariable< R, S, t >::generateSampleValue (double _p) [inline],[virtual]`

Generates sample value using actual random variable model.

Implements [GoSUM::CModelVariable](#).

6.75.2.16 `template<class R , class S , typename t > virtual bool GoSUM::TModelVariable< R, S, t >::isDistributionDefined () const [inline],[virtual]`

Returns true if distribution is defined, false otherwise.

Implements [GoSUM::CModelVariable](#).

6.75.2.17 `template<class R , class S , typename t > virtual double GoSUM::TModelVariable< R, S, t >::maxValue () const [inline],[virtual]`

Returns maximal variable value.

Implements [GoSUM::CModelVariable](#).

6.75.2.18 `template<class R , class S , typename t > virtual double GoSUM::TModelVariable< R, S, t >::minValue () const [inline],[virtual]`

Returns minimal variable value.

Implements [GoSUM::CModelVariable](#).

6.75.2.19 `template<class R , class S , typename t > t GoSUM::TModelVariable< R, S, t >::normalizedHistogram (Array< t, Dynamic, 1 > & _x, ArrayXd & _H) const [inline]`

Returns normalized histogram (x,H).

6.75.2.20 `template<class R , class S , typename t > virtual CRandomVariable* GoSUM::TModelVariable< R, S, t >::randomVariable () const [inline],[virtual]`

Returns random variable.

Implements [GoSUM::CModelVariable](#).

6.75.2.21 `template<class R , class S , typename t > virtual void GoSUM::TModelVariable< R, S, t >::readSampleValue (std::ifstream & _ifs, int _at) [inline],[virtual]`

Reads particular sample value from input file stream.

Implements [GoSUM::CModelVariable](#).

6.75.2.22 `template<class R , class S , typename t > virtual CSample* GoSUM::TModelVariable< R, S, t >::sample () const [inline],[virtual]`

Returns sample.

Implements [GoSUM::CModelVariable](#).

6.75.2.23 `template<class R , class S , typename t > virtual int GoSUM::TModelVariable< R, S, t >::sampleSize () const [inline],[virtual]`

Returns sample size.

Implements [GoSUM::CModelVariable](#).

6.75.2.24 `template<class R, class S, typename t> virtual double GoSUM::TModelVariable< R, S, t >::sampleValue (int _at) const [inline],[virtual]`

Returns particular sample value.

Implements [GoSUM::CModelVariable](#).

6.75.2.25 `template<class R, class S, typename t> template<class Archive > void GoSUM::TModelVariable< R, S, t >::serialize (Archive & ar, const unsigned int version) [inline],[protected]`

Reimplemented from [GoSUM::CModelVariable](#).

6.75.2.26 `template<class R, class S, typename t> void GoSUM::TModelVariable< R, S, t >::setDistributionParameters (t _dp1, t _dp2 = 0) [inline]`

Set parameters of the theoretical distribution.

6.75.2.27 `template<class R, class S, typename t> virtual void GoSUM::TModelVariable< R, S, t >::setEmpiricalDistribution () [pure virtual]`

Compute distribution of the actual random variable from empirical sample data.

Reimplemented from [GoSUM::CModelVariable](#).

Implemented in [GoSUM::CContinuousMV](#), and [GoSUM::CDiscreteMV](#).

6.75.2.28 `template<class R, class S, typename t> virtual void GoSUM::TModelVariable< R, S, t >::setSampleSize (int _n) [inline],[virtual]`

Returns number of variables after expansion for analytical model.

Sets sample size.

Implements [GoSUM::CModelVariable](#).

6.75.2.29 `template<class R, class S, typename t> virtual void GoSUM::TModelVariable< R, S, t >::setSampleValue (double _val, int _at) [inline],[virtual]`

Sets particular sample value.

Implements [GoSUM::CModelVariable](#).

6.75.2.30 `template<class R, class S, typename t> virtual void GoSUM::TModelVariable< R, S, t >::setTheoreticalDistribution () [pure virtual]`

Detects distribution and turns random variable to appropriate type.

Reimplemented from [GoSUM::CModelVariable](#).

Implemented in [GoSUM::CContinuousMV](#), and [GoSUM::CDiscreteMV](#).

6.75.2.31 `template<class R, class S, typename t> virtual void GoSUM::TModelVariable< R, S, t >::writeSampleValue (std::ofstream & _ofs, int _at) const [inline],[virtual]`

Writes particular sample value to output file stream.

Implements [GoSUM::CModelVariable](#).

6.75.3 Friends And Related Function Documentation

6.75.3.1 `template<class R, class S, typename t > friend class boost::serialization::access` [friend]

Boost serialization.

6.75.4 Member Data Documentation

6.75.4.1 `template<class R, class S, typename t > R* GoSUM::TModelVariable< R, S, t >::pRV` [protected]

Pointer to a random variable that models behaviour of model variable.

6.75.4.2 `template<class R, class S, typename t > S* GoSUM::TModelVariable< R, S, t >::pS` [protected]

Pointer to the sample data of the model variable.

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

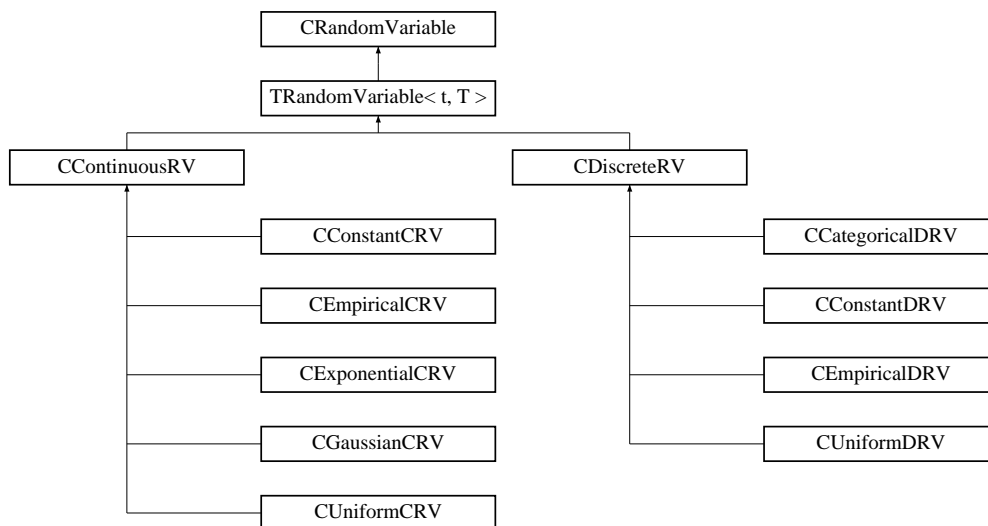
- [C:/Development/core/ModelVariable.h](#)

6.76 TRandomVariable< t, T > Class Template Reference

Template of an abstract class - covers two types of random variables: discrete ($t=\text{int}, T=\text{CDiscreteSample}$) or continuous ($t=\text{double}, T=\text{CContinuousSample}$).

```
#include <RandomVariable.h>
```

Inheritance diagram for TRandomVariable< t, T >:



Public Member Functions

- [TRandomVariable](#) ()
- virtual [~TRandomVariable](#) ()
- [TRandomVariable](#) (const [TRandomVariable](#) &O)
- virtual void [setDistribution](#) (t_dp1, t_dp2=0)
Sets distribution parameters of the random variable.

- virtual void [setDistribution](#) (const T &_aS)
Sets distribution of the random variable from sample.
- virtual double [probability](#) (t _x) const =0
Returns probability mass function (PMF) or probability density function (PDF) for the random variable value _x.
- virtual double [cumulative](#) (t _x) const =0
Returns cumulative distribution function (CDF) for the random variable value _x.
- [rvExportMacro](#) (Probability, [probability](#), t) [rvExportMacro](#)(Cumulative
- t [rvExportMacro](#) (Quantile, [quantile](#), double) virtual Array<t
 < Exports cumulative for an array of x values.
- t [exportDomain](#) () const =0
Exports domain of the random variable.

Public Attributes

- [cumulative](#)
- t [Dynamic](#)

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.76.1 Detailed Description

template<typename t, class T>class TRandomVariable< t, T >

Template of an abstract class - covers two types of random variables: discrete (t=int,T=[CDiscreteSample](#)) or continuous (t=double,T=[CContinuousSample](#)).

6.76.2 Constructor & Destructor Documentation

6.76.2.1 template<typename t, class T > TRandomVariable< t, T >::TRandomVariable () [inline]

6.76.2.2 template<typename t, class T > virtual TRandomVariable< t, T >::~~TRandomVariable () [inline],
[virtual]

6.76.2.3 template<typename t, class T > TRandomVariable< t, T >::TRandomVariable (const TRandomVariable< t, T > &O) [inline]

6.76.3 Member Function Documentation

6.76.3.1 template<typename t, class T > virtual double TRandomVariable< t, T >::cumulative (t _x) const [pure
virtual]

Returns cumulative distribution function (CDF) for the random variable value _x.

6.76.3.2 `template<typename t, class T> t TRandomVariable<t, T>::exportDomain () const [pure virtual]`

Exports domain of the random variable.

Implemented in [CEmpiricalCRV](#), [CExponentialCRV](#), [CGaussianCRV](#), [CUniformCRV](#), [CConstantCRV](#), [C-CategoricalDRV](#), [CEmpiricalDRV](#), [CUniformDRV](#), and [CConstantDRV](#).

6.76.3.3 `template<typename t, class T> virtual double TRandomVariable<t, T>::probability (t _x) const [pure virtual]`

Returns probability mass function (PMF) or probability density function (PDF) for the random variable value `_x`.

6.76.3.4 `template<typename t, class T> TRandomVariable<t, T>::rvExportMacro (Probability , probability , t)`

6.76.3.5 `template<typename t, class T> t TRandomVariable<t, T>::rvExportMacro (Quantile , quantile , double)`

< Exports cumulative for an array of x values.

6.76.3.6 `template<typename t, class T> template<class Archive> void TRandomVariable<t, T>::serialize (Archive & ar, const unsigned int version) [inline], [private]`

6.76.3.7 `template<typename t, class T> virtual void TRandomVariable<t, T>::setDistribution (t _dp1, t _dp2 = 0) [inline], [virtual]`

Sets distribution parameters of the random variable.

6.76.3.8 `template<typename t, class T> virtual void TRandomVariable<t, T>::setDistribution (const T & _aS) [inline], [virtual]`

Sets distribution of the random variable from sample.

6.76.4 Friends And Related Function Documentation

6.76.4.1 `template<typename t, class T> friend class boost::serialization::access [friend]`

Boost serialization.

6.76.5 Member Data Documentation

6.76.5.1 `template<typename t, class T> TRandomVariable<t, T>::cumulative`

6.76.5.2 `template<typename t, class T> t TRandomVariable<t, T>::Dynamic`

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

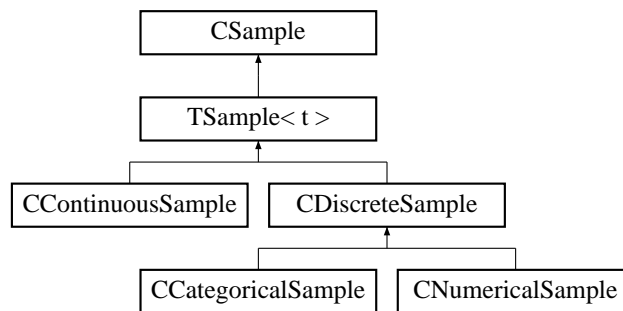
- C:/Development/core/[RandomVariable.h](#)

6.77 TSample< t > Class Template Reference

Template of an abstract class - covers two types of samples: discrete (t=int) or continuous (t=double).

```
#include <Sample.h>
```

Inheritance diagram for TSample< t >:



Public Member Functions

- [TSample](#) ()
- virtual [~TSample](#) ()
- [TSample](#) (const [TSample](#) &O)
- void [clear](#) ()
Clears object.
- virtual void [setSampleSize](#) (int _n)
(Re)sizes the sample, all previous data is lost.
- virtual int [sampleSize](#) () const
Returns sample size (number of data in the sample).
- virtual void [setSampleValue](#) (double _val, int _at)
Sets particular sample data value.
- t [sampleValue](#) (int _at) const
Returns particular sample data value.
- virtual void [readSampleValue](#) (std::ifstream &_ifs, int _at)
Reads particular sample data from input file stream.
- virtual void [writeSampleValue](#) (std::ofstream &_ofs, int _at) const
< Writes particular sample data to output file stream.
- bool [isConstant](#) () const
Returns true if it is a constant sample, false otherwise.
- t [minValue](#) () const
Returns minimal sample data value.
- t [maxValue](#) () const
Returns maximal sample data value.
- t [normalizedHistogram](#) (Array< t, Dynamic, 1 > &_x, ArrayXd &_H) const
Returns normalized histogram (x,H).
- void [cumulativeHistogram](#) (Array< t, Dynamic, 1 > &_x, ArrayXd &_cH) const
Returns cumulative histogram (x,cH).
- virtual t [exportHistogram](#) (Array< t, Dynamic, 1 > &_x, ArrayXd &_H) const =0
Returns histogram (x,H).
- virtual void [exportSubHistogram](#) (ArrayXd &_subH, const std::vector< int > &subset) const =0
Exports histogram relative to a subset of sample data.
- Array< t, Dynamic, 1 > [exportSample](#) () const
Exports sample.
- virtual std::vector< int > [select](#) (double _left, double _right) const
< Returns indices of those sample values that are between _left and _right.
- virtual void [eraseSelected](#) (const std::vector< int > &sel)
Erases sample values on positions defiend by sel.

Protected Member Functions

- `template<class Archive >`
void [serialize](#) (Archive &ar, const unsigned int version)

Protected Attributes

- `Array< t, Dynamic, 1 >` [sample](#)
Array that holds sample data.
- `Array< t, Dynamic, 1 >` [x](#)
Array that holds x-coordinate of the normalized histogram.
- `ArrayXd` [H](#)
- `ArrayXd` [cH](#)
Array that holds normalized histogram.
- `t` [range](#)

Private Member Functions

- virtual void [computeNormalizedHistogram](#) (int _n)=0
Computes normalized histogram from sample data.
- virtual void [computeNormalizedSubHistogram](#) (ArrayXd &_subH, const std::vector< int > &subset) const =0
Computes normalized histogram from a subset of sample data.

Friends

- class [boost::serialization::access](#)
Boost serialization.

Additional Inherited Members

6.77.1 Detailed Description

`template<typename t>class TSample< t >`

Template of an abstract class - covers two types of samples: discrete (t=int) or continuous (t=double).

6.77.2 Constructor & Destructor Documentation

6.77.2.1 `template<typename t> TSample< t >::TSample ()` `[inline]`

6.77.2.2 `template<typename t> virtual TSample< t >::~~TSample ()` `[inline]`, `[virtual]`

6.77.2.3 `template<typename t> TSample< t >::TSample (const TSample< t > & O)` `[inline]`

6.77.3 Member Function Documentation

6.77.3.1 `template<typename t> void TSample< t >::clear ()` `[inline]`, `[virtual]`

Clears object.

Implements [CSample](#).

6.77.3.2 `template<typename t> virtual void TSample< t >::computeNormalizedHistogram (int _n) [private],
[pure virtual]`

Computes normalized histogram from sample data.

6.77.3.3 `template<typename t> virtual void TSample< t >::computeNormalizedSubHistogram (ArrayXd & _subH, const
std::vector< int > & subset) const [private],[pure virtual]`

Computes normalized histogram from a subset of sample data.

6.77.3.4 `template<typename t> void TSample< t >::cummulativeHistogram (Array< t, Dynamic, 1 > & _x, ArrayXd & _cH
) const [inline]`

Returns cummulative histogram (x,cH).

6.77.3.5 `template<typename t> virtual void TSample< t >::eraseSelected (const std::vector< int > & sel)
[inline],[virtual]`

Erases sample values on positions defiend by sel.

Parameters

<i>sel</i>	Erases sample values on positions defiend by sel.
------------	---

Implements [CSample](#).

6.77.3.6 `template<typename t> virtual t TSample< t >::exportHistogram (Array< t, Dynamic, 1 > & _x, ArrayXd & _H)
const [pure virtual]`

Returns histogram (x,H).

6.77.3.7 `template<typename t> Array<t,Dynamic,1> TSample< t >::exportSample () const [inline]`

Exports sample.

6.77.3.8 `template<typename t> virtual void TSample< t >::exportSubHistogram (ArrayXd & _subH, const std::vector< int
> & subset) const [pure virtual]`

Exports histogram relative to a subset of sample data.

Implemented in [CContinuousSample](#), and [CDiscreteSample](#).

6.77.3.9 `template<typename t> bool TSample< t >::isConstant () const [inline]`

Returns true if it is a constant sample, false otherwise.

6.77.3.10 `template<typename t> t TSample< t >::maxValue () const [inline]`

Returns maximal sample data value.

Reimplemented in [CCategoricalSample](#), and [CNumericalSample](#).

6.77.3.11 `template<typename t> t TSample< t >::minValue () const [inline]`

Returns minimal sample data value.

Reimplemented in [CCategoricalSample](#), and [CNumericalSample](#).

6.77.3.12 `template<typename t> t TSample< t >::normalizedHistogram (Array< t, Dynamic, 1 > & _x, ArrayXd & _H) const [inline]`

Returns normalized histogram (x,H).

6.77.3.13 `template<typename t> virtual void TSample< t >::readSampleValue (std::ifstream & _ifs, int _at) [inline],[virtual]`

Reads particular sample data from input file stream.

Parameters

<code>_at</code>	Reads particular sample data from input file stream.
------------------	--

Implements [CSample](#).

Reimplemented in [CCategoricalSample](#), and [CNumericalSample](#).

6.77.3.14 `template<typename t> virtual int TSample< t >::sampleSize () const [inline],[virtual]`

Returns sample size (number of data in the sample).

Implements [CSample](#).

6.77.3.15 `template<typename t> t TSample< t >::sampleValue (int _at) const [inline]`

Returns particular sample data value.

6.77.3.16 `template<typename t> virtual std::vector<int> TSample< t >::select (double _left, double _right) const [inline],[virtual]`

< Returns indices of those sample values that are between _left and _right.

Implements [CSample](#).

6.77.3.17 `template<typename t> template<class Archive > void TSample< t >::serialize (Archive & ar, const unsigned int version) [inline],[protected]`

Reimplemented from [CSample](#).

Reimplemented in [CContinuousSample](#), and [CDiscreteSample](#).

6.77.3.18 `template<typename t> virtual void TSample< t >::setSampleSize (int _n) [inline],[virtual]`

(Re)sizes the sample, all previous data is lost.

Implements [CSample](#).

Reimplemented in [CCategoricalSample](#), and [CNumericalSample](#).

6.77.3.19 `template<typename t> virtual void TSample< t >::setSampleValue (double _val, int _at) [inline], [virtual]`

Sets particular sample data value.

Implements [CSample](#).

6.77.3.20 `template<typename t> virtual void TSample< t >::writeSampleValue (std::ofstream & _ofs, int _at) const [inline], [virtual]`

< Writes particular sample data to output file stream.

Implements [CSample](#).

Reimplemented in [CCategoricalSample](#), and [CNumericalSample](#).

6.77.4 Friends And Related Function Documentation

6.77.4.1 `template<typename t> friend class boost::serialization::access [friend]`

Boost serialization.

6.77.5 Member Data Documentation

6.77.5.1 `template<typename t> ArrayXd TSample< t >::cH [protected]`

Array that holds normalized histogram.

6.77.5.2 `template<typename t> ArrayXd TSample< t >::H [protected]`

6.77.5.3 `template<typename t> t TSample< t >::range [protected]`

Range of the sample data.

6.77.5.4 `template<typename t> Array<t,Dynamic,1> TSample< t >::sample [protected]`

Array that holds sample data.

6.77.5.5 `template<typename t> Array<t,Dynamic,1> TSample< t >::x [protected]`

Array that holds x-coordinate of the normalized histogram.

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

- C:/Development/core/[Sample.h](#)

Chapter 7

File Documentation

7.1 C:/Development/core/AnalyticalModel.cpp File Reference

```
#include "AnalyticalModel.h"
```

Classes

- struct [GoSUM](#)

7.2 C:/Development/core/AnalyticalModel.h File Reference

```
#include "Model.h"  
#include "MADSOptimization.h"  
#include "svm.h"
```

Classes

- class [GoSUM::CSingleAM](#)
Base class for the analytical model for single output state, interface between analytical model and SVM.
- class [GoSUM::CCSvcSAM](#)
Class for the analytical model for single output state, SVC type.
- class [GoSUM::CNuSvcSAM](#)
Class for the analytical model for single output state, nu-SVC type.
- class [GoSUM::COneClassSAM](#)
Class for the analytical model for single output state, one class type.
- class [GoSUM::CEpsSvrSAM](#)
Class for the analytical model for single output state, epsilon-SVR type.
- class [GoSUM::CNuSvrSAM](#)
Class for the analytical model for single output state, nu-SVR type.
- class [GoSUM::CA analyticalModel](#)
Class for analytical representation of the model.

Namespaces

- namespace [GoSUM](#)
Namespace for [GoSUM](#) model.

7.3 C:/Development/core/Constraints.cpp File Reference

```
#include "Constraints.h"  
#include "Model.h"
```

7.4 C:/Development/core/Constraints.h File Reference

```
#include "Utilities.h"  
#include "VariousMath.h"  
#include "fparser.hh"
```

Classes

- class [GoSUM::CConstraints](#)
Class for the constraints with parser functions.
- class [GoSUM::CModelConstraints](#)

Namespaces

- namespace [GoSUM](#)
Namespace for [GoSUM](#) model.

7.5 C:/Development/core/Container.cpp File Reference

```
#include "Container.h"
```

7.6 C:/Development/core/Container.h File Reference

```
#include "Reduction.h"  
#include "MADSOptimization.h"  
#include "GAOptimization.h"  
#include "ParserOptimizationProblem.h"  
#include "OriginalModel.h"  
#include "Hypercube.h"
```

Classes

- class [GoSUM::CContainer](#)
Class for the [GoSUM](#) main project.

Namespaces

- namespace [GoSUM](#)
Namespace for [GoSUM](#) model.

7.7 C:/Development/core/Ds.cpp File Reference

```
#include "Ds.h"
```

Functions

- `Array< long double, Dynamic, 1 >` [prepareDSample](#) (unsigned int *M*)

7.7.1 Function Documentation

7.7.1.1 `Array<long double,Dynamic,1> prepareDSample (unsigned int M)`

7.8 C:/Development/core/Ds.h File Reference

```
#include <Eigen/Dense>
```

Classes

- class [DSn](#)

Functions

- `Array< long double, Dynamic, 1 >` [prepareDSample](#) (unsigned int *M*)

7.8.1 Function Documentation

7.8.1.1 `Array<long double,Dynamic,1> prepareDSample (unsigned int M)`

7.9 C:/Development/core/FFTWLibrary.cpp File Reference

```
#include "FFTWLibrary.h"
```

7.10 C:/Development/core/FFTWLibrary.h File Reference

```
#include "Utilities.h"  
#include <windows.h>  
#include <QLibrary>  
#include <fftw3.h>
```

Classes

- class [CFFTW](#)

Class interface for FFTW's libfftw library.

7.11 C:/Development/core/GAOptimization.cpp File Reference

```
#include "GAOptimization.h"
#include "GRealGenome.h"
```

Macros

- #define [INstantiate_Real_Genome](#)

7.11.1 Macro Definition Documentation

7.11.1.1 #define [INstantiate_Real_Genome](#)

7.12 C:/Development/core/GAOptimization.h File Reference

```
#include "ParserOptimizationProblem.h"
#include "ga.h"
```

Classes

- class [CGAModelOptimization](#)

Class for the genetic algorithm, i.e. interface for GALIB.

7.13 C:/Development/core/Hypercube.cpp File Reference

```
#include "Hypercube.h"
#include "Model.h"
```

7.14 C:/Development/core/Hypercube.h File Reference

```
#include "RandomGenerators.h"
#include "Model.h"
```

Classes

- class [GoSUM::CHypercube](#)
- class [GoSUM::CModelHypercube](#)
- class [GoSUM::CDSampleHC](#)
- class [GoSUM::CMonteCarloHC](#)

- class [GoSUM::CCVoronoiHC](#)
- class [GoSUM::CLCVoronoiHC](#)

Namespaces

- namespace [GoSUM](#)
Namespace for [GoSUM](#) model.

7.15 C:/Development/core/MADSOptimization.cpp File Reference

```
#include "MADSOptimization.h"
```

7.16 C:/Development/core/MADSOptimization.h File Reference

```
#include "OptimizationProblem.h"  
#include <Mads.hpp>
```

Classes

- class [CMADS](#)
Class for the mesh addaptive direct serach, i.e. interface for NOMAD.
- class [CMADSEvaluator](#)
Subclass of the [NOMAD::Evaluator](#) class.

7.17 C:/Development/core/MatlabLibrary.cpp File Reference

```
#include "MatlabLibrary.h"  
#include <windows.h>
```

7.18 C:/Development/core/MatlabLibrary.h File Reference

```
#include "Utilities.h"  
#include <QLibrary>  
#include <mat.h>  
#include <engine.h>
```

Classes

- class [CMATLAB](#)
Class interface for Matlab's libmat and libmx dynamic libraries.

Typedefs

- typedef MATFile *(* [matftype1](#))(const char *, const char *)
Typedefs for functions in Matlab's libmat dynamic library.
- typedef int(* [matftype2](#))(MATFile *)
- typedef int(* [matftype3](#))(MATFile *, const char *, const mxArray *)
- typedef mxArray *(* [matftype4](#))(MATFile *, const char *)
- typedef mxArray *(* [mxftype1](#))(mwSize, mwSize, mxComplexity)
Typedefs for functions in Matlab's libmx dynamic library.
- typedef void(* [mxftype2](#))(mxArray *)
- typedef double *(* [mxftype3](#))(const mxArray *)
- typedef void(* [mxftype4](#))(mxArray *pa, double *pr)
- typedef size_t(* [mxftype5](#))(const mxArray *)
- typedef Engine *(* [engftype1](#))(const char *, void *, int *)
Typedefs for functions in Matlab's libeng dynamic library.
- typedef Engine *(* [engftype2](#))(const char *)
- typedef int(* [engftype3](#))(Engine *)
- typedef int(* [engftype4](#))(Engine *, bool *)
- typedef int(* [engftype5](#))(Engine *, bool)
- typedef int(* [engftype6](#))(Engine *, const char *)
- typedef mxArray *(* [engftype7](#))(Engine *, const char *)
- typedef int(* [engftype8](#))(Engine *, const char *, const mxArray *)
- typedef int(* [engftype9](#))(Engine *, char *, int)

7.18.1 Typedef Documentation

7.18.1.1 typedef Engine *(* [engftype1](#))(const char *, void *, int *)

Typedefs for functions in Matlab's libeng dynamic library.

7.18.1.2 typedef Engine *(* [engftype2](#))(const char *)

7.18.1.3 typedef int(* [engftype3](#))(Engine *)

7.18.1.4 typedef int(* [engftype4](#))(Engine *, bool *)

7.18.1.5 typedef int(* [engftype5](#))(Engine *, bool)

7.18.1.6 typedef int(* [engftype6](#))(Engine *, const char *)

7.18.1.7 typedef mxArray *(* [engftype7](#))(Engine *, const char *)

7.18.1.8 typedef int(* [engftype8](#))(Engine *, const char *, const mxArray *)

7.18.1.9 typedef int(* [engftype9](#))(Engine *, char *, int)

7.18.1.10 typedef MATFile *(* [matftype1](#))(const char *, const char *)

Typedefs for functions in Matlab's libmat dynamic library.

7.18.1.11 `typedef int>(* matftype2)(MATFile *)`

7.18.1.12 `typedef int>(* matftype3)(MATFile *, const char *, const mxArray *)`

7.18.1.13 `typedef mxArray>(* matftype4)(MATFile *, const char *)`

7.18.1.14 `typedef mxArray>(* mxftype1)(mwSize,mwSize,mxComplexity)`

Typedefs for functions in Matlab's libmx dynamic library.

7.18.1.15 `typedef void(* mxftype2)(mxArray *)`

7.18.1.16 `typedef double(* mxftype3)(const mxArray *)`

7.18.1.17 `typedef void(* mxftype4)(mxArray *pa, double *pr)`

7.18.1.18 `typedef size_t(* mxftype5)(const mxArray *)`

7.19 C:/Development/core/Model.cpp File Reference

```
#include "Model.h"
#include "Hypercube.h"
#include "OriginalModel.h"
```

7.20 C:/Development/core/Model.h File Reference

```
#include "ModelVariable.h"
#include "Constraints.h"
```

Classes

- class [GoSUM::CModelVariables](#)
Class for the vector of model variables (inputs & outputs).
- class [GoSUM::CInputParameters](#)
Class for [GoSUM](#) input parameters.
- class [GoSUM::COutputStates](#)
Class for [GoSUM](#) output states.

Namespaces

- namespace [GoSUM](#)
Namespace for [GoSUM](#) model.

7.21 C:/Development/core/ModelVariable.cpp File Reference

```
#include "ModelVariable.h"
```

7.22 C:/Development/core/ModelVariable.h File Reference

```
#include "Probability.h"
```

Classes

- class [GoSUM::CModelVariable](#)
Class for any variable in the [GoSUM](#) model (input or output).
- class [GoSUM::TModelVariable< R, S, t >](#)
- class [GoSUM::CDiscreteMV](#)
- class [GoSUM::CContinuousMV](#)

Namespaces

- namespace [GoSUM](#)
Namespace for [GoSUM](#) model.

7.23 C:/Development/core/OptimizationProblem.cpp File Reference

```
#include "OptimizationProblem.h"
```

7.24 C:/Development/core/OptimizationProblem.h File Reference

```
#include "Utilities.h"
```

Classes

- class [COptimizationVariable](#)
- class [COptimizationProblem](#)
Class for the optimization problem.

7.25 C:/Development/core/OriginalModel.cpp File Reference

```
#include "OriginalModel.h"  
#include <QFileInfo>
```

Variables

- enum
[GoSUM::CEvaluator::evaluatortype](#) [GoSUM](#)

7.25.1 Variable Documentation

7.25.1.1 enum GoSUM::CReduction::cutofftype GoSUM

7.26 C:/Development/core/OriginalModel.h File Reference

```
#include <QProcess>
#include <QFile>
#include "Model.h"
#include "MatlabLibrary.h"
```

Classes

- class [GoSUM::CEvaluator](#)
Class for [GoSUM](#) model evaluator.
- class [GoSUM::CModelEvaluator](#)
Class for [GoSUM](#) model evaluator.
- class [GoSUM::CExeEvaluator](#)
Template for the evaluator with some file i/o and .exe derived from [CModelEvaluator](#).
- class [GoSUM::CExeAsciiEvaluator](#)
Class for the evaluator with ascii i/o and .exe.
- class [GoSUM::CExeMatEvaluator](#)
Class for the evaluator with Matlab .mat i/o and .exe.
- class [GoSUM::CMatlabShellEvaluator](#)
Class for the evaluator through Matlab engine and .mat i/o.
- class [GoSUM::CMatlabEngineEvaluator](#)
Class for the evaluator through Matlab engine derived from [CModelEvaluator](#).

Namespaces

- namespace [GoSUM](#)
Namespace for [GoSUM](#) model.

7.27 C:/Development/core/ParserOptimizationProblem.cpp File Reference

```
#include "ParserOptimizationProblem.h"
```

7.28 C:/Development/core/ParserOptimizationProblem.h File Reference

```
#include "OptimizationProblem.h"
#include "fparser.hh"
#include "AnalyticalModel.h"
```

Classes

- class [GoSUM::CParserOptimizationProblem](#)
Class for the optimization problem with parser functions.
- class [GoSUM::CModelOptimizationVariable](#)
- class [GoSUM::CSimpleOptimizationProblem](#)
Class for the optimization problem based only on [GoSUM](#) input parameters.
- class [GoSUM::COriginalOptimizationProblem](#)
Class for the optimization problem based on [GoSUM](#) analytical model.
- class [GoSUM::CA analyticalOptimizationProblem](#)
Class for the optimization problem based on [GoSUM](#) analytical model.

Namespaces

- namespace [GoSUM](#)
Namespace for [GoSUM](#) model.

7.29 C:/Development/core/Plot2D.cpp File Reference

```
#include "Plot2D.h"
```

7.30 C:/Development/core/Plot2D.h File Reference

```
#include <Eigen/Dense>
#include <QLayout>
#include "qwt_plot.h"
#include "qwt_plot_marker.h"
#include "qwt_plot_curve.h"
#include "qwt_plot_layout.h"
#include "qwt_legend.h"
#include "qwt_legend_item.h"
#include "qwt_symbol.h"
#include "qwt_plot_grid.h"
#include "qwt_plot_histogram.h"
#include "qwt_plot_spectrogram.h"
#include "qwt_color_map.h"
#include "qwt_matrix_raster_data.h"
#include "qwt_scale_widget.h"
#include "qwt_plot_picker.h"
#include "qwt_picker_machine.h"
#include "qwt_plot_renderer.h"
```

Classes

- class [CPlot2D](#)

7.31 C:/Development/core/Probability.cpp File Reference

```
#include "Probability.h"
```

Functions

- bool [chiSquareTest](#) (const [CContinuousRV](#) &_aRV, const [CContinuousSample](#) &_aS, const double chi_alpha)
- double [KolmogorovSmirnovTest](#) (const [CContinuousRV](#) &_aRV, const [CContinuousSample](#) &_aS)

7.31.1 Function Documentation

7.31.1.1 bool [chiSquareTest](#) (const [CContinuousRV](#) & *aRV*, const [CContinuousSample](#) & *aS*, const double *chi_alpha*)

brief Function that implements chi-sqaure test to determine if a continuous random variable is a good fit to a continuous sample.

7.31.1.2 double [KolmogorovSmirnovTest](#) (const [CContinuousRV](#) & *aRV*, const [CContinuousSample](#) & *aS*)

brief Function that implements Kolmogorov-Smirnov test to determine if a continuous random variable is a good fit to a continuous sample.

7.32 C:/Development/core/Probability.h File Reference

```
#include "RandomVariable.h"
```

Functions

- bool [chiSquareTest](#) (const [CContinuousRV](#) &_aRV, const [CContinuousSample](#) &_aS, const double chi_alpha)
- double [KolmogorovSmirnovTest](#) (const [CContinuousRV](#) &_aRV, const [CContinuousSample](#) &_aS)

7.32.1 Function Documentation

7.32.1.1 bool [chiSquareTest](#) (const [CContinuousRV](#) & *aRV*, const [CContinuousSample](#) & *aS*, const double *chi_alpha*)

brief Function that implements chi-sqaure test to determine if a continuous random variable is a good fit to a continuous sample.

7.32.1.2 double [KolmogorovSmirnovTest](#) (const [CContinuousRV](#) & *aRV*, const [CContinuousSample](#) & *aS*)

brief Function that implements Kolmogorov-Smirnov test to determine if a continuous random variable is a good fit to a continuous sample.

7.33 C:/Development/core/ran.h File Reference

Classes

- class [Ran](#)

Typedefs

- typedef unsigned long long int [Ullong](#)

7.33.1 Typedef Documentation

7.33.1.1 typedef unsigned long long int Ullong

7.34 C:/Development/core/RandomGenerators.cpp File Reference

```
#include "RandomGenerators.h"
```

Macros

- #define [IA](#) 16807L
- #define [IM](#) 2147483647L
- #define [AM](#) (1.0/[IM](#))
- #define [IQ](#) 127773L
- #define [IR](#) 2836L
- #define [NTAB](#) 32
- #define [NDIV](#) (1+([IM](#)-1)/[NTAB](#))
- #define [EPS](#) 1.2e-7
- #define [RNMx](#) (1.0-[EPS](#))
- #define [IM1](#) 2147483563L
- #define [IM2](#) 2147483399L
- #define [AM](#) (1.0/[IM1](#))
- #define [IMM1](#) ([IM1](#)-1)
- #define [IA1](#) 40014L
- #define [IA2](#) 40692L
- #define [IQ1](#) 53668L
- #define [IQ2](#) 52774L
- #define [IR1](#) 12211L
- #define [IR2](#) 3791
- #define [NTAB](#) 32
- #define [NDIV](#) (1+[IMM1](#)/[NTAB](#))
- #define [EPS](#) 1.2e-7
- #define [RNMx](#) (1.0-[EPS](#))
- #define [MBIG](#) 1000000000
- #define [MSEED](#) 161803398
- #define [MZ](#) 0
- #define [FAC](#) (1.0/[MBIG](#))

7.34.1 Macro Definition Documentation

7.34.1.1 `#define AM (1.0/IM)`

7.34.1.2 `#define AM (1.0/IM1)`

7.34.1.3 `#define EPS 1.2e-7`

7.34.1.4 `#define EPS 1.2e-7`

7.34.1.5 `#define FAC (1.0/MBIG)`

7.34.1.6 `#define IA 16807L`

7.34.1.7 `#define IA1 40014L`

7.34.1.8 `#define IA2 40692L`

7.34.1.9 `#define IM 2147483647L`

7.34.1.10 `#define IM1 2147483563L`

7.34.1.11 `#define IM2 2147483399L`

7.34.1.12 `#define IMM1 (IM1-1)`

7.34.1.13 `#define IQ 127773L`

7.34.1.14 `#define IQ1 53668L`

7.34.1.15 `#define IQ2 52774L`

7.34.1.16 `#define IR 2836L`

7.34.1.17 `#define IR1 12211L`

7.34.1.18 `#define IR2 3791`

7.34.1.19 `#define MBIG 1000000000`

7.34.1.20 `#define MSEED 161803398`

7.34.1.21 `#define MZ 0`

7.34.1.22 `#define NDIV (1+(IM-1)/NTAB)`

7.34.1.23 `#define NDIV (1+IMM1/NTAB)`

7.34.1.24 `#define NTAB 32`

7.34.1.25 `#define NTAB 32`

7.34.1.26 `#define RNMIX (1.0-EPS)`

7.34.1.27 `#define RNMIX (1.0-EPS)`

7.35 C:/Development/core/RandomGenerators.h File Reference

```
#include <iostream>
#include <Eigen/Dense>
```

Classes

- class [CBaseRNG](#)
Abstract base class for all uniform random number generators (RNG).
- class [CLargePeriodRNG](#)
Class for uniform RNG with large period = 3.138×10^{57} .
- class [CDynamicSystemRNG](#)
Class for dynamic system uniform RNG.
- class [CSmallPeriodRNG](#)
- class [CMediumPeriodRNG](#)
- class [CNLCRNG](#)
- class [CNomadRNG](#)
Class for NOMAD RNG with period = $2^{96}-1$.
- class [CRandomGenerator](#)
Class that holds all implemented uniform RNGs as static. and allows switching between them.

Macros

- `#define` [UINT32_MAX](#) 0xffffffff
- `#define` [NTAB](#) 32
Class for uniform RNG with small period = 10^8 .
- `#define` [NTAB](#) 32
Class for uniform RNG with small period = 10^8 .
- `#define` [NTAB](#) 32
Class for uniform RNG with small period = 10^8 .

7.35.1 Macro Definition Documentation

7.35.1.1 `#define` NTAB 32

Class for uniform RNG with small period = 10^8 .

Class for uniform RNG which is not linearly congruential.

Class for uniform RNG with medium period = 2×10^{18} .

7.35.1.2 `#define` NTAB 32

Class for uniform RNG with small period = 10^8 .

Class for uniform RNG which is not linearly congruential.

Class for uniform RNG with medium period = 2×10^{18} .

7.35.1.3 `#define NTAB 32`

Class for uniform RNG with small period = 10^8 .

Class for uniform RNG which is not linearly congruential.

Class for uniform RNG with medium period = $2 * 10^{18}$.

7.35.1.4 `#define UINT32_MAX 0xffffffff`

7.36 C:/Development/core/RandomVariable.cpp File Reference

```
#include "RandomVariable.h"
#include "FFTWLibrary.h"
```

7.37 C:/Development/core/RandomVariable.h File Reference

```
#include "Sample.h"
```

Classes

- class [CRandomVariable](#)
Abstract base class for all random variables.
- class [TRandomVariable< t, T >](#)
Template of an abstract class - covers two types of random variables: discrete ($t=int, T=CDiscreteSample$) or continuous ($t=double, T=CContinuousSample$).
- class [CDiscreteRV](#)
- class [CContinuousRV](#)
- class [CConstantDRV](#)
Class for constant discrete random variables derived from discrete random variables.
- class [CUniformDRV](#)
Class for uniform discrete random variables derived from discrete random variables.
- class [CEmpiricalDRV](#)
Class for categorical discrete random variables derived from discrete random variables.
- class [CCategoricalDRV](#)
Class for categorical discrete random variables derived from discrete random variables.
- class [CConstantCRV](#)
Class for constant continuous random variables derived from continuous random variables.
- class [CUniformCRV](#)
Class for uniform continuous random variables derived from continuous random variables.
- class [CGaussianCRV](#)
Class for Gaussian continuous random variables derived from continuous random variables.
- class [CExponentialCRV](#)
Class for exponential continuous random variables derived from continuous random variables.
- class [CEmpiricalCRV](#)
Class for empirical continuous random variables derived from continuous random variables.

Macros

- `#define rvExportMacro(F, f, arg)`

Typedefs

- typedef [TRandomVariable](#)< int, [CDiscreteSample](#) > [CTDiscreteRV](#)

Abstract class for all discrete random variables derived as specialization for t=int, T=[TDiscreteSample](#).

- typedef [TRandomVariable](#)
< double, [CContinuousSample](#) > [CTContinuousRV](#)

Abstract class for all continuous random variables derived as specialization with t=double, T=[CContinuousSample](#).

7.37.1 Macro Definition Documentation

7.37.1.1 #define rvExportMacro(F, f, arg)

Value:

```
public: \
ArrayXd export##F(const Array<arg,Dynamic,1> &_x) const \
{
    \
    int i,n=_x.size(); \
    ArrayXd _f(n); \
    for ( i=0; i<n; i++ ) _f(i)=f(_x(i)); \
    return _f; \
}
```

7.37.2 Typedef Documentation

7.37.2.1 typedef TRandomVariable<double,CContinuousSample> CTContinuousRV

Abstract class for all continuous random variables derived as specialization with t=double, T=[CContinuousSample](#).

7.37.2.2 typedef TRandomVariable<int,CDiscreteSample> CTDiscreteRV

Abstract class for all discrete random variables derived as specialization for t=int, T=[TDiscreteSample](#).

7.38 C:/Development/core/Reduction.cpp File Reference

```
#include "Reduction.h"
```

Functions

- bool [compareFirst](#) (std::pair< double, int > pair1, std::pair< double, int > pair2)

Variables

- enum
[GoSUM::CReduction::reductiontype](#) [GoSUM](#)

7.38.1 Function Documentation

7.38.1.1 bool compareFirst (std::pair< double, int > pair1, std::pair< double, int > pair2)

7.38.2 Variable Documentation

7.38.2.1 enum GoSUM::CReduction::cutofftype GoSUM

7.39 C:/Development/core/Reduction.h File Reference

```
#include "SensitivityAnalysis.h"
```

Classes

- class [GoSUM::CReduction](#)
Class for sensitivity analysis of the model.

Namespaces

- namespace [GoSUM](#)
Namespace for [GoSUM](#) model.

7.40 C:/Development/core/Sample.cpp File Reference

```
#include "Sample.h"
```

7.41 C:/Development/core/Sample.h File Reference

```
#include "VariousMath.h"
```

Classes

- class [CSample](#)
Abstract class for all samples.
- class [TSample< t >](#)
Template of an abstract class - covers two types of samples: discrete (t=int) or continuous (t=double).
- class [CDiscreteSample](#)
- class [CContinuousSample](#)
- class [CNumericalSample](#)
Abstract class for numerical discrete samples.
- class [CCategoricalSample](#)
Abstract class for categorical discrete samples.

Typedefs

- typedef [TSample< int >](#) [CTDiscreteSample](#)
Abstract class for all discrete samples derived as specialization with t=int.
- typedef [TSample< double >](#) [CTContinuousSample](#)
Abstract class for all continuous samples derived as specialization with t=double.

7.41.1 Typedef Documentation

7.41.1.1 `typedef TSample<double> CTContinuousSample`

Abstract class for all continuous samples derived as specialization with `t=double`.

7.41.1.2 `typedef TSample<int> CTDiscreteSample`

Abstract class for all discrete samples derived as specialization with `t=int`.

7.42 C:/Development/core/Script.cpp File Reference

```
#include "Script.h"
```

7.43 C:/Development/core/Script.h File Reference

```
#include "Container.h"
```

Classes

- class [GoSUM::CScript](#)
Class for the [GoSUM](#) script format.

Namespaces

- namespace [GoSUM](#)
Namespace for [GoSUM](#) model.

7.44 C:/Development/core/SensitivityAnalysis.cpp File Reference

```
#include "SensitivityAnalysis.h"  
#include "Hypercube.h"
```

7.45 C:/Development/core/SensitivityAnalysis.h File Reference

```
#include "AnalyticalModel.h"
```

Classes

- class [GoSUM::CSensitivityAnalysis](#)
Class for sensitivity analysis of the model.

Namespaces

- namespace [GoSUM](#)

Namespace for [GoSUM](#) model.

7.46 C:/Development/core/Utilities.cpp File Reference

```
#include "Utilities.h"  
#include <windows.h>
```

Functions

- std::wstring [string2WideString](#) (const std::string &s)

Some utility functions.

7.46.1 Function Documentation

7.46.1.1 std::wstring string2WideString (const std::string & s)

Some utility functions.

Converts string to wide string.

7.47 C:/Development/core/Utilities.h File Reference

```
#include <cmath>
```

```

#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <algorithm>
#include <utility>
#include <queue>
#include <boost/math/special_functions/erf.hpp>
#include <boost/math/distributions/chi_squared.hpp>
#include <boost/ptr_container/ptr_vector.hpp>
#include <boost/serialization/split_member.hpp>
#include <boost/serialization/nvp.hpp>
#include <boost/serialization/base_object.hpp>
#include <boost/serialization/utility.hpp>
#include <boost/serialization/vector.hpp>
#include <boost/serialization/string.hpp>
#include <boost/serialization/export.hpp>
#include <boost/ptr_container/serialize_ptr_vector.hpp>
#include <boost/config.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/archive/binary_iarchive.hpp>
#include <boost/archive/binary_oarchive.hpp>
#include <boost/archive/xml_iarchive.hpp>
#include <boost/archive/xml_oarchive.hpp>
#include <boost/thread.hpp>
#include <boost/signal.hpp>
#include <boost/bind.hpp>
#include <Eigen/Dense>

```

Namespaces

- namespace [boost](#)
- namespace [boost::serialization](#)

Macros

- `#define` [_USE_MATH_DEFINES](#)
- `#define` [SIGNALSLIB_HPP_INCLUDED](#)
- `#define` [boostSerializeArrayXdiMacro\(A\)](#)
Boost serialization for some Eigen Arrays.
- `#define` [boostSerializeArrayXXdiMacro\(B\)](#)

Functions

- `std::wstring` [string2WideString](#) (const std::string &s)
Some utility functions.

7.47.1 Macro Definition Documentation

7.47.1.1 `#define` [_USE_MATH_DEFINES](#)

7.47.1.2 #define boostSerializeArrayXdiMacro(A)

Value:

```
template<class Archive> \
void save(Archive &ar, const A &a, const unsigned int version) \
{   int i,n=int(a.size()); \
    ar << boost::serialization::make_nvp("size",n); \
    for ( i=0; i<n; i++ ) ar << boost::serialization::make_nvp("data",a(i)); \
} \
template<class Archive> \
void load(Archive &ar, A &a, const unsigned int version) \
{   int i,n; \
    ar >> boost::serialization::make_nvp("size",n); \
    a.resize(n); \
    for ( i=0; i<n; i++ ) ar >> boost::serialization::make_nvp("data",a(i)); \
}
```

Boost serialization for some Eigen Arrays.

7.47.1.3 #define boostSerializeArrayXXdiMacro(B)

Value:

```
template<class Archive> \
void save(Archive &ar, const B &a, const unsigned int version) \
{   int i,j,n=int(a.rows()),m=int(a.cols()); \
    ar << boost::serialization::make_nvp("rows",n); \
    ar << boost::serialization::make_nvp("cols",m); \
    for ( i=0; i<n; i++ ) \
        for ( j=0; j<m; j++ ) \
            ar << boost::serialization::make_nvp("data",a(i,j)); \
} \
template<class Archive> \
void load(Archive &ar, B &a, const unsigned int version) \
{   int i,j,n,m; \
    ar >> boost::serialization::make_nvp("rows",n); \
    ar >> boost::serialization::make_nvp("cols",m); \
    a.resize(n,m); \
    for ( i=0; i<n; i++ ) \
        for ( j=0; j<m; j++ ) \
            ar >> boost::serialization::make_nvp("data",a(i,j)); \
}
```

7.47.1.4 #define SIGNALSLIB_HPP_INCLUDED

7.47.2 Function Documentation

7.47.2.1 std::wstring string2WideString (const std::string & s)

Some utility functions.

Converts string to wide string.

7.48 C:/Development/core/VariousMath.cpp File Reference

```
#include "VariousMath.h"
```

Functions

- double [linearInterpolation](#) (const ArrayXd &X, const ArrayXd &Y, double x)

$$\text{Linear interpolation } Y(x) = Y_l + ((Y_r - Y_l) / (X_r - X_l)) * (x - X_l).$$

- void `cumulativeSum` (ArrayXd &a)
Returns cummulative sum of elements of the array.
- double `variance` (ArrayXd &a)
Returns variance of elements of the array.
- ArrayXi `permutation` (int N)
Generates one random permutation of (1,...,N).
- ArrayXXi `permutations` (int n, int N)
Generates n random permutations of (1,...,N).
- double `squareDistance` (const ArrayXd &a, const ArrayXd &b)
Returns squared distance between a and b as points.
- int `findNearest` (const std::vector< ArrayXd > &v, const ArrayXd &y)
Returns index of the point beetween points v nearest to the point y.

7.48.1 Function Documentation

7.48.1.1 void cumulativeSum (ArrayXd & a)

Returns cummulative sum of elements of the array.

7.48.1.2 int findNearest (const std::vector< ArrayXd > & v, const ArrayXd & y)

Returns index of the point beetween points v nearest to the point y.

7.48.1.3 double linearInterpolation (const ArrayXd & X, const ArrayXd & Y, double x)

Linear interpolation $Y(x)=Yl+((Yr-Yl)/(Xr-Xl))*(x-Xl)$.

7.48.1.4 ArrayXi permutation (int N)

Generates one random permutation of (1,...,N).

7.48.1.5 ArrayXXi permutations (int n, int N)

Generates n random permutations of (1,...,N).

7.48.1.6 double squareDistance (const ArrayXd & a, const ArrayXd & b)

Returns squared distance between a and b as points.

7.48.1.7 double variance (ArrayXd & a)

Returns variance of elements of the array.

7.49 C:/Development/core/VariousMath.h File Reference

```
#include "Utilities.h"
#include "ran.h"
#include "Ds.h"
```


Classes

- class [Cx_ZetaGammamax](#)
Class for the function $f(x)=x\text{-zeta}\cdot\text{gamma}^{[7]}(x)$.

Macros

- `#define TINY 100*(std::numeric_limits<double>::epsilon())`

Functions

- `template<class T > double findRoot (const T &fnc, double a, double b)`
Finds $_x$ in the interval (a,b) such that $f(_x)=0$.
- `ArrayXi permutation (int N)`
Generates one random permutation of $(1,...,N)$.
- `ArrayXXi permutations (int n, int N)`
Generates n random permutations of $(1,...,N)$.
- `double variance (ArrayXd &a)`
Returns variance of elements of the array.
- `double linearInterpolation (const ArrayXd &X, const ArrayXd &Y, double x)`
Linear interpolation $Y(x)=Y_l+((Y_r-Y_l)/(X_r-X_l))(x-X_l)$.*
- `void cumulativeSum (ArrayXd &a)`
Returns cummulative sum of elements of the array.
- `double squareDistance (const ArrayXd &a, const ArrayXd &b)`
Returns squared distance between a and b as points.
- `int findNearest (const std::vector< ArrayXd > &v, const ArrayXd &y)`
Returns index of the point beetween points v nearest to the point y .

7.49.1 Macro Definition Documentation

7.49.1.1 `#define TINY 100*(std::numeric_limits<double>::epsilon())`

7.49.2 Function Documentation

7.49.2.1 `void cumulativeSum (ArrayXd & a)`

Returns cummulative sum of elements of the array.

7.49.2.2 `int findNearest (const std::vector< ArrayXd > & v, const ArrayXd & y)`

Returns index of the point beetween points v nearest to the point y .

7.49.2.3 `template<class T > double findRoot (const T & fnc, double a, double b)`

Finds $_x$ in the interval (a,b) such that $f(_x)=0$.

Returns

True if root is found, false otherwise.

7.49.2.4 `double linearInterpolation (const ArrayXd & X, const ArrayXd & Y, double x)`

Linear interpolation $Y(x)=Y_l+((Y_r-Y_l)/(X_r-X_l))*(x-X_l)$.

7.49.2.5 `ArrayXi permutation (int N)`

Generates one random permutation of $(1,...,N)$.

7.49.2.6 `ArrayXXi permutations (int n, int N)`

Generates n random permutations of $(1,...,N)$.

7.49.2.7 `double squareDistance (const ArrayXd & a, const ArrayXd & b)`

Returns squared distance between a and b as points.

7.49.2.8 `double variance (ArrayXd & a)`

Returns variance of elements of the array.

Index

- ~CA analyticalModel
 - GoSUM::CA analyticalModel, [15](#)
- ~CA analyticalOptimizationProblem
 - GoSUM::CA analyticalOptimizationProblem, [18](#)
- ~CCSvcSAM
 - GoSUM::CCSvcSAM, [64](#)
- ~CCVoronoiHC
 - GoSUM::CCVoronoiHC, [65](#)
- ~CCategoricalDRV
 - CCategoricalDRV, [21](#)
- ~CCategoricalSample
 - CCategoricalSample, [24](#)
- ~CConstantCRV
 - CConstantCRV, [27](#)
- ~CConstantDRV
 - CConstantDRV, [30](#)
- ~CConstraints
 - GoSUM::CConstraints, [34](#)
- ~CContainer
 - GoSUM::CContainer, [43](#)
- ~CContinuousSample
 - CContinuousSample, [62](#)
- ~CDSampleHC
 - GoSUM::CDSampleHC, [72](#)
- ~CDiscreteSample
 - CDiscreteSample, [70](#)
- ~CDynamicSystemRNG
 - CDynamicSystemRNG, [73](#)
- ~CEmpiricalCRV
 - CEmpiricalCRV, [76](#)
- ~CEmpiricalDRV
 - CEmpiricalDRV, [80](#)
- ~CEpsSvrSAM
 - GoSUM::CEpsSvrSAM, [83](#)
- ~CEvaluator
 - GoSUM::CEvaluator, [85](#)
- ~CExeAsciiEvaluator
 - GoSUM::CExeAsciiEvaluator, [88](#)
- ~CExeEvaluator
 - GoSUM::CExeEvaluator, [90](#)
- ~CExeMatEvaluator
 - GoSUM::CExeMatEvaluator, [92](#)
- ~CExponentialCRV
 - CExponentialCRV, [95](#)
- ~CFFTW
 - CFFTW, [97](#)
- ~CGAModelOptimization
 - CGAModelOptimization, [99](#)
- ~CGaussianCRV
 - CGaussianCRV, [102](#)
- ~CHypercube
 - GoSUM::CHypercube, [106](#)
- ~CInputParameters
 - GoSUM::CInputParameters, [109](#)
- ~CLCVoronoiHC
 - GoSUM::CLCVoronoiHC, [113](#)
- ~CLargePeriodRNG
 - CLargePeriodRNG, [111](#)
- ~CMADS
 - CMADS, [115](#)
- ~CMADSEvaluator
 - CMADSEvaluator, [118](#)
- ~CMATLAB
 - CMATLAB, [120](#)
- ~CMatlabEngineEvaluator
 - GoSUM::CMatlabEngineEvaluator, [125](#)
- ~CMatlabShellEvaluator
 - GoSUM::CMatlabShellEvaluator, [127](#)
- ~CMediumPeriodRNG
 - CMediumPeriodRNG, [129](#)
- ~CModelConstraints
 - GoSUM::CModelConstraints, [130](#)
- ~CModelEvaluator
 - GoSUM::CModelEvaluator, [133](#)
- ~CModelHypercube
 - GoSUM::CModelHypercube, [135](#)
- ~CModelOptimizationVariable
 - GoSUM::CModelOptimizationVariable, [137](#)
- ~CModelVariable
 - GoSUM::CModelVariable, [140](#)
- ~CModelVariables
 - GoSUM::CModelVariables, [146](#)
- ~CMonteCarloHC
 - GoSUM::CMonteCarloHC, [151](#)
- ~CNLCRNG
 - CNLCRNG, [152](#)
- ~CNomadRNG
 - CNomadRNG, [154](#)
- ~CNuSvcSAM
 - GoSUM::CNuSvcSAM, [158](#)
- ~CNuSvrSAM
 - GoSUM::CNuSvrSAM, [159](#)
- ~CNumericalSample
 - CNumericalSample, [156](#)
- ~COneClassSAM
 - GoSUM::COneClassSAM, [160](#)
- ~COptimizationProblem
 - COptimizationProblem, [163](#)

- ~COptimizationVariable
 - COptimizationVariable, 168
- ~COriginalOptimizationProblem
 - GoSUM::COriginalOptimizationProblem, 170
- ~COutputStates
 - GoSUM::COutputStates, 172
- ~CParserOptimizationProblem
 - GoSUM::CParserOptimizationProblem, 175
- ~CPlot2D
 - CPlot2D, 179
- ~CRandomVariable
 - CRandomVariable, 186
- ~CReduction
 - GoSUM::CReduction, 190
- ~CSample
 - CSample, 195
- ~CScript
 - GoSUM::CScript, 198
- ~CSensitivityAnalysis
 - GoSUM::CSensitivityAnalysis, 210
- ~CSimpleOptimizationProblem
 - GoSUM::CSimpleOptimizationProblem, 216
- ~CSingleAM
 - GoSUM::CSingleAM, 221
- ~CSmallPeriodRNG
 - CSmallPeriodRNG, 225
- ~CUniformCRV
 - CUniformCRV, 227
- ~CUniformDRV
 - CUniformDRV, 231
- ~DSn
 - DSn, 234
- ~TModelVariable
 - GoSUM::TModelVariable, 240
- ~TRandomVariable
 - TRandomVariable, 245
- ~TSample
 - TSample, 248
- _USE_MATH_DEFINES
 - Utilities.h, 272
- a
 - CUniformCRV, 229
 - CUniformDRV, 233
- a2
 - Cx_ZetaGammax, 233
- AM
 - RandomGenerators.cpp, 265
- aader
 - GoSUM::CSensitivityAnalysis, 213
- absoluteAverageDerivative
 - GoSUM::CSensitivityAnalysis, 211
- absoluteaveragederivative
 - GoSUM::CReduction, 190
- add
 - GoSUM::CModelVariables, 146
- addConstraintExpression
 - GoSUM::CParserOptimizationProblem, 175
- addDefaultVariable
 - GoSUM::CContainer, 43
- addExpression
 - GoSUM::CConstraints, 34
- addInput
 - GoSUM::CContainer, 43
 - GoSUM::CScript, 198
- addInputs
 - GoSUM::CContainer, 44
 - GoSUM::CScript, 199
- addOptimizationConstraint
 - GoSUM::CContainer, 44
 - GoSUM::CScript, 199
- addOutput
 - GoSUM::CContainer, 44
 - GoSUM::CScript, 199
- addOutputs
 - GoSUM::CContainer, 44
 - GoSUM::CScript, 199
- addVariable
 - COptimizationProblem, 163
 - GoSUM::CContainer, 44
 - GoSUM::CScript, 200
- addVariables
 - GoSUM::CContainer, 44
 - GoSUM::CScript, 200
- ader
 - GoSUM::CSensitivityAnalysis, 213
- alpha2
 - GoSUM::CHypercube, 107
- analytical
 - GoSUM::CContainer, 56
- analyticalModel
 - GoSUM::CContainer, 44
- anova
 - GoSUM::CReduction, 190
- ArrayXd2NOMADPoint
 - CMADS, 115
- averageDerivative
 - GoSUM::CSensitivityAnalysis, 211
- averagederivative
 - GoSUM::CReduction, 190
- b
 - CUniformCRV, 229
 - CUniformDRV, 233
- baseName
 - GoSUM::CModelVariables, 146
- basename
 - GoSUM::CModelVariables, 150
- beta
 - CExponentialCRV, 97
- beta2
 - GoSUM::CHypercube, 107
- boost, 11
 - boost::serialization, 11
- boost::serialization::access
 - CCategoricalDRV, 23
 - CCategoricalSample, 25
 - CConstantCRV, 28

- CConstantDRV, [32](#)
- CContinuousRV, [60](#)
- CContinuousSample, [63](#)
- CDiscreteRV, [69](#)
- CDiscreteSample, [71](#)
- CEmpiricalCRV, [78](#)
- CEmpiricalDRV, [81](#)
- CExponentialCRV, [96](#)
- CGAModelOptimization, [99](#)
- CGaussianCRV, [103](#)
- CMADS, [117](#)
- CNumericalSample, [157](#)
- COptimizationProblem, [166](#)
- COptimizationVariable, [169](#)
- CRandomVariable, [188](#)
- CSample, [196](#)
- CUniformCRV, [229](#)
- CUniformDRV, [232](#)
- GoSUM::CA analyticalModel, [16](#)
- GoSUM::CA analyticalOptimizationProblem, [18](#)
- GoSUM::CConstraints, [35](#)
- GoSUM::CContainer, [56](#)
- GoSUM::CContinuousMV, [59](#)
- GoSUM::CCSvcSAM, [64](#)
- GoSUM::CCVoronoiHC, [66](#)
- GoSUM::CDiscreteMV, [68](#)
- GoSUM::CDSampleHC, [72](#)
- GoSUM::CEpsSvrSAM, [83](#)
- GoSUM::CEvaluator, [86](#)
- GoSUM::CExeAsciiEvaluator, [88](#)
- GoSUM::CExeEvaluator, [91](#)
- GoSUM::CExeMatEvaluator, [93](#)
- GoSUM::CHypercube, [107](#)
- GoSUM::CInputParameters, [110](#)
- GoSUM::CLCVoronoiHC, [114](#)
- GoSUM::CMatlabEngineEvaluator, [126](#)
- GoSUM::CMatlabShellEvaluator, [128](#)
- GoSUM::CModelConstraints, [131](#)
- GoSUM::CModelEvaluator, [133](#)
- GoSUM::CModelHypercube, [136](#)
- GoSUM::CModelOptimizationVariable, [138](#)
- GoSUM::CModelVariable, [143](#)
- GoSUM::CModelVariables, [150](#)
- GoSUM::CMonteCarloHC, [152](#)
- GoSUM::CNuSvcSAM, [158](#)
- GoSUM::CNuSvrSAM, [159](#)
- GoSUM::COneClassSAM, [161](#)
- GoSUM::COriginalOptimizationProblem, [171](#)
- GoSUM::COutputStates, [173](#)
- GoSUM::CParserOptimizationProblem, [177](#)
- GoSUM::CREduction, [192](#)
- GoSUM::CSensitivityAnalysis, [213](#)
- GoSUM::CSimpleOptimizationProblem, [218](#)
- GoSUM::CSingleAM, [222](#)
- GoSUM::TModelVariable, [244](#)
- TRandomVariable, [246](#)
- TSample, [251](#)
- boostSerializeArrayXXdiMacro
- Utilities.h, [273](#)
- boostSerializeArrayXdiMacro
- Utilities.h, [272](#)
- C
 - CGAModelOptimization, [99](#)
 - GoSUM::CSingleAM, [223](#)
- c
 - CConstantDRV, [32](#)
- CRandomGenerator
 - dynamicsystem, [181](#)
 - largeperiod, [181](#)
 - mediumperiod, [181](#)
 - nomads, [181](#)
 - nonlinearcongruential, [181](#)
 - smallperiod, [181](#)
- CRandomVariable
 - categorical, [185](#)
 - constantcontinuous, [185](#)
 - constantdiscrete, [185](#)
 - empiricalcontinuous, [185](#)
 - empiricaldiscrete, [185](#)
 - exponential, [185](#)
 - gaussian, [185](#)
 - uniformcontinuous, [185](#)
 - uniformdiscrete, [185](#)
- C:/Development/core/AnalyticalModel.cpp, [253](#)
- C:/Development/core/AnalyticalModel.h, [253](#)
- C:/Development/core/Constraints.cpp, [254](#)
- C:/Development/core/Constraints.h, [254](#)
- C:/Development/core/Container.cpp, [254](#)
- C:/Development/core/Container.h, [254](#)
- C:/Development/core/Ds.cpp, [255](#)
- C:/Development/core/Ds.h, [255](#)
- C:/Development/core/FFTWLibrary.cpp, [255](#)
- C:/Development/core/FFTWLibrary.h, [255](#)
- C:/Development/core/GAOptimization.cpp, [256](#)
- C:/Development/core/GAOptimization.h, [256](#)
- C:/Development/core/Hypercube.cpp, [256](#)
- C:/Development/core/Hypercube.h, [256](#)
- C:/Development/core/MADSOptimization.cpp, [257](#)
- C:/Development/core/MADSOptimization.h, [257](#)
- C:/Development/core/MatlabLibrary.cpp, [257](#)
- C:/Development/core/MatlabLibrary.h, [257](#)
- C:/Development/core/Model.cpp, [259](#)
- C:/Development/core/Model.h, [259](#)
- C:/Development/core/ModelVariable.cpp, [259](#)
- C:/Development/core/ModelVariable.h, [260](#)
- C:/Development/core/OptimizationProblem.cpp, [260](#)
- C:/Development/core/OptimizationProblem.h, [260](#)
- C:/Development/core/OriginalModel.cpp, [260](#)
- C:/Development/core/OriginalModel.h, [261](#)
- C:/Development/core/ParserOptimizationProblem.cpp, [261](#)
- C:/Development/core/ParserOptimizationProblem.h, [261](#)
- C:/Development/core/Plot2D.cpp, [262](#)
- C:/Development/core/Plot2D.h, [262](#)
- C:/Development/core/Probability.cpp, [263](#)

- C:/Development/core/Probability.h, 263
- C:/Development/core/RandomGenerators.cpp, 264
- C:/Development/core/RandomGenerators.h, 266
- C:/Development/core/RandomVariable.cpp, 267
- C:/Development/core/RandomVariable.h, 267
- C:/Development/core/Reduction.cpp, 268
- C:/Development/core/Reduction.h, 269
- C:/Development/core/Sample.cpp, 269
- C:/Development/core/Sample.h, 269
- C:/Development/core/Script.cpp, 270
- C:/Development/core/Script.h, 270
- C:/Development/core/SensitivityAnalysis.cpp, 270
- C:/Development/core/SensitivityAnalysis.h, 270
- C:/Development/core/Utilities.cpp, 271
- C:/Development/core/Utilities.h, 271
- C:/Development/core/VariousMath.cpp, 273
- C:/Development/core/VariousMath.h, 274
- C:/Development/core/ran.h, 264
- CA analyticalModel
 - GoSUM::CA analyticalModel, 15
- CA analyticalOptimizationProblem
 - GoSUM::CA analyticalOptimizationProblem, 18
- CBaseRNG, 19
 - rnd, 19
 - rndi, 19
 - setSeed, 19
- CCSvcSAM
 - GoSUM::CCSvcSAM, 64
- CCVoronoiHC
 - GoSUM::CCVoronoiHC, 65
- CCategoricalDRV, 20
 - ~CCategoricalDRV, 21
 - boost::serialization::access, 23
 - CCategoricalDRV, 21
 - CCategoricalDRV, 21
 - cdf, 23
 - cumulative, 21
 - distributionName, 21
 - distributionType, 21
 - doQuantile, 21
 - expandedSize, 22
 - expectedValue, 22
 - exportDomain, 22
 - isDistributionDefined, 22
 - maxValue, 22
 - minValue, 22
 - p, 23
 - probability, 22
 - serialize, 22
 - setDistribution, 22
 - variance, 22
- CCategoricalSample, 23
 - ~CCategoricalSample, 24
 - boost::serialization::access, 25
 - CCategoricalSample, 24
 - categories, 25
 - CCategoricalSample, 24
 - maxValue, 24
 - minValue, 24
 - readSampleValue, 24
 - serialize, 25
 - setSampleSize, 25
 - variance, 25
 - writeSampleValue, 25
- CConstantCRV, 25
 - ~CConstantCRV, 27
 - boost::serialization::access, 28
 - CConstantCRV, 27
 - CConstantCRV, 27
 - constantValue, 27
 - cumulative, 27
 - distributionName, 27
 - distributionType, 27
 - doQuantile, 27
 - expectedValue, 27
 - exportDomain, 28
 - maxValue, 28
 - minValue, 28
 - probability, 28
 - serialize, 28
 - setConstantValue, 28
 - setDistribution, 28
 - variance, 28
 - xc, 29
- CConstantDRV, 29
 - ~CConstantDRV, 30
 - boost::serialization::access, 32
 - c, 32
 - CConstantDRV, 30
 - CConstantDRV, 30
 - constantValue, 30
 - cumulative, 30
 - distributionName, 30
 - distributionType, 31
 - doQuantile, 31
 - expectedValue, 31
 - exportDomain, 31
 - maxValue, 31
 - minValue, 31
 - probability, 31
 - serialize, 31
 - setConstantValue, 31
 - setDistribution, 31
 - variance, 32
- CConstraints
 - GoSUM::CConstraints, 34
- CContainer
 - GoSUM::CContainer, 43
- CContinuousMV
 - GoSUM::CContinuousMV, 58, 59
- CContinuousRV, 59
 - boost::serialization::access, 60
 - CContinuousRV, 60
 - CContinuousRV, 60
 - probability, 60
 - quantile, 60

- serialize, 60
- CContinuousSample, 61
 - ~CContinuousSample, 62
 - boost::serialization::access, 63
 - CContinuousSample, 62
 - CContinuousSample, 62
 - computeMeanAndStandardDeviation, 62
 - computeNormalizedHistogram, 62
 - computeNormalizedSubHistogram, 62
 - computeStatistics, 62
 - exportHistogram, 62
 - exportSubHistogram, 62
 - hasDiscreteData, 63
 - mean, 63
 - mu, 63
 - serialize, 63
 - standardDeviation, 63
 - stdv, 63
 - variance, 63
- CDSampleHC
 - GoSUM::CDSampleHC, 72
- CDiscreteMV
 - GoSUM::CDiscreteMV, 67
- CDiscreteRV, 68
 - boost::serialization::access, 69
 - CDiscreteRV, 68
 - CDiscreteRV, 68
 - quantile, 68
 - serialize, 69
- CDiscreteSample, 69
 - ~CDiscreteSample, 70
 - boost::serialization::access, 71
 - CDiscreteSample, 70
 - CDiscreteSample, 70
 - computeNormalizedHistogram, 70
 - computeNormalizedSubHistogram, 70
 - computeStatistics, 70
 - exportHistogram, 70
 - exportSubHistogram, 70
 - serialize, 70
 - variance, 71
- CDynamicSystemRNG, 72
 - ~CDynamicSystemRNG, 73
 - CDynamicSystemRNG, 73
 - CDynamicSystemRNG, 73
 - i, 74
 - j, 74
 - k, 74
 - Long, 73
 - n, 74
 - prepare, 74
 - rnd, 74
 - rndi, 74
 - setSeed, 74
 - w, 74
 - x, 74
- CEmpiricalCRV, 75
 - ~CEmpiricalCRV, 76
- boost::serialization::access, 78
- CEmpiricalCRV, 76
 - cdf, 78
 - CEmpiricalCRV, 76
 - cumulative, 76
 - distributionName, 76
 - distributionType, 76
 - doQuantile, 77
 - expectedValue, 77
 - exportDomain, 77
 - isDistributionDefined, 77
 - maxValue, 77
 - mean, 77
 - minValue, 77
 - mu, 78
 - p, 78
 - pbandwidth, 78
 - probability, 77
 - qbandwidth, 78
 - serialize, 77
 - setDistribution, 77
 - standardDeviation, 78
 - stdv, 78
 - variance, 78
 - x, 78
- CEmpiricalDRV, 78
 - ~CEmpiricalDRV, 80
 - boost::serialization::access, 81
 - CEmpiricalDRV, 80
 - cdf, 82
 - CEmpiricalDRV, 80
 - cumulative, 80
 - distributionName, 80
 - distributionType, 80
 - doQuantile, 80
 - expandedSize, 80
 - expectedValue, 80
 - exportDomain, 81
 - isDistributionDefined, 81
 - maxValue, 81
 - minValue, 81
 - p, 82
 - probability, 81
 - serialize, 81
 - setDistribution, 81
 - variance, 81
- CEpsSvrSAM
 - GoSUM::CEpsSvrSAM, 83
- CEvaluator
 - GoSUM::CEvaluator, 85
- CExeAsciiEvaluator
 - GoSUM::CExeAsciiEvaluator, 88
- CExeEvaluator
 - GoSUM::CExeEvaluator, 90
- CExeMatEvaluator
 - GoSUM::CExeMatEvaluator, 92
- CExponentialCRV, 93
 - ~CExponentialCRV, 95

- beta, [97](#)
- boost::serialization::access, [96](#)
- CExponentialCRV, [95](#)
- CExponentialCRV, [95](#)
- cumulative, [95](#)
- distributionName, [95](#)
- distributionType, [95](#)
- doQuantile, [95](#)
- expectedValue, [95](#)
- exportDomain, [95](#)
- lambda, [97](#)
- maxValue, [96](#)
- minValue, [96](#)
- probability, [96](#)
- rateParameter, [96](#)
- serialize, [96](#)
- setDistribution, [96](#)
- setRateParameter, [96](#)
- variance, [96](#)
- CFFTW, [97](#)
 - ~CFFTW, [97](#)
 - CFFTW, [97](#)
 - CFFTW, [97](#)
 - discreteCosineTransform, [97](#)
 - inverseDiscreteCosineTransform, [97](#)
- CGAModelOptimization, [98](#)
 - ~CGAModelOptimization, [99](#)
 - boost::serialization::access, [99](#)
 - C, [99](#)
 - CGAModelOptimization, [99](#)
 - CGAModelOptimization, [99](#)
 - Fitness, [99](#)
 - GAGenome2ArrayXd, [99](#)
 - optimize, [99](#)
 - pOP, [99](#)
 - popsiz, [100](#)
 - populationSize, [99](#)
 - progressStepsSize, [99](#)
 - serialize, [99](#)
 - setPopulationSize, [99](#)
 - Xbest, [100](#)
 - ybest, [100](#)
- CGaussianCRV, [100](#)
 - ~CGaussianCRV, [102](#)
 - boost::serialization::access, [103](#)
 - CGaussianCRV, [102](#)
 - CGaussianCRV, [102](#)
 - cumulative, [102](#)
 - distributionName, [102](#)
 - distributionType, [102](#)
 - doQuantile, [102](#)
 - expectedValue, [102](#)
 - exportDomain, [102](#)
 - maxValue, [102](#)
 - mean, [102](#)
 - minValue, [102](#)
 - mu, [103](#)
 - probability, [103](#)
 - serialize, [103](#)
 - setDistribution, [103](#)
 - setMean, [103](#)
 - setStandardDeviation, [103](#)
 - sigma, [103](#)
 - sigma2, [103](#)
 - sigmasqrt2, [104](#)
 - standardDeviation, [103](#)
 - variance, [103](#)
- cH
 - TSample, [251](#)
- CHypercube
 - GoSUM::CHypercube, [106](#)
- CInputParameters
 - GoSUM::CInputParameters, [109](#)
- CLCVoronoiHC
 - GoSUM::CLCVoronoiHC, [113](#)
- CLargePeriodRNG, [110](#)
 - ~CLargePeriodRNG, [111](#)
 - CLargePeriodRNG, [111](#)
 - CLargePeriodRNG, [111](#)
 - int64, [111](#)
 - rnd, [111](#)
 - rndi, [111](#)
 - setSeed, [112](#)
 - u, [112](#)
 - Ullong, [111](#)
 - v, [112](#)
 - w, [112](#)
- CMADS, [114](#)
 - ~CMADS, [115](#)
 - ArrayXd2NOMADPoint, [115](#)
 - boost::serialization::access, [117](#)
 - CMADS, [115](#)
 - CMADS, [115](#)
 - evaluationSize, [115](#)
 - ims, [117](#)
 - initalMeshSize, [116](#)
 - initialLHSearch, [116](#)
 - iterationLHSearch, [116](#)
 - lh0, [117](#)
 - lhi, [117](#)
 - maxeval, [117](#)
 - minimalPollSize, [116](#)
 - mps, [117](#)
 - NOMADPoint2ArrayXd, [116](#)
 - optimize, [116](#)
 - pOP, [117](#)
 - progressStepsSize, [116](#)
 - serialize, [116](#)
 - setEvaluationSize, [116](#)
 - setInitialLHSearch, [116](#)
 - setInitialMeshSize, [116](#)
 - setIterationLHSearch, [116](#)
 - setMinimalPollSize, [116](#)
 - Xbest, [117](#)
 - ybest, [117](#)
- CMADSEvaluator, [117](#)

- ~CMADSEvaluator, 118
- CMADSEvaluator, 118
- CMADSEvaluator, 118
- eval_x, 118
- pOP, 118
- CMATLAB, 118
 - ~CMATLAB, 120
 - CMATLAB, 120
 - CMATLAB, 120
 - engClose, 120
 - engEvalString, 120
 - engGetVariable, 121
 - engGetVisible, 121
 - engOpen, 121
 - engOpenSingleUse, 121
 - engOutputBuffer, 121
 - engPutVariable, 121
 - engSetVisible, 121
 - libeng, 122
 - libmat, 122
 - libmx, 122
 - matClose, 121
 - matGet, 121
 - matGetVariable, 121
 - matOpen, 121
 - matPut, 121
 - matPutVariable, 122
 - mxCreateDoubleMatrix, 122
 - mxDestroyArray, 122
 - mxGetNumberOfElements, 122
 - mxGetPr, 122
 - pEngClose, 122
 - pEngEvalString, 123
 - pEngGetVariable, 123
 - pEngGetVisible, 123
 - pEngOpen, 123
 - pEngOpenSingleUse, 123
 - pEngOutputBuffer, 123
 - pEngPutVariable, 123
 - pEngSetVisible, 123
 - pMatClose, 123
 - pMatGetVariable, 123
 - pMatOpen, 123
 - pMatPutVariable, 123
 - pMxCreateDoubleMatrix, 124
 - pMxDestroyArray, 124
 - pMxGetNumberOfElements, 124
 - pMxGetPr, 124
 - pMxSetPr, 124
 - Path, 122
 - path, 122
 - SetPath, 122
- CMatlabEngineEvaluator
 - GoSUM::CMatlabEngineEvaluator, 125
- CMatlabShellEvaluator
 - GoSUM::CMatlabShellEvaluator, 127
- CMediumPeriodRNG, 128
 - ~CMediumPeriodRNG, 129
- CMediumPeriodRNG, 129
- idum, 129
- idum2, 129
- iv2, 129
- iy2, 129
- rnd, 129
- rndi, 129
- setSeed, 129
- CModelConstraints
 - GoSUM::CModelConstraints, 130
- CModelEvaluator
 - GoSUM::CModelEvaluator, 133
- CModelHypercube
 - GoSUM::CModelHypercube, 135
- CModelOptimizationVariable
 - GoSUM::CModelOptimizationVariable, 137
- CModelVariable
 - GoSUM::CModelVariable, 140
- CModelVariables
 - GoSUM::CModelVariables, 146
- CMonteCarloHC
 - GoSUM::CMonteCarloHC, 151
- CNLCRNG, 152
 - ~CNLCRNG, 152
 - CNLCRNG, 152
 - CNLCRNG, 152
 - inext, 153
 - inextp, 153
 - ma, 153
 - rnd, 152
 - rndi, 153
 - setSeed, 153
- CNomadRNG, 153
 - ~CNomadRNG, 154
 - CNomadRNG, 154
 - CNomadRNG, 154
 - rnd, 154
 - rndi, 154
 - setSeed, 154
 - x, 154
 - y, 154
 - z, 154
- CNuSvcSAM
 - GoSUM::CNuSvcSAM, 158
- CNuSvrSAM
 - GoSUM::CNuSvrSAM, 159
- CNumericalSample, 155
 - ~CNumericalSample, 156
 - boost::serialization::access, 157
 - CNumericalSample, 156
 - CNumericalSample, 156
 - maxValue, 156
 - minValue, 156
 - readSampleValue, 156
 - serialize, 156
 - setSampleSize, 156
 - values, 157

- variance, 156
- writeSampleValue, 156
- COneClassSAM
 - GoSUM::COneClassSAM, 160
- COptimizationProblem, 161
 - ~COptimizationProblem, 163
 - addVariable, 163
 - boost::serialization::access, 166
 - COptimizationProblem, 163
 - clear, 163
 - clearHistory, 163
 - closeOptimization, 163
 - constraint, 163
 - constraintsSize, 164
 - COptimizationProblem, 163
 - dimension, 164
 - emptyHistory, 164
 - eraseVariable, 164
 - evaluate, 164
 - exportObjectiveHistory, 164
 - history, 166
 - historySize, 164
 - initialValue, 164
 - isFeasible, 164
 - isMaximization, 164
 - isMinimization, 165
 - lowerBound, 165
 - minimize, 166
 - objective, 165
 - objectiveHistory, 165
 - openOptimization, 165
 - ovs, 167
 - serialize, 165
 - setInitialValue, 165
 - setLowerBound, 165
 - setMaximization, 166
 - setMinimization, 166
 - setUpperBound, 166
 - upperBound, 166
 - variableHistory, 166
 - writeHistory, 166
- COptimizationVariable, 167
 - ~COptimizationVariable, 168
 - boost::serialization::access, 169
 - COptimizationVariable, 168
 - COptimizationVariable, 168
 - initialValue, 168
 - lowerBound, 168
 - serialize, 168
 - set, 168
 - setInitialValue, 168
 - setLowerBound, 168
 - setUpperBound, 168
 - upperBound, 168
 - x0, 169
 - xL, 169
 - xU, 169
- COriginalOptimizationProblem
 - GoSUM::COriginalOptimizationProblem, 170
- COutputStates
 - GoSUM::COutputStates, 172
- CParserOptimizationProblem
 - GoSUM::CParserOptimizationProblem, 175
- CPlot2D, 178
 - ~CPlot2D, 179
 - CPlot2D, 179
 - canvas, 179
 - clear, 179
 - CPlot2D, 179
 - getPlot, 179
 - newColorBarPlot, 179
 - newColorMap, 179
 - newPlot, 179
 - plot, 180
 - plotWithHistogram, 179
 - plotWithPen, 179
 - plotWithSpectrogram, 179
 - plotWithSymbol, 179, 180
 - replot, 180
 - setTitles, 180
- CRandomGenerator, 180
 - ds, 183
 - lp, 183
 - mad, 183
 - mp, 183
 - nlc, 183
 - p, 183
 - Rnd, 182
 - RndInt, 182
 - rngtype, 181
 - Set, 182
 - SetDynamicSystem, 182
 - SetLargePeriod, 182
 - SetMediumPeriod, 182
 - SetNomads, 182
 - SetNonlinearCongruential, 182
 - SetSeed, 182
 - SetSmallPeriod, 182
 - sp, 183
 - Type, 182
 - type, 183
- CRandomVariable, 183
 - ~CRandomVariable, 186
 - boost::serialization::access, 188
 - CRandomVariable, 186
 - CRandomVariable, 186
 - distributionName, 186
 - DistributionParametersSize, 186
 - distributionType, 186
 - distributiontype, 185
 - doQuantile, 186
 - expandedSize, 186
 - expectedValue, 186
 - IsContinuous, 186
 - IsDiscrete, 186
 - isDistributionDefined, 186

- IsDistributionType, [187](#)
- IsEmpirical, [187](#)
- IsTheoretical, [187](#)
- maxValue, [187](#)
- minValue, [187](#)
- quantile, [187](#)
- serialize, [187](#)
- Type, [187](#)
- variance, [187](#)
- CReduction
 - GoSUM::CReduction, [190](#)
- CSample, [194](#)
 - ~CSample, [195](#)
 - boost::serialization::access, [196](#)
 - CSample, [195](#)
 - clear, [195](#)
 - computeStatistics, [195](#)
 - CSample, [195](#)
 - eraseSelected, [195](#)
 - maxdiscretesize, [196](#)
 - readSampleValue, [195](#)
 - sampleSize, [195](#)
 - select, [195](#)
 - serialize, [195](#)
 - setSampleSize, [196](#)
 - setSampleValue, [196](#)
 - variance, [196](#)
 - writeSampleValue, [196](#)
- CScript
 - GoSUM::CScript, [198](#)
- CSensitivityAnalysis
 - GoSUM::CSensitivityAnalysis, [210](#)
- CSimpleOptimizationProblem
 - GoSUM::CSimpleOptimizationProblem, [216](#)
- CSingleAM
 - GoSUM::CSingleAM, [221](#)
- CSmallPeriodRNG, [224](#)
 - ~CSmallPeriodRNG, [225](#)
 - CSmallPeriodRNG, [225](#)
 - CSmallPeriodRNG, [225](#)
 - idum1, [225](#)
 - iv1, [225](#)
 - iy1, [225](#)
 - rnd, [225](#)
 - rndi, [225](#)
 - setSeed, [225](#)
- CTContinuousMV
 - GoSUM, [237](#)
- CTContinuousRV
 - RandomVariable.h, [268](#)
- CTContinuousSample
 - Sample.h, [270](#)
- CTDiscreteMV
 - GoSUM, [237](#)
- CTDiscreteRV
 - RandomVariable.h, [268](#)
- CTDiscreteSample
 - Sample.h, [270](#)
- CUniformCRV, [226](#)
 - ~CUniformCRV, [227](#)
 - a, [229](#)
 - b, [229](#)
 - boost::serialization::access, [229](#)
 - CUniformCRV, [227](#)
 - cumulative, [227](#)
 - CUniformCRV, [227](#)
 - d, [229](#)
 - distributionName, [227](#)
 - distributionType, [227](#)
 - doQuantile, [227](#)
 - expectedValue, [228](#)
 - exportDomain, [228](#)
 - lowerBound, [228](#)
 - maxValue, [228](#)
 - minValue, [228](#)
 - probability, [228](#)
 - serialize, [228](#)
 - setDistribution, [228](#)
 - setLowerBound, [228](#)
 - setUpperBound, [228](#)
 - upperBound, [229](#)
 - variance, [229](#)
- CUniformDRV, [229](#)
 - ~CUniformDRV, [231](#)
 - a, [233](#)
 - b, [233](#)
 - boost::serialization::access, [232](#)
 - CUniformDRV, [231](#)
 - cumulative, [231](#)
 - CUniformDRV, [231](#)
 - distributionName, [231](#)
 - distributionType, [231](#)
 - doQuantile, [231](#)
 - expectedValue, [231](#)
 - exportDomain, [231](#)
 - lowerBound, [231](#)
 - maxValue, [232](#)
 - minValue, [232](#)
 - n, [233](#)
 - probability, [232](#)
 - serialize, [232](#)
 - setDistribution, [232](#)
 - setLowerBound, [232](#)
 - setUpperBound, [232](#)
 - upperBound, [232](#)
 - variance, [232](#)
- canvas
 - CPlot2D, [179](#)
- cast2ContinuousMV
 - GoSUM::CModelVariable, [140](#)
- cast2DiscreteMV
 - GoSUM::CModelVariable, [140](#)
- castExportSample
 - GoSUM::CModelVariable, [140](#)
- categorical
 - CRandomVariable, [185](#)

- categories
 - CCategoricalSample, 25
- cdf
 - CCategoricalDRV, 23
 - CEmpiricalCRV, 78
 - CEmpiricalDRV, 82
- centralize
 - GoSUM::CCVoronoiHC, 65
- checkHCPoints
 - GoSUM::CModelHypercube, 135
- chiSquareTest
 - Probability.cpp, 263
 - Probability.h, 263
- clear
 - COptimizationProblem, 163
 - CPlot2D, 179
 - CSample, 195
 - GoSUM::CA analyticalModel, 15
 - GoSUM::CConstraints, 34
 - GoSUM::CContainer, 44
 - GoSUM::CEvaluator, 85
 - GoSUM::CModelConstraints, 130
 - GoSUM::CModelVariables, 146
 - GoSUM::CReduction, 190
 - GoSUM::CSensitivityAnalysis, 211
 - GoSUM::CSimpleOptimizationProblem, 216
 - TSample, 248
- clearConstraintExpressions
 - GoSUM::CParserOptimizationProblem, 175
- clearForReducing
 - GoSUM::CContainer, 44
- clearHistory
 - COptimizationProblem, 163
 - GoSUM::CSimpleOptimizationProblem, 216
- clearLearningResults
 - GoSUM::CContainer, 45
- clearOptimizationResults
 - GoSUM::CContainer, 45
- clearResults
 - GoSUM::CContainer, 45
- clearSample
 - GoSUM::CModelVariable, 140
 - GoSUM::TModelVariable, 240
- clearSamples
 - GoSUM::CModelVariables, 146
- clearSamplingResults
 - GoSUM::CContainer, 45
- clearSensitivityResults
 - GoSUM::CContainer, 45
- clone
 - GoSUM::CModelVariable, 140
- cloneVariable
 - GoSUM::CContainer, 45
- closeEvaluation
 - GoSUM::CExeEvaluator, 90
 - GoSUM::CMatlabEngineEvaluator, 125
 - GoSUM::CModelEvaluator, 133
 - GoSUM::COutputStates, 172
- closeOptimization
 - COptimizationProblem, 163
 - GoSUM::COriginalOptimizationProblem, 170
 - GoSUM::CParserOptimizationProblem, 175
 - GoSUM::CSimpleOptimizationProblem, 216
- CloseSVM
 - GoSUM::CSingleAM, 221
- cmds
 - GoSUM::CScript, 207
- coef0
 - GoSUM::CSingleAM, 223
- coeff
 - GoSUM::CSingleAM, 223
- compareFirst
 - Reduction.cpp, 268
- compcoo
 - GoSUM::CLCVoronoiHC, 113
- computeDerivativeSensitivities
 - GoSUM::CSensitivityAnalysis, 211
- computeMeanAndStandardDeviation
 - CContinuousSample, 62
- computeNormalizedHistogram
 - CContinuousSample, 62
 - CDiscreteSample, 70
 - TSample, 248
- computeNormalizedSubHistogram
 - CContinuousSample, 62
 - CDiscreteSample, 70
 - TSample, 249
- computeSensitivities
 - GoSUM::CContainer, 45
 - GoSUM::CScript, 200
 - GoSUM::CSensitivityAnalysis, 211
- computeStatistics
 - CContinuousSample, 62
 - CDiscreteSample, 70
 - CSample, 195
 - GoSUM::CModelVariable, 140
 - GoSUM::CModelVariables, 146
 - GoSUM::TModelVariable, 240
- computeVarianceSensitivities
 - GoSUM::CSensitivityAnalysis, 211
- computeVarianceSensitivities1
 - GoSUM::CSensitivityAnalysis, 211
- computingProgressed
 - GoSUM::CSensitivityAnalysis, 213
- constantValue
 - CConstantCRV, 27
 - CConstantDRV, 30
- constantcontinuous
 - CRandomVariable, 185
- constantdiscrete
 - CRandomVariable, 185
- constraint
 - COptimizationProblem, 163
 - GoSUM::CParserOptimizationProblem, 175
 - GoSUM::CSingleAM, 221
- constraintExpression

- GoSUM::CParserOptimizationProblem, 175
- constraints
 - GoSUM::CInputParameters, 109
- constraintsSatisfied
 - GoSUM::CConstraints, 34
- constraintsSize
 - COptimizationProblem, 164
 - GoSUM::CParserOptimizationProblem, 176
 - GoSUM::CSingleAM, 221
- containsDeclaredEmpiricalViarables
 - GoSUM::CContainer, 45
- containsEmpiricalViarables
 - GoSUM::CContainer, 45
- containsNamedDeclaredEmpiricalViarables
 - GoSUM::CContainer, 45
- containsNamedEmpiricalViarables
 - GoSUM::CContainer, 45
- containsNamedTheoreticalViarables
 - GoSUM::CContainer, 46
- containsPredictionSamples
 - GoSUM::CContainer, 46
- containsTheoreticalViarables
 - GoSUM::CContainer, 46
- coo
 - GoSUM::CLCVoronoiHC, 114
- couplingError
 - GoSUM::CSensitivityAnalysis, 211
- cummulativeHistogram
 - TSample, 249
- cumulative
 - CCategoricalDRV, 21
 - CConstantCRV, 27
 - CConstantDRV, 30
 - CEmpiricalCRV, 76
 - CEmpiricalDRV, 80
 - CExponentialCRV, 95
 - CGaussianCRV, 102
 - CUniformCRV, 227
 - CUniformDRV, 231
 - TRandomVariable, 245, 246
- cumulativeSum
 - VariousMath.cpp, 274
 - VariousMath.h, 275
- cutip
 - GoSUM::CReduction, 193
- cutoffSize
 - GoSUM::CReduction, 190
- CutoffType
 - GoSUM::CReduction, 191
- cutoffValue
 - GoSUM::CReduction, 191
- cutoffsize
 - GoSUM::CReduction, 190
- cutofftype
 - GoSUM::CReduction, 190
- cutoffvalue
 - GoSUM::CReduction, 190
- cutos
 - GoSUM::CReduction, 193
- cuttype
 - GoSUM::CReduction, 193
- cutval
 - GoSUM::CReduction, 193
- cvoronoi
 - GoSUM::CHypercube, 105
- CvtIterationSize
 - GoSUM::CHypercube, 106
- CvtNewCenterCoefficient
 - GoSUM::CHypercube, 106
- CvtOldCenterCoefficient
 - GoSUM::CHypercube, 106
- CvtPointsSize
 - GoSUM::CHypercube, 106
- Cx_ZetaGammax, 233
 - a2, 233
 - Cx_ZetaGammax, 233
 - Cx_ZetaGammax, 233
 - evaluate, 233
 - i2, 234
 - N, 234
- d
 - CUniformCRV, 229
- dDenormalize
 - GoSUM::CModelVariables, 147
- dNormalize
 - GoSUM::CModelVariables, 147
- DSn, 234
 - ~DSn, 234
 - DSn, 234
 - doub, 235
 - DSn, 234
 - i, 235
 - j, 235
 - k, 235
 - Long, 234
 - n, 235
 - w, 235
 - x, 235
- dataanalysis
 - GoSUM::CContainer, 43
- degree
 - GoSUM::CSingleAM, 223
- denormalize
 - GoSUM::CModelVariables, 147
- derivative
 - GoSUM::CReduction, 190
- derivativeSensitivity
 - GoSUM::CSensitivityAnalysis, 211
- df
 - GoSUM::CSingleAM, 221
- dimension
 - COptimizationProblem, 164
- discreteCosineTransform
 - CFFTW, 97
- distributionName
 - CCategoricalDRV, 21

- CConstantCRV, [27](#)
- CConstantDRV, [30](#)
- CEmpiricalCRV, [76](#)
- CEmpiricalDRV, [80](#)
- CExponentialCRV, [95](#)
- CGaussianCRV, [102](#)
- CRandomVariable, [186](#)
- CUniformCRV, [227](#)
- CUniformDRV, [231](#)
- GoSUM::CModelVariable, [141](#)
- GoSUM::TModelVariable, [240](#)
- DistributionParametersSize
 - CRandomVariable, [186](#)
- distributionType
 - CCategoricalDRV, [21](#)
 - CConstantCRV, [27](#)
 - CConstantDRV, [31](#)
 - CEmpiricalCRV, [76](#)
 - CEmpiricalDRV, [80](#)
 - CExponentialCRV, [95](#)
 - CGaussianCRV, [102](#)
 - CRandomVariable, [186](#)
 - CUniformCRV, [227](#)
 - CUniformDRV, [231](#)
 - GoSUM::CModelVariable, [141](#)
 - GoSUM::TModelVariable, [240](#)
- distributiontype
 - CRandomVariable, [185](#)
- doGenerate
 - GoSUM::CCVoronoiHC, [65](#)
 - GoSUM::CDSampleHC, [72](#)
 - GoSUM::CLCVoronoiHC, [113](#)
 - GoSUM::CModelHypercube, [135](#)
 - GoSUM::CMonteCarloHC, [151](#)
- doQuantile
 - CCategoricalDRV, [21](#)
 - CConstantCRV, [27](#)
 - CConstantDRV, [31](#)
 - CEmpiricalCRV, [77](#)
 - CEmpiricalDRV, [80](#)
 - CExponentialCRV, [95](#)
 - CGaussianCRV, [102](#)
 - CRandomVariable, [186](#)
 - CUniformCRV, [227](#)
 - CUniformDRV, [231](#)
- doub
 - DSn, [235](#)
 - Ran, [237](#)
- ds
 - CRandomGenerator, [183](#)
- Ds.cpp
 - prepareDSample, [255](#)
- Ds.h
 - prepareDSample, [255](#)
- dsample
 - GoSUM::CHypercube, [105](#)
- Dynamic
 - TRandomVariable, [246](#)
- dynamicsystem
 - CRandomGenerator, [181](#)
- EPS
 - RandomGenerators.cpp, [265](#)
- empiricalcontinuous
 - CRandomVariable, [185](#)
- empiricaldiscrete
 - CRandomVariable, [185](#)
- empty
 - GoSUM::CA analyticalModel, [15](#)
 - GoSUM::CModelVariables, [147](#)
 - GoSUM::CSensitivityAnalysis, [211](#)
- emptyAbsoluteAverageDerivativeSensitivity
 - GoSUM::CSensitivityAnalysis, [211](#)
- emptyAverageDerivativeSensitivity
 - GoSUM::CSensitivityAnalysis, [211](#)
- emptyDerivativeSensitivity
 - GoSUM::CSensitivityAnalysis, [211](#)
- emptyFirstOrderANOVA
 - GoSUM::CSensitivityAnalysis, [212](#)
- emptyHistory
 - COptimizationProblem, [164](#)
- emptyLearningResults
 - GoSUM::CContainer, [46](#)
- emptyOptimizationResults
 - GoSUM::CContainer, [46](#)
- emptyResults
 - GoSUM::CContainer, [46](#)
- emptySamples
 - GoSUM::CModelVariables, [147](#)
- emptySamplingResults
 - GoSUM::CContainer, [46](#)
- emptySelectedSamples
 - GoSUM::CContainer, [46](#)
- emptySensitivityResults
 - GoSUM::CContainer, [46](#)
- emptyVarianceSensitivity
 - GoSUM::CSensitivityAnalysis, [212](#)
- engClose
 - CMATLAB, [120](#)
- engEvalString
 - CMATLAB, [120](#)
- engGetVariable
 - CMATLAB, [121](#)
- engGetVisible
 - CMATLAB, [121](#)
- engOpen
 - CMATLAB, [121](#)
- engOpenSingleUse
 - CMATLAB, [121](#)
- engOutputBuffer
 - CMATLAB, [121](#)
- engPutVariable
 - CMATLAB, [121](#)
- engSetVisible
 - CMATLAB, [121](#)
- engftype1
 - MatlabLibrary.h, [258](#)

- engftype2
 - MatlabLibrary.h, 258
- engftype3
 - MatlabLibrary.h, 258
- engftype4
 - MatlabLibrary.h, 258
- engftype5
 - MatlabLibrary.h, 258
- engftype6
 - MatlabLibrary.h, 258
- engftype7
 - MatlabLibrary.h, 258
- engftype8
 - MatlabLibrary.h, 258
- engftype9
 - MatlabLibrary.h, 258
- ep
 - GoSUM::CMatlabEngineEvaluator, 126
- eps1
 - GoSUM::CSensitivityAnalysis, 213
- eps2
 - GoSUM::CSensitivityAnalysis, 213
- eps3
 - GoSUM::CSensitivityAnalysis, 213
- epsilonsvr
 - GoSUM::CA analyticalModel, 14
- erase
 - GoSUM::CModelVariables, 147
- eraseConstraintExpression
 - GoSUM::CParserOptimizationProblem, 176
- eraseExpression
 - GoSUM::CConstraints, 34
- eraseNonSelectedVariables
 - GoSUM::CReduction, 191
- eraseSelected
 - CSample, 195
 - TSample, 249
- eraseSelectedSamples
 - GoSUM::CContainer, 46
 - GoSUM::CModelVariables, 147
- eraseVariable
 - COptimizationProblem, 164
 - GoSUM::CContainer, 46
- etype
 - GoSUM::CA analyticalModel, 16
 - GoSUM::CHypercube, 107
- eval_x
 - CMADSEvaluator, 118
- evaluate
 - COptimizationProblem, 164
 - Cx_ZetaGammamax, 233
 - GoSUM::CConstraints, 34
 - GoSUM::COutputStates, 172
 - GoSUM::CSimpleOptimizationProblem, 217
- evaluateMaximalSensitivities
 - GoSUM::CReduction, 191
- evaluateOutputs
 - GoSUM::CContainer, 46
- GoSUM::CScript, 200
- evaluateOutputsSize
 - GoSUM::CExeEvaluator, 90
 - GoSUM::CMatlabEngineEvaluator, 125
 - GoSUM::CModelEvaluator, 133
- evaluatingFinished
 - GoSUM::CContainer, 56
- evaluatingProgressed
 - GoSUM::CEvaluator, 86
- evaluationSize
 - CMADS, 115
- evaluator
 - GoSUM::CContainer, 56
- evaluatorType
 - GoSUM::CEvaluator, 85
- exe
 - GoSUM::CEvaluator, 86
- exeascii
 - GoSUM::CEvaluator, 85
- exemat
 - GoSUM::CEvaluator, 85
- expandNTuple
 - GoSUM::CModelVariables, 148
- expandedNTuple
 - GoSUM::CModelVariables, 147
- expandedSize
 - CCategoricalDRV, 22
 - CEmpiricalDRV, 80
 - CRandomVariable, 186
 - GoSUM::CModelVariable, 141
 - GoSUM::CModelVariables, 148
 - GoSUM::TModelVariable, 240
- expectedValue
 - CCategoricalDRV, 22
 - CConstantCRV, 27
 - CConstantDRV, 31
 - CEmpiricalCRV, 77
 - CEmpiricalDRV, 80
 - CExponentialCRV, 95
 - CGaussianCRV, 102
 - CRandomVariable, 186
 - CUniformCRV, 228
 - CUniformDRV, 231
 - GoSUM::CModelVariable, 141
 - GoSUM::TModelVariable, 240
- exponential
 - CRandomVariable, 185
- exportANOVA1
 - GoSUM::CScript, 200
- exportAbsoluteAverageDerivative
 - GoSUM::CContainer, 47
 - GoSUM::CScript, 200
- exportAverageDerivative
 - GoSUM::CContainer, 47
 - GoSUM::CScript, 200
- exportCumulative
 - GoSUM::TModelVariable, 240
- exportDerivativeSensitivity

- GoSUM::CContainer, 47
- GoSUM::CScript, 200
- exportDomain
 - CCategoricalDRV, 22
 - CConstantCRV, 28
 - CConstantDRV, 31
 - CEmpiricalCRV, 77
 - CEmpiricalDRV, 81
 - CExponentialCRV, 95
 - CGaussianCRV, 102
 - CUniformCRV, 228
 - CUniformDRV, 231
 - GoSUM::TModelVariable, 241
 - TRandomVariable, 245
- exportExpectedCurve
 - GoSUM::CA analyticalModel, 15
- exportFirstOrderANOVA
 - GoSUM::CContainer, 47
- exportHistogram
 - CContinuousSample, 62
 - CDiscreteSample, 70
 - GoSUM::TModelVariable, 241
 - TSample, 249
- exportInputSamples
 - GoSUM::CContainer, 47
 - GoSUM::CScript, 200
- exportIntersectionCurve
 - GoSUM::CA analyticalModel, 15
- exportObjectiveHistory
 - COptimizationProblem, 164
- exportOptimizationHistory
 - GoSUM::CContainer, 47
 - GoSUM::CScript, 201
- exportOptimizationMethod
 - GoSUM::CContainer, 47
 - GoSUM::CScript, 201
- exportOutputSamples
 - GoSUM::CContainer, 47
 - GoSUM::CScript, 201
- exportPredictionOutputSamples
 - GoSUM::CContainer, 47
 - GoSUM::CScript, 201
- exportProbability
 - GoSUM::TModelVariable, 241
- exportQuantile
 - GoSUM::TModelVariable, 241
- exportSample
 - GoSUM::TModelVariable, 241
 - TSample, 249
- exportSamples
 - GoSUM::CContainer, 47
- exportSubHistogram
 - CContinuousSample, 62
 - CDiscreteSample, 70
 - GoSUM::TModelVariable, 241
 - TSample, 249
- exportVarianceSensitivity
 - GoSUM::CContainer, 47
- GoSUM::CScript, 201
- expression
 - GoSUM::CConstraints, 34
- ext
 - GoSUM::CEvaluator, 87
- ExternalEvaluator
 - GoSUM::CEvaluator, 85
- ExternalEvaluatorFolder
 - GoSUM::CEvaluator, 85
- f
 - GoSUM::CConstraints, 35
 - GoSUM::CParserOptimizationProblem, 177
 - GoSUM::CSingleAM, 221
- FAC
 - RandomGenerators.cpp, 265
- fexpr
 - GoSUM::CParserOptimizationProblem, 177
- find
 - GoSUM::CModelVariables, 148
- findConnectedTo
 - GoSUM::CSimpleOptimizationProblem, 217
- findInExpressions
 - GoSUM::CConstraints, 34
- findNearest
 - VariousMath.cpp, 274
 - VariousMath.h, 275
- findRoot
 - VariousMath.h, 275
- firstOrderANOVA
 - GoSUM::CSensitivityAnalysis, 212
- Fitness
 - CGAModelOptimization, 99
- g
 - GoSUM::CConstraints, 35
 - GoSUM::CParserOptimizationProblem, 177
- GAGenome2ArrayXd
 - CGAModelOptimization, 99
- GAMO
 - GoSUM::CContainer, 56
- ga
 - GoSUM::CContainer, 43
- gaOptimizer
 - GoSUM::CContainer, 47
- gamma
 - GoSUM::CSingleAM, 223
- gaussian
 - CRandomVariable, 185
- generate
 - GoSUM::CModelHypercube, 135
- generateHCPoint
 - GoSUM::CModelHypercube, 135
- generateHCPnts
 - GoSUM::CModelHypercube, 136
- generateModelPoints
 - GoSUM::CModelHypercube, 136
- generateSampleValue
 - GoSUM::CModelVariable, 141

- GoSUM::TModelVariable, 241
- generateSamples
 - GoSUM::CInputParameters, 109
- generatingProgressed
 - GoSUM::CHypercube, 107
- getPlot
 - CPlot2D, 179
- gexpr
 - GoSUM::CConstraints, 35
 - GoSUM::CParserOptimizationProblem, 177
- GoSUM::CA analyticalModel
 - epsilonsvr, 14
 - nusvr, 14
- GoSUM::CContainer
 - dataanalysis, 43
 - ga, 43
 - learneddataoptimization, 43
 - learnedmodeloptimization, 43
 - mads, 43
 - modelanalysis, 43
 - modeloptimization, 43
 - samplegeneration, 43
 - simpleoptimization, 43
- GoSUM::CEvaluator
 - exeascii, 85
 - exemat, 85
 - matlabengine, 85
 - matlabshell, 85
- GoSUM::CHypercube
 - cvoronoi, 105
 - dsample, 105
 - lcvoronoi, 105
 - montecarlo, 105
- GoSUM::CReduction
 - absoluteaveragederivative, 190
 - anova, 190
 - averagederivative, 190
 - cutoffsize, 190
 - cutoffvalue, 190
 - derivative, 190
 - variance, 190
- GoSUM, 11, 235
 - CTContinuousMV, 237
 - CTDiscreteMV, 237
 - NULL, 237
 - OriginalModel.cpp, 261
 - paction, 237
 - Reduction.cpp, 269
- GoSUM::CA analyticalModel, 13
 - ~CA analyticalModel, 15
 - boost::serialization::access, 16
 - CA analyticalModel, 15
 - clear, 15
 - empty, 15
 - etype, 16
 - exportExpectedCurve, 15
 - exportIntersectionCurve, 15
 - inputSamples, 15
 - learn, 15
 - outputSamples, 15
 - pIP, 16
 - pOS, 16
 - predict, 15
 - predictDerivative, 15
 - predictDerivatives, 15
 - predictValue, 15
 - predictValues, 16
 - progressSlot, 16
 - sam, 16
 - serialize, 16
 - setInputSamples, 16
 - setProgressSlot, 16
 - setSvmType, 16
 - svmType, 16
 - svmtyp, 14
 - X, 17
 - Y, 17
- GoSUM::CA analyticalOptimizationProblem, 17
 - ~CA analyticalOptimizationProblem, 18
 - boost::serialization::access, 18
 - CA analyticalOptimizationProblem, 18
 - inputPoint2ModelPoint, 18
 - pAM, 18
 - serialize, 18
- GoSUM::CCSvcSAM, 63
 - ~CCSvcSAM, 64
 - boost::serialization::access, 64
 - CCSvcSAM, 64
 - serialize, 64
- GoSUM::CCVoronoiHC, 65
 - ~CCVoronoiHC, 65
 - boost::serialization::access, 66
 - CCVoronoiHC, 65
 - centralize, 65
 - doGenerate, 65
 - serialize, 66
- GoSUM::CConstraints, 32
 - ~CConstraints, 34
 - addExpression, 34
 - boost::serialization::access, 35
 - CConstraints, 34
 - clear, 34
 - constraintsSatisfied, 34
 - eraseExpression, 34
 - evaluate, 34
 - expression, 34
 - f, 35
 - findInExpressions, 34
 - g, 35
 - gexpr, 35
 - names, 35
 - parseExpressions, 34
 - roundoffEquality, 34
 - serialize, 34
 - setExpression, 34
 - setVariableNames, 35

- size, 35
- validateExpressions, 35
- GoSUM::CContainer, 35
 - ~CContainer, 43
 - addDefaultVariable, 43
 - addInput, 43
 - addInputs, 44
 - addOptimizationConstraint, 44
 - addOutput, 44
 - addOutputs, 44
 - addVariable, 44
 - addVariables, 44
 - analytical, 56
 - analyticalModel, 44
 - boost::serialization::access, 56
 - CContainer, 43
 - clear, 44
 - clearForReducing, 44
 - clearLearningResults, 45
 - clearOptimizationResults, 45
 - clearResults, 45
 - clearSamplingResults, 45
 - clearSensitivityResults, 45
 - cloneVariable, 45
 - computeSensitivities, 45
 - containsDeclaredEmpiricalVariables, 45
 - containsEmpiricalVariables, 45
 - containsNamedDeclaredEmpiricalVariables, 45
 - containsNamedEmpiricalVariables, 45
 - containsNamedTheoreticalVariables, 46
 - containsPredictionSamples, 46
 - containsTheoreticalVariables, 46
 - emptyLearningResults, 46
 - emptyOptimizationResults, 46
 - emptyResults, 46
 - emptySamplingResults, 46
 - emptySelectedSamples, 46
 - emptySensitivityResults, 46
 - eraseSelectedSamples, 46
 - eraseVariable, 46
 - evaluateOutputs, 46
 - evaluatingFinished, 56
 - evaluator, 56
 - exportAbsoluteAverageDerivative, 47
 - exportAverageDerivative, 47
 - exportDerivativeSensitivity, 47
 - exportFirstOrderANOVA, 47
 - exportInputSamples, 47
 - exportOptimizationHistory, 47
 - exportOptimizationMethod, 47
 - exportOutputSamples, 47
 - exportPredictionOutputSamples, 47
 - exportSamples, 47
 - exportVarianceSensitivity, 47
 - GAMO, 56
 - gaOptimizer, 47
 - hycube, 56
 - hyperCube, 48
 - importDeclaredEmpiricalVariables, 48
 - importEmpiricalVariables, 48
 - importInitialValue, 48
 - importInputs, 48
 - importLowerBound, 48
 - importModelConstraints, 48
 - importNamedDeclaredEmpiricalVariables, 48
 - importNamedEmpiricalVariables, 48
 - importNamedTheoreticalVariables, 48
 - importOptimizationConstraints, 48
 - importOutputs, 49
 - importPredictionInputSamples, 49
 - importTheoreticalVariables, 49
 - importUpperBound, 49
 - importVariables, 49
 - inconsts, 56
 - inputParameter, 49
 - inputParameters, 49
 - inputs, 56
 - inputsSize, 49
 - isSampleSelected, 49
 - join, 49
 - learnEvaluationSize, 49
 - learnModel, 50
 - learningFinished, 56
 - learningStepsSize, 49
 - load, 50
 - loadTxt, 50
 - loadXml, 50
 - longProjectName, 50
 - MADS, 56
 - madsOptimizer, 50
 - maximize, 50
 - minimize, 50
 - modelConstraints, 50
 - modelEvaluator, 50
 - omType, 56
 - optimizationMethod, 50
 - OptimizationMethodType, 50
 - optimizationProblem, 51
 - optimizationStepsSize, 51
 - optimizationmethodtype, 43
 - optimize, 51
 - optimizingFinished, 56
 - outputState, 51
 - outputStates, 51
 - outputs, 57
 - outputsSize, 51
 - pOP, 57
 - predict, 51
 - predictMean, 51
 - prjName, 57
 - prjPath, 57
 - prjType, 57
 - projectName, 51
 - projectPath, 51
 - ProjectType, 51
 - projectType, 51

- projecttype, 43
- readyToComputeSensitivities, 52
- readyToEvaluateOutputs, 52
- readyToLearnModel, 52
- readyToOptimize, 52
- readyToReduceModel, 52
- readyToSampleInputs, 52
- readyToSetOptimization, 52
- reduce, 52
- reduced, 57
- reducer, 52
- resampleInputs, 52
- resamplingFinished, 57
- resetOptimizationVariable, 52
- save, 52
- saveTxt, 53
- saveXml, 53
- selectSamples, 53
- selectedSamples, 53
- selectedsamples, 57
- sensitivity, 57
- sensitivityAnalysis, 53
- sensitivityComputingFinished, 57
- separateBySelection, 53
- serialize, 53
- setInitialValue, 53
- setLowerBound, 53
- setMadsInitMeshSize, 53
- setMadsLHSearch, 53
- setMadsMaxEvaluation, 53
- setMadsMinPollSize, 53
- setMatLabPath, 54
- setModelEvaluator, 54
- setObjective, 54
- setOptimizationMethod, 54
- setProjectName, 54
- setProjectPath, 54
- setProjectType, 54
- setRNG, 55
- setReductionCutoffSize, 54
- setReductionCutoffValue, 54
- setReductionOutputs, 54
- setReductionType, 54
- setResampleSize, 54
- setResampleType, 55
- setSensitivityOptions, 55
- setThreadSize, 55
- setUpperBound, 55
- setVoronoiOptions, 55
- startEvaluating, 55
- startLearning, 55
- startOptimizing, 55
- startResampling, 55
- startSensitivityComputing, 55
- thrd, 57
- GoSUM::CContinuousMV, 58
 - boost::serialization::access, 59
 - CContinuousMV, 58, 59
 - hasDiscreteData, 59
 - isContinuous, 59
 - isDiscrete, 59
 - serialize, 59
 - setEmpiricalDistribution, 59
 - setTheoreticalDistribution, 59
- GoSUM::CDSampleHC, 71
 - ~CDSampleHC, 72
 - boost::serialization::access, 72
 - CDSampleHC, 72
 - doGenerate, 72
 - serialize, 72
- GoSUM::CDiscreteMV, 66
 - boost::serialization::access, 68
 - CDiscreteMV, 67
 - isContinuous, 67
 - isDiscrete, 67
 - serialize, 67
 - setEmpiricalDistribution, 67
 - setTheoreticalDistribution, 67
- GoSUM::CEpsSvrSAM, 82
 - ~CEpsSvrSAM, 83
 - boost::serialization::access, 83
 - CEpsSvrSAM, 83
 - openOptimization, 83
 - optimizationPoint2SVMParam, 83
 - serialize, 83
- GoSUM::CEvaluator, 83
 - ~CEvaluator, 85
 - boost::serialization::access, 86
 - CEvaluator, 85
 - clear, 85
 - evaluatingProgressed, 86
 - evaluatorType, 85
 - exe, 86
 - ext, 87
 - ExternalEvaluator, 85
 - ExternalEvaluatorFolder, 85
 - in, 87
 - New, 86
 - out, 87
 - path, 87
 - serialize, 86
 - SetExternalEvaluator, 86
 - SetThreadSize, 86
 - SetType, 86
 - ThreadSize, 86
 - trdN, 87
 - Type, 86
 - type, 87
- GoSUM::CExeAsciiEvaluator, 87
 - ~CExeAsciiEvaluator, 88
 - boost::serialization::access, 88
 - CExeAsciiEvaluator, 88
 - readFrom, 88
 - readSizeFrom, 88
 - serialize, 88
 - writeTo, 88

- GoSUM::CExeEvaluator, 89
 - ~CExeEvaluator, 90
 - boost::serialization::access, 91
 - CExeEvaluator, 90
 - closeEvaluation, 90
 - evaluateOutputsSize, 90
 - openEvaluation, 90
 - operator(), 90
 - pP, 91
 - readFrom, 90
 - readSizeFrom, 90
 - serialize, 91
 - systemCommand, 91
 - writeTo, 91
- GoSUM::CExeMatEvaluator, 91
 - ~CExeMatEvaluator, 92
 - boost::serialization::access, 93
 - CExeMatEvaluator, 92
 - matlab, 93
 - readFrom, 93
 - readSizeFrom, 93
 - serialize, 93
 - writeTo, 93
- GoSUM::CHypercube, 104
 - ~CHypercube, 106
 - alpha2, 107
 - beta2, 107
 - boost::serialization::access, 107
 - CHypercube, 106
 - CvtIterationSize, 106
 - CvtNewCenterCoefficient, 106
 - CvtOldCenterCoefficient, 106
 - CvtPointsSize, 106
 - etype, 107
 - generatingProgressed, 107
 - hctype, 105
 - maxiter, 108
 - New, 106
 - ProgressSize, 106
 - q, 108
 - serialize, 106
 - SetCvtIterationSize, 106
 - SetCvtNewCenterCoefficient, 106
 - SetCvtOldCenterCoefficient, 106
 - SetCvtPointsSize, 106
 - SetType, 107
 - SetVoronoiOptions, 107
 - Type, 107
 - VoronoiOptions, 107
- GoSUM::CInputParameters, 108
 - ~CInputParameters, 109
 - boost::serialization::access, 110
 - CInputParameters, 109
 - constraints, 109
 - generateSamples, 109
 - pIC, 110
 - progressSlot, 110
 - RSsize, 110
 - resampleSize, 109
 - resampleStepsSize, 109
 - serialize, 110
 - setProgressSlot, 110
 - setResampleSize, 110
- GoSUM::CLCVoronoiHC, 112
 - ~CLCVoronoiHC, 113
 - boost::serialization::access, 114
 - CLCVoronoiHC, 113
 - compcoo, 113
 - coo, 114
 - doGenerate, 113
 - latinize, 113
 - serialize, 113
- GoSUM::CMatlabEngineEvaluator, 124
 - ~CMatlabEngineEvaluator, 125
 - boost::serialization::access, 126
 - CMatlabEngineEvaluator, 125
 - closeEvaluation, 125
 - ep, 126
 - evaluateOutputsSize, 125
 - matlab, 126
 - openEvaluation, 125
 - operator(), 126
 - serialize, 126
- GoSUM::CMatlabShellEvaluator, 126
 - ~CMatlabShellEvaluator, 127
 - boost::serialization::access, 128
 - CMatlabShellEvaluator, 127
 - serialize, 127
 - systemCommand, 127
- GoSUM::CModelConstraints, 130
 - ~CModelConstraints, 130
 - boost::serialization::access, 131
 - CModelConstraints, 130
 - clear, 130
 - isConstrained, 131
 - pIP, 131
 - serialize, 131
 - set, 131
 - setVariableNames, 131
 - status, 131
- GoSUM::CModelEvaluator, 131
 - ~CModelEvaluator, 133
 - boost::serialization::access, 133
 - CModelEvaluator, 133
 - closeEvaluation, 133
 - evaluateOutputsSize, 133
 - I, 134
 - openEvaluation, 133
 - operator(), 133
 - pIP, 134
 - pOS, 134
 - serialize, 133
 - trdI, 134
- GoSUM::CModelHypercube, 134
 - ~CModelHypercube, 135
 - boost::serialization::access, 136

- CModelHypercube, 135
- checkHCPoints, 135
- doGenerate, 135
- generate, 135
- generateHCPoint, 135
- generateHCPoints, 136
- generateModelPoints, 136
- hcPoints2ModelPoints, 136
- maxtries, 136
- pIC, 136
- pIP, 136
- serialize, 136
- GoSUM::CModelOptimizationVariable, 137
 - ~CModelOptimizationVariable, 137
 - boost::serialization::access, 138
 - CModelOptimizationVariable, 137
 - inputParameter, 137
 - pip, 138
 - reset, 137
 - serialize, 138
- GoSUM::CModelVariable, 138
 - ~CModelVariable, 140
 - boost::serialization::access, 143
 - CModelVariable, 140
 - cast2ContinuousMV, 140
 - cast2DiscreteMV, 140
 - castExportSample, 140
 - clearSample, 140
 - clone, 140
 - computeStatistics, 140
 - distributionName, 141
 - distributionType, 141
 - expandedSize, 141
 - expectedValue, 141
 - generateSampleValue, 141
 - isContinuous, 141
 - isDiscrete, 141
 - isDistributionDefined, 141
 - maxValue, 142
 - minValue, 142
 - name, 142
 - randomVariable, 142
 - readSampleValue, 142
 - sample, 142
 - sampleSize, 142
 - sampleValue, 142
 - serialize, 142
 - setDistribution, 142
 - setEmpiricalDistribution, 143
 - setName, 143
 - setSampleSize, 143
 - setSampleValue, 143
 - setTheoreticalDistribution, 143
 - uname, 143
 - writeSampleValue, 143
- GoSUM::CModelVariables, 144
 - ~CModelVariables, 146
 - add, 146
 - baseName, 146
 - basename, 150
 - boost::serialization::access, 150
 - CModelVariables, 146
 - clear, 146
 - clearSamples, 146
 - computeStatistics, 146
 - dDenormalize, 147
 - dNormalize, 147
 - denormalize, 147
 - empty, 147
 - emptySamples, 147
 - erase, 147
 - eraseSelectedSamples, 147
 - expandNTuple, 148
 - expandedNTuple, 147
 - expandedSize, 148
 - find, 148
 - hcPoint2ModelPoint, 148
 - insert, 148
 - isModelVariableName, 148
 - mvs, 150
 - nTuple, 148
 - nameToFind, 150
 - nextName, 148
 - normalize, 148
 - operator(), 148, 149
 - rename, 149
 - replace, 149
 - sampleSize, 149
 - samples, 149
 - scl, 150
 - serialize, 149
 - setBaseName, 149
 - setDistributions, 149
 - setNTuple, 149
 - setNormalization, 149
 - setSamples, 149
 - size, 150
 - trn, 150
- GoSUM::CMonteCarloHC, 150
 - ~CMonteCarloHC, 151
 - boost::serialization::access, 152
 - CMonteCarloHC, 151
 - doGenerate, 151
 - serialize, 151
- GoSUM::CNUSvcSAM, 157
 - ~CNUSvcSAM, 158
 - boost::serialization::access, 158
 - CNUSvcSAM, 158
 - serialize, 158
- GoSUM::CNUSvrSAM, 158
 - ~CNUSvrSAM, 159
 - boost::serialization::access, 159
 - CNUSvrSAM, 159
 - openOptimization, 159
 - optimizationPoint2SVMPParam, 159
 - serialize, 159

- GoSUM::COneClassSAM, 160
 - ~COneClassSAM, 160
 - boost::serialization::access, 161
 - COneClassSAM, 160
 - serialize, 160
- GoSUM::COriginalOptimizationProblem, 169
 - ~COriginalOptimizationProblem, 170
 - boost::serialization::access, 171
 - COriginalOptimizationProblem, 170
 - closeOptimization, 170
 - inputPoint2ModelPoint, 170
 - openOptimization, 170
 - pOS, 171
 - serialize, 171
 - setVariableNames, 171
- GoSUM::COutputStates, 171
 - ~COutputStates, 172
 - boost::serialization::access, 173
 - COutputStates, 172
 - closeEvaluation, 172
 - evaluate, 172
 - openEvaluation, 172
 - pE, 173
 - progressSlot, 173
 - serialize, 173
 - setEvaluatorCompatible, 173
 - setProgressSlot, 173
- GoSUM::CParserOptimizationProblem, 173
 - ~CParserOptimizationProblem, 175
 - addConstraintExpression, 175
 - boost::serialization::access, 177
 - CParserOptimizationProblem, 175
 - clearConstraintExpressions, 175
 - closeOptimization, 175
 - constraint, 175
 - constraintExpression, 175
 - constraintsSize, 176
 - eraseConstraintExpression, 176
 - f, 177
 - fexpr, 177
 - g, 177
 - gexpr, 177
 - names, 177
 - objective, 176
 - objectiveExpression, 176
 - openOptimization, 176
 - optimizingProgressed, 177
 - parseExpressions, 176
 - roundoffEquality, 176
 - serialize, 176
 - setConstraintExpression, 176
 - setObjectiveExpression, 176
 - setVariableNames, 177
 - validateExpressions, 177
- GoSUM::CReduction, 188
 - ~CReduction, 190
 - boost::serialization::access, 192
 - CReduction, 190
 - clear, 190
 - cutip, 193
 - cutoffSize, 190
 - CutoffType, 191
 - cutoffValue, 191
 - cutofftype, 190
 - cutos, 193
 - cuttype, 193
 - cutval, 193
 - eraseNonSelectedVariables, 191
 - evaluateMaximalSensitivities, 191
 - initialize, 191
 - isSelectedInputParameter, 191
 - isSelectedOutputState, 191
 - maximalSensitivity, 191
 - pAM, 193
 - pIP, 193
 - pOS, 193
 - pSA, 193
 - rankIP, 193
 - rankOS, 193
 - redtype, 193
 - ReductionType, 191
 - reductiontype, 190
 - selectInputParameters, 191
 - selectOutputState, 192
 - selectOutputs, 192
 - sensitivityIndex, 192
 - serialize, 192
 - setCutoffSize, 192
 - SetCutoffType, 192
 - setCutoffValue, 192
 - SetReductionType, 192
 - setReductionType, 192
- GoSUM::CScript, 196
 - ~CScript, 198
 - addInput, 198
 - addInputs, 199
 - addOptimizationConstraint, 199
 - addOutput, 199
 - addOutputs, 199
 - addVariable, 200
 - addVariables, 200
 - CScript, 198
 - cmds, 207
 - computeSensitivities, 200
 - evaluateOutputs, 200
 - exportANOVA1, 200
 - exportAbsoluteAverageDerivative, 200
 - exportAverageDerivative, 200
 - exportDerivativeSensitivity, 200
 - exportInputSamples, 200
 - exportOptimizationHistory, 201
 - exportOptimizationMethod, 201
 - exportOutputSamples, 201
 - exportPredictionOutputSamples, 201
 - exportVarianceSensitivity, 201
 - importInputs, 201

- importOutputs, 202
- importPredictionInputSamples, 202
- importVariables, 202
- learnModel, 202
- line2Path, 202
- load, 202
- loadTxt, 202
- loadXml, 202
- maximize, 202
- minimize, 203
- noeq1, 207
- pContainer, 208
- padd, 207
- padds, 207
- pimport, 208
- predict, 203
- read, 203
- reduce, 203
- resampleInputs, 203
- save, 203
- saveTxt, 203
- saveXml, 203
- setConstInitialValue, 203
- setConstLowerBound, 204
- setConstUpperBound, 204
- setCoreSize, 204
- setInitialValue, 204
- setLowerBound, 204
- setMadsInitMeshSize, 204
- setMadsLHSearch, 204
- setMadsMaxEvaluation, 205
- setMadsMinPollSize, 205
- setMatLabPath, 205
- setModelEvaluator, 205
- setObjective, 205
- setOptimizationMethod, 205
- setProjectName, 205
- setProjectPath, 205
- setProjectType, 206
- setRNG, 207
- setReductionCutoffSize, 206
- setReductionCutoffValue, 206
- setReductionOutputs, 206
- setReductionType, 206
- setResampleSize, 206
- setResampleType, 206
- setSensitivityOptions, 207
- setUpperBound, 207
- setVoronoiOptions, 207
- GoSUM::CSensitivityAnalysis, 208
 - ~CSensitivityAnalysis, 210
 - aader, 213
 - absoluteAverageDerivative, 211
 - ader, 213
 - averageDerivative, 211
 - boost::serialization::access, 213
 - CSensitivityAnalysis, 210
 - clear, 211
 - computeDerivativeSensitivities, 211
 - computeSensitivities, 211
 - computeVarianceSensitivities, 211
 - computeVarianceSensitivities1, 211
 - computingProgressed, 213
 - couplingError, 211
 - derivativeSensitivity, 211
 - empty, 211
 - emptyAbsoluteAverageDerivativeSensitivity, 211
 - emptyAverageDerivativeSensitivity, 211
 - emptyDerivativeSensitivity, 211
 - emptyFirstOrderANOVA, 212
 - emptyVarianceSensitivity, 212
 - eps1, 213
 - eps2, 213
 - eps3, 213
 - firstOrderANOVA, 212
 - pAM, 214
 - pIP, 214
 - pOS, 214
 - prepare, 212
 - progressStepsSize, 212
 - RSSize, 214
 - resampleSize, 212
 - sample1, 214
 - sample2, 214
 - secondOrderANOVA, 212
 - secondOrderANOVA Pairs, 212
 - sensitivityError, 212
 - separabilityError, 212
 - serialize, 212
 - setCouplingError, 212
 - setResampleSize, 212
 - setSensitivityError, 213
 - setSensitivityOptions, 213
 - setSeparabilityError, 213
 - si1, 214
 - si1der, 214
 - si2, 214
 - siT, 214
 - varianceSensitivity, 213
 - y, 214
 - ymu, 214
 - yvar, 214
- GoSUM::CSimpleOptimizationProblem, 215
 - ~CSimpleOptimizationProblem, 216
 - boost::serialization::access, 218
 - CSimpleOptimizationProblem, 216
 - clear, 216
 - clearHistory, 216
 - closeOptimization, 216
 - evaluate, 217
 - findConnectedTo, 217
 - inputPoint2ModelPoint, 217
 - isFeasible, 217
 - isModelVariable, 217
 - openOptimization, 217
 - ovind, 218

- pip, 218
 - pipToFind, 218
 - samples, 218
 - serialize, 217
 - setVariableNames, 217
 - uvind, 218
 - variable, 217, 218
 - variableName, 218
- GoSUM::CSingleAM, 218
 - ~CSingleAM, 221
 - boost::serialization::access, 222
 - C, 223
 - CSingleAM, 221
 - CloseSVM, 221
 - coef0, 223
 - coeff, 223
 - constraint, 221
 - constraintsSize, 221
 - degree, 223
 - df, 221
 - f, 221
 - gamma, 223
 - learn, 221
 - learnfrac, 223
 - LearningFraction, 221
 - learningProgressed, 223
 - MaximalLearningSize, 221
 - maxlearnsize, 223
 - model, 223
 - modelVariables, 221
 - New, 221
 - Nsvm, 223
 - nu, 223
 - objective, 222
 - openOptimization, 222
 - OpenSVM, 222
 - optimizationPoint2SVMParam, 222
 - osi, 223
 - p, 223
 - pip, 223
 - pOS, 224
 - param, 223
 - perm, 223
 - prob, 224
 - rbfKernel, 222
 - serialize, 222
 - SetLearningFraction, 222
 - SetMaximalLearningSize, 222
 - xspace, 224
 - Xsv, 224
- GoSUM::TModelVariable
 - ~TModelVariable, 240
 - boost::serialization::access, 244
 - clearSample, 240
 - computeStatistics, 240
 - distributionName, 240
 - distributionType, 240
 - expandedSize, 240
 - expectedValue, 240
 - exportCumulative, 240
 - exportDomain, 241
 - exportHistogram, 241
 - exportProbability, 241
 - exportQuantile, 241
 - exportSample, 241
 - exportSubHistogram, 241
 - generateSampleValue, 241
 - isDistributionDefined, 241
 - maxValue, 242
 - minValue, 242
 - normalizedHistogram, 242
 - pRV, 244
 - pS, 244
 - randomVariable, 242
 - readSampleValue, 242
 - sample, 242
 - sampleSize, 242
 - sampleValue, 242
 - serialize, 243
 - setDistributionParameters, 243
 - setEmpiricalDistribution, 243
 - setSampleSize, 243
 - setSampleValue, 243
 - setTheoreticalDistribution, 243
 - TModelVariable, 240
 - writeSampleValue, 243
- GoSUM::TModelVariable< R, S, t >, 238
- H
 - TSample, 251
- hasDiscreteData
 - CContinuousSample, 63
 - GoSUM::CContinuousMV, 59
- hcPoint2ModelPoint
 - GoSUM::CModelVariables, 148
- hcPoints2ModelPoints
 - GoSUM::CModelHypercube, 136
- hctype
 - GoSUM::CHypercube, 105
- history
 - COptimizationProblem, 166
- historySize
 - COptimizationProblem, 164
- hycube
 - GoSUM::CContainer, 56
- hyperCube
 - GoSUM::CContainer, 48
- I
 - GoSUM::CModelEvaluator, 134
- i
 - CDynamicSystemRNG, 74
 - DSn, 235
- i2
 - Cx_ZetaGammax, 234
- IA
 - RandomGenerators.cpp, 265

- IA1
 - RandomGenerators.cpp, [265](#)
- IA2
 - RandomGenerators.cpp, [265](#)
- IM
 - RandomGenerators.cpp, [265](#)
- IM1
 - RandomGenerators.cpp, [265](#)
- IM2
 - RandomGenerators.cpp, [265](#)
- IMM1
 - RandomGenerators.cpp, [265](#)
- IQ
 - RandomGenerators.cpp, [265](#)
- IQ1
 - RandomGenerators.cpp, [265](#)
- IQ2
 - RandomGenerators.cpp, [265](#)
- IR
 - RandomGenerators.cpp, [265](#)
- IR1
 - RandomGenerators.cpp, [265](#)
- IR2
 - RandomGenerators.cpp, [265](#)
- idum
 - CMediumPeriodRNG, [129](#)
- idum1
 - CSmallPeriodRNG, [225](#)
- idum2
 - CMediumPeriodRNG, [129](#)
- importDeclaredEmpiricalVariables
 - GoSUM::CContainer, [48](#)
- importEmpiricalVariables
 - GoSUM::CContainer, [48](#)
- importInitialValue
 - GoSUM::CContainer, [48](#)
- importInputs
 - GoSUM::CContainer, [48](#)
 - GoSUM::CScript, [201](#)
- importLowerBound
 - GoSUM::CContainer, [48](#)
- importModelConstraints
 - GoSUM::CContainer, [48](#)
- importNamedDeclaredEmpiricalVariables
 - GoSUM::CContainer, [48](#)
- importNamedEmpiricalVariables
 - GoSUM::CContainer, [48](#)
- importNamedTheoreticalVariables
 - GoSUM::CContainer, [48](#)
- importOptimizationConstraints
 - GoSUM::CContainer, [48](#)
- importOutputs
 - GoSUM::CContainer, [49](#)
 - GoSUM::CScript, [202](#)
- importPredictionInputSamples
 - GoSUM::CContainer, [49](#)
 - GoSUM::CScript, [202](#)
- importTheoreticalVariables
 - GoSUM::CContainer, [49](#)
- importUpperBound
 - GoSUM::CContainer, [49](#)
- importVariables
 - GoSUM::CContainer, [49](#)
 - GoSUM::CScript, [202](#)
- ims
 - CMADS, [117](#)
- in
 - GoSUM::CEvaluator, [87](#)
- inconsts
 - GoSUM::CContainer, [56](#)
- inext
 - CNLCRNG, [153](#)
- inextp
 - CNLCRNG, [153](#)
- initMeshSize
 - CMADS, [116](#)
- initialLHSearch
 - CMADS, [116](#)
- initialValue
 - COptimizationProblem, [164](#)
 - COptimizationVariable, [168](#)
- initialize
 - GoSUM::CReduction, [191](#)
- inputParameter
 - GoSUM::CContainer, [49](#)
 - GoSUM::CModelOptimizationVariable, [137](#)
- inputParameters
 - GoSUM::CContainer, [49](#)
- inputPoint2ModelPoint
 - GoSUM::CA analyticalOptimizationProblem, [18](#)
 - GoSUM::COriginalOptimizationProblem, [170](#)
 - GoSUM::CSimpleOptimizationProblem, [217](#)
- inputSamples
 - GoSUM::CA analyticalModel, [15](#)
- inputs
 - GoSUM::CContainer, [56](#)
- inputsSize
 - GoSUM::CContainer, [49](#)
- insert
 - GoSUM::CModelVariables, [148](#)
- int32
 - Ran, [237](#)
- int64
 - CLargePeriodRNG, [111](#)
 - Ran, [237](#)
- inverseDiscreteCosineTransform
 - CFFTW, [97](#)
- isConstant
 - TSample, [249](#)
- isConstrained
 - GoSUM::CModelConstraints, [131](#)
- IsContinuous
 - CRandomVariable, [186](#)
- isContinuous
 - GoSUM::CContinuousMV, [59](#)
 - GoSUM::CDiscreteMV, [67](#)

- GoSUM::CModelVariable, [141](#)
- IsDiscrete
 - CRandomVariable, [186](#)
- isDiscrete
 - GoSUM::CContinuousMV, [59](#)
 - GoSUM::CDiscreteMV, [67](#)
 - GoSUM::CModelVariable, [141](#)
- isDistributionDefined
 - CCategoricalDRV, [22](#)
 - CEmpiricalCRV, [77](#)
 - CEmpiricalDRV, [81](#)
 - CRandomVariable, [186](#)
 - GoSUM::CModelVariable, [141](#)
 - GoSUM::TModelVariable, [241](#)
- IsDistributionType
 - CRandomVariable, [187](#)
- IsEmpirical
 - CRandomVariable, [187](#)
- isFeasible
 - COptimizationProblem, [164](#)
 - GoSUM::CSimpleOptimizationProblem, [217](#)
- isMaximization
 - COptimizationProblem, [164](#)
- isMinimization
 - COptimizationProblem, [165](#)
- isModelVariable
 - GoSUM::CSimpleOptimizationProblem, [217](#)
- isModelVariableName
 - GoSUM::CModelVariables, [148](#)
- isSampleSelected
 - GoSUM::CContainer, [49](#)
- isSelectedInputParameter
 - GoSUM::CReduction, [191](#)
- isSelectedOutputState
 - GoSUM::CReduction, [191](#)
- IsTheoretical
 - CRandomVariable, [187](#)
- iterationLHSearch
 - CMADS, [116](#)
- iv1
 - CSmallPeriodRNG, [225](#)
- iv2
 - CMediumPeriodRNG, [129](#)
- iy1
 - CSmallPeriodRNG, [225](#)
- iy2
 - CMediumPeriodRNG, [129](#)
- j
 - CDynamicSystemRNG, [74](#)
 - DSn, [235](#)
- join
 - GoSUM::CContainer, [49](#)
- k
 - CDynamicSystemRNG, [74](#)
 - DSn, [235](#)
- KolmogorovSmirnovTest
 - Probability.cpp, [263](#)
- Probability.h, [263](#)
- lambda
 - CExponentialCRV, [97](#)
- largeperiod
 - CRandomGenerator, [181](#)
- latinize
 - GoSUM::CLCVoronoiHC, [113](#)
- lcvoronoi
 - GoSUM::CHypercube, [105](#)
- learn
 - GoSUM::CA analyticalModel, [15](#)
 - GoSUM::CSingleAM, [221](#)
- learnEvaluationSize
 - GoSUM::CContainer, [49](#)
- learnModel
 - GoSUM::CContainer, [50](#)
 - GoSUM::CScript, [202](#)
- learneddataoptimization
 - GoSUM::CContainer, [43](#)
- learnedmodeloptimization
 - GoSUM::CContainer, [43](#)
- learnfrac
 - GoSUM::CSingleAM, [223](#)
- learningFinished
 - GoSUM::CContainer, [56](#)
- LearningFraction
 - GoSUM::CSingleAM, [221](#)
- learningProgressed
 - GoSUM::CSingleAM, [223](#)
- learningStepsSize
 - GoSUM::CContainer, [49](#)
- lh0
 - CMADS, [117](#)
- lhi
 - CMADS, [117](#)
- libeng
 - CMATLAB, [122](#)
- libmat
 - CMATLAB, [122](#)
- libmx
 - CMATLAB, [122](#)
- line2Path
 - GoSUM::CScript, [202](#)
- linearInterpolation
 - VariousMath.cpp, [274](#)
 - VariousMath.h, [275](#)
- load
 - GoSUM::CContainer, [50](#)
 - GoSUM::CScript, [202](#)
- loadTxt
 - GoSUM::CContainer, [50](#)
 - GoSUM::CScript, [202](#)
- loadXml
 - GoSUM::CContainer, [50](#)
 - GoSUM::CScript, [202](#)
- Long
 - CDynamicSystemRNG, [73](#)
 - DSn, [234](#)

- longProjectName
 - GoSUM::CContainer, [50](#)
- lowerBound
 - COptimizationProblem, [165](#)
 - COptimizationVariable, [168](#)
 - CUniformCRV, [228](#)
 - CUniformDRV, [231](#)
- lp
 - CRandomGenerator, [183](#)
- MADS
 - GoSUM::CContainer, [56](#)
- MBIG
 - RandomGenerators.cpp, [265](#)
- MSEED
 - RandomGenerators.cpp, [265](#)
- MZ
 - RandomGenerators.cpp, [265](#)
- ma
 - CNLCRNG, [153](#)
- mad
 - CRandomGenerator, [183](#)
- mads
 - GoSUM::CContainer, [43](#)
- madsOptimizer
 - GoSUM::CContainer, [50](#)
- matClose
 - CMATLAB, [121](#)
- matGet
 - CMATLAB, [121](#)
- matGetVariable
 - CMATLAB, [121](#)
- matOpen
 - CMATLAB, [121](#)
- matPut
 - CMATLAB, [121](#)
- matPutVariable
 - CMATLAB, [122](#)
- matftype1
 - MatlabLibrary.h, [258](#)
- matftype2
 - MatlabLibrary.h, [258](#)
- matftype3
 - MatlabLibrary.h, [259](#)
- matftype4
 - MatlabLibrary.h, [259](#)
- matlab
 - GoSUM::CExeMatEvaluator, [93](#)
 - GoSUM::CMatlabEngineEvaluator, [126](#)
- MatlabLibrary.h
 - engftype1, [258](#)
 - engftype2, [258](#)
 - engftype3, [258](#)
 - engftype4, [258](#)
 - engftype5, [258](#)
 - engftype6, [258](#)
 - engftype7, [258](#)
 - engftype8, [258](#)
 - engftype9, [258](#)
- matftype1, [258](#)
- matftype2, [258](#)
- matftype3, [259](#)
- matftype4, [259](#)
- mxftype1, [259](#)
- mxftype2, [259](#)
- mxftype3, [259](#)
- mxftype4, [259](#)
- mxftype5, [259](#)
- matlabengine
 - GoSUM::CEvaluator, [85](#)
- matlabshell
 - GoSUM::CEvaluator, [85](#)
- maxValue
 - CCategoricalDRV, [22](#)
 - CCategoricalSample, [24](#)
 - CConstantCRV, [28](#)
 - CConstantDRV, [31](#)
 - CEmpiricalCRV, [77](#)
 - CEmpiricalDRV, [81](#)
 - CExponentialCRV, [96](#)
 - CGaussianCRV, [102](#)
 - CNumericalSample, [156](#)
 - CRandomVariable, [187](#)
 - CUniformCRV, [228](#)
 - CUniformDRV, [232](#)
 - GoSUM::CModelVariable, [142](#)
 - GoSUM::TModelVariable, [242](#)
 - TSample, [249](#)
- maxdiscretesize
 - CSample, [196](#)
- maxeval
 - CMADS, [117](#)
- MaximalLearningSize
 - GoSUM::CSingleAM, [221](#)
- maximalSensitivity
 - GoSUM::CReduction, [191](#)
- maximize
 - GoSUM::CContainer, [50](#)
 - GoSUM::CScript, [202](#)
- maxiter
 - GoSUM::CHypercube, [108](#)
- maxlearnsize
 - GoSUM::CSingleAM, [223](#)
- maxtries
 - GoSUM::CModelHypercube, [136](#)
- mean
 - CContinuousSample, [63](#)
 - CEmpiricalCRV, [77](#)
 - CGaussianCRV, [102](#)
- mediumperiod
 - CRandomGenerator, [181](#)
- minValue
 - CCategoricalDRV, [22](#)
 - CCategoricalSample, [24](#)
 - CConstantCRV, [28](#)
 - CConstantDRV, [31](#)
 - CEmpiricalCRV, [77](#)

- CEmpiricalDRV, [81](#)
- CExponentialCRV, [96](#)
- CGaussianCRV, [102](#)
- CNumericalSample, [156](#)
- CRandomVariable, [187](#)
- CUniformCRV, [228](#)
- CUniformDRV, [232](#)
- GoSUM::CModelVariable, [142](#)
- GoSUM::TModelVariable, [242](#)
- TSample, [249](#)
- minimalPollSize
 - CMADS, [116](#)
- minimize
 - COptimizationProblem, [166](#)
 - GoSUM::CContainer, [50](#)
 - GoSUM::CScript, [203](#)
- model
 - GoSUM::CSingleAM, [223](#)
- modelConstraints
 - GoSUM::CContainer, [50](#)
- modelEvaluator
 - GoSUM::CContainer, [50](#)
- modelVariables
 - GoSUM::CSingleAM, [221](#)
- modelanalysis
 - GoSUM::CContainer, [43](#)
- modeloptimization
 - GoSUM::CContainer, [43](#)
- montecarlo
 - GoSUM::CHypercube, [105](#)
- mp
 - CRandomGenerator, [183](#)
- mps
 - CMADS, [117](#)
- mu
 - CContinuousSample, [63](#)
 - CEmpiricalCRV, [78](#)
 - CGaussianCRV, [103](#)
- mvs
 - GoSUM::CModelVariables, [150](#)
- mxCreateDoubleMatrix
 - CMATLAB, [122](#)
- mxDestroyArray
 - CMATLAB, [122](#)
- mxGetNumberOfElements
 - CMATLAB, [122](#)
- mxGetPr
 - CMATLAB, [122](#)
- mxftype1
 - MatlabLibrary.h, [259](#)
- mxftype2
 - MatlabLibrary.h, [259](#)
- mxftype3
 - MatlabLibrary.h, [259](#)
- mxftype4
 - MatlabLibrary.h, [259](#)
- mxftype5
 - MatlabLibrary.h, [259](#)
- N
 - Cx_ZetaGammamax, [234](#)
- n
 - CDynamicSystemRNG, [74](#)
 - CUniformDRV, [233](#)
 - DSn, [235](#)
- NDIV
 - RandomGenerators.cpp, [265](#)
- NOMADPoint2ArrayXd
 - CMADS, [116](#)
- NTAB
 - RandomGenerators.cpp, [265](#)
 - RandomGenerators.h, [266](#)
- nTuple
 - GoSUM::CModelVariables, [148](#)
- NULL
 - GoSUM, [237](#)
- name
 - GoSUM::CModelVariable, [142](#)
- nameToFind
 - GoSUM::CModelVariables, [150](#)
- names
 - GoSUM::CConstraints, [35](#)
 - GoSUM::CParserOptimizationProblem, [177](#)
- New
 - GoSUM::CEvaluator, [86](#)
 - GoSUM::CHypercube, [106](#)
 - GoSUM::CSingleAM, [221](#)
- newColorBarPlot
 - CPlot2D, [179](#)
- newColorMap
 - CPlot2D, [179](#)
- newPlot
 - CPlot2D, [179](#)
- nextName
 - GoSUM::CModelVariables, [148](#)
- nlc
 - CRandomGenerator, [183](#)
- noeqI
 - GoSUM::CScript, [207](#)
- nomads
 - CRandomGenerator, [181](#)
- nonlinearcongruential
 - CRandomGenerator, [181](#)
- normalize
 - GoSUM::CModelVariables, [148](#)
- normalizedHistogram
 - GoSUM::TModelVariable, [242](#)
 - TSample, [250](#)
- Nsvm
 - GoSUM::CSingleAM, [223](#)
- nu
 - GoSUM::CSingleAM, [223](#)
- nusvr
 - GoSUM::CA analyticalModel, [14](#)
- objective
 - COptimizationProblem, [165](#)
 - GoSUM::CParserOptimizationProblem, [176](#)

- GoSUM::CSingleAM, [222](#)
- objectiveExpression
 - GoSUM::CParserOptimizationProblem, [176](#)
- objectiveHistory
 - COptimizationProblem, [165](#)
- omType
 - GoSUM::CContainer, [56](#)
- openEvaluation
 - GoSUM::CExeEvaluator, [90](#)
 - GoSUM::CMatlabEngineEvaluator, [125](#)
 - GoSUM::CModelEvaluator, [133](#)
 - GoSUM::COutputStates, [172](#)
- openOptimization
 - COptimizationProblem, [165](#)
 - GoSUM::CEpsSvrSAM, [83](#)
 - GoSUM::CNuSvrSAM, [159](#)
 - GoSUM::COriginalOptimizationProblem, [170](#)
 - GoSUM::CParserOptimizationProblem, [176](#)
 - GoSUM::CSimpleOptimizationProblem, [217](#)
 - GoSUM::CSingleAM, [222](#)
- OpenSVM
 - GoSUM::CSingleAM, [222](#)
- operator()
 - GoSUM::CExeEvaluator, [90](#)
 - GoSUM::CMatlabEngineEvaluator, [126](#)
 - GoSUM::CModelEvaluator, [133](#)
 - GoSUM::CModelVariables, [148](#), [149](#)
- optimizationMethod
 - GoSUM::CContainer, [50](#)
- OptimizationMethodType
 - GoSUM::CContainer, [50](#)
- optimizationPoint2SVMPParam
 - GoSUM::CEpsSvrSAM, [83](#)
 - GoSUM::CNuSvrSAM, [159](#)
 - GoSUM::CSingleAM, [222](#)
- optimizationProblem
 - GoSUM::CContainer, [51](#)
- optimizationStepsSize
 - GoSUM::CContainer, [51](#)
- optimizationmethodtype
 - GoSUM::CContainer, [43](#)
- optimize
 - CGAModelOptimization, [99](#)
 - CMADS, [116](#)
 - GoSUM::CContainer, [51](#)
- optimizingFinished
 - GoSUM::CContainer, [56](#)
- optimizingProgressed
 - GoSUM::CParserOptimizationProblem, [177](#)
- OriginalModel.cpp
 - GoSUM, [261](#)
- osi
 - GoSUM::CSingleAM, [223](#)
- out
 - GoSUM::CEvaluator, [87](#)
- outputSamples
 - GoSUM::CAnalyticalModel, [15](#)
- outputState
 - GoSUM::CContainer, [51](#)
- outputStates
 - GoSUM::CContainer, [51](#)
- outputs
 - GoSUM::CContainer, [57](#)
- outputsSize
 - GoSUM::CContainer, [51](#)
- ovind
 - GoSUM::CSimpleOptimizationProblem, [218](#)
- ovs
 - COptimizationProblem, [167](#)
- p
 - CCategoricalDRV, [23](#)
 - CEmpiricalCRV, [78](#)
 - CEmpiricalDRV, [82](#)
 - CRandomGenerator, [183](#)
 - GoSUM::CSingleAM, [223](#)
- pAM
 - GoSUM::CAnalyticalOptimizationProblem, [18](#)
 - GoSUM::CReduction, [193](#)
 - GoSUM::CSensitivityAnalysis, [214](#)
- pContainer
 - GoSUM::CScript, [208](#)
- pE
 - GoSUM::COutputStates, [173](#)
- pEngClose
 - CMATLAB, [122](#)
- pEngEvalString
 - CMATLAB, [123](#)
- pEngGetVariable
 - CMATLAB, [123](#)
- pEngGetVisible
 - CMATLAB, [123](#)
- pEngOpen
 - CMATLAB, [123](#)
- pEngOpenSingleUse
 - CMATLAB, [123](#)
- pEngOutputBuffer
 - CMATLAB, [123](#)
- pEngPutVariable
 - CMATLAB, [123](#)
- pEngSetVisible
 - CMATLAB, [123](#)
- pIC
 - GoSUM::CInputParameters, [110](#)
 - GoSUM::CModelHypercube, [136](#)
- pIP
 - GoSUM::CAnalyticalModel, [16](#)
 - GoSUM::CModelConstraints, [131](#)
 - GoSUM::CModelEvaluator, [134](#)
 - GoSUM::CModelHypercube, [136](#)
 - GoSUM::CReduction, [193](#)
 - GoSUM::CSensitivityAnalysis, [214](#)
 - GoSUM::CSimpleOptimizationProblem, [218](#)
 - GoSUM::CSingleAM, [223](#)
- pMatClose
 - CMATLAB, [123](#)
- pMatGetVariable

- CMATLAB, 123
- pMatOpen
 - CMATLAB, 123
- pMatPutVariable
 - CMATLAB, 123
- pMxCreateDoubleMatrix
 - CMATLAB, 124
- pMxDestroyArray
 - CMATLAB, 124
- pMxGetNumberOfElements
 - CMATLAB, 124
- pMxGetPr
 - CMATLAB, 124
- pMxSetPr
 - CMATLAB, 124
- pOP
 - CGAModelOptimization, 99
 - CMADS, 117
 - CMADSEvaluator, 118
 - GoSUM::CContainer, 57
- pOS
 - GoSUM::CA analyticalModel, 16
 - GoSUM::CModelEvaluator, 134
 - GoSUM::COriginalOptimizationProblem, 171
 - GoSUM::CReduction, 193
 - GoSUM::CSensitivityAnalysis, 214
 - GoSUM::CSingleAM, 224
- pP
 - GoSUM::CExeEvaluator, 91
- pRV
 - GoSUM::TModelVariable, 244
- pS
 - GoSUM::TModelVariable, 244
- pSA
 - GoSUM::CReduction, 193
- paction
 - GoSUM, 237
- padd
 - GoSUM::CScript, 207
- padds
 - GoSUM::CScript, 207
- param
 - GoSUM::CSingleAM, 223
- parseExpressions
 - GoSUM::CConstraints, 34
 - GoSUM::CParserOptimizationProblem, 176
- Path
 - CMATLAB, 122
- path
 - CMATLAB, 122
 - GoSUM::CEvaluator, 87
- pbandwidth
 - CEmpiricalCRV, 78
- perm
 - GoSUM::CSingleAM, 223
- permutation
 - VariousMath.cpp, 274
 - VariousMath.h, 276
- permutations
 - VariousMath.cpp, 274
 - VariousMath.h, 276
- pimport
 - GoSUM::CScript, 208
- pip
 - GoSUM::CModelOptimizationVariable, 138
- pipToFind
 - GoSUM::CSimpleOptimizationProblem, 218
- plot
 - CPlot2D, 180
- plotWithHistogram
 - CPlot2D, 179
- plotWithPen
 - CPlot2D, 179
- plotWithSpectrogram
 - CPlot2D, 179
- plotWithSymbol
 - CPlot2D, 179, 180
- popsze
 - CGAModelOptimization, 100
- populationSize
 - CGAModelOptimization, 99
- predict
 - GoSUM::CA analyticalModel, 15
 - GoSUM::CContainer, 51
 - GoSUM::CScript, 203
- predictDerivative
 - GoSUM::CA analyticalModel, 15
- predictDerivatives
 - GoSUM::CA analyticalModel, 15
- predictMean
 - GoSUM::CContainer, 51
- predictValue
 - GoSUM::CA analyticalModel, 15
- predictValues
 - GoSUM::CA analyticalModel, 16
- prepare
 - CDynamicSystemRNG, 74
 - GoSUM::CSensitivityAnalysis, 212
- prepareDSample
 - Ds.cpp, 255
 - Ds.h, 255
- prjName
 - GoSUM::CContainer, 57
- prjPath
 - GoSUM::CContainer, 57
- prjType
 - GoSUM::CContainer, 57
- prob
 - GoSUM::CSingleAM, 224
- probability
 - CCategoricalDRV, 22
 - CConstantCRV, 28
 - CConstantDRV, 31
 - CContinuousRV, 60
 - CEmpiricalCRV, 77
 - CEmpiricalDRV, 81

- CExponentialCRV, 96
- CGaussianCRV, 103
- CUniformCRV, 228
- CUniformDRV, 232
- TRandomVariable, 246
- Probability.cpp
 - chiSquareTest, 263
 - KolmogorovSmirnovTest, 263
- Probability.h
 - chiSquareTest, 263
 - KolmogorovSmirnovTest, 263
- ProgressSize
 - GoSUM::CHypercube, 106
- progressSlot
 - GoSUM::CA analyticalModel, 16
 - GoSUM::CInputParameters, 110
 - GoSUM::COutputStates, 173
- progressStepsSize
 - CGAModelOptimization, 99
 - CMADS, 116
 - GoSUM::CSensitivityAnalysis, 212
- projectName
 - GoSUM::CContainer, 51
- projectPath
 - GoSUM::CContainer, 51
- ProjectType
 - GoSUM::CContainer, 51
- projectType
 - GoSUM::CContainer, 51
- projecttype
 - GoSUM::CContainer, 43
- q
 - GoSUM::CHypercube, 108
- qbandwidth
 - CEmpiricalCRV, 78
- quantile
 - CContinuousRV, 60
 - CDiscreteRV, 68
 - CRandomVariable, 187
- RNMx
 - RandomGenerators.cpp, 265
- RSSize
 - GoSUM::CInputParameters, 110
 - GoSUM::CSensitivityAnalysis, 214
- Ran, 237
 - doub, 237
 - int32, 237
 - int64, 237
 - Ran, 237
 - u, 238
 - v, 238
 - w, 238
- ran.h
 - Ullong, 264
- RandomGenerators.cpp
 - AM, 265
 - EPS, 265
 - FAC, 265
 - IA, 265
 - IA1, 265
 - IA2, 265
 - IM, 265
 - IM1, 265
 - IM2, 265
 - IMM1, 265
 - IQ, 265
 - IQ1, 265
 - IQ2, 265
 - IR, 265
 - IR1, 265
 - IR2, 265
 - MBIG, 265
 - MSEED, 265
 - MZ, 265
 - NDIV, 265
 - NTAB, 265
 - RNMx, 265
- RandomGenerators.h
 - NTAB, 266
 - UINT32_MAX, 267
- randomVariable
 - GoSUM::CModelVariable, 142
 - GoSUM::TModelVariable, 242
- RandomVariable.h
 - CTContinuousRV, 268
 - CTDiscreteRV, 268
 - rvExportMacro, 268
- range
 - TSample, 251
- rankIP
 - GoSUM::CReduction, 193
- rankOS
 - GoSUM::CReduction, 193
- rateParameter
 - CExponentialCRV, 96
- rbfKernel
 - GoSUM::CSingleAM, 222
- read
 - GoSUM::CScript, 203
- readFrom
 - GoSUM::CExeAsciiEvaluator, 88
 - GoSUM::CExeEvaluator, 90
 - GoSUM::CExeMatEvaluator, 93
- readSampleValue
 - CCategoricalSample, 24
 - CNumericalSample, 156
 - CSample, 195
 - GoSUM::CModelVariable, 142
 - GoSUM::TModelVariable, 242
 - TSample, 250
- readSizeFrom
 - GoSUM::CExeAsciiEvaluator, 88
 - GoSUM::CExeEvaluator, 90
 - GoSUM::CExeMatEvaluator, 93
- readyToComputeSensitivities

- GoSUM::CContainer, 52
- readyToEvaluateOutputs
 - GoSUM::CContainer, 52
- readyToLearnModel
 - GoSUM::CContainer, 52
- readyToOptimize
 - GoSUM::CContainer, 52
- readyToReduceModel
 - GoSUM::CContainer, 52
- readyToSampleInputs
 - GoSUM::CContainer, 52
- readyToSetOptimization
 - GoSUM::CContainer, 52
- redtype
 - GoSUM::CReduction, 193
- reduce
 - GoSUM::CContainer, 52
 - GoSUM::CScript, 203
- reduced
 - GoSUM::CContainer, 57
- reducer
 - GoSUM::CContainer, 52
- Reduction.cpp
 - compareFirst, 268
 - GoSUM, 269
- ReductionType
 - GoSUM::CReduction, 191
- reductiontype
 - GoSUM::CReduction, 190
- rename
 - GoSUM::CModelVariables, 149
- replace
 - GoSUM::CModelVariables, 149
- replot
 - CPlot2D, 180
- resampleInputs
 - GoSUM::CContainer, 52
 - GoSUM::CScript, 203
- resampleSize
 - GoSUM::CInputParameters, 109
 - GoSUM::CSensitivityAnalysis, 212
- resampleStepsSize
 - GoSUM::CInputParameters, 109
- resamplingFinished
 - GoSUM::CContainer, 57
- reset
 - GoSUM::CModelOptimizationVariable, 137
- resetOptimizationVariable
 - GoSUM::CContainer, 52
- Rnd
 - CRandomGenerator, 182
- rnd
 - CBaseRNG, 19
 - CDynamicSystemRNG, 74
 - CLargePeriodRNG, 111
 - CMediumPeriodRNG, 129
 - CNLCRNG, 152
 - CNomadRNG, 154
 - CSmallPeriodRNG, 225
- RndInt
 - CRandomGenerator, 182
- rndi
 - CBaseRNG, 19
 - CDynamicSystemRNG, 74
 - CLargePeriodRNG, 111
 - CMediumPeriodRNG, 129
 - CNLCRNG, 153
 - CNomadRNG, 154
 - CSmallPeriodRNG, 225
- rngtype
 - CRandomGenerator, 181
- roundoffEquality
 - GoSUM::CConstraints, 34
 - GoSUM::CParserOptimizationProblem, 176
- rvExportMacro
 - RandomVariable.h, 268
 - TRandomVariable, 246
- sam
 - GoSUM::CA analyticalModel, 16
- sample
 - GoSUM::CModelVariable, 142
 - GoSUM::TModelVariable, 242
 - TSample, 251
- Sample.h
 - CTContinuousSample, 270
 - CTDiscreteSample, 270
- sample1
 - GoSUM::CSensitivityAnalysis, 214
- sample2
 - GoSUM::CSensitivityAnalysis, 214
- sampleSize
 - CSample, 195
 - GoSUM::CModelVariable, 142
 - GoSUM::CModelVariables, 149
 - GoSUM::TModelVariable, 242
 - TSample, 250
- sampleValue
 - GoSUM::CModelVariable, 142
 - GoSUM::TModelVariable, 242
 - TSample, 250
- samplegeneration
 - GoSUM::CContainer, 43
- samples
 - GoSUM::CModelVariables, 149
 - GoSUM::CSimpleOptimizationProblem, 218
- save
 - GoSUM::CContainer, 52
 - GoSUM::CScript, 203
- saveTxt
 - GoSUM::CContainer, 53
 - GoSUM::CScript, 203
- saveXml
 - GoSUM::CContainer, 53
 - GoSUM::CScript, 203
- scl
 - GoSUM::CModelVariables, 150

- secondOrderANOVA
 - GoSUM::CSensitivityAnalysis, 212
- secondOrderANOVAPairs
 - GoSUM::CSensitivityAnalysis, 212
- select
 - CSample, 195
 - TSample, 250
- selectInputParameters
 - GoSUM::CReduction, 191
- selectOutputState
 - GoSUM::CReduction, 192
- selectOutputs
 - GoSUM::CReduction, 192
- selectSamples
 - GoSUM::CContainer, 53
- selectedSamples
 - GoSUM::CContainer, 53
- selectedsamples
 - GoSUM::CContainer, 57
- sensitivity
 - GoSUM::CContainer, 57
- sensitivityAnalysis
 - GoSUM::CContainer, 53
- sensitivityComputingFinished
 - GoSUM::CContainer, 57
- sensitivityError
 - GoSUM::CSensitivityAnalysis, 212
- sensitivityIndex
 - GoSUM::CReduction, 192
- separabilityError
 - GoSUM::CSensitivityAnalysis, 212
- separateBySelection
 - GoSUM::CContainer, 53
- serialize
 - CCategoricalDRV, 22
 - CCategoricalSample, 25
 - CConstantCRV, 28
 - CConstantDRV, 31
 - CContinuousRV, 60
 - CContinuousSample, 63
 - CDiscreteRV, 69
 - CDiscreteSample, 70
 - CEmpiricalCRV, 77
 - CEmpiricalDRV, 81
 - CExponentialCRV, 96
 - CGAModelOptimization, 99
 - CGaussianCRV, 103
 - CMADS, 116
 - CNumericalSample, 156
 - COptimizationProblem, 165
 - COptimizationVariable, 168
 - CRandomVariable, 187
 - CSample, 195
 - CUniformCRV, 228
 - CUniformDRV, 232
 - GoSUM::CANalyticalModel, 16
 - GoSUM::CANalyticalOptimizationProblem, 18
 - GoSUM::CConstraints, 34
 - GoSUM::CContainer, 53
 - GoSUM::CContinuousMV, 59
 - GoSUM::CCSvcSAM, 64
 - GoSUM::CCVoronoiHC, 66
 - GoSUM::CDiscreteMV, 67
 - GoSUM::CDSampleHC, 72
 - GoSUM::CEpsSvrSAM, 83
 - GoSUM::CEvaluator, 86
 - GoSUM::CExeAsciiEvaluator, 88
 - GoSUM::CExeEvaluator, 91
 - GoSUM::CExeMatEvaluator, 93
 - GoSUM::CHypercube, 106
 - GoSUM::CInputParameters, 110
 - GoSUM::CLCVoronoiHC, 113
 - GoSUM::CMatlabEngineEvaluator, 126
 - GoSUM::CMatlabShellEvaluator, 127
 - GoSUM::CModelConstraints, 131
 - GoSUM::CModelEvaluator, 133
 - GoSUM::CModelHypercube, 136
 - GoSUM::CModelOptimizationVariable, 138
 - GoSUM::CModelVariable, 142
 - GoSUM::CModelVariables, 149
 - GoSUM::CMonteCarloHC, 151
 - GoSUM::CNuSvcSAM, 158
 - GoSUM::CNuSvrSAM, 159
 - GoSUM::COneClassSAM, 160
 - GoSUM::COriginalOptimizationProblem, 171
 - GoSUM::COutputStates, 173
 - GoSUM::CParserOptimizationProblem, 176
 - GoSUM::CReduction, 192
 - GoSUM::CSensitivityAnalysis, 212
 - GoSUM::CSimpleOptimizationProblem, 217
 - GoSUM::CSingleAM, 222
 - GoSUM::TModelVariable, 243
 - TRandomVariable, 246
 - TSample, 250
- Set
 - CRandomGenerator, 182
- set
 - COptimizationVariable, 168
 - GoSUM::CModelConstraints, 131
- setBaseName
 - GoSUM::CModelVariables, 149
- setConstInitialValue
 - GoSUM::CScript, 203
- setConstLowerBound
 - GoSUM::CScript, 204
- setConstUpperBound
 - GoSUM::CScript, 204
- setConstantValue
 - CConstantCRV, 28
 - CConstantDRV, 31
- setConstraintExpression
 - GoSUM::CParserOptimizationProblem, 176
- setCoreSize
 - GoSUM::CScript, 204
- setCouplingError
 - GoSUM::CSensitivityAnalysis, 212

- setCutoffSize
 - GoSUM::CReduction, [192](#)
- SetCutoffType
 - GoSUM::CReduction, [192](#)
- setCutoffValue
 - GoSUM::CReduction, [192](#)
- SetCvtIterationSize
 - GoSUM::CHypercube, [106](#)
- SetCvtNewCenterCoefficient
 - GoSUM::CHypercube, [106](#)
- SetCvtOldCenterCoefficient
 - GoSUM::CHypercube, [106](#)
- SetCvtPointsSize
 - GoSUM::CHypercube, [106](#)
- setDistribution
 - CCategoricalDRV, [22](#)
 - CConstantCRV, [28](#)
 - CConstantDRV, [31](#)
 - CEmpiricalCRV, [77](#)
 - CEmpiricalDRV, [81](#)
 - CExponentialCRV, [96](#)
 - CGaussianCRV, [103](#)
 - CUniformCRV, [228](#)
 - CUniformDRV, [232](#)
 - GoSUM::CModelVariable, [142](#)
 - TRandomVariable, [246](#)
- setDistributionParameters
 - GoSUM::TModelVariable, [243](#)
- setDistributions
 - GoSUM::CModelVariables, [149](#)
- SetDynamicSystem
 - CRandomGenerator, [182](#)
- setEmpiricalDistribution
 - GoSUM::CContinuousMV, [59](#)
 - GoSUM::CDiscreteMV, [67](#)
 - GoSUM::CModelVariable, [143](#)
 - GoSUM::TModelVariable, [243](#)
- setEvaluationSize
 - CMADS, [116](#)
- setEvaluatorCompatible
 - GoSUM::COutputStates, [173](#)
- setExpression
 - GoSUM::CConstraints, [34](#)
- SetExternalEvaluator
 - GoSUM::CEvaluator, [86](#)
- setInitialLHSearch
 - CMADS, [116](#)
- setInitialMeshSize
 - CMADS, [116](#)
- setInitialValue
 - COptimizationProblem, [165](#)
 - COptimizationVariable, [168](#)
 - GoSUM::CContainer, [53](#)
 - GoSUM::CScript, [204](#)
- setInputSamples
 - GoSUM::CAnalyticalModel, [16](#)
- setIterationLHSearch
 - CMADS, [116](#)
- SetLargePeriod
 - CRandomGenerator, [182](#)
- SetLearningFraction
 - GoSUM::CSingleAM, [222](#)
- setLowerBound
 - COptimizationProblem, [165](#)
 - COptimizationVariable, [168](#)
 - CUniformCRV, [228](#)
 - CUniformDRV, [232](#)
 - GoSUM::CContainer, [53](#)
 - GoSUM::CScript, [204](#)
- setMadsInitMeshSize
 - GoSUM::CContainer, [53](#)
 - GoSUM::CScript, [204](#)
- setMadsLHSearch
 - GoSUM::CContainer, [53](#)
 - GoSUM::CScript, [204](#)
- setMadsMaxEvaluation
 - GoSUM::CContainer, [53](#)
 - GoSUM::CScript, [205](#)
- setMadsMinPollSize
 - GoSUM::CContainer, [53](#)
 - GoSUM::CScript, [205](#)
- setMatLabPath
 - GoSUM::CContainer, [54](#)
 - GoSUM::CScript, [205](#)
- SetMaximalLearningSize
 - GoSUM::CSingleAM, [222](#)
- setMaximization
 - COptimizationProblem, [166](#)
- setMean
 - CGaussianCRV, [103](#)
- SetMediumPeriod
 - CRandomGenerator, [182](#)
- setMinimalPollSize
 - CMADS, [116](#)
- setMinimization
 - COptimizationProblem, [166](#)
- setModelEvaluator
 - GoSUM::CContainer, [54](#)
 - GoSUM::CScript, [205](#)
- setNTuple
 - GoSUM::CModelVariables, [149](#)
- setName
 - GoSUM::CModelVariable, [143](#)
- SetNomads
 - CRandomGenerator, [182](#)
- SetNonlinearCongruential
 - CRandomGenerator, [182](#)
- setNormalization
 - GoSUM::CModelVariables, [149](#)
- setObjective
 - GoSUM::CContainer, [54](#)
 - GoSUM::CScript, [205](#)
- setObjectiveExpression
 - GoSUM::CParserOptimizationProblem, [176](#)
- setOptimizationMethod
 - GoSUM::CContainer, [54](#)

- GoSUM::CScript, 205
- SetPath
 - CMATLAB, 122
- setPopulationSize
 - CGAModelOptimization, 99
- setProgressSlot
 - GoSUM::CA analyticalModel, 16
 - GoSUM::CInputParameters, 110
 - GoSUM::COutputStates, 173
- setProjectName
 - GoSUM::CContainer, 54
 - GoSUM::CScript, 205
- setProjectPath
 - GoSUM::CContainer, 54
 - GoSUM::CScript, 205
- setProjectType
 - GoSUM::CContainer, 54
 - GoSUM::CScript, 206
- setRNG
 - GoSUM::CContainer, 55
 - GoSUM::CScript, 207
- setRateParameter
 - CExponentialCRV, 96
- setReductionCutoffSize
 - GoSUM::CContainer, 54
 - GoSUM::CScript, 206
- setReductionCutoffValue
 - GoSUM::CContainer, 54
 - GoSUM::CScript, 206
- setReductionOutputs
 - GoSUM::CContainer, 54
 - GoSUM::CScript, 206
- SetReductionType
 - GoSUM::CReduction, 192
- setReductionType
 - GoSUM::CContainer, 54
 - GoSUM::CReduction, 192
 - GoSUM::CScript, 206
- setResampleSize
 - GoSUM::CContainer, 54
 - GoSUM::CInputParameters, 110
 - GoSUM::CScript, 206
 - GoSUM::CSensitivityAnalysis, 212
- setResampleType
 - GoSUM::CContainer, 55
 - GoSUM::CScript, 206
- setSampleSize
 - CCategoricalSample, 25
 - CNumericalSample, 156
 - CSample, 196
 - GoSUM::CModelVariable, 143
 - GoSUM::TModelVariable, 243
 - TSample, 250
- setSampleValue
 - CSample, 196
 - GoSUM::CModelVariable, 143
 - GoSUM::TModelVariable, 243
 - TSample, 250
- setSamples
 - GoSUM::CModelVariables, 149
- SetSeed
 - CRandomGenerator, 182
- setSeed
 - CBaseRNG, 19
 - CDynamicSystemRNG, 74
 - CLargePeriodRNG, 112
 - CMediumPeriodRNG, 129
 - CNLCRNG, 153
 - CNomadRNG, 154
 - CSmallPeriodRNG, 225
- setSensitivityError
 - GoSUM::CSensitivityAnalysis, 213
- setSensitivityOptions
 - GoSUM::CContainer, 55
 - GoSUM::CScript, 207
 - GoSUM::CSensitivityAnalysis, 213
- setSeparabilityError
 - GoSUM::CSensitivityAnalysis, 213
- SetSmallPeriod
 - CRandomGenerator, 182
- setStandardDeviation
 - CGaussianCRV, 103
- setSvmType
 - GoSUM::CA analyticalModel, 16
- setTheoreticalDistribution
 - GoSUM::CContinuousMV, 59
 - GoSUM::CDiscreteMV, 67
 - GoSUM::CModelVariable, 143
 - GoSUM::TModelVariable, 243
- SetThreadSize
 - GoSUM::CEvaluator, 86
- setThreadSize
 - GoSUM::CContainer, 55
- setTitles
 - CPlot2D, 180
- SetType
 - GoSUM::CEvaluator, 86
 - GoSUM::CHypercube, 107
- setUpperBound
 - COptimizationProblem, 166
 - COptimizationVariable, 168
 - CUniformCRV, 228
 - CUniformDRV, 232
 - GoSUM::CContainer, 55
 - GoSUM::CScript, 207
- setVariableNames
 - GoSUM::CConstraints, 35
 - GoSUM::CModelConstraints, 131
 - GoSUM::COriginalOptimizationProblem, 171
 - GoSUM::CParserOptimizationProblem, 177
 - GoSUM::CSimpleOptimizationProblem, 217
- SetVoronoiOptions
 - GoSUM::CHypercube, 107
- setVoronoiOptions
 - GoSUM::CContainer, 55
 - GoSUM::CScript, 207

- si1
 - GoSUM::CSensitivityAnalysis, 214
- si1der
 - GoSUM::CSensitivityAnalysis, 214
- si2
 - GoSUM::CSensitivityAnalysis, 214
- siT
 - GoSUM::CSensitivityAnalysis, 214
- sigma
 - CGaussianCRV, 103
- sigma2
 - CGaussianCRV, 103
- sigmasqrt2
 - CGaussianCRV, 104
- simpleoptimization
 - GoSUM::CContainer, 43
- size
 - GoSUM::CConstraints, 35
 - GoSUM::CModelVariables, 150
- smallperiod
 - CRandomGenerator, 181
- sp
 - CRandomGenerator, 183
- squareDistance
 - VariousMath.cpp, 274
 - VariousMath.h, 276
- standardDeviation
 - CContinuousSample, 63
 - CEmpiricalCRV, 78
 - CGaussianCRV, 103
- startEvaluating
 - GoSUM::CContainer, 55
- startLearning
 - GoSUM::CContainer, 55
- startOptimizing
 - GoSUM::CContainer, 55
- startResampling
 - GoSUM::CContainer, 55
- startSensitivityComputing
 - GoSUM::CContainer, 55
- status
 - GoSUM::CModelConstraints, 131
- stdv
 - CContinuousSample, 63
 - CEmpiricalCRV, 78
- string2WideString
 - Utilities.cpp, 271
 - Utilities.h, 273
- svmType
 - GoSUM::CANalyticalModel, 16
- svmtyp
 - GoSUM::CANalyticalModel, 14
- systemCommand
 - GoSUM::CExeEvaluator, 91
 - GoSUM::CMatlabShellEvaluator, 127
- TINY
 - VariousMath.h, 275
- TModelVariable
 - GoSUM::TModelVariable, 240
- TRandomVariable
 - ~TRandomVariable, 245
 - boost::serialization::access, 246
 - cumulative, 245, 246
 - Dynamic, 246
 - exportDomain, 245
 - probability, 246
 - rvExportMacro, 246
 - serialize, 246
 - setDistribution, 246
 - TRandomVariable, 245
 - TRandomVariable, 245
- TRandomVariable< t, T >, 244
- TSample
 - ~TSample, 248
 - boost::serialization::access, 251
 - cH, 251
 - clear, 248
 - computeNormalizedHistogram, 248
 - computeNormalizedSubHistogram, 249
 - cummulativeHistogram, 249
 - eraseSelected, 249
 - exportHistogram, 249
 - exportSample, 249
 - exportSubHistogram, 249
 - H, 251
 - isConstant, 249
 - maxValue, 249
 - minValue, 249
 - normalizedHistogram, 250
 - range, 251
 - readSampleValue, 250
 - sample, 251
 - sampleSize, 250
 - sampleValue, 250
 - select, 250
 - serialize, 250
 - setSampleSize, 250
 - setSampleValue, 250
 - TSample, 248
 - TSample, 248
 - writeSampleValue, 251
 - x, 251
- TSample< t >, 246
- thrd
 - GoSUM::CContainer, 57
- ThreadSize
 - GoSUM::CEvaluator, 86
- trdl
 - GoSUM::CModelEvaluator, 134
- trdN
 - GoSUM::CEvaluator, 87
- trn
 - GoSUM::CModelVariables, 150
- Type
 - CRandomGenerator, 182
 - CRandomVariable, 187

- GoSUM::CEvaluator, [86](#)
- GoSUM::CHypercube, [107](#)
- type
 - CRandomGenerator, [183](#)
 - GoSUM::CEvaluator, [87](#)
- u
 - CLargePeriodRNG, [112](#)
 - Ran, [238](#)
- UINT32_MAX
 - RandomGenerators.h, [267](#)
- Ullong
 - CLargePeriodRNG, [111](#)
 - ran.h, [264](#)
- uname
 - GoSUM::CModelVariable, [143](#)
- uniformcontinuous
 - CRandomVariable, [185](#)
- uniformdiscrete
 - CRandomVariable, [185](#)
- upperBound
 - COptimizationProblem, [166](#)
 - COptimizationVariable, [168](#)
 - CUniformCRV, [229](#)
 - CUniformDRV, [232](#)
- Utilities.cpp
 - string2WideString, [271](#)
- Utilities.h
 - _USE_MATH_DEFINES, [272](#)
 - boostSerializeArrayXXdiMacro, [273](#)
 - boostSerializeArrayXdiMacro, [272](#)
 - string2WideString, [273](#)
- uvind
 - GoSUM::CSimpleOptimizationProblem, [218](#)
- v
 - CLargePeriodRNG, [112](#)
 - Ran, [238](#)
- validateExpressions
 - GoSUM::CConstraints, [35](#)
 - GoSUM::CParserOptimizationProblem, [177](#)
- values
 - CNumericalSample, [157](#)
- variable
 - GoSUM::CSimpleOptimizationProblem, [217](#), [218](#)
- variableHistory
 - COptimizationProblem, [166](#)
- variableName
 - GoSUM::CSimpleOptimizationProblem, [218](#)
- variance
 - CCategoricalDRV, [22](#)
 - CCategoricalSample, [25](#)
 - CConstantCRV, [28](#)
 - CConstantDRV, [32](#)
 - CContinuousSample, [63](#)
 - CDiscreteSample, [71](#)
 - CEmpiricalCRV, [78](#)
 - CEmpiricalDRV, [81](#)
 - CExponentialCRV, [96](#)
 - CGaussianCRV, [103](#)
 - CNumericalSample, [156](#)
 - CRandomVariable, [187](#)
 - CSample, [196](#)
 - CUniformCRV, [229](#)
 - CUniformDRV, [232](#)
 - GoSUM::CReduction, [190](#)
 - VariousMath.cpp, [274](#)
 - VariousMath.h, [276](#)
- varianceSensitivity
 - GoSUM::CSensitivityAnalysis, [213](#)
- VariousMath.cpp
 - cumulativeSum, [274](#)
 - findNearest, [274](#)
 - linearInterpolation, [274](#)
 - permutation, [274](#)
 - permutations, [274](#)
 - squareDistance, [274](#)
 - variance, [274](#)
- VariousMath.h
 - cumulativeSum, [275](#)
 - findNearest, [275](#)
 - findRoot, [275](#)
 - linearInterpolation, [275](#)
 - permutation, [276](#)
 - permutations, [276](#)
 - squareDistance, [276](#)
 - TINY, [275](#)
 - variance, [276](#)
- VoronoiOptions
 - GoSUM::CHypercube, [107](#)
- w
 - CDynamicSystemRNG, [74](#)
 - CLargePeriodRNG, [112](#)
 - DSn, [235](#)
 - Ran, [238](#)
- writeHistory
 - COptimizationProblem, [166](#)
- writeSampleValue
 - CCategoricalSample, [25](#)
 - CNumericalSample, [156](#)
 - CSample, [196](#)
 - GoSUM::CModelVariable, [143](#)
 - GoSUM::TModelVariable, [243](#)
 - TSample, [251](#)
- writeTo
 - GoSUM::CExeAsciiEvaluator, [88](#)
 - GoSUM::CExeEvaluator, [91](#)
 - GoSUM::CExeMatEvaluator, [93](#)
- X
 - GoSUM::CAnalyticalModel, [17](#)
- x
 - CDynamicSystemRNG, [74](#)
 - CEmpiricalCRV, [78](#)
 - CNomadRNG, [154](#)
 - DSn, [235](#)
 - TSample, [251](#)

x0
 COptimizationVariable, [169](#)

xL
 COptimizationVariable, [169](#)

xU
 COptimizationVariable, [169](#)

Xbest
 CGAModelOptimization, [100](#)
 CMADS, [117](#)

xc
 CConstantCRV, [29](#)

xspace
 GoSUM::CSingleAM, [224](#)

Xsv
 GoSUM::CSingleAM, [224](#)

Y
 GoSUM::CA analyticalModel, [17](#)

y
 CNomadRNG, [154](#)
 GoSUM::CSensitivityAnalysis, [214](#)

ybest
 CGAModelOptimization, [100](#)
 CMADS, [117](#)

ymu
 GoSUM::CSensitivityAnalysis, [214](#)

yvar
 GoSUM::CSensitivityAnalysis, [214](#)

z
 CNomadRNG, [154](#)