

## GuidewireSimulation

Generated by Doxygen 1.9.1



<b>1 Requirements Traceability</b>	<b>1</b>
<b>2 Namespace Index</b>	<b>5</b>
2.1 Packages	5
<b>3 Hierarchical Index</b>	<b>7</b>
3.1 Class Hierarchy	7
<b>4 Class Index</b>	<b>9</b>
4.1 Class List	9
<b>5 File Index</b>	<b>11</b>
5.1 File List	11
<b>6 Namespace Documentation</b>	<b>13</b>
6.1 GuidewireSim Namespace Reference	13
<b>7 Class Documentation</b>	<b>15</b>
7.1 GuidewireSim.ConstraintSolvingStep Class Reference	15
7.1.1 Detailed Description	16
7.1.2 Member Function Documentation	16
7.1.2.1 Awake()	16
7.1.2.2 CorrectBendTwistPredictions()	16
7.1.2.3 CorrectStretchPredictions()	17
7.1.2.4 SolveBendTwistConstraint()	17
7.1.2.5 SolveBendTwistConstraints()	18
7.1.2.6 SolveBendTwistConstraintsInBilateralOrder()	19
7.1.2.7 SolveBendTwistConstraintsInNaiveOrder()	20
7.1.2.8 SolveStretchConstraint()	20
7.1.2.9 SolveStretchConstraints()	21
7.1.2.10 SolveStretchConstraintsInBilateralOrder()	22
7.1.2.11 SolveStretchConstraintsInNaiveOrder()	23
7.1.3 Member Data Documentation	23
7.1.3.1 deltaOrientation	23
7.1.3.2 deltaOrientationOne	24
7.1.3.3 deltaOrientationTwo	24
7.1.3.4 deltaPositionOne	24
7.1.3.5 deltaPositionTwo	24
7.1.3.6 executeInBilateralOrder	24
7.1.3.7 mathHelper	24
7.2 GuidewireSim.DirectorsDrawer Class Reference	25
7.2.1 Detailed Description	25
7.2.2 Member Function Documentation	25
7.2.2.1 Awake()	26

7.2.2.2 CalculateArrowHeadPositions()	26
7.2.2.3 DrawArrowHeadConnectionLines()	26
7.2.2.4 DrawArrowHeadLines()	26
7.2.2.5 DrawDirectors()	27
7.2.2.6 Update()	27
7.2.3 Member Data Documentation	27
7.2.3.1 arrowHeadAngle	27
7.2.3.2 arrowHeadPercentage	28
7.2.3.3 directorColors	28
7.2.3.4 directorOneColor	28
7.2.3.5 directorThreeColor	28
7.2.3.6 directorTwoColor	28
7.2.3.7 scaleFactor	29
7.2.3.8 simulationLoop	29
7.3 GuidewireSim.ForceTestPerformer Class Reference	29
7.3.1 Detailed Description	30
7.3.2 Member Function Documentation	30
7.3.2.1 Awake()	30
7.3.2.2 PerformForceTestFour()	30
7.3.2.3 PerformForceTestOne()	30
7.3.2.4 PerformForceTests()	30
7.3.2.5 PerformForceTestThree()	30
7.3.2.6 PerformForceTestTwo()	31
7.3.2.7 PerformSingleLoopTest()	31
7.3.2.8 Start()	31
7.3.3 Member Data Documentation	31
7.3.3.1 doForceTestFour	31
7.3.3.2 doForceTestOne	31
7.3.3.3 doForceTestThree	32
7.3.3.4 doForceTestTwo	32
7.3.3.5 doSingleLoopTest	32
7.3.3.6 pullForceTestThree	32
7.3.3.7 simulationLoop	32
7.4 GuidewireSim.InitializationStep Class Reference	32
7.4.1 Detailed Description	33
7.4.2 Member Function Documentation	33
7.4.2.1 Awake()	33
7.4.2.2 InitCylinderAngularVelocities()	33
7.4.2.3 InitCylinderExternalTorques()	34
7.4.2.4 InitCylinderOrientationPredictions()	34
7.4.2.5 InitCylinderOrientations()	34
7.4.2.6 InitCylinderPositions()	35

7.4.2.7 InitCylinderScalarWeights()	35
7.4.2.8 InitDirectors()	35
7.4.2.9 InitDiscreteRestDarbouxVectors()	36
7.4.2.10 InitInertiaTensor()	37
7.4.2.11 InitInverseInertiaTensor()	37
7.4.2.12 InitSphereExternalForces()	37
7.4.2.13 InitSphereInverseMasses()	37
7.4.2.14 InitSpherePositionPredictions()	38
7.4.2.15 InitSpherePositions()	38
7.4.2.16 InitSphereVelocities()	39
7.4.2.17 InitWorldSpaceBasis()	39
7.4.3 Member Data Documentation	39
7.4.3.1 materialDensity	39
7.4.3.2 materialRadius	39
7.4.3.3 mathHelper	40
7.5 GuidewireSim.MathHelper Class Reference	40
7.5.1 Detailed Description	41
7.5.2 Member Function Documentation	41
7.5.2.1 BendTwistConstraintDeviation()	41
7.5.2.2 CalculateCylinderPositions()	41
7.5.2.3 DarbouxSignFactor()	42
7.5.2.4 DiscreteDarbouxVector()	42
7.5.2.5 EmbeddedVector()	43
7.5.2.6 GetGaussianRandomNumber()	43
7.5.2.7 ImaginaryPart()	44
7.5.2.8 MatrixVectorMultiplication()	44
7.5.2.9 QuaternionConversionFromBSM()	45
7.5.2.10 QuaternionConversionToBSM()	45
7.5.2.11 QuaternionLength()	45
7.5.2.12 RandomUnitQuaternion()	46
7.5.2.13 RodElementLength()	46
7.5.2.14 RodElementLengthDeviation()	47
7.5.2.15 SquaredNorm()	47
7.5.2.16 StretchConstraintDeviation()	47
7.5.2.17 UpdateDirectors()	48
7.5.2.18 VectorLength()	49
7.6 GuidewireSim.ObjectSetter Class Reference	49
7.6.1 Detailed Description	50
7.6.2 Member Function Documentation	50
7.6.2.1 Awake()	50
7.6.2.2 SetCylinderOrientations()	50
7.6.2.3 SetCylinderPositions()	50

7.6.2.4 SetSpherePositions()	52
7.6.3 Member Data Documentation	52
7.6.3.1 mathHelper	52
7.7 GuidewireSim.PredictionStep Class Reference	52
7.7.1 Detailed Description	53
7.7.2 Member Function Documentation	53
7.7.2.1 Awake()	53
7.7.2.2 PredictAngularVelocities()	53
7.7.2.3 PredictCylinderOrientations()	54
7.7.2.4 PredictSpherePositions()	54
7.7.2.5 PredictSphereVelocities()	55
7.7.3 Member Data Documentation	55
7.7.3.1 mathHelper	56
7.8 GuidewireSim.SimulationLoop Class Reference	56
7.8.1 Detailed Description	58
7.8.2 Member Function Documentation	58
7.8.2.1 AdoptCalculations()	58
7.8.2.2 Awake()	58
7.8.2.3 FixedUpdate()	58
7.8.2.4 PerformConstraintSolvingStep()	59
7.8.2.5 PerformInitializationStep()	59
7.8.2.6 PerformPredictionStep()	59
7.8.2.7 PerformSimulationLoop()	60
7.8.2.8 PerformUpdateStep()	60
7.8.2.9 Start()	60
7.8.3 Member Data Documentation	60
7.8.3.1 constraintSolvingStep	60
7.8.3.2 cylinderAngularVelocities	60
7.8.3.3 cylinderExternalTorques	61
7.8.3.4 cylinderOrientationPredictions	61
7.8.3.5 cylinderOrientations	61
7.8.3.6 cylinderPositions	61
7.8.3.7 cylinders	61
7.8.3.8 cylinderScalarWeights	62
7.8.3.9 directors	62
7.8.3.10 discreteRestDarbouxVectors	62
7.8.3.11 inertiaTensor	62
7.8.3.12 initializationStep	63
7.8.3.13 inverseInertiaTensor	63
7.8.3.14 mathHelper	63
7.8.3.15 objectSetter	63
7.8.3.16 predictionStep	63

7.8.3.17 rodElementLength . . . . .	63
7.8.3.18 solveBendTwistConstraints . . . . .	64
7.8.3.19 solveStretchConstraints . . . . .	64
7.8.3.20 sphereExternalForces . . . . .	64
7.8.3.21 sphereInverseMasses . . . . .	64
7.8.3.22 spherePositionPredictions . . . . .	64
7.8.3.23 spherePositions . . . . .	64
7.8.3.24 spheres . . . . .	65
7.8.3.25 sphereVelocities . . . . .	65
7.8.3.26 updateStep . . . . .	65
7.8.3.27 worldSpaceBasis . . . . .	65
7.8.4 Property Documentation . . . . .	65
7.8.4.1 ConstraintSolverSteps . . . . .	65
7.8.4.2 CylinderCount . . . . .	66
7.8.4.3 ExecuteSingleLoopTest . . . . .	66
7.8.4.4 SpheresCount . . . . .	66
7.9 GuidewireSim.StressTestPerformer Class Reference . . . . .	66
7.9.1 Detailed Description . . . . .	67
7.9.2 Member Function Documentation . . . . .	67
7.9.2.1 Awake() . . . . .	67
7.9.2.2 PerformStressTestOne() . . . . .	67
7.9.2.3 PerformStressTests() . . . . .	67
7.9.2.4 Start() . . . . .	68
7.9.3 Member Data Documentation . . . . .	68
7.9.3.1 doStressTestOne . . . . .	68
7.9.3.2 simulationLoop . . . . .	68
7.10 GuidewireSim.TorqueTestPerformer Class Reference . . . . .	68
7.10.1 Detailed Description . . . . .	69
7.10.2 Member Function Documentation . . . . .	69
7.10.2.1 Awake() . . . . .	69
7.10.2.2 PerformTorqueTestOne() . . . . .	69
7.10.2.3 PerformTorqueTests() . . . . .	69
7.10.2.4 PerformTorqueTestThree() . . . . .	69
7.10.2.5 PerformTorqueTestTwo() . . . . .	70
7.10.2.6 Start() . . . . .	70
7.10.3 Member Data Documentation . . . . .	70
7.10.3.1 doTorqueTestOne . . . . .	70
7.10.3.2 doTorqueTestThree . . . . .	71
7.10.3.3 doTorqueTestTwo . . . . .	71
7.10.3.4 pullTorque . . . . .	71
7.10.3.5 simulationLoop . . . . .	71
7.11 UnitTest_SolveBendTwistConstraint Class Reference . . . . .	71

7.11.1 Detailed Description	72
7.11.2 Member Function Documentation	72
7.11.2.1 PerformUnitTests()	72
7.11.2.2 Test_SolveBendTwistConstraint()	72
7.11.3 Member Data Documentation	73
7.11.3.1 constraintSolverSteps	73
7.11.3.2 sampleSize	73
7.12 UnitTest_SolveStretchConstraint Class Reference	73
7.12.1 Detailed Description	74
7.12.2 Member Function Documentation	74
7.12.2.1 PerformUnitTests()	74
7.12.2.2 PickRandomPositions()	74
7.12.2.3 Test_SolveStretchConstraint()	75
7.12.3 Member Data Documentation	75
7.12.3.1 constraintSolverSteps	75
7.12.3.2 maximalDistanceOffset	76
7.12.3.3 rodElementLength	76
7.12.3.4 sampleSize	76
7.13 GuidewireSim.UpdateStep Class Reference	76
7.13.1 Detailed Description	77
7.13.2 Member Function Documentation	77
7.13.2.1 Awake()	77
7.13.2.2 UpdateCylinderAngularVelocities()	77
7.13.2.3 UpdateCylinderOrientations()	78
7.13.2.4 UpdateSpherePositions()	78
7.13.2.5 UpdateSphereVelocities()	78
7.13.3 Member Data Documentation	79
7.13.3.1 mathHelper	79
<b>8 File Documentation</b>	<b>81</b>
8.1 ConstraintSolvingStep.cs File Reference	81
8.1.1 Typedef Documentation	81
8.1.1.1 BSM	81
8.2 DirectorsDrawer.cs File Reference	81
8.3 ForceTestPerformer.cs File Reference	82
8.4 InitializationStep.cs File Reference	82
8.4.1 Typedef Documentation	82
8.4.1.1 BSM	82
8.5 MathHelper.cs File Reference	82
8.5.1 Typedef Documentation	83
8.5.1.1 BSM	83
8.6 ObjectSetter.cs File Reference	83

8.6.1 Typedef Documentation . . . . .	83
8.6.1.1 BSM . . . . .	83
8.7 PredictionStep.cs File Reference . . . . .	83
8.7.1 Typedef Documentation . . . . .	84
8.7.1.1 BSM . . . . .	84
8.8 SimulationLoop.cs File Reference . . . . .	84
8.8.1 Typedef Documentation . . . . .	84
8.8.1.1 BSM . . . . .	84
8.9 StressTestPerformer.cs File Reference . . . . .	84
8.10 TorqueTestPerformer.cs File Reference . . . . .	85
8.11 UnitTest_SolveBendTwistConstraint.cs File Reference . . . . .	85
8.11.1 Typedef Documentation . . . . .	85
8.11.1.1 BSM . . . . .	85
8.12 UnitTest_SolveStretchConstraint.cs File Reference . . . . .	85
8.12.1 Typedef Documentation . . . . .	86
8.12.1.1 BSM . . . . .	86
8.13 UpdateStep.cs File Reference . . . . .	86
8.13.1 Typedef Documentation . . . . .	86
8.13.1.1 BSM . . . . .	86
<b>Index</b>	<b>87</b>



# Chapter 1

## Requirements Traceability

**Member** [GuidewireSim.ConstraintSolvingStep.CorrectBendTwistPredictions](#) (int cylinderIndex, BSM.Quaternion[] cylinderOrientationPredictions)

The relevant entries of `cylinderOrientationPredictions` should be unit quaternions, i.e. have length approximately equal to one.

After the quaternion predictions got corrected, they should again be unit quaternions, i.e. have length approximately equal to one.

**Member** [GuidewireSim.ConstraintSolvingStep.CorrectStretchPredictions](#) (int sphereIndex, Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions)

The relevant entries of `cylinderOrientationPredictions` should be unit quaternions, i.e. have length approximately equal to one.

After the quaternion prediction got corrected, it should again be a unit quaternions, i.e. have length approximately equal to one.

**Member** [GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraint](#) (BSM.Quaternion orientationOne, BSM.Quaternion orientationTwo, Vector3 discreteRestDarbouxVector, float rodElementLength, out BSM.Quaternion deltaOrientationOne, out BSM.Quaternion deltaOrientationTwo, float inertiaWeightOne=1f, float inertiaWeightTwo=1f)

`orientationOne` and `orientationTwo` should be unit quaternions, i.e. have length approximately equal to one.

`rodElementLength` should be positive.

`inertiaWeightOne` and `inertiaWeightTwo` should be values between 0 and 1.

**Member** [GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraints](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] discreteRestDarbouxVectors, float rodElementLength)

`cylinderCount` should be at least one.

`rodElementLength` should be positive.

Executes the constraint solving step in bilateral interleaving order if [executeInBilateralOrder](#) and otherwise in naive order.

**Member** [GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraintsInBilateralOrder](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] discreteRestDarbouxVectors, float rodElementLength)

`cylinderCount` should be at least one.

`rodElementLength` should be positive.

**Member** [GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraintsInNaiveOrder](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] discreteRestDarbouxVectors, float rodElementLength)

`cylinderCount` should be at least one.

`rodElementLength` should be positive.

**Member [GuidewireSim.ConstraintSolvingStep.SolveStretchConstraint](#)** (Vector3 particlePositionOne, Vector3 particlePositionTwo, BSM.Quaternion orientation, BSM.Quaternion e\_3, float rodElementLength, out Vector3 deltaPositionOne, out Vector3 deltaPositionTwo, out BSM.Quaternion deltaOrientation, float inverseMassOne=1f, float inverseMassTwo=1f, float inertiaWeight=1f)

orientation should be a unit quaternions, i.e. have length approximately equal to one.

e\_3 should be a unit quaternions, i.e. have length approximately equal to one.

rodElementLength should be positive.

inverseMassOne, inverseMassTwo and inertiaWeight should be values between 0 and 1.

**Member [GuidewireSim.ConstraintSolvingStep.SolveStretchConstraints](#)** (Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions, int spheresCount, BSM.Quaternion[] worldSpaceBasis, float rodElementLength)

spheresCount should be at least one.

rodElementLength should be positive.

Executes the constraint solving step in bilateral interleaving order if [executeInBilateralOrder](#) and otherwise in naive order.

**Member [GuidewireSim.ConstraintSolvingStep.SolveStretchConstraintsInBilateralOrder](#)** (Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions, int spheresCount, float rodElementLength, BSM.Quaternion e\_3)

spheresCount should be at least one.

rodElementLength should be positive.

**Member [GuidewireSim.ConstraintSolvingStep.SolveStretchConstraintsInNaiveOrder](#)** (Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions, int spheresCount, float rodElementLength, BSM.Quaternion e\_3)

spheresCount should be at least one.

rodElementLength should be positive.

**Member [GuidewireSim.DirectorsDrawer.CalculateArrowHeadPositions](#)** (Vector3 startPosition, Vector3 endPosition)

arrowHeadPositions has a length of 4.

**Member [GuidewireSim.DirectorsDrawer.DrawArrowHeadLines](#)** (int directorIndex, Vector3 endPosition, Vector3[] arrowHeadPositions)

arrowHeadPositions has a length of 4.

**Member [GuidewireSim.InitializationStep.InitDiscreteRestDarbouxVectors](#)** (int cylinderCount, BSM.Quaternion[] cylinderOrientations, out Vector3[] discreteRestDarbouxVectors, float rodElementLength)

cylinderCount should be at least one.

rodElementLength should be positive.

**Member [GuidewireSim.InitializationStep.InitSpherePositions](#)** (GameObject[] spheres, int spheresCount, out Vector3[] spherePositions)

spheresCount should be at least one.

**Member [GuidewireSim.MathHelper.MatrixVectorMultiplication](#)** (float[,] matrix, Vector3 vector)

matrix must be a  $3 \times 3$  matrix.

**Member [GuidewireSim.MathHelper.RandomUnitQuaternion](#)** ()

The length of the drawn quaternion is approximately equal to one.

**Member [GuidewireSim.SimulationLoop.AdoptCalculations](#)** ()

Sets the positions of the GameObjects [spheres](#) to [spherePositions](#).

Calculates [cylinderPositions](#) based on [spherePositions](#).

Sets the positions of the GameObjects [cylinders](#) to [cylinderPositions](#).

Sets the rotations of the GameObjects [cylinders](#) to [cylinderOrientations](#).

**Member [GuidewireSim.SimulationLoop.FixedUpdate](#)** ()

Execute the simulation loop if and only if [ExecuteSingleLoopTest](#) is false.

**Member `GuidewireSim.SimulationLoop.PerformConstraintSolvingStep ()`**

Performs the constraint solving of every constraint `#solverStep` many times.

Solve stretch constraints, if and only if `solveStretchConstraints` is true.

Solve bend twist constraints, if and only if `solveBendTwistConstraints` is true.

If `solveStretchConstraints`, then `SpheresCount` is at least two.

If `solveStretchConstraints`, then `CylinderCount` is at least one.

If `solveBendTwistConstraints`, then `SpheresCount` is at least three.

If `solveBendTwistConstraints`, then `CylinderCount` is at least two.

If `solveStretchConstraints`, after the step is complete the deviation between the actual rod element length and the default (rest state) `rodElementLength` should be close to zero.

If `solveStretchConstraints`, after the step is complete the deviation of the stretch constraint to zero should be close to zero.

**Member `GuidewireSim.SimulationLoop.PerformInitializationStep ()`**

Set `SpheresCount` to the length of `spheres`.

Set `CylinderCount` to the length of `cylinders`.

Call every init method of `initializationStep`.

**Member `GuidewireSim.SimulationLoop.PerformPredictionStep ()`**

Predict the `sphereVelocities`.

Predict the `spherePositionPredictions`.

Predict the `cylinderAngularVelocities`.

Predict the `cylinderOrientationPredictions`.

**Member `GuidewireSim.SimulationLoop.PerformUpdateStep ()`**

Update `sphereVelocities`.

Update `spherePositions`.

Update `cylinderAngularVelocities`.

Update `cylinderOrientations`.

Update `directors`.

**Member `GuidewireSim.StressTestPerformer.PerformStressTestOne (float applyForceTime=1f)`**

Output a log message when no further forces are applied to the guidewire.

**Member `GuidewireSim.TorqueTestPerformer.PerformTorqueTestThree (Vector3 pullTorque, float applyTorqueTime=10f)`**

Output a log message when no further torques are applied to the guidewire.

**Member `GuidewireSim.TorqueTestPerformer.PerformTorqueTestTwo (Vector3 pullTorque, float applyTorqueTime=1f)`**

Output a log message when no further torques are applied to the guidewire.

**Member `UnitTest_SolveBendTwistConstraint.Test_SolveBendTwistConstraint (int iterations, BSM.Quaternion orientationOne, BSM.Quaternion orientationTwo, float rodElementLength, Vector3 discreteRestDarbouxVector, GuidewireSim.MathHelper mathHelper, GuidewireSim.ConstraintSolvingStep constraintSolvingStep)`**

`orientationOne` and `orientationTwo` are still unit quaternions at the end of the test.

The deviation between the bend twist constraint and zero is lower than a reasonable tolerance, i.e. close to zero., which means that the algorithm of `SolveBendTwistConstraint()` converges towards the fulfillment of the bend twist constraint.

**Member `UnitTest_SolveStretchConstraint.PickRandomPositions (out Vector3 particlePositionOne, out Vector3 particlePositionTwo)`**

Picks the first particle position uniformly distributed so that  $x, y, z \in [-5, 5]$ .

Picks a distance between the two particles that is uniformly distributed in the interval  $[rodElementLength - maximalDistanceOffset, rodElementLength + maximalDistanceOffset]$ .

Picks the second particle position uniformly distributed on the surface of the sphere with center `particlePositionOne` and radius `startDistance`.

**Member** `UnitTest_SolveStretchConstraint.Test_SolveStretchConstraint` (int iterations, Vector3 `particlePositionOne`, Vector3 `particlePositionTwo`, BSM.Quaternion `orientation`, `GuidewireSim.MathHelper.mathHelper`, `GuidewireSim.ConstraintSolvingStep.constraintSolvingStep`)

`orientation` is still a unit quaternion at the end of the test.

The deviation between the stretch constraint and zero is lower than the tolerance 0.1, which means that the algorithm of `SolveStretchConstraint()` converges towards the fulfillment of the stretch constraint.

The deviation between the actual distance of `particlePositionOne` and `particlePositionTwo` and the rest rod element length is lower than a reasonable tolerance, i.e. close to zero.

## Chapter 2

# Namespace Index

### 2.1 Packages

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

<a href="#">GuidewireSim</a> . . . . .	13
--	----



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

MonoBehaviour	
GuidewireSim.ConstraintSolvingStep . . . . .	15
GuidewireSim.DirectorsDrawer . . . . .	25
GuidewireSim.ForceTestPerformer . . . . .	29
GuidewireSim.InitializationStep . . . . .	32
GuidewireSim.MathHelper . . . . .	40
GuidewireSim.ObjectSetter . . . . .	49
GuidewireSim.PredictionStep . . . . .	52
GuidewireSim.SimulationLoop . . . . .	56
GuidewireSim.StressTestPerformer . . . . .	66
GuidewireSim.TorqueTestPerformer . . . . .	68
GuidewireSim.UpdateStep . . . . .	76
UnitTest_SolveBendTwistConstraint . . . . .	71
UnitTest_SolveStretchConstraint . . . . .	73



## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">GuidewireSim.ConstraintSolvingStep</a>	15
<a href="#">GuidewireSim.DirectorsDrawer</a>	25
<a href="#">GuidewireSim.ForceTestPerformer</a>	29
<a href="#">GuidewireSim.InitializationStep</a>	32
<a href="#">GuidewireSim.MathHelper</a>	40
<a href="#">GuidewireSim.ObjectSetter</a>	49
<a href="#">GuidewireSim.PredictionStep</a>	52
<a href="#">GuidewireSim.SimulationLoop</a>	56
<a href="#">GuidewireSim.StressTestPerformer</a>	66
<a href="#">GuidewireSim.TorqueTestPerformer</a>	68
<a href="#">UnitTest_SolveBendTwistConstraint</a>	71
<a href="#">UnitTest_SolveStretchConstraint</a>	73
<a href="#">GuidewireSim.UpdateStep</a>	76



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">ConstraintSolvingStep.cs</a>	81
<a href="#">DirectorsDrawer.cs</a>	81
<a href="#">ForceTestPerformer.cs</a>	82
<a href="#">InitializationStep.cs</a>	82
<a href="#">MathHelper.cs</a>	82
<a href="#">ObjectSetter.cs</a>	83
<a href="#">PredictionStep.cs</a>	83
<a href="#">SimulationLoop.cs</a>	84
<a href="#">StressTestPerformer.cs</a>	84
<a href="#">TorqueTestPerformer.cs</a>	85
<a href="#">UnitTest_SolveBendTwistConstraint.cs</a>	85
<a href="#">UnitTest_SolveStretchConstraint.cs</a>	85
<a href="#">UpdateStep.cs</a>	86



## Chapter 6

# Namespace Documentation

### 6.1 GuidewireSim Namespace Reference

#### Classes

- class [ConstraintSolvingStep](#)
- class [DirectorsDrawer](#)
- class [ForceTestPerformer](#)
- class [InitializationStep](#)
- class [MathHelper](#)
- class [ObjectSetter](#)
- class [PredictionStep](#)
- class [SimulationLoop](#)
- class [StressTestPerformer](#)
- class [TorqueTestPerformer](#)
- class [UpdateStep](#)

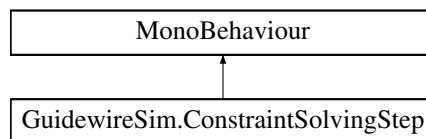


## Chapter 7

# Class Documentation

### 7.1 GuidewireSim.ConstraintSolvingStep Class Reference

Inheritance diagram for GuidewireSim.ConstraintSolvingStep:



#### Public Member Functions

- void [SolveStretchConstraints](#) (Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions, int spheresCount, BSM.Quaternion[] worldSpaceBasis, float rodElementLength)
- void [SolveBendTwistConstraints](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] discreteRestDarbouxVectors, float rodElementLength)
- void [SolveStretchConstraint](#) (Vector3 particlePositionOne, Vector3 particlePositionTwo, BSM.Quaternion orientation, BSM.Quaternion e\_3, float rodElementLength, out Vector3 [deltaPositionOne](#), out Vector3 [deltaPositionTwo](#), out BSM.Quaternion [deltaOrientation](#), float inverseMassOne=1f, float inverseMassTwo=1f, float inertiaWeight=1f)
- void [SolveBendTwistConstraint](#) (BSM.Quaternion orientationOne, BSM.Quaternion orientationTwo, Vector3 discreteRestDarbouxVector, float rodElementLength, out BSM.Quaternion [deltaOrientationOne](#), out BSM.Quaternion [deltaOrientationTwo](#), float inertiaWeightOne=1f, float inertiaWeightTwo=1f)

#### Private Member Functions

- void [Awake](#) ()
- void [SolveStretchConstraintsInBilateralOrder](#) (Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions, int spheresCount, float rodElementLength, BSM.Quaternion e\_3)
- void [SolveStretchConstraintsInNaiveOrder](#) (Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions, int spheresCount, float rodElementLength, BSM.Quaternion e\_3)
- void [SolveBendTwistConstraintsInBilateralOrder](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] discreteRestDarbouxVectors, float rodElementLength)
- void [SolveBendTwistConstraintsInNaiveOrder](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] discreteRestDarbouxVectors, float rodElementLength)
- void [CorrectStretchPredictions](#) (int sphereIndex, Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions)
- void [CorrectBendTwistPredictions](#) (int cylinderIndex, BSM.Quaternion[] cylinderOrientationPredictions)

## Private Attributes

- [MathHelper mathHelper](#)  
The component [MathHelper](#) that provides math related helper functions.
- Vector3 [deltaPositionOne](#) = new Vector3()  
The correction of *particlePositionOne* in method [SolveStretchConstraint\(\)](#).
- Vector3 [deltaPositionTwo](#) = new Vector3()  
The correction of *particlePositionTwo* in method [SolveStretchConstraint\(\)](#).
- BSM.Quaternion [deltaOrientation](#) = new BSM.Quaternion()  
The correction of *orientation* in method [SolveStretchConstraint\(\)](#).
- BSM.Quaternion [deltaOrientationOne](#) = new BSM.Quaternion()  
The correction of *orientationOne* in method [SolveBendTwistConstraint\(\)](#).
- BSM.Quaternion [deltaOrientationTwo](#) = new BSM.Quaternion()  
The correction of *orientationTwo* in method [SolveBendTwistConstraint\(\)](#).
- bool [executeInBilateralOrder](#) = false  
Whether to solve both constraints in bilateral interleaving order. Naive order is used when false.

### 7.1.1 Detailed Description

This class executes and implements various algorithms of the constraint solving step of the algorithm and manages all coherent data.

### 7.1.2 Member Function Documentation

#### 7.1.2.1 Awake()

```
void GuidewireSim.ConstraintSolvingStep.Awake ( ) [private]
```

#### 7.1.2.2 CorrectBendTwistPredictions()

```
void GuidewireSim.ConstraintSolvingStep.CorrectBendTwistPredictions (
    int cylinderIndex,
    BSM.Quaternion[] cylinderOrientationPredictions ) [private]
```

Corrects the predictions of the bend twist constraint by adding *deltaOrientationOne* and *deltaOrientationTwo*.

#### Note

Note that *deltaOrientationOne* and *deltaOrientationTwo* may have a length unequal one by definition.

## Parameters

<i>cylinderIndex</i>	The index of the first element of <i>cylinderOrientationPredictions</i> that gets corrected.
<i>cylinderOrientationPredictions</i>	The array of orientation predictions of which two quaternions get corrected in this method.

**Requirements** The relevant entries of *cylinderOrientationPredictions* should be unit quaternions, i.e. have length approximately equal to one.

After the quaternion predictions got corrected, they should again be unit quaternions, i.e. have length approximately equal to one.

## 7.1.2.3 CorrectStretchPredictions()

```
void GuidewireSim.ConstraintSolvingStep.CorrectStretchPredictions (
    int sphereIndex,
    Vector3[] spherePositionPredictions,
    BSM.Quaternion[] cylinderOrientationPredictions ) [private]
```

Corrects the predictions of the stretch constraint by adding *deltaPositionOne*, *deltaPositionTwo* and *deltaOrientation*.

## Note

Note that *deltaOrientation* may has a length unequal one by definition.

## Parameters

<i>sphereIndex</i>	The index of the first element of <i>spherePositionPredictions</i> that gets corrected.
<i>spherePositionPredictions</i>	The array of position predictions of which two positions get corrected in this method.
<i>cylinderOrientationPredictions</i>	The array of orientation predictions of which one quaternions gets corrected in this method.

**Requirements** The relevant entries of *cylinderOrientationPredictions* should be unit quaternions, i.e. have length approximately equal to one.

After the quaternion prediction got corrected, it should again be a unit quaternions, i.e. have length approximately equal to one.

## 7.1.2.4 SolveBendTwistConstraint()

```
void GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraint (
    BSM.Quaternion orientationOne,
```

```

BSM.Quaternion orientationTwo,
Vector3 discreteRestDarbouxVector,
float rodElementLength,
out BSM.Quaternion deltaOrientationOne,
out BSM.Quaternion deltaOrientationTwo,
float inertiaWeightOne = 1f,
float inertiaWeightTwo = 1f )

```

Solves the bend twist constraint by calculating the corrections `deltaOrientationOne` and `deltaOrientationTwo`.

#### Note

To be more precise, the bend twist constraint is not solved but minimized, i.e. the constraint will after correcting with the corrections be closer to zero.

#### Parameters

	<i>orientationOne</i>	The first orientation quaternion prediction of the orientation element to be corrected.
	<i>orientationTwo</i>	The second orientation quaternion prediction of the orientation element to be corrected.
	<i>discreteRestDarbouxVector</i>	The discrete Darboux Vector at the rest configuration, i.e. at frame 0.
	<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.
out	<i>deltaOrientationOne</i>	The correction of <code>orientationOne</code> .
out	<i>deltaOrientationTwo</i>	The correction of <code>orientationTwo</code> .
	<i>inertiaWeightOne</i>	The inertia weight scalar for <code>orientationOne</code> . Use a value of 1 for a moving orientation and 0 for a fixed orientation.
	<i>inertiaWeightTwo</i>	The inertia weight scalar for <code>orientationTwo</code> . Use a value of 1 for a moving orientation and 0 for a fixed orientation.

**Requirements** `orientationOne` and `orientationTwo` should be unit quaternions, i.e. have length approximately equal to one.

`rodElementLength` should be positive.

`inertiaWeightOne` and `inertiaWeightTwo` should be values between 0 and 1.

#### 7.1.2.5 SolveBendTwistConstraints()

```

void GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraints (
    BSM.Quaternion[] cylinderOrientationPredictions,
    int cylinderCount,
    Vector3[] discreteRestDarbouxVectors,
    float rodElementLength )

```

Is responsible for executing one iteration of the constraint solving step for the bend twist constraint, i.e. corrects each orientation prediction one time.

#### Note

Can be executed in naive order or bilateral interleaving order.

## Parameters

<i>cylinderOrientationPredictions</i>	The array of orientation predictions that get corrected in this step.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>discreteRestDarbouxVectors</i>	The array of all discrete Darboux Vectors at the rest configuration, i.e. at frame 0. Has (n-1) elements, if n is the number of orientations of the guidewire, because the darboux vector is taken of two adjacent orientations.
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.

**Requirements** *cylinderCount* should be at least one.

*rodElementLength* should be positive.

Executes the constraint solving step in bilateral interleaving order if [executeInBilateralOrder](#) and otherwise in naive order.

## 7.1.2.6 SolveBendTwistConstraintsInBilateralOrder()

```
void GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraintsInBilateralOrder (
    BSM.Quaternion[] cylinderOrientationPredictions,
    int cylinderCount,
    Vector3[] discreteRestDarbouxVectors,
    float rodElementLength ) [private]
```

Is responsible for executing one iteration of the constraint solving step for the bend twist constraint in bilateral order, i.e. corrects each orientation prediction one time.

## Note

You can read more about bilateral order in the 2016 paper "Position and Orientation Based Cosserat Rods".

## Attention

The index shifting of this algorithm is not easy to understand, but got deeply tested.

## Parameters

<i>cylinderOrientationPredictions</i>	The array of orientation predictions that get corrected in this step.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>discreteRestDarbouxVectors</i>	The array of all discrete Darboux Vectors at the rest configuration, i.e. at frame 0. Has (n-1) elements, if n is the number of orientations of the guidewire, because the darboux vector is taken of two adjacent orientations.
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.

**Requirements** `cylinderCount` should be at least one.  
`rodElementLength` should be positive.

#### 7.1.2.7 SolveBendTwistConstraintsInNaiveOrder()

```
void GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraintsInNaiveOrder (
    BSM.Quaternion[] cylinderOrientationPredictions,
    int cylinderCount,
    Vector3[] discreteRestDarbouxVectors,
    float rodElementLength ) [private]
```

Is responsible for executing one iteration of the constraint solving step for the bend twist constraint in naive order, i.e. corrects each orientation prediction one time.

#### Note

Naive order means the predictions are updated beginning from one end of the guidewire to the other end of the guidewire.

#### Parameters

<i>cylinderOrientationPredictions</i>	The array of orientation predictions that get corrected in this step.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>discreteRestDarbouxVectors</i>	The array of all discrete Darboux Vectors at the rest configuration, i.e. at frame 0. Has (n-1) elements, if n is the number of orientations of the guidewire, because the darboux vector is taken of two adjacent orientations.
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.

**Requirements** `cylinderCount` should be at least one.  
`rodElementLength` should be positive.

#### 7.1.2.8 SolveStretchConstraint()

```
void GuidewireSim.ConstraintSolvingStep.SolveStretchConstraint (
    Vector3 particlePositionOne,
    Vector3 particlePositionTwo,
    BSM.Quaternion orientation,
    BSM.Quaternion e_3,
    float rodElementLength,
    out Vector3 deltaPositionOne,
    out Vector3 deltaPositionTwo,
    out BSM.Quaternion deltaOrientation,
    float inverseMassOne = 1f,
```

```
float inverseMassTwo = 1f,
float inertiaWeight = 1f )
```

Solves the stretch constraint by calculating the corrections `deltaPositionOne` and `deltaPositionTwo`, `deltaOrientation`.

#### Note

To be more precise, the stretch constraint is not solved but minimized, i.e. the constraint will after correcting with the corrections be closer to zero.

#### Parameters

	<i>particlePositionOne</i>	The first particle position prediction of the centerline element to be corrected.
	<i>particlePositionTwo</i>	The second particle position prediction of the centerline element to be corrected.
	<i>orientation</i>	The orientation quaternion prediction of the orientation element between the particle positions to be corrected.
	<i>e_3</i>	The third basis vector of the world space coordinates embedded as a quaternion with scalar part 0.
	<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.
out	<i>deltaPositionOne</i>	The correction of <i>particlePositionOne</i> .
out	<i>deltaPositionTwo</i>	The correction of <i>particlePositionTwo</i> .
out	<i>deltaOrientation</i>	The correction of <i>orientation</i> .
	<i>inverseMassOne</i>	The inverse mass scalar for <i>particlePositionOne</i> . Use a value of 1 for a moving particle and 0 for a fixed particle.
	<i>inverseMassTwo</i>	The inverse mass scalar for <i>particlePositionTwo</i> . Use a value of 1 for a moving particle and 0 for a fixed particle.
	<i>inertiaWeight</i>	The inertia weight scalar for <i>orientation</i> . Use a value of 1 for a moving orientation and 0 for a fixed orientation.

**Requirements** `orientation` should be a unit quaternions, i.e. have length approximately equal to one.  
`e_3` should be a unit quaternions, i.e. have length approximately equal to one.  
`rodElementLength` should be positive.  
`inverseMassOne`, `inverseMassTwo` and `inertiaWeight` should be values between 0 and 1.

#### 7.1.2.9 SolveStretchConstraints()

```
void GuidewireSim.ConstraintSolvingStep.SolveStretchConstraints (
    Vector3[] spherePositionPredictions,
    BSM.Quaternion[] cylinderOrientationPredictions,
    int spheresCount,
    BSM.Quaternion[] worldSpaceBasis,
    float rodElementLength )
```

Is responsible for executing one iteration of the constraint solving step for the stretch constraint, i.e. corrects each particle position prediction one time and also each orientation prediction one time.

#### Note

Can be executed in naive order or bilateral interleaving order.

## Parameters

<i>spherePositionPredictions</i>	The array of position predictions that get corrected in this step.
<i>cylinderOrientationPredictions</i>	The array of orientation predictions that get corrected in this step.
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>worldSpaceBasis</i>	The three basis vectors of the world coordinate system as embedded quaternions with scalar part 0.
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.

**Requirements** *spheresCount* should be at least one.

*rodElementLength* should be positive.

Executes the constraint solving step in bilateral interleaving order if [executeInBilateralOrder](#) and otherwise in naive order.

### 7.1.2.10 SolveStretchConstraintsInBilateralOrder()

```
void GuidewireSim.ConstraintSolvingStep.SolveStretchConstraintsInBilateralOrder (
    Vector3[] spherePositionPredictions,
    BSM.Quaternion[] cylinderOrientationPredictions,
    int spheresCount,
    float rodElementLength,
    BSM.Quaternion e_3 ) [private]
```

Executes one iteration of the constraint solving step for the stretch constraint in bilateral order, i.e. corrects each particle position prediction one time and also each orientation prediction one time.

#### Note

You can read more about bilateral order in the 2016 paper "Position and Orientation Based Cosserat Rods".

#### Attention

The index shifting of this algorithm is not easy to understand, but got deeply tested.

## Parameters

<i>spherePositionPredictions</i>	The array of position predictions that get corrected in this step.
<i>cylinderOrientationPredictions</i>	The array of orientation predictions that get corrected in this step.
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.
<i>e_3</i>	The third basis vector of the world coordinate system as embedded quaternions with scalar part 0.

**Requirements** `spheresCount` should be at least one.  
`rodElementLength` should be positive.

### 7.1.2.11 SolveStretchConstraintsInNaiveOrder()

```
void GuidewireSim.ConstraintSolvingStep.SolveStretchConstraintsInNaiveOrder (
    Vector3[] spherePositionPredictions,
    BSM.Quaternion[] cylinderOrientationPredictions,
    int spheresCount,
    float rodElementLength,
    BSM.Quaternion e_3 ) [private]
```

Executes one iteration of the constraint solving step for the stretch constraint in naive order, i.e. corrects each particle position prediction one time and also each orientation prediction one time.

#### Note

Naive order means the predictions are updated beginning from one end of the guidewire to the other end of the guidewire.

#### Parameters

<i>spherePositionPredictions</i>	The array of position predictions that get corrected in this step.
<i>cylinderOrientationPredictions</i>	The array of orientation predictions that get corrected in this step.
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.
<i>e_3</i>	The third basis vector of the world coordinate system as embedded quaternions with scalar part 0.

**Requirements** `spheresCount` should be at least one.  
`rodElementLength` should be positive.

## 7.1.3 Member Data Documentation

### 7.1.3.1 deltaOrientation

```
BSM.Quaternion GuidewireSim.ConstraintSolvingStep.deltaOrientation = new BSM.Quaternion()
[private]
```

The correction of `orientation` in method [SolveStretchConstraint\(\)](#).

### 7.1.3.2 deltaOrientationOne

```
BSM.Quaternion GuidewireSim.ConstraintSolvingStep.deltaOrientationOne = new BSM.Quaternion()  
[private]
```

The correction of `orientationOne` in method [SolveBendTwistConstraint\(\)](#).

### 7.1.3.3 deltaOrientationTwo

```
BSM.Quaternion GuidewireSim.ConstraintSolvingStep.deltaOrientationTwo = new BSM.Quaternion()  
[private]
```

The correction of `orientationTwo` in method [SolveBendTwistConstraint\(\)](#).

### 7.1.3.4 deltaPositionOne

```
Vector3 GuidewireSim.ConstraintSolvingStep.deltaPositionOne = new Vector3() [private]
```

The correction of `particlePositionOne` in method [SolveStretchConstraint\(\)](#).

### 7.1.3.5 deltaPositionTwo

```
Vector3 GuidewireSim.ConstraintSolvingStep.deltaPositionTwo = new Vector3() [private]
```

The correction of `particlePositionTwo` in method [SolveStretchConstraint\(\)](#).

### 7.1.3.6 executeInBilateralOrder

```
bool GuidewireSim.ConstraintSolvingStep.executeInBilateralOrder = false [private]
```

Whether to solve both constraints in bilateral interleaving order. Naive order is used when false.

### 7.1.3.7 mathHelper

```
MathHelper GuidewireSim.ConstraintSolvingStep.mathHelper [private]
```

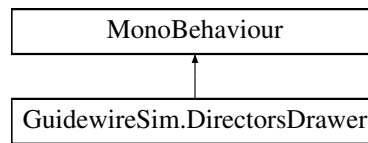
The component [MathHelper](#) that provides math related helper functions.

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

- [ConstraintSolvingStep.cs](#)

## 7.2 GuidewireSim.DirectorsDrawer Class Reference

Inheritance diagram for GuidewireSim.DirectorsDrawer:



### Private Member Functions

- void [Awake](#) ()
- void [Update](#) ()
- void [DrawDirectors](#) (Vector3[] cylinderPositions, Vector3[] directors)
- Vector3[] [CalculateArrowHeadPositions](#) (Vector3 startPosition, Vector3 endPosition)
- void [DrawArrowHeadLines](#) (int directorIndex, Vector3 endPosition, Vector3[] arrowHeadPositions)
- void [DrawArrowHeadConnectionLines](#) (int directorIndex, Vector3[] arrowHeadPositions)

### Private Attributes

- [SimulationLoop](#) simulationLoop  
*The component [SimulationLoop](#).*
- float [scaleFactor](#)  
*The scale factor that gets multiplied to the length of the respective director.*
- float [arrowHeadAngle](#)  
*The angle spread of the arrow head.*
- float [arrowHeadPercentage](#)  
*The percentage of the length of the arrow that the arrow head covers.*
- Color [directorOneColor](#) = Color.green  
*The color that the lines representing the first director are drawn with.*
- Color [directorTwoColor](#) = Color.blue  
*The color that the lines representing the second director are drawn with.*
- Color [directorThreeColor](#) = Color.red  
*The color that the lines representing the third director are drawn with.*
- Color[] [directorColors](#) = new Color[3] {Color.red, Color.green, Color.blue}

### 7.2.1 Detailed Description

This class represents each orientation by drawing all of its directors as arrows in each frame.

### 7.2.2 Member Function Documentation

### 7.2.2.1 Awake()

```
void GuidewireSim.DirectorsDrawer.Awake ( ) [private]
```

### 7.2.2.2 CalculateArrowHeadPositions()

```
Vector3 [ ] GuidewireSim.DirectorsDrawer.CalculateArrowHeadPositions (
    Vector3 startPosition,
    Vector3 endPosition ) [private]
```

Calculates the end position of each line of each arrow head. E.g. an arrow head consists of four lines, each of them starting at `endPosition` and spreading in different directions to form the shape of an arrow tip.

#### Parameters

<i>startPosition</i>	The start position of the director, i.e. the position of the orientation.
<i>endPosition</i>	The position of the tip of the arrow head.

#### Returns

The end positions of the four lines that form the arrow head.

**Requirements** `arrowHeadPositions` has a length of 4.

### 7.2.2.3 DrawArrowHeadConnectionLines()

```
void GuidewireSim.DirectorsDrawer.DrawArrowHeadConnectionLines (
    int directorIndex,
    Vector3[] arrowHeadPositions ) [private]
```

Draws the four lines that connect the arrow head tips with each other. E.g. draws the line from `arrowHeadPositions 0` and `arrowHeadPositions 1`.

#### Parameters

<i>directorIndex</i>	The index of the director under consideration.
<i>arrowHeadPositions</i>	The end positions of the four lines that form the arrow head.

### 7.2.2.4 DrawArrowHeadLines()

```
void GuidewireSim.DirectorsDrawer.DrawArrowHeadLines (
    int directorIndex,
```

```
Vector3 endPosition,
Vector3[] arrowHeadPositions ) [private]
```

Draws the four lines that form the arrow head for the director that corresponds to `directorIndex`.

#### Parameters

<i>directorIndex</i>	The index of the director under consideration.
<i>endPosition</i>	The position of the tip of the arrow head.
<i>arrowHeadPositions</i>	The end positions of the four lines that form the arrow head.

**Requirements** `arrowHeadPositions` has a length of 4.

### 7.2.2.5 DrawDirectors()

```
void GuidewireSim.DirectorsDrawer.DrawDirectors (
    Vector3[] cylinderPositions,
    Vector3 directors[][] ) [private]
```

Draws the director basis of each orientation element as arrows.

#### Parameters

<i>cylinderPositions</i>	The center of mass of each cylinder, i.e. the position of each orientation element.
<i>directors</i>	The orthonormal basis of each orientation element / cylinder, also called directors.

### 7.2.2.6 Update()

```
void GuidewireSim.DirectorsDrawer.Update ( ) [private]
```

## 7.2.3 Member Data Documentation

### 7.2.3.1 arrowHeadAngle

```
float GuidewireSim.DirectorsDrawer.arrowHeadAngle [private]
```

The angle spread of the arrow head.

### 7.2.3.2 arrowHeadPercentage

```
float GuidewireSim.DirectorsDrawer.arrowHeadPercentage [private]
```

The percentage of the length of the arrow that the arrow head covers.

### 7.2.3.3 directorColors

```
Color [ ] GuidewireSim.DirectorsDrawer.directorColors = new Color[3] {Color.red, Color.green, Color.blue} [private]
```

The color that the lines representing the three directors are drawn with.

#### Note

The i-th director is drawn in the i-th Color.

### 7.2.3.4 directorOneColor

```
Color GuidewireSim.DirectorsDrawer.directorOneColor = Color.green [private]
```

The color that the lines representing the first director are drawn with.

### 7.2.3.5 directorThreeColor

```
Color GuidewireSim.DirectorsDrawer.directorThreeColor = Color.red [private]
```

The color that the lines representing the third director are drawn with.

### 7.2.3.6 directorTwoColor

```
Color GuidewireSim.DirectorsDrawer.directorTwoColor = Color.blue [private]
```

The color that the lines representing the second director are drawn with.

### 7.2.3.7 scaleFactor

```
float GuidewireSim.DirectorsDrawer.scaleFactor [private]
```

The scale factor that gets multiplied to the length of the respective director.

### 7.2.3.8 simulationLoop

```
SimulationLoop GuidewireSim.DirectorsDrawer.simulationLoop [private]
```

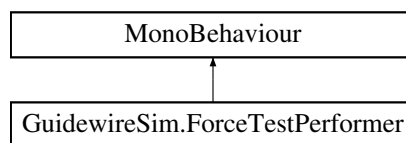
The component [SimulationLoop](#).

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

- [DirectorsDrawer.cs](#)

## 7.3 GuidewireSim.ForceTestPerformer Class Reference

Inheritance diagram for GuidewireSim.ForceTestPerformer:



### Private Member Functions

- void [Awake](#) ()
- void [Start](#) ()
- void [PerformForceTests](#) ()
- void [PerformForceTestOne](#) ()
- void [PerformForceTestTwo](#) ()
- IEnumerator [PerformForceTestThree](#) (Vector3 pullForce, float applyForceTime=1f)
- void [PerformForceTestFour](#) ()
- void [PerformSingleLoopTest](#) ()

### Private Attributes

- [SimulationLoop](#) [simulationLoop](#)  
The [SimulationLoop](#) component that executes all steps of the simulation loop.
- bool [doForceTestOne](#) = false  
Whether to run Force Test One. This test applies gravity to all spheres.
- bool [doForceTestTwo](#) = false  
Whether to run Force Test Two. This test applies an external force to one end of the guidewire.
- bool [doForceTestThree](#) = false
- bool [doForceTestFour](#) = false
- bool [doSingleLoopTest](#) = false
- Vector3 [pullForceTestThree](#) = new Vector3(0f, 3f, 0f)  
External force that is applied in Force Test Three.

### 7.3.1 Detailed Description

This class enables the user to test the impact of external forces with one button within the Unity inspector.

### 7.3.2 Member Function Documentation

#### 7.3.2.1 Awake()

```
void GuidewireSim.ForceTestPerformer.Awake ( ) [private]
```

#### 7.3.2.2 PerformForceTestFour()

```
void GuidewireSim.ForceTestPerformer.PerformForceTestFour ( ) [private]
```

Performs force test four. This test applies an external force to one end of the guidewire and the opposite force at the other end of the guidewire.

#### 7.3.2.3 PerformForceTestOne()

```
void GuidewireSim.ForceTestPerformer.PerformForceTestOne ( ) [private]
```

Performs force test one. This test applies gravity to all spheres.

#### 7.3.2.4 PerformForceTests()

```
void GuidewireSim.ForceTestPerformer.PerformForceTests ( ) [private]
```

Performs each Force Test whose respective serialized boolean is set to true in the Unity inspector.

#### 7.3.2.5 PerformForceTestThree()

```
IEnumerator GuidewireSim.ForceTestPerformer.PerformForceTestThree (
    Vector3 pullForce,
    float applyForceTime = 1f ) [private]
```

Performs force test three. This test applies an external force to one end of the guidewire for a fixed amount of time and then the opposite force at the same sphere for the same amount of time.

#### Parameters

<i>applyForceTime</i>	For how many seconds to apply the force to the particles.
-----------------------	---

#### 7.3.2.6 PerformForceTestTwo()

```
void GuidewireSim.ForceTestPerformer.PerformForceTestTwo ( ) [private]
```

Performs force test two. This test applies an external force to one end of the guidewire.

#### 7.3.2.7 PerformSingleLoopTest()

```
void GuidewireSim.ForceTestPerformer.PerformSingleLoopTest ( ) [private]
```

Performs the single loop test. This test shifts one end of the guidewire and runs the simulation for exactly one loop iteration to test constraint solving.

##### Note

Position of particle one stays at (0, 0, 0), while the section particle shifts to about (10, 2, 0). Expected result is that both particles move a bit towards each other and reestablish a distance of 10 between them.

#### 7.3.2.8 Start()

```
void GuidewireSim.ForceTestPerformer.Start ( ) [private]
```

### 7.3.3 Member Data Documentation

#### 7.3.3.1 doForceTestFour

```
bool GuidewireSim.ForceTestPerformer.doForceTestFour = false [private]
```

Whether to run Force Test Four. This test applies an external force to one end of the guidewire and the opposite force at the other end of the guidewire.

#### 7.3.3.2 doForceTestOne

```
bool GuidewireSim.ForceTestPerformer.doForceTestOne = false [private]
```

Whether to run Force Test One. This test applies gravity to all spheres.

### 7.3.3.3 doForceTestThree

```
bool GuidewireSim.ForceTestPerformer.doForceTestThree = false [private]
```

Whether to run Force Test Three. This test applies an external force to one end of the guidewire for a fixed amount of time and then the opposite force at the same sphere for the same amount of time.

### 7.3.3.4 doForceTestTwo

```
bool GuidewireSim.ForceTestPerformer.doForceTestTwo = false [private]
```

Whether to run Force Test Two. This test applies an external force to one end of the guidewire.

### 7.3.3.5 doSingleLoopTest

```
bool GuidewireSim.ForceTestPerformer.doSingleLoopTest = false [private]
```

Whether to run the Single Loop Test. This test shifts one end of the guidewire and runs the simulation for exactly one loop iteration to test constraint solving.

### 7.3.3.6 pullForceTestThree

```
Vector3 GuidewireSim.ForceTestPerformer.pullForceTestThree = new Vector3(0f, 3f, 0f) [private]
```

External force that is applied in Force Test Three.

### 7.3.3.7 simulationLoop

```
SimulationLoop GuidewireSim.ForceTestPerformer.simulationLoop [private]
```

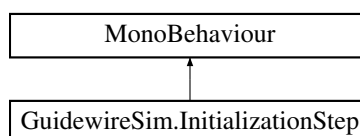
The [SimulationLoop](#) component that executes all steps of the simulation loop.

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

- [ForceTestPerformer.cs](#)

## 7.4 GuidewireSim.InitializationStep Class Reference

Inheritance diagram for GuidewireSim.InitializationStep:



## Public Member Functions

- void [InitSpherePositions](#) (GameObject[] spheres, int spheresCount, out Vector3[] spherePositions)
- void [InitSphereVelocities](#) (int spheresCount, out Vector3[] sphereVelocities)
- void [InitSphereInverseMasses](#) (int spheresCount, out float[] sphereInverseMasses)
- void [InitCylinderPositions](#) (int cylinderCount, Vector3[] spherePositions, out Vector3[] cylinderPositions)
- void [InitCylinderOrientations](#) (int cylinderCount, out BSM.Quaternion[] cylinderOrientations)
- void [InitDiscreteRestDarbouxVectors](#) (int cylinderCount, BSM.Quaternion[] cylinderOrientations, out Vector3[] discreteRestDarbouxVectors, float rodElementLength)
- void [InitCylinderAngularVelocities](#) (int cylinderCount, out Vector3[] cylinderAngularVelocities)
- void [InitCylinderScalarWeights](#) (int cylinderCount, out float[] cylinderScalarWeights)
- void [InitSphereExternalForces](#) (int spheresCount, out Vector3[] sphereExternalForces)
- void [InitSpherePositionPredictions](#) (int spheresCount, out Vector3[] spherePositionPredictions)
- void [InitCylinderOrientationPredictions](#) (int cylinderCount, out BSM.Quaternion[] cylinderOrientationPredictions)
- void [InitInertiaTensor](#) (out float[,] inertiaTensor)
- void [InitInverseInertiaTensor](#) (float[,] inertiaTensor, out float[,] inverseInertiaTensor)
- void [InitCylinderExternalTorques](#) (int cylinderCount, out Vector3[] cylinderExternalTorques)
- void [InitWorldSpaceBasis](#) (out BSM.Quaternion[] worldSpaceBasis)
- void [InitDirectors](#) (int cylinderCount, BSM.Quaternion[] worldSpaceBasis, out Vector3[][] directors)

## Private Member Functions

- void [Awake](#) ()

## Private Attributes

- [MathHelper](#) `mathHelper`  
*The component [MathHelper](#) that provides math related helper functions.*
- float [materialDensity](#) = 7860
- float [materialRadius](#) = 0.001f

### 7.4.1 Detailed Description

This class is responsible for initializing all data with their initial values of the simulation at the start of the simulation.

### 7.4.2 Member Function Documentation

#### 7.4.2.1 Awake()

```
void GuidewireSim.InitializationStep.Awake ( ) [private]
```

#### 7.4.2.2 InitCylinderAngularVelocities()

```
void GuidewireSim.InitializationStep.InitCylinderAngularVelocities (
    int cylinderCount,
    out Vector3[] cylinderAngularVelocities )
```

Initializes `cylinderAngularVelocities` with the default value of zero at the start of the simulation.

## Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
out	<i>cylinderAngularVelocities</i>	The angular velocity of the current frame of each orientation element/ cylinder.

**7.4.2.3 InitCylinderExternalTorques()**

```
void GuidewireSim.InitializationStep.InitCylinderExternalTorques (
    int cylinderCount,
    out Vector3[] cylinderExternalTorques )
```

Initializes *cylinderExternalTorques* with the default value of zero at the start of the simulation.

## Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
out	<i>cylinderExternalTorques</i>	The sum of all current external torques that are applied per orientation element/ cylinder.

**7.4.2.4 InitCylinderOrientationPredictions()**

```
void GuidewireSim.InitializationStep.InitCylinderOrientationPredictions (
    int cylinderCount,
    out BSM.Quaternion[] cylinderOrientationPredictions )
```

Initializes *cylinderOrientationPredictions* with the default value of (0f, 0f, 0f, 1f) which equals the quaternion identity at the start of the simulation.

## Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
out	<i>cylinderOrientationPredictions</i>	The prediction of the orientation of each cylinder at its center of mass.

**7.4.2.5 InitCylinderOrientations()**

```
void GuidewireSim.InitializationStep.InitCylinderOrientations (
    int cylinderCount,
    out BSM.Quaternion[] cylinderOrientations )
```

Initializes `cylinderOrientations` with the default value of (0f, 0f, 0f, 1f) which equals the quaternion identity at the start of the simulation.

#### Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientations</code> .
out	<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.

#### 7.4.2.6 InitCylinderPositions()

```
void GuidewireSim.InitializationStep.InitCylinderPositions (
    int cylinderCount,
    Vector3[] spherePositions,
    out Vector3[] cylinderPositions )
```

Initializes `cylinderPositions` as middle points of the positions of `spheres` at the start of the simulation.

#### Parameters

	<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <code>spherePositionPredictions</code> .
	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
out	<i>cylinderPositions</i>	The position/ center of mass of each cylinder.

#### 7.4.2.7 InitCylinderScalarWeights()

```
void GuidewireSim.InitializationStep.InitCylinderScalarWeights (
    int cylinderCount,
    out float[] cylinderScalarWeights )
```

Initializes `cylinderScalarWeights` with the default value of one at the start of the simulation.

#### Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
out	<i>cylinderScalarWeights</i>	The constant scalar weights of each orientation/ quaternion similar to <code>sphereInverseMasses</code> .

#### 7.4.2.8 InitDirectors()

```
void GuidewireSim.InitializationStep.InitDirectors (
```

```

    int cylinderCount,
    BSM.Quaternion[] worldSpaceBasis,
    out Vector3 directors[][] )

```

Initializes the `directors` array of arrays. The zero-th array defines all first directors of each director basis and so on.

#### Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
	<i>worldSpaceBasis</i>	The three basis vectors of the world coordinate system.
out	<i>directors</i>	The orthonormal basis of each orientation element / cylinder, also called directors.

#### Note

Example: The (i, j)th element holds the (i-1)th director of orientation element j.

#### 7.4.2.9 InitDiscreteRestDarbouxVectors()

```

void GuidewireSim.InitializationStep.InitDiscreteRestDarbouxVectors (
    int cylinderCount,
    BSM.Quaternion[] cylinderOrientations,
    out Vector3[] discreteRestDarbouxVectors,
    float rodElementLength )

```

Calculates the discrete darboux vector for each orientation pair (two adjacent orientations) at its rest configuration, i.e. at frame 0.

#### Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
	<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.
out	<i>discreteRestDarbouxVectors</i>	The array of all discrete Darboux Vectors at the rest configuration, i.e. at frame 0. Has (n-1) elements, if n is the number of orientations of the guidewire, because the darboux vector is taken of two adjacent orientations.
	<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.

**Requirements** `cylinderCount` should be at least one.  
`rodElementLength` should be positive.

#### Note

All cylinder orientations must be computed for frame 0 first.

**7.4.2.10 InitInertiaTensor()**

```
void GuidewireSim.InitializationStep.InitInertiaTensor (
    out float inertiaTensor[,] )
```

Initializes `inertiaTensor` so that all elements except the diagonal ones are zero. The first and second diagonal entry equal  $\rho * \pi * \frac{r^2}{4}$ , and the third diagonal entry equals  $\rho * \pi * \frac{r^2}{2}$ .

**Parameters**

out	<i>inertiaTensor</i>	The inertia tensor. Entries are approximates as in the CoRdE paper.
-----	----------------------	---

**7.4.2.11 InitInverseInertiaTensor()**

```
void GuidewireSim.InitializationStep.InitInverseInertiaTensor (
    float inertiaTensor[,],
    out float inverseInertiaTensor[,] )
```

Initializes `inverseInertiaTensor` as the inverse of `inertiaTensor`.

**Parameters**

	<i>inertiaTensor</i>	The inertia tensor. Entries are approximates as in the CoRdE paper.
out	<i>inverseInertiaTensor</i>	The inverse of <code>inertiaTensor</code> .

**7.4.2.12 InitSphereExternalForces()**

```
void GuidewireSim.InitializationStep.InitSphereExternalForces (
    int spheresCount,
    out Vector3[] sphereExternalForces )
```

Initializes `sphereExternalForces` with the default value of zero at the start of the simulation.

**Parameters**

	<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <code>spherePositionPredictions</code> .
out	<i>sphereExternalForces</i>	The sum of all current external forces that are applied per particle/ sphere.

**7.4.2.13 InitSphereInverseMasses()**

```
void GuidewireSim.InitializationStep.InitSphereInverseMasses (
```

```

    int spheresCount,
    out float[] sphereInverseMasses )

```

Initializes `sphereInverseMasses` with the default value of one at the start of the simulation.

#### Parameters

	<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <code>spherePositionPredictions</code> .
out	<i>sphereInverseMasses</i>	The constant inverse masses of each sphere.

#### 7.4.2.14 InitSpherePositionPredictions()

```

void GuidewireSim.InitializationStep.InitSpherePositionPredictions (
    int spheresCount,
    out Vector3[] spherePositionPredictions )

```

Initializes `spherePositionPredictions` with the default value of zero at the start of the simulation.

#### Parameters

	<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <code>spherePositionPredictions</code> .
out	<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere.

#### 7.4.2.15 InitSpherePositions()

```

void GuidewireSim.InitializationStep.InitSpherePositions (
    GameObject[] spheres,
    int spheresCount,
    out Vector3[] spherePositions )

```

Initializes `spherePositions` with the positions of `spheres` at the start of the simulation.

#### Parameters

	<i>spheres</i>	All spheres that are part of the guidewire.
	<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <code>spherePositionPredictions</code> .
out	<i>spherePositions</i>	The position at the current frame of each sphere.

**Requirements** `spheresCount` should be at least one.

#### 7.4.2.16 InitSphereVelocities()

```
void GuidewireSim.InitializationStep.InitSphereVelocities (
    int spheresCount,
    out Vector3[] sphereVelocities )
```

Initializes *sphereVelocities* with the default value of zero at the start of the simulation.

##### Parameters

	<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
out	<i>sphereVelocities</i>	The velocity of the current frame of each sphere.

##### Note

Velocities are set to zero at the start of the simulation.

#### 7.4.2.17 InitWorldSpaceBasis()

```
void GuidewireSim.InitializationStep.InitWorldSpaceBasis (
    out BSM.Quaternion[] worldSpaceBasis )
```

Initializes the world space basis vectors (1, 0, 0), (0, 1, 0), (0, 0, 1) as embedded quaternions with scalar part zero.

##### Parameters

out	<i>worldSpaceBasis</i>	The three basis vectors of the world coordinate system.
-----	------------------------	---

### 7.4.3 Member Data Documentation

#### 7.4.3.1 materialDensity

```
float GuidewireSim.InitializationStep.materialDensity = 7860 [private]
```

The density of the rod material. The value 7960 is taken from Table 2 of the CoRdE paper.

#### 7.4.3.2 materialRadius

```
float GuidewireSim.InitializationStep.materialRadius = 0.001f [private]
```

The radius of the cross-section of the rod. The value 0.001 or 1mm is taken from Table 2 of the CoRdE paper.

### 7.4.3.3 mathHelper

`MathHelper` `GuidewireSim.InitializationStep.mathHelper` [private]

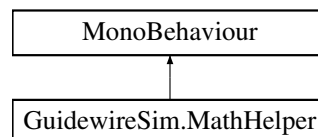
The component `MathHelper` that provides math related helper functions.

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

- [InitializationStep.cs](#)

## 7.5 GuidewireSim.MathHelper Class Reference

Inheritance diagram for `GuidewireSim.MathHelper`:



### Public Member Functions

- void [CalculateCylinderPositions](#) (int cylinderCount, Vector3[] spherePositions, Vector3[] cylinderPositions)
- Vector3 [MatrixVectorMultiplication](#) (float[,] matrix, Vector3 vector)
- BSM.Quaternion [EmbeddedVector](#) (Vector3 vector)
- Vector3 [ImaginaryPart](#) (BSM.Quaternion quaternion)
- Quaternion [QuaternionConversionFromBSM](#) (BSM.Quaternion bsmQuaternion)
- BSM.Quaternion [QuaternionConversionToBSM](#) (Quaternion quaternion)
- Vector3 [DiscreteDarbouxVector](#) (BSM.Quaternion orientationOne, BSM.Quaternion orientationTwo, float rodElementLength)
- float [DarbouxSignFactor](#) (Vector3 currentDarbouxVector, Vector3 restDarbouxVector)
- float [VectorLength](#) (Vector3 vector)
- float [QuaternionLength](#) (BSM.Quaternion quaternion)
- float [RodElementLengthDeviation](#) (Vector3 particlePositionOne, Vector3 particlePositionTwo, float defaultRodElementLength)
- float [StretchConstraintDeviation](#) (Vector3 particlePositionOne, Vector3 particlePositionTwo, BSM.Quaternion orientation, BSM.Quaternion e\_3, float rodElementLength, bool logIntermediateResults=false)
- float [BendTwistConstraintDeviation](#) (BSM.Quaternion orientationOne, BSM.Quaternion orientationTwo, float rodElementLength, Vector3 discreteRestDarbouxVector, bool logIntermediateResults=false)
- float [RodElementLength](#) (Vector3 particlePositionOne, Vector3 particlePositionTwo)
- Vector3[][] [UpdateDirectors](#) (int cylinderCount, BSM.Quaternion[] cylinderOrientations, Vector3[][] directors, BSM.Quaternion[] worldSpaceBasis)
- BSM.Quaternion [RandomUnitQuaternion](#) ()
- float [GetGaussianRandomNumber](#) ()

### Private Member Functions

- float [SquaredNorm](#) (Vector3 vector)

## 7.5.1 Detailed Description

This class provides various helper methods for calculation.

## 7.5.2 Member Function Documentation

### 7.5.2.1 BendTwistConstraintDeviation()

```
float GuidewireSim.MathHelper.BendTwistConstraintDeviation (
    BSM.Quaternion orientationOne,
    BSM.Quaternion orientationTwo,
    float rodElementLength,
    Vector3 discreteRestDarbouxVector,
    bool logIntermediateResults = false )
```

Returns the deviation of the bend twist constraint from zero.

#### Parameters

<i>orientationOne</i>	<i>q</i> of the equation (32).
<i>orientationTwo</i>	<i>u</i> of the equation (32).
<i>rodElementLength</i>	The Rod Element Length between <i>orientationOne</i> and <i>orientationTwo</i> . Used to calculate $\ell$ of the equation (32).
<i>discreteRestDarbouxVector</i>	$\mathcal{K}^0$ of the equation (32).
<i>logIntermediateResults</i>	Whether to output several logs that contain intermediate results of the calculation. Default is false.

#### Returns

The Deviation of the calculated bend twist constraint and zero.

#### Note

Check the Position and Orientation Based Cosserat Rods Paper (2016), equation (32), for more information on the bend twist constraint.

### 7.5.2.2 CalculateCylinderPositions()

```
void GuidewireSim.MathHelper.CalculateCylinderPositions (
    int cylinderCount,
    Vector3[] spherePositions,
    Vector3[] cylinderPositions )
```

Calculates *cylinderPositions* as the middle points of two adjacent spheres.

**Parameters**

<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
<i>spherePositions</i>	The position at the current frame of each sphere.
<i>cylinderPositions</i>	The position/ center of mass of each cylinder.

**Note**

`cylinderPositions` is not marked as an out parameter, since `cylinderPositions` is not initialized in this method, but its values are changed.

**7.5.2.3 DarbouxSignFactor()**

```
float GuidewireSim.MathHelper.DarbouxSignFactor (
    Vector3 currentDarbouxVector,
    Vector3 restDarbouxVector )
```

Calculates the sign factor of the current discrete Darboux Vector and the rest Darboux Vector of the same orientations.

**Note**

Check the Position and Orientation Based Cosserat Rods Paper (2016) for more information on the sign factor.

**Parameters**

<i>currentDarbouxVector</i>	The discrete Darboux Vector of two fixed orientations at the current frame.
<i>restDarbouxVector</i>	The rest Darboux Vector of the same two orientations at frame 0.

**Returns**

The Sign Factor between these two entities.

**7.5.2.4 DiscreteDarbouxVector()**

```
Vector3 GuidewireSim.MathHelper.DiscreteDarbouxVector (
    BSM.Quaternion orientationOne,
    BSM.Quaternion orientationTwo,
    float rodElementLength )
```

Calculates the discrete Darboux Vector of two adjacent orientations `orientationOne`, `orientationTwo`.

## Parameters

<i>orientationOne</i>	The orientation with the lower index, e.g. $i$ .
<i>orientationTwo</i>	The orientation with the higher index, e.g. $i + 1$ .
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.

## Returns

The discrete Darboux Vector between `orientationOne` and `orientationTwo`.

## Note

There is only `cylinderCount - 1` many darboux vectors. The  $i$ -th Darboux Vector is between orientation  $i$  and orientation  $i+1$ .

## Attention

The order in which the orientations are entered matters. The Darboux Vector of  $q_1, q_2$  is not the same as the Darboux Vector of  $q_2, q_1$ .

## 7.5.2.5 EmbeddedVector()

```
BSM.Quaternion GuidewireSim.MathHelper.EmbeddedVector (
    Vector3 vector )
```

Returns a quaternion that is the embedded vector with scalar part zero.

## Example

$$(x, y, z) \mapsto (x, y, z, 0).$$

## Parameters

<i>vector</i>	The vector to be embedded.
---------------	----------------------------

## Returns

The quaternion that is the embedded vector with scalar part zero.

## 7.5.2.6 GetGaussianRandomNumber()

```
float GuidewireSim.MathHelper.GetGaussianRandomNumber ( )
```

Provides a sample from  $\mathcal{N}(0, 1)$  by using the Marsaglia polar method to transform a uniform distribution to a normal distribution.

**Returns**

A sample from  $\mathcal{N}(0, 1)$ .

**Note**

To understand this method, google the Marsaglia polar method. Note that unity does not provide a function to generate a random number following a gaussian distribution.

**7.5.2.7 ImaginaryPart()**

```
Vector3 GuidewireSim.MathHelper.ImaginaryPart (
    BSM.Quaternion quaternion )
```

Returns the imaginary part of a quaternion.

**Example**

$$(x, y, z, w) \mapsto (x, y, z).$$
**Parameters**

<i>quaternion</i>	The quaternion whose imaginary part to return.
-------------------	--

**Returns**

The imaginary part of a quaternion.

**7.5.2.8 MatrixVectorMultiplication()**

```
Vector3 GuidewireSim.MathHelper.MatrixVectorMultiplication (
    float matrix[],
    Vector3 vector )
```

Calculates the multiplication of  $Mx$ , where  $M$  is the `matrix`, and  $x$  is the `vector` input.

**Parameters**

<i>matrix</i>	The matrix to be multiplied with the vector.
<i>vector</i>	The matrix to be multiplied with the matrix.

**Returns**

The multiplication  $Mx$ .

**Requirements** `matrix` must be a  $3 \times 3$  matrix.

#### 7.5.2.9 QuaternionConversionFromBSM()

```
Quaternion GuidewireSim.MathHelper.QuaternionConversionFromBSM (
    BSM.Quaternion bsmQuaternion )
```

Takes as input a BSM.Quaternion and returns a UnityEngine.Quaternion.

##### Parameters

<i>bsmQuaternion</i>	The BSM.Quaternion to be converted.
----------------------	-------------------------------------

##### Returns

The converted UnityEngine.Quaternion.

#### 7.5.2.10 QuaternionConversionToBSM()

```
BSM.Quaternion GuidewireSim.MathHelper.QuaternionConversionToBSM (
    Quaternion quaternion )
```

Takes as input a UnityEngine.Quaternion and returns a BSM.Quaternion.

##### Parameters

<i>bsmQuaternion</i>	The UnityEngine.Quaternion to be converted.
----------------------	---

##### Returns

The converted BSM.Quaternion.

#### 7.5.2.11 QuaternionLength()

```
float GuidewireSim.MathHelper.QuaternionLength (
    BSM.Quaternion quaternion )
```

Returns the quaternion length of `quaternion`, i.e.  $\sqrt{x^2 + y^2 + z^2 + w^2}$ .

**Parameters**

<i>quaternion</i>	The quaternion whose length to return.
-------------------	--

**Returns**

The quaternion length of `quaternion`.

**7.5.2.12 RandomUnitQuaternion()**

```
BSM.Quaternion GuidewireSim.MathHelper.RandomUnitQuaternion ( )
```

Provides a random unit quaternion drawn from a gaussian distribution.

**Returns**

A random unit quaternion drawn from a gaussian distribution.

**Note**

This works by drawing four random, gaussian distributed, numbers, and filling the components of the quaternion with these numbers. Mathematically, this is equal to drawing a quaternion from a gaussian distribution in  $\mathcal{R}^4$ , since the joint distribution of gaussian samples is again gaussian.

**Requirements** The length of the drawn quaternion is approximately equal to one.

**7.5.2.13 RodElementLength()**

```
float GuidewireSim.MathHelper.RodElementLength (
    Vector3 particlePositionOne,
    Vector3 particlePositionTwo )
```

Calculates the rod element length between `particlePositionOne` and `particlePositionTwo`.

**Parameters**

<i>particlePositionOne</i>	The first particle of the rod element under consideration.
<i>particlePositionTwo</i>	The first particle of the rod element under consideration.

**Returns**

The rod element length.

### 7.5.2.14 RodElementLengthDeviation()

```
float GuidewireSim.MathHelper.RodElementLengthDeviation (
    Vector3 particlePositionOne,
    Vector3 particlePositionTwo,
    float defaultRodElementLength )
```

Returns the deviation between the actual distance of *particlePositionOne* and *particlePositionTwo* (current Rod Element Length) and the *defaultRodElementLength*.

#### Parameters

<i>particlePositionOne</i>	The first particle under consideration for the rod element length.
<i>particlePositionTwo</i>	The first particle under consideration for the rod element length.
<i>defaultRodElementLength</i>	The rod element length at rest state (i.e. frame 0) between these two particles.

#### Returns

The deviation between the actual rod element length and the default rod element length.

### 7.5.2.15 SquaredNorm()

```
float GuidewireSim.MathHelper.SquaredNorm (
    Vector3 vector ) [private]
```

Returns the squared norm of a *vector*.

#### Parameters

<i>vector</i>	The vector whose squared norm to return.
---------------	--

#### Returns

The Squared norm of *vector*.

### 7.5.2.16 StretchConstraintDeviation()

```
float GuidewireSim.MathHelper.StretchConstraintDeviation (
    Vector3 particlePositionOne,
    Vector3 particlePositionTwo,
    BSM.Quaternion orientation,
    BSM.Quaternion e_3,
    float rodElementLength,
    bool logIntermediateResults = false )
```

Returns the deviation of the stretch constraint from zero.

## Parameters

<i>particlePositionOne</i>	$p_1$ of the equation (31).
<i>particlePositionTwo</i>	$p_2$ of the equation (31).
<i>orientation</i>	$q$ of the equation (31).
<i>e_3</i>	$e_3$ of the equation (31).
<i>rodElementLength</i>	$l$ of the equation (31).
<i>logIntermediateResults</i>	Whether to output several logs that contain intermediate results of the calculation. Default is false.

## Returns

The Deviation of the calculated stretch constraint and zero.

## Note

Check the Position and Orientation Based Cosserat Rods Paper (2016), equation (31), for more information on the stretch constraint.

## 7.5.2.17 UpdateDirectors()

```
Vector3 [][] GuidewireSim.MathHelper.UpdateDirectors (
    int cylinderCount,
    BSM.Quaternion[] cylinderOrientations,
    Vector3 directors[][],
    BSM.Quaternion[] worldSpaceBasis )
```

Updates all directors of each orientation at the update step of the simulation loop.

## Example

The directors  $d_1, d_2, d_3$  are calculated as  $d_i = q \cdot e_i \cdot \bar{q}$  for each orientation  $q$ , where  $e_i$  is the  $i$ -th world space basis vector. In quaternion calculus, this means rotating  $e_i$  by  $q$ .

## Parameters

<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.
<i>directors</i>	The orthonormal basis of each orientation element / cylinder, also called directors.
<i>worldSpaceBasis</i>	The three basis vectors of the world coordinate system.

## Returns

All directors of each orientation.

### 7.5.2.18 VectorLength()

```
float GuidewireSim.MathHelper.VectorLength (
    Vector3 vector )
```

Returns the vector length of `vector`, i.e.  $\sqrt{x_1^2 + x_2^2 + x_3^2}$  for a three-dimensional vector.

#### Parameters

<code>vector</code>	The vector whose length to return.
---------------------	------------------------------------

#### Returns

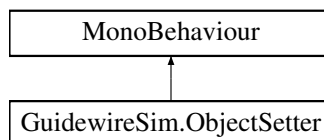
The vector length of `vector`.

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

- [MathHelper.cs](#)

## 7.6 GuidewireSim.ObjectSetter Class Reference

Inheritance diagram for GuidewireSim.ObjectSetter:



### Public Member Functions

- void [SetSpherePositions](#) (GameObject[] spheres, int spheresCount, Vector3[] spherePositions)
- void [SetCylinderPositions](#) (GameObject[] cylinders, int cylinderCount, Vector3[] cylinderPositions)
- void [SetCylinderOrientations](#) (GameObject[] cylinders, int cylinderCount, BSM.Quaternion[] cylinderOrientations, Vector3[][] directors)

### Private Member Functions

- void [Awake](#) ()

### Private Attributes

- [MathHelper](#) `mathHelper`

The component [MathHelper](#) that provides math related helper functions.

## 7.6.1 Detailed Description

This class is responsible for setting the transformation positions of the GameObjects in the scene to their respective simulation data like `spherePositions`.

## 7.6.2 Member Function Documentation

### 7.6.2.1 Awake()

```
void GuidewireSim.ObjectSetter.Awake ( ) [private]
```

### 7.6.2.2 SetCylinderOrientations()

```
void GuidewireSim.ObjectSetter.SetCylinderOrientations (
    GameObject[] cylinders,
    int cylinderCount,
    BSM.Quaternion[] cylinderOrientations,
    Vector3 directors[][] )
```

Rotates each cylinder GameObject such that its centerline is parallel with the line segment that is spanned by the two adjacent sphere's center of masses.

#### Parameters

<i>cylinders</i>	All cylinders that are part of the guidewire.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.
<i>directors</i>	The orthonormal basis of each orientation element / cylinder, also called directors.

#### Note

`appliedTransformation` is the rotation that aligns the y-axis of the cylinder with the z-axis of the orientations (the third director). This is needed, because the y-axis of the cylinder is parallel with its centerline, while the z-axis of the orientations (the third director) is also defined as being parallel with the cylinder's centerline. Thus `appliedTransformation` is necessary.

### 7.6.2.3 SetCylinderPositions()

```
void GuidewireSim.ObjectSetter.SetCylinderPositions (
    GameObject[] cylinders,
```

```
int cylinderCount,  
Vector3[] cylinderPositions )
```

Sets the positions of the GameObjects `cylinders` to their respective `cylinderPositions`.

## Parameters

<i>cylinders</i>	All cylinders that are part of the guidewire.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
<i>cylinderPositions</i>	The position/ center of mass of each cylinder.

## 7.6.2.4 SetSpherePositions()

```
void GuidewireSim.ObjectSetter.SetSpherePositions (
    GameObject[] spheres,
    int spheresCount,
    Vector3[] spherePositions )
```

Sets the positions of the GameObjects `spheres` to their respective `spherePositions`.

## Parameters

<i>spheres</i>	All spheres that are part of the guidewire.
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <code>spherePositionPredictions</code> .
<i>spherePositions</i>	The position at the current frame of each sphere.

## 7.6.3 Member Data Documentation

## 7.6.3.1 mathHelper

`MathHelper` `GuidewireSim.ObjectSetter.mathHelper` [private]

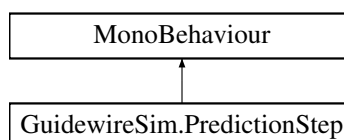
The component `MathHelper` that provides math related helper functions.

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

- [ObjectSetter.cs](#)

## 7.7 GuidewireSim.PredictionStep Class Reference

Inheritance diagram for `GuidewireSim.PredictionStep`:



## Public Member Functions

- Vector3[] [PredictSphereVelocities](#) (Vector3[] sphereVelocities, float[] sphereInverseMasses, Vector3[] sphereExternalForces)
- Vector3[] [PredictSpherePositions](#) (Vector3[] spherePositionPredictions, int spheresCount, Vector3[] spherePositions, Vector3[] sphereVelocities)
- Vector3[] [PredictAngularVelocities](#) (Vector3[] cylinderAngularVelocities, int cylinderCount, float[,] inertiaTensor, Vector3[] cylinderExternalTorques, float[,] inverseInertiaTensor)
- BSM.Quaternion[] [PredictCylinderOrientations](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] cylinderAngularVelocities, BSM.Quaternion[] cylinderOrientations)

## Private Member Functions

- void [Awake](#) ()

## Private Attributes

- [MathHelper](#) mathHelper

*The component [MathHelper](#) that provides math related helper functions.*

### 7.7.1 Detailed Description

This class implements the prediction step of the algorithm.

### 7.7.2 Member Function Documentation

#### 7.7.2.1 Awake()

```
void GuidewireSim.PredictionStep.Awake ( ) [private]
```

#### 7.7.2.2 PredictAngularVelocities()

```
Vector3 [] GuidewireSim.PredictionStep.PredictAngularVelocities (
    Vector3[] cylinderAngularVelocities,
    int cylinderCount,
    float inertiaTensor[,],
    Vector3[] cylinderExternalTorques,
    float inverseInertiaTensor[,])
```

Calculates the predictions for the angular velocities for the prediction step of the algorithm.

**Parameters**

<i>cylinderAngularVelocities</i>	The angular velocity of the current frame of each orientation element/ cylinder.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>inertiaTensor</i>	The inertia tensor. Entries are approximates as in the CoRdE paper.
<i>cylinderExternalTorques</i>	The sum of all current external torques that are applied per orientation element/ cylinder.
<i>inverseInertiaTensor</i>	The inverse of <i>inertiaTensor</i> .

**Returns**

The angular velocity of the current frame of each orientation element/ cylinder, i.e. *cylinderAngularVelocities*.

**Note**

The predictions are again stored in *cylinderAngularVelocities*.

**7.7.2.3 PredictCylinderOrientations()**

```
BSM.Quaternion [] GuidewireSim.PredictionStep.PredictCylinderOrientations (
    BSM.Quaternion[] cylinderOrientationPredictions,
    int cylinderCount,
    Vector3[] cylinderAngularVelocities,
    BSM.Quaternion[] cylinderOrientations )
```

Calculates the predictions for the cylinder orientations for the prediction step of the algorithm.

**Parameters**

<i>cylinderOrientationPredictions</i>	The prediction of the orientation of each cylinder at its center of mass.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>cylinderAngularVelocities</i>	The angular velocity of the current frame of each orientation element/ cylinder.
<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.

**Returns**

The prediction of the orientation of each cylinder at its center of mass, i.e. *cylinderOrientationPredictions*.

**7.7.2.4 PredictSpherePositions()**

```
Vector3 [] GuidewireSim.PredictionStep.PredictSpherePositions (
    Vector3[] spherePositionPredictions,
```

```
int spheresCount,
Vector3[] spherePositions,
Vector3[] sphereVelocities )
```

Calculates the predictions for the sphere positions for the prediction step of the algorithm.

#### Parameters

<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere (in this case of the last frame).
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>spherePositions</i>	The position at the current frame of each sphere.
<i>sphereVelocities</i>	The velocity of the current frame of each sphere.

#### Returns

The prediction of the position at the current frame of each sphere, i.e. *spherePositionPredictions*.

#### 7.7.2.5 PredictSphereVelocities()

```
Vector3 [] GuidewireSim.PredictionStep.PredictSphereVelocities (
    Vector3[] sphereVelocities,
    float[] sphereInverseMasses,
    Vector3[] sphereExternalForces )
```

Calculates the predictions for the sphere velocities for the prediction step of the algorithm.

#### Parameters

<i>sphereVelocities</i>	The velocity of the current frame of each sphere.
<i>sphereInverseMasses</i>	The constant inverse masses of each sphere.
<i>sphereExternalForces</i>	The sum of all current external forces that are applied per particle/ sphere.

#### Returns

The predictions of the positions of the spheres, i.e. *spherePositionPredictions*.

#### Note

The predictions are again stored in *sphereVelocities*.

### 7.7.3 Member Data Documentation

### 7.7.3.1 mathHelper

`MathHelper` GuidewireSim.PredictionStep.mathHelper [private]

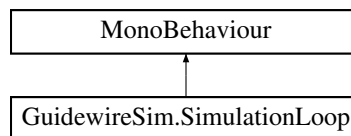
The component `MathHelper` that provides math related helper functions.

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

- [PredictionStep.cs](#)

## 7.8 GuidewireSim.SimulationLoop Class Reference

Inheritance diagram for GuidewireSim.SimulationLoop:



### Public Member Functions

- void [PerformSimulationLoop](#) ()

### Public Attributes

- Vector3[] [spherePositions](#)  
*The position at the current frame of each sphere.*
- Vector3[] [sphereVelocities](#)  
*The velocity of the current frame of each sphere. Initalized with zero entries.*
- float[] [sphereInverseMasses](#)
- Vector3[] [sphereExternalForces](#)  
*The sum of all current external forces that are applied per particle/ sphere.*
- Vector3[] [cylinderPositions](#)  
*The center of mass of each cylinder.*
- float[] [cylinderScalarWeights](#)
- Vector3[] [cylinderExternalTorques](#)  
*The sum of all current external torques that are applied per orientation element/ cylinder.*
- Vector3[][] [directors](#)
- bool [solveStretchConstraints](#) = true  
*Whether or not to perform the constraint solving of the stretch constraint.*
- bool [solveBendTwistConstraints](#) = true  
*Whether or not to perform the constraint solving of the bend twist constraint.*

## Properties

- int [ConstraintSolverSteps](#) = 100 [get, set]
- bool [ExecuteSingleLoopTest](#) = false [get, set]
- int [SpheresCount](#) [get, private set]  
The count of all [spheres](#) of the guidewire.
- int [CylinderCount](#) [get, private set]  
The count of all [cylinders](#) of the guidewire.

## Private Member Functions

- void [Awake](#) ()
- void [Start](#) ()
- void [FixedUpdate](#) ()
- void [PerformInitializationStep](#) ()
- void [PerformPredictionStep](#) ()
- void [PerformConstraintSolvingStep](#) ()
- void [PerformUpdateStep](#) ()
- void [AdoptCalculations](#) ()

## Private Attributes

- [InitializationStep](#) initializationStep  
The component [InitializationStep](#) that is responsible for initializing the simulation.
- [PredictionStep](#) predictionStep  
The component [PredictionStep](#) that is responsible for executing the Prediction Step of the algorithm.
- [ConstraintSolvingStep](#) constraintSolvingStep
- [UpdateStep](#) updateStep  
The component [UpdateStep](#) that is responsible for executing the Update Step of the algorithm.
- [ObjectSetter](#) objectSetter  
The component [ObjectSetter](#) that is responsible for setting all positions and rotations the the GameObjects.
- [MathHelper](#) mathHelper  
The component [MathHelper](#) that provides math related helper functions.
- GameObject[] [spheres](#)
- GameObject[] [cylinders](#)
- Vector3[] [spherePositionPredictions](#)  
The prediction of the position at the current frame of each sphere.
- BSM.Quaternion[] [cylinderOrientations](#)  
The orientation of each cylinder at its center of mass.
- BSM.Quaternion[] [cylinderOrientationPredictions](#)  
The prediction of the orientation of each cylinder at its center of mass.
- Vector3[] [discreteRestDarbouxVectors](#)
- Vector3[] [cylinderAngularVelocities](#)
- float[,] [inertiaTensor](#)  
The inertia tensor. Entries are approximates as in the CoRdE paper.
- float[,] [inverseInertiaTensor](#)  
The inverse of [inertiaTensor](#).
- BSM.Quaternion[] [worldSpaceBasis](#)
- float [rodElementLength](#) = 10f

## 7.8.1 Detailed Description

This class executes the outer simulation loop of the algorithm and calls the implementations of each algorithm step and manages all coherent data.

## 7.8.2 Member Function Documentation

### 7.8.2.1 AdoptCalculations()

```
void GuidewireSim.SimulationLoop.AdoptCalculations ( ) [private]
```

Adopts the data to the Unity GameObjects. For example, sets the positions of the GameObjects [spheres](#) to [spherePositions](#).

**Requirements** Sets the positions of the GameObjects [spheres](#) to [spherePositions](#).  
Calculates [cylinderPositions](#) based on [spherePositions](#).  
Sets the positions of the GameObjects [cylinders](#) to [cylinderPositions](#).  
Sets the rotations of the GameObjects [cylinders](#) to [cylinderOrientations](#).

### 7.8.2.2 Awake()

```
void GuidewireSim.SimulationLoop.Awake ( ) [private]
```

### 7.8.2.3 FixedUpdate()

```
void GuidewireSim.SimulationLoop.FixedUpdate ( ) [private]
```

**Requirements** Execute the simulation loop if and only if [ExecuteSingleLoopTest](#) is false.

#### 7.8.2.4 PerformConstraintSolvingStep()

```
void GuidewireSim.SimulationLoop.PerformConstraintSolvingStep ( ) [private]
```

Performs the constraint solving step of the algorithm.

**Requirements** Performs the constraint solving of every constraint `#solverStep` many times.

- Solve stretch constraints, if and only if `solveStretchConstraints` is true.
- Solve bend twist constraints, if and only if `solveBendTwistConstraints` is true.
- If `solveStretchConstraints`, then `SpheresCount` is at least two.
- If `solveStretchConstraints`, then `CylinderCount` is at least one.
- If `solveBendTwistConstraints`, then `SpheresCount` is at least three.
- If `solveBendTwistConstraints`, then `CylinderCount` is at least two.
- If `solveStretchConstraints`, after the step is complete the deviation between the actual rod element length and the default (rest state) `rodElementLength` should be close to zero.
- If `solveStretchConstraints`, after the step is complete the deviation of the stretch constraint to zero should be close to zero.

#### 7.8.2.5 PerformInitializationStep()

```
void GuidewireSim.SimulationLoop.PerformInitializationStep ( ) [private]
```

Calls every step that is mandatory to declare and initialize all data.

**Requirements** Set `SpheresCount` to the length of `spheres`.

- Set `CylinderCount` to the length of `cylinders`.
- Call every init method of `initializationStep`.

#### 7.8.2.6 PerformPredictionStep()

```
void GuidewireSim.SimulationLoop.PerformPredictionStep ( ) [private]
```

Performs the prediction step of the algorithm.

**Requirements** Predict the `sphereVelocities`.

- Predict the `spherePositionPredictions`.
- Predict the `cylinderAngularVelocities`.
- Predict the `cylinderOrientationPredictions`.

### 7.8.2.7 PerformSimulationLoop()

```
void GuidewireSim.SimulationLoop.PerformSimulationLoop ( )
```

Performs the outer simulation loop of the algorithm.

#### Note

In a late version, CollisionDetection and GenerateCollisionConstraints will be added to the algorithm.

### 7.8.2.8 PerformUpdateStep()

```
void GuidewireSim.SimulationLoop.PerformUpdateStep ( ) [private]
```

Performs the update step of the algorithm.

**Requirements** Update [sphereVelocities](#).  
Update [spherePositions](#).  
Update [cylinderAngularVelocities](#).  
Update [cylinderOrientations](#).  
Update [directors](#).

### 7.8.2.9 Start()

```
void GuidewireSim.SimulationLoop.Start ( ) [private]
```

## 7.8.3 Member Data Documentation

### 7.8.3.1 constraintSolvingStep

```
ConstraintSolvingStep GuidewireSim.SimulationLoop.constraintSolvingStep [private]
```

The component [ConstraintSolvingStep](#) that is responsible for correcting the predictions with the collision and model constraints.

### 7.8.3.2 cylinderAngularVelocities

```
Vector3 [ ] GuidewireSim.SimulationLoop.cylinderAngularVelocities [private]
```

The angular velocity of the current frame of each orientation element/ cylinder. Initalized with zero entries.

### 7.8.3.3 cylinderExternalTorques

```
Vector3 [] GuidewireSim.SimulationLoop.cylinderExternalTorques
```

The sum of all current external torques that are applied per orientation element/ cylinder.

### 7.8.3.4 cylinderOrientationPredictions

```
BSM.Quaternion [] GuidewireSim.SimulationLoop.cylinderOrientationPredictions [private]
```

The prediction of the orientation of each cylinder at its center of mass.

### 7.8.3.5 cylinderOrientations

```
BSM.Quaternion [] GuidewireSim.SimulationLoop.cylinderOrientations [private]
```

The orientation of each cylinder at its center of mass.

### 7.8.3.6 cylinderPositions

```
Vector3 [] GuidewireSim.SimulationLoop.cylinderPositions
```

The center of mass of each cylinder.

### 7.8.3.7 cylinders

```
GameObject [] GuidewireSim.SimulationLoop.cylinders [private]
```

All cylinders that are part of the guidewire.

#### Attention

The order in which the cylinders are assigned matters. Assign them such that two adjacent cylinders are adjacent in the array as well.

### 7.8.3.8 cylinderScalarWeights

```
float [ ] GuidewireSim.SimulationLoop.cylinderScalarWeights
```

The constant scalar weights of each orientation/ quaternion similar to [sphereInverseMasses](#).

#### Note

Set to 1 for moving orientations (so that angular motion can be applied) and to 0 for fixed orientations.

### 7.8.3.9 directors

```
Vector3 [ ] [ ] GuidewireSim.SimulationLoop.directors
```

The orthonormal basis of each orientation element / cylinder, also called directors.

#### Note

In the 0th row are the first directors of each orientation element, not in the 1th row. Example: The (i, j)th element holds the (i-1)th director of orientation element j.

### 7.8.3.10 discreteRestDarbouxVectors

```
Vector3 [ ] GuidewireSim.SimulationLoop.discreteRestDarbouxVectors [private]
```

The discrete Darboux Vector at the rest configuration, i.e. at frame 0.

#### Note

It is important to only take the imaginary part in the calculation for the discrete Darboux Vector, thus we only save it as a Vector3. To use it in a quaternion setting, embedd the Vector3 with scalar part 0, i.e. with `EmbeddedVector()`.

#### Attention

There is only `CylinderCount - 1` many darboux vectors. The i-th Darboux Vector is between orientation i and orientation i+1.

### 7.8.3.11 inertiaTensor

```
float [,] GuidewireSim.SimulationLoop.inertiaTensor [private]
```

The inertia tensor. Entries are approximates as in the CoRdE paper.

### 7.8.3.12 initializationStep

`InitializationStep` GuidewireSim.SimulationLoop.initializationStep [private]

The component `InitializationStep` that is responsible for initializing the simulation.

### 7.8.3.13 inverseInertiaTensor

`float [,]` GuidewireSim.SimulationLoop.inverseInertiaTensor [private]

The inverse of `inertiaTensor`.

### 7.8.3.14 mathHelper

`MathHelper` GuidewireSim.SimulationLoop.mathHelper [private]

The component `MathHelper` that provides math related helper functions.

### 7.8.3.15 objectSetter

`ObjectSetter` GuidewireSim.SimulationLoop.objectSetter [private]

The component `ObjectSetter` that is responsible for setting all positions and rotations the the GameObjects.

### 7.8.3.16 predictionStep

`PredictionStep` GuidewireSim.SimulationLoop.predictionStep [private]

The component `PredictionStep` that is responsible for executing the Prediction Step of the algorithm.

### 7.8.3.17 rodElementLength

`float` GuidewireSim.SimulationLoop.rodElementLength = 10f [private]

The distance between two spheres, also the distance between two orientations. Also the length of one cylinder.

#### Note

This should be two times the radius of a sphere.

#### Attention

Make sure that the guidewire setup fulfills that the distance between two adjacent spheres is `rodElementLength`.

#### 7.8.3.18 solveBendTwistConstraints

```
bool GuidewireSim.SimulationLoop.solveBendTwistConstraints = true
```

Whether or not to perform the constraint solving of the bend twist constraint.

#### 7.8.3.19 solveStretchConstraints

```
bool GuidewireSim.SimulationLoop.solveStretchConstraints = true
```

Whether or not to perform the constraint solving of the stretch constraint.

#### 7.8.3.20 sphereExternalForces

```
Vector3 [ ] GuidewireSim.SimulationLoop.sphereExternalForces
```

The sum of all current external forces that are applied per particle/ sphere.

#### 7.8.3.21 sphereInverseMasses

```
float [ ] GuidewireSim.SimulationLoop.sphereInverseMasses
```

The constant inverse masses of each sphere.

##### Note

Set to 1 for moving spheres and to 0 for fixed spheres.

#### 7.8.3.22 spherePositionPredictions

```
Vector3 [ ] GuidewireSim.SimulationLoop.spherePositionPredictions [private]
```

The prediction of the position at the current frame of each sphere.

#### 7.8.3.23 spherePositions

```
Vector3 [ ] GuidewireSim.SimulationLoop.spherePositions
```

The position at the current frame of each sphere.

#### 7.8.3.24 spheres

```
GameObject [] GuidewireSim.SimulationLoop.spheres [private]
```

All spheres that are part of the guidewire.

##### Attention

The order in which the spheres are assigned matters. Assign them such that two adjacent spheres are adjacent in the array as well.

#### 7.8.3.25 sphereVelocities

```
Vector3 [] GuidewireSim.SimulationLoop.sphereVelocities
```

The velocity of the current frame of each sphere. Initalized with zero entries.

#### 7.8.3.26 updateStep

```
UpdateStep GuidewireSim.SimulationLoop.updateStep [private]
```

The component [UpdateStep](#) that is responsible for executing the Update Step of the algorithm.

#### 7.8.3.27 worldSpaceBasis

```
BSM.Quaternion [] GuidewireSim.SimulationLoop.worldSpaceBasis [private]
```

The three basis vectors of the world coordinate system as embedded quaternions with scalar part 0. E.g. the first basis vector is (1, 0, 0), the second (0, 1, 0) and the third (0, 0, 1).

### 7.8.4 Property Documentation

#### 7.8.4.1 ConstraintSolverSteps

```
int GuidewireSim.SimulationLoop.ConstraintSolverSteps = 100 [get], [set]
```

How often the constraint solver iterates over each constraint during the Constraint Solving Step.

##### Attention

This value must be positive.

#### 7.8.4.2 CylinderCount

```
int GuidewireSim.SimulationLoop.CylinderCount [get], [private set]
```

The count of all [cylinders](#) of the guidewire.

#### 7.8.4.3 ExecuteSingleLoopTest

```
bool GuidewireSim.SimulationLoop.ExecuteSingleLoopTest = false [get], [set]
```

Whether or not to execute the Single Loop Test, in which the outer simulation loop is exactly executed once.

#### 7.8.4.4 SpheresCount

```
int GuidewireSim.SimulationLoop.SpheresCount [get], [private set]
```

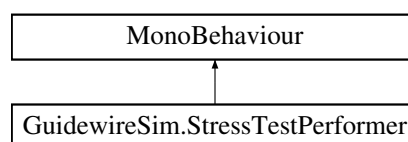
The count of all [spheres](#) of the guidewire.

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

- [SimulationLoop.cs](#)

## 7.9 GuidewireSim.StressTestPerformer Class Reference

Inheritance diagram for GuidewireSim.StressTestPerformer:



### Private Member Functions

- void [Awake](#) ()
- void [Start](#) ()
- void [PerformStressTests](#) ()
- IEnumerator [PerformStressTestOne](#) (float applyForceTime=1f)

### Private Attributes

- [SimulationLoop](#) [simulationLoop](#)  
The [SimulationLoop](#) component that executes all steps of the simulation loop.
- bool [doStressTestOne](#) = false

## 7.9.1 Detailed Description

This class enables the user to test the impact of multiple external forces and external torques with one button within the Unity inspector.

### Attention

In the current version, the user is not able to fix positions or orientations of the guidewire, which is necessary e.g. for stress test one.

## 7.9.2 Member Function Documentation

### 7.9.2.1 Awake()

```
void GuidewireSim.StressTestPerformer.Awake ( ) [private]
```

### 7.9.2.2 PerformStressTestOne()

```
IEnumerator GuidewireSim.StressTestPerformer.PerformStressTestOne (
    float applyForceTime = 1f ) [private]
```

Performs stress test one. This test fixes the position of one end of the guidewire, and applies `pullForce` at the other end for `applyForceTime` seconds, and then applies `- pullForce` for another `applyForceTime` seconds.

#### Parameters

<code>applyForceTime</code>	For how many seconds to apply the force to the particles.
-----------------------------	---

### Attention

In the current version, the user is not able to fix positions or orientations of the guidewire, which is necessary e.g. for stress test one.

**Requirements** Output a log message when no further forces are applied to the guidewire.

### 7.9.2.3 PerformStressTests()

```
void GuidewireSim.StressTestPerformer.PerformStressTests ( ) [private]
```

Performs each Stress Test whose respective serialized boolean is set to true in the Unity inspector.

#### 7.9.2.4 Start()

```
void GuidewireSim.StressTestPerformer.Start ( ) [private]
```

### 7.9.3 Member Data Documentation

#### 7.9.3.1 doStressTestOne

```
bool GuidewireSim.StressTestPerformer.doStressTestOne = false [private]
```

Whether to run Stress Test One. This test fixes the position of one end of the guidewire, and applies `pullForce` at the other end for `applyForceTime` seconds, and then applies `- pullForce` for another `applyForceTime` seconds.

#### 7.9.3.2 simulationLoop

```
SimulationLoop GuidewireSim.StressTestPerformer.simulationLoop [private]
```

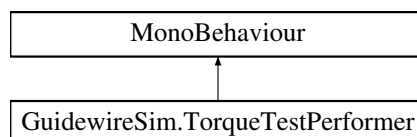
The [SimulationLoop](#) component that executes all steps of the simulation loop.

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

- [StressTestPerformer.cs](#)

## 7.10 GuidewireSim.TorqueTestPerformer Class Reference

Inheritance diagram for GuidewireSim.TorqueTestPerformer:



### Private Member Functions

- void [Awake](#) ()
- void [Start](#) ()
- void [PerformTorqueTests](#) ()
- void [PerformTorqueTestOne](#) ()
- IEnumerator [PerformTorqueTestTwo](#) (Vector3 [pullTorque](#), float `applyTorqueTime=1f`)
- IEnumerator [PerformTorqueTestThree](#) (Vector3 [pullTorque](#), float `applyTorqueTime=10f`)

## Private Attributes

- [SimulationLoop](#) `simulationLoop`  
*The [SimulationLoop](#) component that executes all steps of the simulation loop.*
- `Vector3` `pullTorque` = new `Vector3(0f, 0.3f, 0f)`
- `bool` `doTorqueTestOne` = false  
*Whether to run Torque Test One. This test applies an external torque to one end of the guidewire.*
- `bool` `doTorqueTestTwo` = false
- `bool` `doTorqueTestThree` = false

### 7.10.1 Detailed Description

This class enables the user to test the impact of external torques with one button within the Unity inspector.

### 7.10.2 Member Function Documentation

#### 7.10.2.1 Awake()

```
void GuidewireSim.TorqueTestPerformer.Awake ( ) [private]
```

#### 7.10.2.2 PerformTorqueTestOne()

```
void GuidewireSim.TorqueTestPerformer.PerformTorqueTestOne ( ) [private]
```

Performs torque test one. This test applies an external torque to one end of the guidewire.

#### 7.10.2.3 PerformTorqueTests()

```
void GuidewireSim.TorqueTestPerformer.PerformTorqueTests ( ) [private]
```

Performs each Torque Test whose respective serialized boolean is set to true in the Unity inspector.

#### 7.10.2.4 PerformTorqueTestThree()

```
IEnumerator GuidewireSim.TorqueTestPerformer.PerformTorqueTestThree (
    Vector3 pullTorque,
    float applyTorqueTime = 10f ) [private]
```

Performs torque test three. This test applies an external torque to one end of the guidewire and at the same time the opposite torque at the other end of the guidewire. The applied torque starts at 0 and linearly interpolates until it reaches `pullTorque` at `applyTorqueTime` seconds.

## Parameters

<i>pullTorque</i>	The external torque that is applied to one end of the guidewire.
<i>applyTorqueTime</i>	For how many seconds to apply the torque to the orientations.

**Requirements** Output a log message when no further torques are applied to the guidewire.

### 7.10.2.5 PerformTorqueTestTwo()

```
IEnumerator GuidewireSim.TorqueTestPerformer.PerformTorqueTestTwo (
    Vector3 pullTorque,
    float applyTorqueTime = 1f ) [private]
```

Performs torque test two. This test applies an external torque to one end of the guidewire for a fixed amount of time and then the opposite torque at the same orientation for the same amount of time.

## Parameters

<i>pullTorque</i>	The external torque that is applied to one end of the guidewire.
<i>applyTorqueTime</i>	For how many seconds to apply the torque to the orientations.

**Requirements** Output a log message when no further torques are applied to the guidewire.

### 7.10.2.6 Start()

```
void GuidewireSim.TorqueTestPerformer.Start ( ) [private]
```

## 7.10.3 Member Data Documentation

### 7.10.3.1 doTorqueTestOne

```
bool GuidewireSim.TorqueTestPerformer.doTorqueTestOne = false [private]
```

Whether to run Torque Test One. This test applies an external torque to one end of the guidewire.

### 7.10.3.2 doTorqueTestThree

```
bool GuidewireSim.TorqueTestPerformer.doTorqueTestThree = false [private]
```

Whether to run Torque Test Three. This test applies an external torque to one end of the guidewire and at the same time the opposite torque at the other end of the guidewire. The applied torque starts at 0 and linearly interpolates until it reaches `pullTorque` at `applyTorqueTime` seconds.

### 7.10.3.3 doTorqueTestTwo

```
bool GuidewireSim.TorqueTestPerformer.doTorqueTestTwo = false [private]
```

Whether to run Torque Test Two. This test applies an external torque to one end of the guidewire for a fixed amount of time and then the opposite torque at the same orientation for the same amount of time.

### 7.10.3.4 pullTorque

```
Vector3 GuidewireSim.TorqueTestPerformer.pullTorque = new Vector3(0f, 0.3f, 0f) [private]
```

The external torque that is applied to the respective parts of the guidewire, depending on the test.

### 7.10.3.5 simulationLoop

```
SimulationLoop GuidewireSim.TorqueTestPerformer.simulationLoop [private]
```

The [SimulationLoop](#) component that executes all steps of the simulation loop.

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

- [TorqueTestPerformer.cs](#)

## 7.11 UnitTest\_SolveBendTwistConstraint Class Reference

### Public Member Functions

- [IEnumerator](#) [PerformUnitTests](#) ()

### Private Member Functions

- void [Test\\_SolveBendTwistConstraint](#) (int iterations, BSM.Quaternion orientationOne, BSM.Quaternion orientationTwo, float rodElementLength, Vector3 discreteRestDarbouxVector, [GuidewireSim.MathHelper](#) mathHelper, [GuidewireSim.ConstraintSolvingStep](#) constraintSolvingStep)

### Private Attributes

- int [sampleSize](#) = 10
- int [constraintSolverSteps](#) = 50

*How often the constraint solver iterates over each constraint during the Constraint Solving Step.*

### 7.11.1 Detailed Description

This class provides unit tests that test the method `SolveBendTwistConstraint()` of `ConstraintSolvingStep`. Executing this test once generates `sampleSize` many random value pairs and executes the unit test with each of these pairs.

### 7.11.2 Member Function Documentation

#### 7.11.2.1 PerformUnitTests()

```
IEnumerator UnitTest_SolveBendTwistConstraint.PerformUnitTests ( )
```

Arranges all necessary data, generates `sampleSize` many random value pairs, and then passes all data to `Test_SolveBendTwistConstraint()`, where the unit tests are executed.

#### Note

Only tests the case that all rod elements are aligned at rest state. If you want to test deformed rods at rest state, change `discreteRestDarbouxVector` accordingly.

#### 7.11.2.2 Test\_SolveBendTwistConstraint()

```
void UnitTest_SolveBendTwistConstraint.Test_SolveBendTwistConstraint (
    int iterations,
    BSM.Quaternion orientationOne,
    BSM.Quaternion orientationTwo,
    float rodElementLength,
    Vector3 discreteRestDarbouxVector,
    GuidewireSim.MathHelper mathHelper,
    GuidewireSim.ConstraintSolvingStep constraintSolvingStep ) [private]
```

Executes `SolveBendTwistConstraint()` of `ConstraintSolvingStep` `iterations` many times for one values pair, and then asserts whether the results of the algorithm of `SolveBendTwistConstraint()` converged towards the expected values.

#### Parameters

<i>iterations</i>	The number of iterations that <code>SolveBendTwistConstraint()</code> of <code>ConstraintSolvingStep</code> is executed.
<i>orientationOne</i>	The first orientation for <code>SolveBendTwistConstraint()</code> .
<i>orientationTwo</i>	The second orientation for <code>SolveBendTwistConstraint()</code> .
<i>rodElementLength</i>	The rod element length for <code>SolveBendTwistConstraint()</code> .
<i>discreteRestDarbouxVector</i>	The discrete Darboux Vector at rest state for <code>SolveBendTwistConstraint()</code> .
<i>mathHelper</i>	The component <code>MathHelper</code> .
<i>constraintSolvingStep</i>	The component <code>ConstraintSolvingStep</code> .

**Requirements** `orientationOne` and `orientationTwo` are still unit quaternions at the end of the test.

The deviation between the bend twist constraint and zero is lower than a reasonable tolerance, i.e. close to zero., which means that the algorithm of `SolveBendTwistConstraint()` converges towards the fulfillment of the bend twist constraint.

### 7.11.3 Member Data Documentation

#### 7.11.3.1 constraintSolverSteps

```
int UnitTest_SolveBendTwistConstraint.constraintSolverSteps = 50 [private]
```

How often the constraint solver iterates over each constraint during the Constraint Solving Step.

#### 7.11.3.2 sampleSize

```
int UnitTest_SolveBendTwistConstraint.sampleSize = 10 [private]
```

The number of value-pairs the test is executed with. E.g. if `sampleSize` is 10, then the unit test is executed with 10 randomly drawn value-pairs. A higher number needs more time to execute.

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

- [UnitTest\\_SolveBendTwistConstraint.cs](#)

## 7.12 UnitTest\_SolveStretchConstraint Class Reference

### Public Member Functions

- `IEnumerator` [PerformUnitTests](#) ()

### Private Member Functions

- `void` [PickRandomPositions](#) (out `Vector3` `particlePositionOne`, out `Vector3` `particlePositionTwo`)
- `void` [Test\\_SolveStretchConstraint](#) (int `iterations`, `Vector3` `particlePositionOne`, `Vector3` `particlePositionTwo`, `BSM.Quaternion` `orientation`, [GuidewireSim.MathHelper](#) `mathHelper`, [GuidewireSim.ConstraintSolvingStep](#) `constraintSolvingStep`)

### Private Attributes

- `float` [maximalDistanceOffset](#) = 1f
- `int` [sampleSize](#) = 10
- `int` [constraintSolverSteps](#) = 1000
 

*How often the constraint solver iterates over each constraint during the Constraint Solving Step.*
- `float` [rodElementLength](#) = 10f
 

*The distance between two spheres, also the distance between two orientations.*

### 7.12.1 Detailed Description

This class provides unit tests that test the method `SolveStretchConstraint()` of `ConstraintSolvingStep`. Executing this test once generates `sampleSize` many random value pairs and executes the unit test with each of these pairs.

### 7.12.2 Member Function Documentation

#### 7.12.2.1 PerformUnitTests()

```
IEnumerator UnitTest_SolveStretchConstraint.PerformUnitTests ( )
```

Arranges all necessary data, generates `sampleSize` many random value pairs, and then passes all data to `Test_SolveStretchConstraint()`, where the unit tests are executed.

#### 7.12.2.2 PickRandomPositions()

```
void UnitTest_SolveStretchConstraint.PickRandomPositions (
    out Vector3 particlePositionOne,
    out Vector3 particlePositionTwo ) [private]
```

Picks the first particle position uniformly distributed with  $x, y, z \in [-5, 5]$  and the second uniformly distributed around the first position with a uniformly distributed distance in  $[rodElementLength - maximalDistanceOffset, rodElementLength + maximalDistanceOffset]$ .

#### Note

The method for picking the second position is inspired by <https://math.stackexchange.com/q/50482>

#### Parameters

out	<i>particlePositionOne</i>	The first particle position that got picked.
out	<i>particlePositionTwo</i>	The second particle position that got picked.

**Requirements** Picks the first particle position uniformly distributed so that  $x, y, z \in [-5, 5]$ .

Picks a distance between the two particles that is uniformly distributed in the interval  $[rodElementLength - maximalDistanceOffset, rodElementLength + maximalDistanceOffset]$ .

Picks the second particle position uniformly distributed on the surface of the sphere with center `particlePositionOne` and radius `startDistance`.

### 7.12.2.3 Test\_SolveStretchConstraint()

```
void UnitTest_SolveStretchConstraint::Test_SolveStretchConstraint (
    int iterations,
    Vector3 particlePositionOne,
    Vector3 particlePositionTwo,
    BSM.Quaternion orientation,
    GuidewireSim.MathHelper mathHelper,
    GuidewireSim.ConstraintSolvingStep constraintSolvingStep ) [private]
```

Executes SolveStretchConstraint() of ConstraintSolvingStep `iterations` many times for one values pair, and then asserts whether the results of the algorithm of SolveStretchConstraint() converged towards the expected values.

#### Parameters

<i>iterations</i>	The number of iterations that SolveStretchConstraint() of ConstraintSolvingStep is executed.
<i>particlePositionOne</i>	The first particle position for SolveStretchConstraint().
<i>particlePositionTwo</i>	The second particle position for SolveStretchConstraint().
<i>orientation</i>	The orientation for SolveStretchConstraint().
<i>mathHelper</i>	The component MathHelper.
<i>constraintSolvingStep</i>	The component ConstraintSolvingStep.

**Requirements** `orientation` is still a unit quaternion at the end of the test.

The deviation between the stretch constraint and zero is lower than the tolerance 0.1, which means that the algorithm of SolveStretchConstraint() converges towards the fulfillment of the stretch constraint.

The deviation between the actual distance of `particlePositionOne` and `particlePositionTwo` and the rest rod element length is lower than a reasonable tolerance, i.e. close to zero.

#### Attention

The fulfillment of the requirement that the rod element length converges towards the rest rod element length depends on the initial deviation of both particle positions from each other and is just a byproduct of converging towards the constraint fulfillment. If this requirement is not fulfilled, the initial offset or the number of iterations was probably simply to high or low, respectively.

## 7.12.3 Member Data Documentation

### 7.12.3.1 constraintSolverSteps

```
int UnitTest_SolveStretchConstraint::constraintSolverSteps = 1000 [private]
```

How often the constraint solver iterates over each constraint during the Constraint Solving Step.

### 7.12.3.2 maximalDistanceOffset

```
float UnitTest_SolveStretchConstraint.maximalDistanceOffset = 1f [private]
```

The maximal deviation from the rest `rodElementLength`.

#### Example

Let `rodElementLength` be 10 and `maximalDistanceOffset` be 2. Then the two random particle positions drawn will have a distance between 8 and 12.

### 7.12.3.3 rodElementLength

```
float UnitTest_SolveStretchConstraint.rodElementLength = 10f [private]
```

The distance between two spheres, also the distance between two orientations.

### 7.12.3.4 sampleSize

```
int UnitTest_SolveStretchConstraint.sampleSize = 10 [private]
```

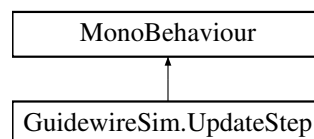
The number of value-pairs the test is executed with. E.g. if `sampleSize` is 10, then the unit test is executed with 10 randomly drawn value-pairs. A higher number needs more time to execute.

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

- [UnitTest\\_SolveStretchConstraint.cs](#)

## 7.13 GuidewireSim.UpdateStep Class Reference

Inheritance diagram for `GuidewireSim.UpdateStep`:



### Public Member Functions

- `Vector3[] UpdateSphereVelocities` (`Vector3[]` sphereVelocities, `int` spheresCount, `Vector3[]` spherePosition↔ Predictions, `Vector3[]` spherePositions)
- `Vector3[] UpdateSpherePositions` (`Vector3[]` spherePositions, `int` spheresCount, `Vector3[]` spherePosition↔ Predictions)
- `Vector3[] UpdateCylinderAngularVelocities` (`Vector3[]` cylinderAngularVelocities, `int` cylinderCount, `BSM.↔ Quaternion[]` cylinderOrientations, `BSM.Quaternion[]` cylinderOrientationPredictions)
- `BSM.Quaternion[] UpdateCylinderOrientations` (`BSM.Quaternion[]` cylinderOrientations, `int` cylinderCount, `BSM.Quaternion[]` cylinderOrientationPredictions)

## Private Member Functions

- void [Awake](#) ()

## Private Attributes

- [MathHelper](#) `mathHelper`

The component [MathHelper](#) that provides math related helper functions.

### 7.13.1 Detailed Description

This class implements the update step of the algorithm.

### 7.13.2 Member Function Documentation

#### 7.13.2.1 Awake()

```
void GuidewireSim.UpdateStep.Awake ( ) [private]
```

#### 7.13.2.2 UpdateCylinderAngularVelocities()

```
Vector3 [ ] GuidewireSim.UpdateStep.UpdateCylinderAngularVelocities (
    Vector3[] cylinderAngularVelocities,
    int cylinderCount,
    BSM.Quaternion[] cylinderOrientations,
    BSM.Quaternion[] cylinderOrientationPredictions )
```

Updates the cylinder angular velocities for the update step of the simulation.

##### Parameters

<i>cylinderAngularVelocities</i>	The angular velocity of the current frame of each orientation element/ cylinder.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.
<i>cylinderOrientationPredictions</i>	The prediction of the orientation of each cylinder at its center of mass.

##### Returns

The angular velocity of the current frame of each orientation element/ cylinder, i.e. *cylinderAngularVelocities*.

### 7.13.2.3 UpdateCylinderOrientations()

```
BSM.Quaternion [ ] GuidewireSim.UpdateStep.UpdateCylinderOrientations (
    BSM.Quaternion[] cylinderOrientations,
    int cylinderCount,
    BSM.Quaternion[] cylinderOrientationPredictions )
```

Updates the cylinder orientations given the current orientation predictions for the update step of the simulation.

#### Parameters

<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>cylinderOrientationPredictions</i>	The prediction of the orientation of each cylinder at its center of mass.

#### Returns

The orientation of each cylinder at its center of mass, i.e. *cylinderOrientations*.

### 7.13.2.4 UpdateSpherePositions()

```
Vector3 [ ] GuidewireSim.UpdateStep.UpdateSpherePositions (
    Vector3[] spherePositions,
    int spheresCount,
    Vector3[] spherePositionPredictions )
```

Updates the sphere positions given the current position predictions.

#### Parameters

<i>spherePositions</i>	The position at the current frame of each sphere.
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere (in this case of the last frame).

#### Returns

The position at the current frame of each sphere, i.e. *spherePositions*.

### 7.13.2.5 UpdateSphereVelocities()

```
Vector3 [ ] GuidewireSim.UpdateStep.UpdateSphereVelocities (
    Vector3[] sphereVelocities,
```

```
int spheresCount,  
Vector3[] spherePositionPredictions,  
Vector3[] spherePositions )
```

Updates the sphere velocities given the current prediction and the current position.

#### Parameters

<i>sphereVelocities</i>	The velocity of the current frame of each sphere.
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere (in this case of the last frame).
<i>spherePositions</i>	The position at the current frame of each sphere.

#### Returns

The velocity of the current frame of each sphere, i.e. *sphereVelocities*.

### 7.13.3 Member Data Documentation

#### 7.13.3.1 mathHelper

[MathHelper](#) GuidewireSim.UpdateStep.mathHelper [private]

The component [MathHelper](#) that provides math related helper functions.

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

- [UpdateStep.cs](#)



## Chapter 8

# File Documentation

### 8.1 ConstraintSolvingStep.cs File Reference

#### Classes

- class [GuidewireSim.ConstraintSolvingStep](#)

#### Namespaces

- namespace [GuidewireSim](#)

#### Typedefs

- using [BSM](#) = BulletSharp.Math

#### 8.1.1 Typedef Documentation

##### 8.1.1.1 BSM

```
using BSM = BulletSharp.Math
```

### 8.2 DirectorsDrawer.cs File Reference

#### Classes

- class [GuidewireSim.DirectorsDrawer](#)

## Namespaces

- namespace [GuidewireSim](#)

## 8.3 ForceTestPerformer.cs File Reference

### Classes

- class [GuidewireSim.ForceTestPerformer](#)

## Namespaces

- namespace [GuidewireSim](#)

## 8.4 InitializationStep.cs File Reference

### Classes

- class [GuidewireSim.InitializationStep](#)

## Namespaces

- namespace [GuidewireSim](#)

## Typedefs

- using [BSM](#) = BulletSharp.Math

### 8.4.1 Typedef Documentation

#### 8.4.1.1 BSM

```
using BSM = BulletSharp.Math
```

## 8.5 MathHelper.cs File Reference

### Classes

- class [GuidewireSim.MathHelper](#)

## Namespaces

- namespace [GuidewireSim](#)

## Typedefs

- using [BSM](#) = BulletSharp.Math

### 8.5.1 Typedef Documentation

#### 8.5.1.1 BSM

```
using BSM = BulletSharp.Math
```

## 8.6 ObjectSetter.cs File Reference

### Classes

- class [GuidewireSim.ObjectSetter](#)

## Namespaces

- namespace [GuidewireSim](#)

## Typedefs

- using [BSM](#) = BulletSharp.Math

### 8.6.1 Typedef Documentation

#### 8.6.1.1 BSM

```
using BSM = BulletSharp.Math
```

## 8.7 PredictionStep.cs File Reference

### Classes

- class [GuidewireSim.PredictionStep](#)

## Namespaces

- namespace [GuidewireSim](#)

## Typedefs

- using [BSM](#) = BulletSharp.Math

### 8.7.1 Typedef Documentation

#### 8.7.1.1 BSM

```
using BSM = BulletSharp.Math
```

## 8.8 SimulationLoop.cs File Reference

### Classes

- class [GuidewireSim.SimulationLoop](#)

## Namespaces

- namespace [GuidewireSim](#)

## Typedefs

- using [BSM](#) = BulletSharp.Math

### 8.8.1 Typedef Documentation

#### 8.8.1.1 BSM

```
using BSM = BulletSharp.Math
```

## 8.9 StressTestPerformer.cs File Reference

### Classes

- class [GuidewireSim.StressTestPerformer](#)

## Namespaces

- namespace [GuidewireSim](#)

## 8.10 TorqueTestPerformer.cs File Reference

### Classes

- class [GuidewireSim.TorqueTestPerformer](#)

## Namespaces

- namespace [GuidewireSim](#)

## 8.11 UnitTest\_SolveBendTwistConstraint.cs File Reference

### Classes

- class [UnitTest\\_SolveBendTwistConstraint](#)

### Typedefs

- using [BSM](#) = BulletSharp.Math

#### 8.11.1 Typedef Documentation

##### 8.11.1.1 BSM

```
using BSM = BulletSharp.Math
```

## 8.12 UnitTest\_SolveStretchConstraint.cs File Reference

### Classes

- class [UnitTest\\_SolveStretchConstraint](#)

### Typedefs

- using [BSM](#) = BulletSharp.Math

### 8.12.1 Typedef Documentation

#### 8.12.1.1 BSM

```
using BSM = BulletSharp.Math
```

## 8.13 UpdateStep.cs File Reference

### Classes

- class [GuidewireSim.UpdateStep](#)

### Namespaces

- namespace [GuidewireSim](#)

### Typedefs

- using [BSM](#) = BulletSharp.Math

### 8.13.1 Typedef Documentation

#### 8.13.1.1 BSM

```
using BSM = BulletSharp.Math
```

# Index

- AdoptCalculations
  - GuidewireSim.SimulationLoop, [58](#)
- arrowHeadAngle
  - GuidewireSim.DirectorsDrawer, [27](#)
- arrowHeadPercentage
  - GuidewireSim.DirectorsDrawer, [27](#)
- Awake
  - GuidewireSim.ConstraintSolvingStep, [16](#)
  - GuidewireSim.DirectorsDrawer, [25](#)
  - GuidewireSim.ForceTestPerformer, [30](#)
  - GuidewireSim.InitializationStep, [33](#)
  - GuidewireSim.ObjectSetter, [50](#)
  - GuidewireSim.PredictionStep, [53](#)
  - GuidewireSim.SimulationLoop, [58](#)
  - GuidewireSim.StressTestPerformer, [67](#)
  - GuidewireSim.TorqueTestPerformer, [69](#)
  - GuidewireSim.UpdateStep, [77](#)
- BendTwistConstraintDeviation
  - GuidewireSim.MathHelper, [41](#)
- BSM
  - ConstraintSolvingStep.cs, [81](#)
  - InitializationStep.cs, [82](#)
  - MathHelper.cs, [83](#)
  - ObjectSetter.cs, [83](#)
  - PredictionStep.cs, [84](#)
  - SimulationLoop.cs, [84](#)
  - UnitTest\_SolveBendTwistConstraint.cs, [85](#)
  - UnitTest\_SolveStretchConstraint.cs, [86](#)
  - UpdateStep.cs, [86](#)
- CalculateArrowHeadPositions
  - GuidewireSim.DirectorsDrawer, [26](#)
- CalculateCylinderPositions
  - GuidewireSim.MathHelper, [41](#)
- ConstraintSolverSteps
  - GuidewireSim.SimulationLoop, [65](#)
- constraintSolverSteps
  - UnitTest\_SolveBendTwistConstraint, [73](#)
  - UnitTest\_SolveStretchConstraint, [75](#)
- constraintSolvingStep
  - GuidewireSim.SimulationLoop, [60](#)
- ConstraintSolvingStep.cs, [81](#)
  - BSM, [81](#)
- CorrectBendTwistPredictions
  - GuidewireSim.ConstraintSolvingStep, [16](#)
- CorrectStretchPredictions
  - GuidewireSim.ConstraintSolvingStep, [17](#)
- cylinderAngularVelocities
  - GuidewireSim.SimulationLoop, [60](#)
- CylinderCount
  - GuidewireSim.SimulationLoop, [65](#)
- cylinderExternalTorques
  - GuidewireSim.SimulationLoop, [60](#)
- cylinderOrientationPredictions
  - GuidewireSim.SimulationLoop, [61](#)
- cylinderOrientations
  - GuidewireSim.SimulationLoop, [61](#)
- cylinderPositions
  - GuidewireSim.SimulationLoop, [61](#)
- cylinders
  - GuidewireSim.SimulationLoop, [61](#)
- cylinderScalarWeights
  - GuidewireSim.SimulationLoop, [61](#)
- DarbouxSignFactor
  - GuidewireSim.MathHelper, [42](#)
- deltaOrientation
  - GuidewireSim.ConstraintSolvingStep, [23](#)
- deltaOrientationOne
  - GuidewireSim.ConstraintSolvingStep, [23](#)
- deltaOrientationTwo
  - GuidewireSim.ConstraintSolvingStep, [24](#)
- deltaPositionOne
  - GuidewireSim.ConstraintSolvingStep, [24](#)
- deltaPositionTwo
  - GuidewireSim.ConstraintSolvingStep, [24](#)
- directorColors
  - GuidewireSim.DirectorsDrawer, [28](#)
- directorOneColor
  - GuidewireSim.DirectorsDrawer, [28](#)
- directors
  - GuidewireSim.SimulationLoop, [62](#)
- DirectorsDrawer.cs, [81](#)
- directorThreeColor
  - GuidewireSim.DirectorsDrawer, [28](#)
- directorTwoColor
  - GuidewireSim.DirectorsDrawer, [28](#)
- DiscreteDarbouxVector
  - GuidewireSim.MathHelper, [42](#)
- discreteRestDarbouxVectors
  - GuidewireSim.SimulationLoop, [62](#)
- doForceTestFour
  - GuidewireSim.ForceTestPerformer, [31](#)
- doForceTestOne
  - GuidewireSim.ForceTestPerformer, [31](#)
- doForceTestThree
  - GuidewireSim.ForceTestPerformer, [31](#)
- doForceTestTwo
  - GuidewireSim.ForceTestPerformer, [32](#)

- doSingleLoopTest
  - GuidewireSim.ForceTestPerformer, 32
- doStressTestOne
  - GuidewireSim.StressTestPerformer, 68
- doTorqueTestOne
  - GuidewireSim.TorqueTestPerformer, 70
- doTorqueTestThree
  - GuidewireSim.TorqueTestPerformer, 70
- doTorqueTestTwo
  - GuidewireSim.TorqueTestPerformer, 71
- DrawArrowHeadConnectionLines
  - GuidewireSim.DirectorsDrawer, 26
- DrawArrowHeadLines
  - GuidewireSim.DirectorsDrawer, 26
- DrawDirectors
  - GuidewireSim.DirectorsDrawer, 27
- EmbeddedVector
  - GuidewireSim.MathHelper, 43
- executeInBilateralOrder
  - GuidewireSim.ConstraintSolvingStep, 24
- ExecuteSingleLoopTest
  - GuidewireSim.SimulationLoop, 66
- FixedUpdate
  - GuidewireSim.SimulationLoop, 58
- ForceTestPerformer.cs, 82
- GetGaussianRandomNumber
  - GuidewireSim.MathHelper, 43
- GuidewireSim, 13
- GuidewireSim.ConstraintSolvingStep, 15
  - Awake, 16
  - CorrectBendTwistPredictions, 16
  - CorrectStretchPredictions, 17
  - deltaOrientation, 23
  - deltaOrientationOne, 23
  - deltaOrientationTwo, 24
  - deltaPositionOne, 24
  - deltaPositionTwo, 24
  - executeInBilateralOrder, 24
  - mathHelper, 24
  - SolveBendTwistConstraint, 17
  - SolveBendTwistConstraints, 18
  - SolveBendTwistConstraintsInBilateralOrder, 19
  - SolveBendTwistConstraintsInNaiveOrder, 20
  - SolveStretchConstraint, 20
  - SolveStretchConstraints, 21
  - SolveStretchConstraintsInBilateralOrder, 22
  - SolveStretchConstraintsInNaiveOrder, 23
- GuidewireSim.DirectorsDrawer, 25
  - arrowHeadAngle, 27
  - arrowHeadPercentage, 27
  - Awake, 25
  - CalculateArrowHeadPositions, 26
  - directorColors, 28
  - directorOneColor, 28
  - directorThreeColor, 28
  - directorTwoColor, 28
- DrawArrowHeadConnectionLines, 26
- DrawArrowHeadLines, 26
- DrawDirectors, 27
  - scaleFactor, 28
  - simulationLoop, 29
  - Update, 27
- GuidewireSim.ForceTestPerformer, 29
  - Awake, 30
  - doForceTestFour, 31
  - doForceTestOne, 31
  - doForceTestThree, 31
  - doForceTestTwo, 32
  - doSingleLoopTest, 32
  - PerformForceTestFour, 30
  - PerformForceTestOne, 30
  - PerformForceTests, 30
  - PerformForceTestThree, 30
  - PerformForceTestTwo, 31
  - PerformSingleLoopTest, 31
  - pullForceTestThree, 32
  - simulationLoop, 32
  - Start, 31
- GuidewireSim.InitializationStep, 32
  - Awake, 33
  - InitCylinderAngularVelocities, 33
  - InitCylinderExternalTorques, 34
  - InitCylinderOrientationPredictions, 34
  - InitCylinderOrientations, 34
  - InitCylinderPositions, 35
  - InitCylinderScalarWeights, 35
  - InitDirectors, 35
  - InitDiscreteRestDarbouxVectors, 36
  - InitInertiaTensor, 36
  - InitInverseInertiaTensor, 37
  - InitSphereExternalForces, 37
  - InitSphereInverseMasses, 37
  - InitSpherePositionPredictions, 38
  - InitSpherePositions, 38
  - InitSphereVelocities, 38
  - InitWorldSpaceBasis, 39
  - materialDensity, 39
  - materialRadius, 39
  - mathHelper, 39
- GuidewireSim.MathHelper, 40
  - BendTwistConstraintDeviation, 41
  - CalculateCylinderPositions, 41
  - DarbouxSignFactor, 42
  - DiscreteDarbouxVector, 42
  - EmbeddedVector, 43
  - GetGaussianRandomNumber, 43
  - ImaginaryPart, 44
  - MatrixVectorMultiplication, 44
  - QuaternionConversionFromBSM, 45
  - QuaternionConversionToBSM, 45
  - QuaternionLength, 45
  - RandomUnitQuaternion, 46
  - RodElementLength, 46
  - RodElementLengthDeviation, 46

- SquaredNorm, [47](#)
- StretchConstraintDeviation, [47](#)
- UpdateDirectors, [48](#)
- VectorLength, [48](#)
- GuidewireSim.ObjectSetter, [49](#)
  - Awake, [50](#)
  - mathHelper, [52](#)
  - SetCylinderOrientations, [50](#)
  - SetCylinderPositions, [50](#)
  - SetSpherePositions, [52](#)
- GuidewireSim.PredictionStep, [52](#)
  - Awake, [53](#)
  - mathHelper, [55](#)
  - PredictAngularVelocities, [53](#)
  - PredictCylinderOrientations, [54](#)
  - PredictSpherePositions, [54](#)
  - PredictSphereVelocities, [55](#)
- GuidewireSim.SimulationLoop, [56](#)
  - AdoptCalculations, [58](#)
  - Awake, [58](#)
  - ConstraintSolverSteps, [65](#)
  - constraintSolvingStep, [60](#)
  - cylinderAngularVelocities, [60](#)
  - CylinderCount, [65](#)
  - cylinderExternalTorques, [60](#)
  - cylinderOrientationPredictions, [61](#)
  - cylinderOrientations, [61](#)
  - cylinderPositions, [61](#)
  - cylinders, [61](#)
  - cylinderScalarWeights, [61](#)
  - directors, [62](#)
  - discreteRestDarbouxVectors, [62](#)
  - ExecuteSingleLoopTest, [66](#)
  - FixedUpdate, [58](#)
  - inertiaTensor, [62](#)
  - initializationStep, [62](#)
  - inverseInertiaTensor, [63](#)
  - mathHelper, [63](#)
  - objectSetter, [63](#)
  - PerformConstraintSolvingStep, [58](#)
  - PerformInitializationStep, [59](#)
  - PerformPredictionStep, [59](#)
  - PerformSimulationLoop, [59](#)
  - PerformUpdateStep, [60](#)
  - predictionStep, [63](#)
  - rodElementLength, [63](#)
  - solveBendTwistConstraints, [63](#)
  - solveStretchConstraints, [64](#)
  - sphereExternalForces, [64](#)
  - sphereInverseMasses, [64](#)
  - spherePositionPredictions, [64](#)
  - spherePositions, [64](#)
  - spheres, [64](#)
  - SpheresCount, [66](#)
  - sphereVelocities, [65](#)
  - Start, [60](#)
  - updateStep, [65](#)
  - worldSpaceBasis, [65](#)
- GuidewireSim.StressTestPerformer, [66](#)
  - Awake, [67](#)
  - doStressTestOne, [68](#)
  - PerformStressTestOne, [67](#)
  - PerformStressTests, [67](#)
  - simulationLoop, [68](#)
  - Start, [67](#)
- GuidewireSim.TorqueTestPerformer, [68](#)
  - Awake, [69](#)
  - doTorqueTestOne, [70](#)
  - doTorqueTestThree, [70](#)
  - doTorqueTestTwo, [71](#)
  - PerformTorqueTestOne, [69](#)
  - PerformTorqueTests, [69](#)
  - PerformTorqueTestThree, [69](#)
  - PerformTorqueTestTwo, [70](#)
  - pullTorque, [71](#)
  - simulationLoop, [71](#)
  - Start, [70](#)
- GuidewireSim.UpdateStep, [76](#)
  - Awake, [77](#)
  - mathHelper, [79](#)
  - UpdateCylinderAngularVelocities, [77](#)
  - UpdateCylinderOrientations, [77](#)
  - UpdateSpherePositions, [78](#)
  - UpdateSphereVelocities, [78](#)
- ImaginaryPart
  - GuidewireSim.MathHelper, [44](#)
- inertiaTensor
  - GuidewireSim.SimulationLoop, [62](#)
- InitCylinderAngularVelocities
  - GuidewireSim.InitializationStep, [33](#)
- InitCylinderExternalTorques
  - GuidewireSim.InitializationStep, [34](#)
- InitCylinderOrientationPredictions
  - GuidewireSim.InitializationStep, [34](#)
- InitCylinderOrientations
  - GuidewireSim.InitializationStep, [34](#)
- InitCylinderPositions
  - GuidewireSim.InitializationStep, [35](#)
- InitCylinderScalarWeights
  - GuidewireSim.InitializationStep, [35](#)
- InitDirectors
  - GuidewireSim.InitializationStep, [35](#)
- InitDiscreteRestDarbouxVectors
  - GuidewireSim.InitializationStep, [36](#)
- initializationStep
  - GuidewireSim.SimulationLoop, [62](#)
- InitializationStep.cs, [82](#)
  - BSM, [82](#)
- InitInertiaTensor
  - GuidewireSim.InitializationStep, [36](#)
- InitInverseInertiaTensor
  - GuidewireSim.InitializationStep, [37](#)
- InitSphereExternalForces
  - GuidewireSim.InitializationStep, [37](#)
- InitSphereInverseMasses
  - GuidewireSim.InitializationStep, [37](#)

- InitSpherePositionPredictions
  - GuidewireSim.InitializationStep, 38
- InitSpherePositions
  - GuidewireSim.InitializationStep, 38
- InitSphereVelocities
  - GuidewireSim.InitializationStep, 38
- InitWorldSpaceBasis
  - GuidewireSim.InitializationStep, 39
- inverseInertiaTensor
  - GuidewireSim.SimulationLoop, 63
- materialDensity
  - GuidewireSim.InitializationStep, 39
- materialRadius
  - GuidewireSim.InitializationStep, 39
- mathHelper
  - GuidewireSim.ConstraintSolvingStep, 24
  - GuidewireSim.InitializationStep, 39
  - GuidewireSim.ObjectSetter, 52
  - GuidewireSim.PredictionStep, 55
  - GuidewireSim.SimulationLoop, 63
  - GuidewireSim.UpdateStep, 79
- MathHelper.cs, 82
  - BSM, 83
- MatrixVectorMultiplication
  - GuidewireSim.MathHelper, 44
- maximalDistanceOffset
  - UnitTest\_SolveStretchConstraint, 75
- objectSetter
  - GuidewireSim.SimulationLoop, 63
- ObjectSetter.cs, 83
  - BSM, 83
- PerformConstraintSolvingStep
  - GuidewireSim.SimulationLoop, 58
- PerformForceTestFour
  - GuidewireSim.ForceTestPerformer, 30
- PerformForceTestOne
  - GuidewireSim.ForceTestPerformer, 30
- PerformForceTests
  - GuidewireSim.ForceTestPerformer, 30
- PerformForceTestThree
  - GuidewireSim.ForceTestPerformer, 30
- PerformForceTestTwo
  - GuidewireSim.ForceTestPerformer, 31
- PerformInitializationStep
  - GuidewireSim.SimulationLoop, 59
- PerformPredictionStep
  - GuidewireSim.SimulationLoop, 59
- PerformSimulationLoop
  - GuidewireSim.SimulationLoop, 59
- PerformSingleLoopTest
  - GuidewireSim.ForceTestPerformer, 31
- PerformStressTestOne
  - GuidewireSim.StressTestPerformer, 67
- PerformStressTests
  - GuidewireSim.StressTestPerformer, 67
- PerformTorqueTestOne
  - GuidewireSim.TorqueTestPerformer, 69
- PerformTorqueTests
  - GuidewireSim.TorqueTestPerformer, 69
- PerformTorqueTestThree
  - GuidewireSim.TorqueTestPerformer, 69
- PerformTorqueTestTwo
  - GuidewireSim.TorqueTestPerformer, 70
- PerformUnitTests
  - UnitTest\_SolveBendTwistConstraint, 72
  - UnitTest\_SolveStretchConstraint, 74
- PerformUpdateStep
  - GuidewireSim.SimulationLoop, 60
- PickRandomPositions
  - UnitTest\_SolveStretchConstraint, 74
- PredictAngularVelocities
  - GuidewireSim.PredictionStep, 53
- PredictCylinderOrientations
  - GuidewireSim.PredictionStep, 54
- predictionStep
  - GuidewireSim.SimulationLoop, 63
- PredictionStep.cs, 83
  - BSM, 84
- PredictSpherePositions
  - GuidewireSim.PredictionStep, 54
- PredictSphereVelocities
  - GuidewireSim.PredictionStep, 55
- pullForceTestThree
  - GuidewireSim.ForceTestPerformer, 32
- pullTorque
  - GuidewireSim.TorqueTestPerformer, 71
- QuaternionConversionFromBSM
  - GuidewireSim.MathHelper, 45
- QuaternionConversionToBSM
  - GuidewireSim.MathHelper, 45
- QuaternionLength
  - GuidewireSim.MathHelper, 45
- RandomUnitQuaternion
  - GuidewireSim.MathHelper, 46
- RodElementLength
  - GuidewireSim.MathHelper, 46
- rodElementLength
  - GuidewireSim.SimulationLoop, 63
  - UnitTest\_SolveStretchConstraint, 76
- RodElementLengthDeviation
  - GuidewireSim.MathHelper, 46
- sampleSize
  - UnitTest\_SolveBendTwistConstraint, 73
  - UnitTest\_SolveStretchConstraint, 76
- scaleFactor
  - GuidewireSim.DirectorsDrawer, 28
- SetCylinderOrientations
  - GuidewireSim.ObjectSetter, 50
- SetCylinderPositions
  - GuidewireSim.ObjectSetter, 50
- SetSpherePositions
  - GuidewireSim.ObjectSetter, 52

- simulationLoop
  - GuidewireSim.DirectorsDrawer, 29
  - GuidewireSim.ForceTestPerformer, 32
  - GuidewireSim.StressTestPerformer, 68
  - GuidewireSim.TorqueTestPerformer, 71
- SimulationLoop.cs, 84
  - BSM, 84
- SolveBendTwistConstraint
  - GuidewireSim.ConstraintSolvingStep, 17
- SolveBendTwistConstraints
  - GuidewireSim.ConstraintSolvingStep, 18
- solveBendTwistConstraints
  - GuidewireSim.SimulationLoop, 63
- SolveBendTwistConstraintsInBilateralOrder
  - GuidewireSim.ConstraintSolvingStep, 19
- SolveBendTwistConstraintsInNaiveOrder
  - GuidewireSim.ConstraintSolvingStep, 20
- SolveStretchConstraint
  - GuidewireSim.ConstraintSolvingStep, 20
- SolveStretchConstraints
  - GuidewireSim.ConstraintSolvingStep, 21
- solveStretchConstraints
  - GuidewireSim.SimulationLoop, 64
- SolveStretchConstraintsInBilateralOrder
  - GuidewireSim.ConstraintSolvingStep, 22
- SolveStretchConstraintsInNaiveOrder
  - GuidewireSim.ConstraintSolvingStep, 23
- sphereExternalForces
  - GuidewireSim.SimulationLoop, 64
- sphereInverseMasses
  - GuidewireSim.SimulationLoop, 64
- spherePositionPredictions
  - GuidewireSim.SimulationLoop, 64
- spherePositions
  - GuidewireSim.SimulationLoop, 64
- spheres
  - GuidewireSim.SimulationLoop, 64
- SpheresCount
  - GuidewireSim.SimulationLoop, 66
- sphereVelocities
  - GuidewireSim.SimulationLoop, 65
- SquaredNorm
  - GuidewireSim.MathHelper, 47
- Start
  - GuidewireSim.ForceTestPerformer, 31
  - GuidewireSim.SimulationLoop, 60
  - GuidewireSim.StressTestPerformer, 67
  - GuidewireSim.TorqueTestPerformer, 70
- StressTestPerformer.cs, 84
- StretchConstraintDeviation
  - GuidewireSim.MathHelper, 47
- Test\_SolveBendTwistConstraint
  - UnitTest\_SolveBendTwistConstraint, 72
- Test\_SolveStretchConstraint
  - UnitTest\_SolveStretchConstraint, 74
- TorqueTestPerformer.cs, 85
- UnitTest\_SolveBendTwistConstraint, 71
  - constraintSolverSteps, 73
  - PerformUnitTests, 72
  - sampleSize, 73
  - Test\_SolveBendTwistConstraint, 72
- UnitTest\_SolveBendTwistConstraint.cs, 85
  - BSM, 85
- UnitTest\_SolveStretchConstraint, 73
  - constraintSolverSteps, 75
  - maximalDistanceOffset, 75
  - PerformUnitTests, 74
  - PickRandomPositions, 74
  - rodElementLength, 76
  - sampleSize, 76
  - Test\_SolveStretchConstraint, 74
- UnitTest\_SolveStretchConstraint.cs, 85
  - BSM, 86
- Update
  - GuidewireSim.DirectorsDrawer, 27
- UpdateCylinderAngularVelocities
  - GuidewireSim.UpdateStep, 77
- UpdateCylinderOrientations
  - GuidewireSim.UpdateStep, 77
- UpdateDirectors
  - GuidewireSim.MathHelper, 48
- UpdateSpherePositions
  - GuidewireSim.UpdateStep, 78
- UpdateSphereVelocities
  - GuidewireSim.UpdateStep, 78
- updateStep
  - GuidewireSim.SimulationLoop, 65
- UpdateStep.cs, 86
  - BSM, 86
- VectorLength
  - GuidewireSim.MathHelper, 48
- worldSpaceBasis
  - GuidewireSim.SimulationLoop, 65