OpenSim::Blankevoort1991Ligament Class Reference

This class implements a nonlinear spring ligament model introduced by Blankevoort et al. (1991) [1] and further described in Smith et al. (2016) [2]. More...

Inheritance diagram for OpenSim::Blankevoort1991Ligament:

OpenSim Properties, Sockets, Outputs, Inputs

Properties (unnamed)

GeometryPath	**GeometryPath** "The set of points defining the path of the ligament" More...

Properties (single-value)

double	**linear_stiffness** "The slope of the linear region of the force-strain curve. " "Units of force/strain (N)." More...
double	**transition_strain** "The strain at which the ligament force-strain curve transitions from " "quadratic to linear. Units of strain. Default value of 0.06 (6%)." More...
double	**damping_coefficient** "The coefficient that multiplies the strain rate when computing the " " damping force. Units of N*s/strain. Default value of 0.003." More...
double	**slack_length** "The length at which ligament begins developing tension. Units of m." More...

Outputs

double	**spring_force** Provides the value of **getSpringForce()** and is available at stage SimTK::Stage::Position . More...
double	**damping_force** Provides the value of **getDampingForce()** and is available at stage SimTK::Stage::Velocity . More...
double	**total_force** Provides the value of **getTotalForce()** and is available at stage SimTK::Stage::Velocity . More...
double	**strain** Provides the value of **getStrain()** and is available at stage SimTK::Stage::Position . More...
double	**strain_rate** Provides the value of **getStrainRate()** and is available at stage SimTK::Stage::Velocity . More...
double	**length** Provides the value of **getLength()** and is available at stage SimTK::Stage::Position . More...
double	**lengthening_speed** Provides the value of **getLengtheningSpeed()** and is available at stage SimTK::Stage::Velocity . More...

OpenSim Properties, Sockets, Outputs, Inputs inherited from OpenSim::Force

OpenSim Properties, Sockets, Outputs, Inputs inherited from OpenSim::Component

Public Member Functions

	Blankevoort1991Ligament ()
	Blankevoort1991Ligament (std::string name, const **PhysicalFrame** &frame1, SimTK::Vec3 point1, const **PhysicalFrame** &frame2, SimTK::Vec3 point2)
	Blankevoort1991Ligament (std::string name, const **PhysicalFrame** &frame1, SimTK::Vec3 point1, const **PhysicalFrame** &frame2, SimTK::Vec3 point2, double **linear_stiffness**, double **slack_length**)
	Blankevoort1991Ligament (std::string name, double **linear_stiffness**, double **slack_length**)
void	**setSlackLengthFromReferenceStrain** (double **strain**, const SimTK::State &reference_state) Set the slack_length property using the strain in the ligament at a known pose (reference state). More...
void	**setSlackLengthFromReferenceForce** (double force, const SimTK::State &reference_state) Set the slack_length property using the absolute spring force (N) in the ligament at a known pose (reference force). More...
void	**setLinearStiffnessForcePerLength** (double **linear_stiffness**) Set the linear_stiffness property using a value in units of force/length (N/m). More...
void	**setDampingCoefficientForceTimePerLength** (double **damping_coefficient**) Set the damping_coefficient property using a value in units of force*time/length (N*s/m). More...
double	**getStrain** (const SimTK::State &state) const
double	**getStrainRate** (const SimTK::State &state) const
double	**getLength** (const SimTK::State &state) const
double	**getLengtheningSpeed** (const SimTK::State &state) const
double	**getSpringForce** (const SimTK::State &state) const
double	**getDampingForce** (const SimTK::State &state) const
double	**getTotalForce** (const SimTK::State &state) const

double	**getLinearStiffnessForcePerLength** () const Get the linear_stiffness property in units of force/length (N/m) More...
double	**getTransitionLength** () const Get the length (m) of the ligament where the model transitions from the toe region to the linear region. More...
double	**getDampingCoefficientForceTimePerLength** () const Get the damping_coefficient in units of force*time/length (N*s/m) More...
double	**computeMomentArm** (const SimTK::State &s, **Coordinate** &aCoord) const
void	**computeForce** (const SimTK::State &s, SimTK::Vector_< **SimTK::SpatialVec** > &bodyForces, SimTK::Vector &generalizedForces) const override Subclasses must implement this method to compute the forces that should be applied to bodies and generalized speeds. More...
double	**computePotentialEnergy** (const SimTK::State &state) const override Subclasses may optionally override this method to compute a contribution to the potential energy of the system. More...
void	**extendPostScale** (const SimTK::State &s, const **ScaleSet** &scaleSet) override Perform any computations that must occur after **ModelComponent::scale()** has been invoked on all ModelComponents in the **Model**. More...
OpenSim::Array< std::string >	**getRecordLabels** () const override Methods to query a **Force** for the value actually applied during simulation. More...
OpenSim::Array< double >	**getRecordValues** (const SimTK::State &state) const override Given SimTK::State object extract all the values necessary to report forces, application location frame, etc. More...

Property-related functions

const **GeometryPath** &	**get_GeometryPath** () const Get the value of the **GeometryPath** property. More...
GeometryPath &	**upd_GeometryPath** () Get a writable reference to the **GeometryPath** property. More...
void	**set_GeometryPath** (const **GeometryPath** &value) Set the value of the **GeometryPath** property. More...
const double &	**get_linear_stiffness** () const Get the value of the **linear_stiffness** property. More...
double &	**upd_linear_stiffness** () Get a writable reference to the **linear_stiffness** property. More...
void	**set_linear_stiffness** (const double &value) Set the value of the **linear_stiffness** property. More...
const double &	**get_transition_strain** () const Get the value of the **transition_strain** property. More...
double &	**upd_transition_strain** () Get a writable reference to the **transition_strain** property. More...
void	**set_transition_strain** (const double &value) Set the value of the **transition_strain** property. More...
const double &	**get_damping_coefficient** () const Get the value of the **damping_coefficient** property. More...
double &	**upd_damping_coefficient** () Get a writable reference to the **damping_coefficient** property. More...
void	**set_damping_coefficient** (const double &value) Set the value of the **damping_coefficient** property. More...
const double &	**get_slack_length** () const Get the value of the **slack_length** property. More...
double &	**upd_slack_length** () Get a writable reference to the **slack_length** property. More...
void	**set_slack_length** (const double &value) Set the value of the **slack_length** property. More...

- **Public Member Functions inherited from OpenSim::Force**
- **Public Member Functions inherited from OpenSim::ModelComponent**
- **Public Member Functions inherited from OpenSim::Component**
- **Public Member Functions inherited from OpenSim::Object**

Auto-generated functions

static **Blankevoort1991Ligament** *	**safeDownCast** (**OpenSim::Object** *obj) For use in MATLAB and Python to access the concrete class. More...
static const std::string &	**getClassName** () This returns "Blankevoort1991Ligament". More...

Blankevoort1991Ligament *	**clone** () const override Create a new heap-allocated copy of the concrete object to which this Object refers. More...
const std::string &	**getConcreteClassName** () const override Returns the class name of the concrete Object-derived class of the actual object referenced by this Object, as a string. More...

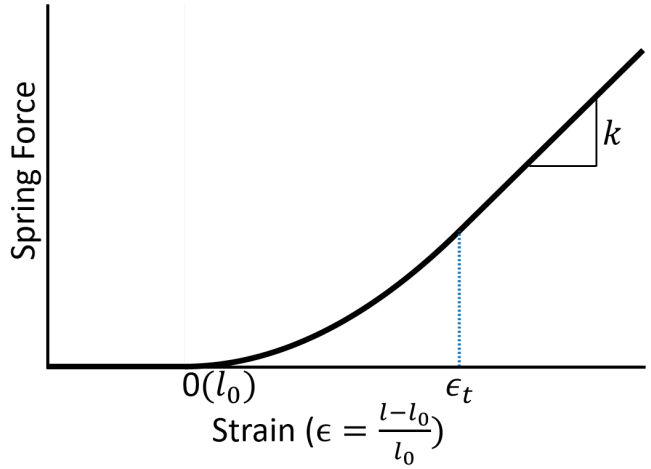
Additional Inherited Members

- **Static Public Member Functions inherited from OpenSim::Force**
- **Static Public Member Functions inherited from OpenSim::ModelComponent**
- **Static Public Member Functions inherited from OpenSim::Component**
- **Static Public Member Functions inherited from OpenSim::Object**

Detailed Description

This class implements a nonlinear spring ligament model introduced by Blankevoort et al. (1991) [1] and further described in Smith et al. (2016) [2].

This model is partially based on the formulation orginally proposed by Wismans et al. (1980) [3]. The ligament is represented as a passive spring with the force-strain relationship described by a quadratic "toe" region at low strains and a linear region at high strains. The toe region represents the uncrimping and alignment of collagen fibers and the linear region represents the subsequent stretching of the aligned fibers. The ligament model also includes a damping force that is only applied if the ligament is stretched beyond the slack length and if the ligament is lengthening. The length of the ligament is l.



Governing Equations

Spring Force:

$$F_{\text{spring}} = \left\{ \begin{array}{cc} 0 & \epsilon < 0 \\ \frac{1}{2\epsilon_t} k\epsilon^2 & 0 \leq \epsilon \leq \epsilon_t \\ k(\epsilon - \frac{\epsilon_t}{2}) & \epsilon > \epsilon_t \end{array} \right\}$$

Damping Force:

$$F_{\text{damping}} = \left\{ \begin{array}{cc} c \cdot \dot{\epsilon} & \epsilon > 0 \, and \, \dot{\epsilon} > 0 \\ 0 & otherwise \end{array} \right\}$$

Total Force:

$$F_{\text{total}} = F_{\text{spring}} + F_{\text{damping}}$$

This **Force** component has the following properties:

- linear stiffness (k): The force/strain (e.g. N) stiffness of the linear region of the ligament model.
- slack_length (l_0): The resting length of the ligament (e.g. m).
- damping coefficient (c): Damping coefficient used in the damping force calculation in units of force*time/strain (e.g. N*s/strain). The default value is 0.003.
- transition_strain (ϵ_t): The strain value where the ligament model transitions from the quadratic toe region to the linear stiffness region. The default value is 0.06 (6%) according to Blankevoort (1991) [1]. This value is widely used in the multibody knee modeling literature [2,4,5,6] and also agrees with some experimental studies [7]. However, other literature suggests the transition strain of ligaments occurs at around 0.03 (3%) strain [8,9]. In reality, the transition strain is likely dependent on the strain rate [10,11], however this effect is not included in this implementation.

The **Blankevoort1991Ligament** implementation is intended to be compatible with common methods in the literature for parameterizing ligament properties. The zero-load length of the ligament is parameterized by the slack_length property, which can be set directly, in meters, using **set_slack_length()**, or using **setSlackLengthFromReferenceStrain()** and **setSlackLengthFromReferenceForce()**. Here, reference strain and reference force are the strain or force in the ligament at a reference pose (state). If you want to compute the strain or force of the ligament in a given pose (state), you can use the **getStrain()** and getForce() methods. The linear_stiffness property has units of force/strain (newton) but can be set and obtained in units of force/length (newton/meter) using **setLinearStiffnessForcePerLength()** and **getLinearStiffnessForcePerLength()**.

When scaling a model (using the **ScaleTool**) that contains a **Blankevoort1991Ligament**, the slack_length property is scaled by the ratio of the entire **GeometryPath** length in the default model pose before and after scaling the bone geometries. This ensures that the strain in the ligament in the default pose is equivilent before and after scaling. Thus, it is important to consider the order of scaling the model and setting the slack_length property for your specific application. The linear_stiffness property is not affected by scaling the model.

References

[1] Blankevoort, L. and Huiskes, R., (1991). Ligament-bone interaction in a three-dimensional model of the knee. J Biomech Eng, 113(3), 263-269

[2] Smith, C.R., Lenhart, R.L., Kaiser, J., Vignos, M.F. and Thelen, D.G., (2016). Influence of ligament properties on tibiofemoral mechanics in walking. J Knee Surg, 29(02), 99-106.

[3] Wismans, J.A.C., Veldpaus, F., Janssen, J., Huson, A. and Struben, P., (1980). A three-dimensional mathematical model of the knee-joint. J Biomech, 13(8), 677-685.

[4] Marra, M. A., Vanheule, V., Fluit, R., Koopman, B. H., Rasmussen, J., Verdonschot, N., & Andersen, M. S. (2015). A subject-specific musculoskeletal modeling framework to predict in vivo mechanics of total knee arthroplasty. Journal of biomechanical engineering, 137(2), 020904.

[5] Guess, T. M., Razu, S., & Jahandar, H. (2016). Evaluation of knee ligament mechanics using computational models. The journal of knee surgery, 29(02), 126-137.

[6] Li, G., Gil, J., Kanamori, A., & Woo, S. Y. (1999). A validated three-dimensional computational model of a human knee joint. Journal of biomechanical engineering, 121(6), 657-662.

[7] Ristaniemi, A., Stenroth, L., Mikkonen, S., & Korhonen, R. K. (2018). Comparison of elastic, viscoelastic and failure tensile material properties of knee ligaments and patellar tendon. Journal of biomechanics, 79, 31-38.

[8] Weiss, J. A., & Gardiner, J. C. (2001). Computational modeling of ligament mechanics. Critical Reviews in Biomedical Engineering, 29(3).

[9] Martin, R. B., Burr, D. B., Sharkey, N. A., & Fyhrie, D. P. (2015). Mechanical properties of ligament and tendon. In Skeletal Tissue Mechanics (pp. 175-225). Springer, New York, NY.

[10] Pioletti, D. P., Rakotomanana, L. R., Benvenuti, J. F., & Leyvraz, P. F. (1998). Viscoelastic constitutive law in large deformations: application to human knee ligaments and tendons. Journal of biomechanics, 31(8), 753-757.

[11] Pioletti, D. P., Rakotomanana, L. R., & Leyvraz, P. F. (1999). Strain rate effect on the mechanical behavior of the anterior cruciate ligament-bone complex. Medical Engineering & Physics, 21(2), 95-100.

[12] Galbusera, F., Freutel, M., D&#65533;rselen, L., D&#65533;Aiuto, M., Croce, D., Villa, T., Sansone, V. & Innocenti, B. (2014). Material models and properties in the finite element analysis of knee ligaments: a literature review. Frontiers in bioengineering and biotechnology, 2, 54.

Author

Colin Smith

Constructor & Destructor Documentation

◆ Blankevoort1991Ligament() [1/4]

```
OpenSim::Blankevoort1991Ligament::Blankevoort1991Ligament ( )
```

◆ Blankevoort1991Ligament() [2/4]

```
OpenSim::Blankevoort1991Ligament::Blankevoort1991Ligament ( std::string                name,
                                                            const PhysicalFrame &  frame1,
                                                            SimTK::Vec3            point1,
                                                            const PhysicalFrame &  frame2,
                                                            SimTK::Vec3            point2
                                                          )
```

◆ Blankevoort1991Ligament() [3/4]

```
OpenSim::Blankevoort1991Ligament::Blankevoort1991Ligament ( std::string                name,
                                                            const PhysicalFrame &      frame1,
                                                            SimTK::Vec3                point1,
                                                            const PhysicalFrame &      frame2,
                                                            SimTK::Vec3                point2,
                                                            double                     linear_stiffness,
                                                            double                     slack_length
                                                          )
```

◆ Blankevoort1991Ligament() [4/4]

```
OpenSim::Blankevoort1991Ligament::Blankevoort1991Ligament ( std::string  name,
                                                            double       linear_stiffness,
                                                            double       slack_length
                                                          )
```

Member Function Documentation

◆ calcDampingForce()

```
double OpenSim::Blankevoort1991Ligament::calcDampingForce ( const SimTK::State &  state ) const
```
protected

◆ calcInverseForceStrainCurve()

```
double OpenSim::Blankevoort1991Ligament::calcInverseForceStrainCurve ( double  force ) const
```
protected

◆ calcSpringForce()

```
double OpenSim::Blankevoort1991Ligament::calcSpringForce ( const SimTK::State &  state ) const
```
protected

◆ calcTotalForce()

```
double OpenSim::Blankevoort1991Ligament::calcTotalForce ( const SimTK::State &  state ) const
```
protected

◆ clone()

```
Blankevoort1991Ligament* OpenSim::Blankevoort1991Ligament::clone ( ) const
```
inline override virtual

Create a new heap-allocated copy of the concrete object to which this Object refers.

It is up to the caller to delete the returned object when no longer needed. Every concrete object deriving from Object implements this pure virtual method automatically, via the declaration macro it invokes (e.g., **OpenSim_DECLARE_CONCRETE_OBJECT()**). Note that the concrete class overrides modify the return type to be a pointer to the *concrete* object; that still overrides the base class method because the return type is covariant with (that is, derives from) Object.

Implements **OpenSim::Force**.

◆ computeForce()

```
void
OpenSim::Blankevoort1991Ligament::computeForce ( const SimTK::State &                    state,
                                                 SimTK::Vector_< SimTK::SpatialVec > &   bodyForces,
                                                 SimTK::Vector &                         generalizedForces
                                               ) const                                   override virtual
```

Subclasses must implement this method to compute the forces that should be applied to bodies and generalized speeds.

This is invoked by **ForceAdapter** to perform the force computation.

Reimplemented from **OpenSim::Force**.

◆ computeMomentArm()

```
double OpenSim::Blankevoort1991Ligament::computeMomentArm ( const SimTK::State &  s,
                                                            Coordinate &          aCoord
                                                          ) const
```

◆ computePotentialEnergy()

```
double OpenSim::Blankevoort1991Ligament::computePotentialEnergy ( const SimTK::State &  state ) const   override virtual
```

Subclasses may optionally override this method to compute a contribution to the potential energy of the system.

The default implementation returns 0, which is appropriate for forces that do not contribute to potential energy.

Reimplemented from **OpenSim::Force**.

◆ extendAddToSystem()

void
OpenSim::Blankevoort1991Ligament::extendAddToSystem (SimTK::MultibodySystem & system) const override protected virtual

Add appropriate Simbody elements (if needed) to the System corresponding to this component and specify needed state resources.

extendAddToSystem() is called when the Simbody System is being created to represent a completed system (model) for computation. That is, connect() will already have been invoked on all components before any **addToSystem()** call is made. Helper methods for adding modeling options, state variables and their derivatives, discrete variables, and cache entries are available and can be called within **extendAddToSystem()** only.

Note that this method is const; you may not modify your model component or the containing model during this call. Any modifications you need should instead be performed in **finalizeFromProperties()** or at the latest connect(), which are non-const. The only exception is that you may need to record access information for resources you create in the *system*, such as an index number. For most Components, **OpenSim** base classes either provide convenience methods or handle indices automatically. Otherwise, you must declare indices as mutable data members so that you can set them here.

If you override this method, be sure to invoke the base class method at the beginning, using code like this:

```
void MyComponent::extendAddToSystem(SimTK::MultibodySystem& system) const {
    // Perform any additions to the system required by your Super
    Super::extendAddToSystem(system);
    // ... your code goes here
}
```

This method assumes that this **Component**'s addToSystem will be invoked before its subcomponents. If you need your subcomponents to be added to the system, first (e.g. require of a **Force** to be anchored to a SimTK::MobilizedBody specified by subcomponents) then you must implement: **extendAddToSystemAfterSubcomponents()**. It is possible to implement both method to add system elements before and then after your subcomponents have added themselves. Caution is required that Simbody elements are not added twice especially when order is unimportant.

Parameters

[in,out] **system** The MultibodySystem being added to.

See also

addModelingOption(), **addStateVariable()**, addDiscreteVariables(), **addCacheVariable()**

Reimplemented from **OpenSim::Component**.

◆ extendFinalizeFromProperties()

void OpenSim::Blankevoort1991Ligament::extendFinalizeFromProperties ( ) override protected virtual

Perform any time-invariant calculations, data structure initializations, or other configuration based on the component's properties to form a functioning (but not yet connected) component.

For example, each property should be checked to ensure that its value is within an acceptable range. When this method returns, the component will be marked as being up-to-date with its properties. Do not perform any configuration that depends on the SimTK::MultibodySystem; it is not available at this point.

If you override this method, be sure to invoke the base class method first, using code like this:

```
void MyComponent::extendFinalizeFromProperties() {
    Super::extendFinalizeFromProperties(); // invoke parent class method
    // ... your code goes here
    // ... catch invalid property values
    // ... initialize any internal data structures
}
```

Reimplemented from **OpenSim::Component**.

◆ extendPostScale()

void OpenSim::Blankevoort1991Ligament::extendPostScale (const SimTK::State & s, const ScaleSet & scaleSet) override virtual

Perform any computations that must occur after **ModelComponent::scale()** has been invoked on all ModelComponents in the **Model**.

This method is virtual and may be implemented by any subclass of **ModelComponent**, but all implementations must begin with a call to `Super::extendPostScale()` to execute the parent class methods before the child class method. The base class implementation in **ModelComponent** does nothing.

See also
postScale()

Reimplemented from **OpenSim::ModelComponent**.

◆ get_damping_coefficient()

const double& OpenSim::Blankevoort1991Ligament::get_damping_coefficient ( ) const inline

Get the value of the **damping_coefficient** property.

◆ get_GeometryPath()

const GeometryPath& OpenSim::Blankevoort1991Ligament::get_GeometryPath ( ) const inline

Get the value of the **GeometryPath** property.

◆ get_linear_stiffness()

const double& OpenSim::Blankevoort1991Ligament::get_linear_stiffness ( ) const inline

Get the value of the **linear_stiffness** property.

◆ get_slack_length()

const double& OpenSim::Blankevoort1991Ligament::get_slack_length ( ) const inline

Get the value of the **slack_length** property.

◆ get_transition_strain()

const double& OpenSim::Blankevoort1991Ligament::get_transition_strain ( ) const inline

Get the value of the **transition_strain** property.

◆ getClassName()

static const std::string& OpenSim::Blankevoort1991Ligament::getClassName ( ) inline static

This returns "Blankevoort1991Ligament".

See **getConcreteClassName()** if you want the class name of the underlying concrete object instead.

◆ getConcreteClassName()

const std::string& OpenSim::Blankevoort1991Ligament::getConcreteClassName ( ) const inline override virtual

Returns the class name of the concrete Object-derived class of the actual object referenced by this Object, as a string.

This is the string that is used as the tag for this concrete object in an XML file. Every concrete class derived from Object automatically overrides this method via the declaration macro it uses. See **getClassName()** to get the class name of the referencing (possibly abstract) class rather than the concrete object.

See also
getClassName()

Implements **OpenSim::Force**.

◆ getDampingCoefficientForceTimePerLength()

double OpenSim::Blankevoort1991Ligament::getDampingCoefficientForceTimePerLength ( ) const

Get the damping_coefficient in units of force*time/length (N*s/m)

◆ getDampingForce()

double OpenSim::Blankevoort1991Ligament::getDampingForce (const SimTK::State & state) const

◆ getLength()

double OpenSim::Blankevoort1991Ligament::getLength (const SimTK::State & state) const

◆ getLengtheningSpeed()

double OpenSim::Blankevoort1991Ligament::getLengtheningSpeed (const SimTK::State & state) const

◆ getLinearStiffnessForcePerLength()

double OpenSim::Blankevoort1991Ligament::getLinearStiffnessForcePerLength ( ) const

Get the linear_stiffness property in units of force/length (N/m)

◆ getRecordLabels()

OpenSim::Array<std::string> OpenSim::Blankevoort1991Ligament::getRecordLabels ( ) const override virtual

Methods to query a **Force** for the value actually applied during simulation.

The names of the quantities (column labels) is returned by this first function **getRecordLabels()**.

Reimplemented from **OpenSim::Force**.

◆ getRecordValues()

OpenSim::Array<double>
OpenSim::Blankevoort1991Ligament::getRecordValues **(const SimTK::State & state) const** override virtual

Given SimTK::State object extract all the values necessary to report forces, application location frame, etc.

used in conjunction with getRecordLabels and should return same size **Array**.

Reimplemented from **OpenSim::Force**.

◆ getSpringForce()

double OpenSim::Blankevoort1991Ligament::getSpringForce (const SimTK::State & state) const

◆ getStrain()

double OpenSim::Blankevoort1991Ligament::getStrain (const SimTK::State & state) const

◆ getStrainRate()

double OpenSim::Blankevoort1991Ligament::getStrainRate (const SimTK::State & state) const

◆ getTotalForce()

double OpenSim::Blankevoort1991Ligament::getTotalForce (const SimTK::State & state) const

◆ getTransitionLength()

double OpenSim::Blankevoort1991Ligament::getTransitionLength ( ) const

Get the length (m) of the ligament where the model transitions from the toe region to the linear region.

This corresponds to the length of the ligament at the transistion_strain.

◆ safeDownCast()

static Blankevoort1991Ligament* OpenSim::Blankevoort1991Ligament::safeDownCast (OpenSim::Object * obj) inline static

For use in MATLAB and Python to access the concrete class.

Example: `cObj = Blankevoort1991Ligament.safeDownCast(obj)`. This is equivalent to `dynamic_cast<Blankevoort1991Ligament*>(obj)` in C++.

◆ set_damping_coefficient()

void OpenSim::Blankevoort1991Ligament::set_damping_coefficient (const double & value) inline

Set the value of the **damping_coefficient** property.

◆ set_GeometryPath()

void OpenSim::Blankevoort1991Ligament::set_GeometryPath (const GeometryPath & value) inline

Set the value of the **GeometryPath** property.

◆ set_linear_stiffness()

void OpenSim::Blankevoort1991Ligament::set_linear_stiffness (const double & value) inline

Set the value of the **linear_stiffness** property.

◆ set_slack_length()

void OpenSim::Blankevoort1991Ligament::set_slack_length (const double & value) inline

Set the value of the **slack_length** property.

◆ set_transition_strain()

void OpenSim::Blankevoort1991Ligament::set_transition_strain (const double & value) inline

Set the value of the **transition_strain** property.

◆ setDampingCoefficientForceTimePerLength()

void OpenSim::Blankevoort1991Ligament::setDampingCoefficientForceTimePerLength (double damping_coefficient)

Set the damping_coefficient property using a value in units of force*time/length (N*s/m).

Note that scaling the model keeps the damping_coefficient property (in units force*time/strain) constant, thus the damping_coefficient in units of force*time/length input to this function will be altered by scaling.

◆ setLinearStiffnessForcePerLength()

void OpenSim::Blankevoort1991Ligament::setLinearStiffnessForcePerLength (double linear_stiffness)

Set the linear_stiffness property using a value in units of force/length (N/m).

Note that scaling the model keeps the linear_stiffness property (in units of force/strain) constant, thus the linear_stiffness in units of force/length input to this function will be altered by scaling.

◆ setSlackLengthFromReferenceForce()

void OpenSim::Blankevoort1991Ligament::setSlackLengthFromReferenceForce (double force,
const SimTK::State & reference_state
)

Set the slack_length property using the absolute spring force (N) in the ligament at a known pose (reference force).

Note that scaling the model will adjust the slack_length property, thus it is important to consider the order of scaling and using this function for your application.

◆ setSlackLengthFromReferenceStrain()

void OpenSim::Blankevoort1991Ligament::setSlackLengthFromReferenceStrain (double strain,
const SimTK::State & reference_state
)

Set the slack_length property using the strain in the ligament at a known pose (reference state).

Note that scaling the model will adjust the slack length property to hold the input reference strain constant if the input reference_state is equal to the default model pose (generated by initSystem()).

◆ upd_damping_coefficient()

double& OpenSim::Blankevoort1991Ligament::upd_damping_coefficient ( ) inline

Get a writable reference to the **damping_coefficient** property.

◆ upd_GeometryPath()

GeometryPath& OpenSim::Blankevoort1991Ligament::upd_GeometryPath ( ) inline

Get a writable reference to the **GeometryPath** property.

◆ upd_linear_stiffness()

double& OpenSim::Blankevoort1991Ligament::upd_linear_stiffness ( ) inline

Get a writable reference to the **linear_stiffness** property.

◆ upd_slack_length()

double& OpenSim::Blankevoort1991Ligament::upd_slack_length ( ) inline

Get a writable reference to the **slack_length** property.

◆ upd_transition_strain()

double& OpenSim::Blankevoort1991Ligament::upd_transition_strain ( ) inline

Get a writable reference to the **transition_strain** property.

OpenSim Property, Socket, Output, Input Documentation

◆ damping_coefficient

double OpenSim::Blankevoort1991Ligament::damping_coefficient

"The coefficient that multiplies the strain rate when computing the " " damping force. Units of N*s/strain. Default value of 0.003."

This property appears in XML files under the tag **<damping_coefficient>**. This property was generated with the **OpenSim_DECLARE_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_damping_coefficient(), **upd_damping_coefficient()**, **set_damping_coefficient()**

◆ damping_force

double OpenSim::Blankevoort1991Ligament::damping_force

Provides the value of **getDampingForce()** and is available at stage SimTK::Stage::Velocity .

This output was generated with the **OpenSim_DECLARE_OUTPUT** macro.

◆ GeometryPath

GeometryPath OpenSim::Blankevoort1991Ligament::GeometryPath

"The set of points defining the path of the ligament"

This property appears in XML files under the tag **<GeometryPath>**. This property was generated with the **OpenSim_DECLARE_UNNAMED_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_GeometryPath(), **upd_GeometryPath()**, **set_GeometryPath()**

◆ length

double OpenSim::Blankevoort1991Ligament::length

Provides the value of **getLength()** and is available at stage SimTK::Stage::Position .

This output was generated with the **OpenSim_DECLARE_OUTPUT** macro.

◆ lengthening_speed

double OpenSim::Blankevoort1991Ligament::lengthening_speed

Provides the value of **getLengtheningSpeed()** and is available at stage SimTK::Stage::Velocity .

This output was generated with the **OpenSim_DECLARE_OUTPUT** macro.

◆ linear_stiffness

double OpenSim::Blankevoort1991Ligament::linear_stiffness

"The slope of the linear region of the force-strain curve. " "Units of force/strain (N)."

This property appears in XML files under the tag **<linear_stiffness>**. This property was generated with the **OpenSim_DECLARE_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_linear_stiffness(), **upd_linear_stiffness()**, **set_linear_stiffness()**

◆ slack_length

double OpenSim::Blankevoort1991Ligament::slack_length

"The length at which ligament begins developing tension. Units of m."

This property appears in XML files under the tag **<slack_length>**. This property was generated with the **OpenSim_DECLARE_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_slack_length(), **upd_slack_length()**, **set_slack_length()**

◆ spring_force

double OpenSim::Blankevoort1991Ligament::spring_force

Provides the value of **getSpringForce()** and is available at stage SimTK::Stage::Position .

This output was generated with the **OpenSim_DECLARE_OUTPUT** macro.

◆ strain

double OpenSim::Blankevoort1991Ligament::strain

Provides the value of **getStrain()** and is available at stage SimTK::Stage::Position .

This output was generated with the **OpenSim_DECLARE_OUTPUT** macro.

◆ strain_rate

double OpenSim::Blankevoort1991Ligament::strain_rate

Provides the value of **getStrainRate()** and is available at stage SimTK::Stage::Velocity .

This output was generated with the **OpenSim_DECLARE_OUTPUT** macro.

◆ total_force

double OpenSim::Blankevoort1991Ligament::total_force

Provides the value of **getTotalForce()** and is available at stage SimTK::Stage::Velocity .

This output was generated with the **OpenSim_DECLARE_OUTPUT** macro.

◆ transition_strain

double OpenSim::Blankevoort1991Ligament::transition_strain

"The strain at which the ligament force-strain curve transitions from " "quadratic to linear. Units of strain. Default value of 0.06 (6%)."

This property appears in XML files under the tag **<transition_strain>**. This property was generated with the **OpenSim_DECLARE_PROPERTY** macro; see **Property** to learn about the property system.

See also
get_transition_strain(), **upd_transition_strain()**, **set_transition_strain()**

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

- OpenSim/Simulation/Model/Blankevoort1991Ligament.h