OpenSim::Smith2018ArticularContactForce Class Reference

This **Force** component models the contact between a pair of triangulated surface meshes (.vtp, .stl, .obj). More...

Inheritance diagram for OpenSim::Smith2018ArticularContactForce:

OpenSim Properties, Sockets, Outputs, Inputs

Properties (single-value)

double	**min_proximity** "The minimum proximity " "that is valid between contacting meshes to limit the search distance " "along the casting_mesh normal ray used for collision detection. Note " "this can be negative if proximity maps should include triangles that " "are not in contact." "Default value set to 0.0 meters." More...
double	**max_proximity** "The maximum proximity " "that is valid between contacting meshes to limit the search distance " "along the casting_mesh normal ray used for collision detection." "Default value set to 0.01 meters." More...
std::string	**elastic_foundation_formulation** "Formulation for depth-pressure relationship: " "'linear' or 'nonlinear'. " "Default value set to 'linear'." More...
bool	**use_lumped_contact_model** "Combine the thickness and the average material properties between " "the Smith2018ContactMeshes for both meshes and use Bei & Fregly 2003 " "lumped parameter Elastic Foundation model." More...

Sockets

Smith2018ContactMesh	**target_mesh** "Target mesh for collision detection." More...
Smith2018ContactMesh	**casting_mesh** "Ray casting mesh for collision detection." More...

Outputs

int	**target_num_contacting_triangles** Provides the value of **getTargetNumContactingTriangles()** and is available at stage SimTK::Stage::Dynamics . More...
int	**casting_num_contacting_triangles** Provides the value of **getCastingNumContactingTriangles()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector	**target_triangle_proximity** Provides the value of **getTargetTriangleProximity()** and is available at stage SimTK::Stage::Position . More...
SimTK::Vector	**casting_triangle_proximity** Provides the value of **getCastingTriangleProximity()** and is available at stage SimTK::Stage::Position . More...
SimTK::Vector	**target_triangle_pressure** Provides the value of **getTargetTrianglePressure()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector	**casting_triangle_pressure** Provides the value of **getCastingTrianglePressure()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector	**target_triangle_potential_energy** Provides the value of **getTargetTrianglePotentialEnergy()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector	**casting_triangle_potential_energy** Provides the value of **getCastingTrianglePotentialEnergy()** and is available at stage SimTK::Stage::Dynamics . More...
double	**target_total_contact_area** Provides the value of **getTargetTotalContactArea()** and is available at stage SimTK::Stage::Position . More...
double	**casting_total_contact_area** Provides the value of **getCastingTotalContactArea()** and is available at stage SimTK::Stage::Position . More...
SimTK::Vector	**target_regional_contact_area** Provides the value of **getTargetRegionalContactArea()** and is available at stage SimTK::Stage::Position . More...
SimTK::Vector	**casting_regional_contact_area** Provides the value of **getCastingRegionalContactArea()** and is available at stage SimTK::Stage::Position . More...
double	**target_total_mean_proximity** Provides the value of **getTargetTotalMeanProximity()** and is available at stage SimTK::Stage::Position . More...
double	**casting_total_mean_proximity** Provides the value of **getCastingTotalMeanProximity()** and is available at stage SimTK::Stage::Position . More...
SimTK::Vector	**target_regional_mean_proximity** Provides the value of **getTargetRegionalMeanProximity()** and is available at stage SimTK::Stage::Position . More...
SimTK::Vector	**casting_regional_mean_proximity** Provides the value of **getCastingRegionalMeanProximity()** and is available at stage SimTK::Stage::Position .

	More...
double	**target_total_max_proximity** Provides the value of **getTargetTotalMaxProximity()** and is available at stage SimTK::Stage::Position . More...
double	**casting_total_max_proximity** Provides the value of **getCastingTotalMaxProximity()** and is available at stage SimTK::Stage::Position . More...
SimTK::Vector	**target_regional_max_proximity** Provides the value of **getTargetRegionalMaxProximity()** and is available at stage SimTK::Stage::Position . More...
SimTK::Vector	**casting_regional_max_proximity** Provides the value of **getCastingRegionalMaxProximity()** and is available at stage SimTK::Stage::Position . More...
double	**target_total_mean_pressure** Provides the value of **getTargetTotalMeanPressure()** and is available at stage SimTK::Stage::Dynamics . More...
double	**casting_total_mean_pressure** Provides the value of **getCastingTotalMeanPressure()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector	**target_regional_mean_pressure** Provides the value of **getTargetRegionalMeanPressure()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector	**casting_regional_mean_pressure** Provides the value of **getCastingRegionalMeanPressure()** and is available at stage SimTK::Stage::Dynamics . More...
double	**target_total_max_pressure** Provides the value of **getTargetTotalMaxPressure()** and is available at stage SimTK::Stage::Dynamics . More...
double	**casting_total_max_pressure** Provides the value of **getCastingTotalMaxPressure()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector	**target_regional_max_pressure** Provides the value of **getTargetRegionalMaxPressure()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector	**casting_regional_max_pressure** Provides the value of **getCastingRegionalMaxPressure()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vec3	**target_total_center_of_proximity** Provides the value of **getTargetTotalCenterOfProximity()** and is available at stage SimTK::Stage::Position . More...
SimTK::Vec3	**casting_total_center_of_proximity** Provides the value of **getCastingTotalCenterOfProximity()** and is available at stage SimTK::Stage::Position . More...
SimTK::Vector_< SimTK::Vec3 >	**target_regional_center_of_proximity** Provides the value of **getTargetRegionalCenterOfProximity()** and is available at stage SimTK::Stage::Position . More...
SimTK::Vector_< SimTK::Vec3 >	**casting_regional_center_of_proximity** Provides the value of **getCastingRegionalCenterOfProximity()** and is available at stage SimTK::Stage::Position . More...
SimTK::Vec3	**target_total_center_of_pressure** Provides the value of **getTargetTotalCenterOfPressure()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vec3	**casting_total_center_of_pressure** Provides the value of **getCastingTotalCenterOfPressure()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector_< SimTK::Vec3 >	**target_regional_center_of_pressure** Provides the value of **getTargetRegionalCenterOfPressure()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector_< SimTK::Vec3 >	**casting_regional_center_of_pressure** Provides the value of **getCastingRegionalCenterOfPressure()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vec3	**target_total_contact_force** Provides the value of **getTargetTotalContactForce()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vec3	**casting_total_contact_force** Provides the value of **getCastingTotalContactForce()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector_< SimTK::Vec3 >	**target_regional_contact_force** Provides the value of **getTargetRegionalContactForce()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector_< SimTK::Vec3 >	**casting_regional_contact_force** Provides the value of **getCastingRegionalContactForce()** and is available at stage SimTK::Stage::Dynamics . More...

SimTK::Vec3	**target_total_contact_moment** Provides the value of **getTargetTotalContactMoment()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vec3	**casting_total_contact_moment** Provides the value of **getCastingTotalContactMoment()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector_< SimTK::Vec3 >	**target_regional_contact_moment** Provides the value of **getTargetRegionalContactMoment()** and is available at stage SimTK::Stage::Dynamics . More...
SimTK::Vector_< SimTK::Vec3 >	**casting_regional_contact_moment** Provides the value of **getCastingRegionalContactMoment()** and is available at stage SimTK::Stage::Dynamics . More...

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

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

Public Member Functions

	Smith2018ArticularContactForce ()
	Smith2018ArticularContactForce (const std::string &name, **Smith2018ContactMesh** &**target_mesh**, **Smith2018ContactMesh** &**casting_mesh**)
int	**getTargetNumContactingTriangles** (const SimTK::State &state) const
int	**getCastingNumContactingTriangles** (const SimTK::State &state) const
SimTK::Vector	**getTargetTriangleProximity** (const SimTK::State &state) const
SimTK::Vector	**getCastingTriangleProximity** (const SimTK::State &state) const
SimTK::Vector	**getTargetTrianglePressure** (const SimTK::State &state) const
SimTK::Vector	**getCastingTrianglePressure** (const SimTK::State &state) const
SimTK::Vector	**getTargetTrianglePotentialEnergy** (const SimTK::State &state) const
SimTK::Vector	**getCastingTrianglePotentialEnergy** (const SimTK::State &state) const
double	**getTargetTotalContactArea** (const SimTK::State &state) const
double	**getCastingTotalContactArea** (const SimTK::State &state) const
SimTK::Vector	**getTargetRegionalContactArea** (const SimTK::State &state) const
SimTK::Vector	**getCastingRegionalContactArea** (const SimTK::State &state) const
double	**getTargetTotalMeanProximity** (const SimTK::State &state) const
double	**getCastingTotalMeanProximity** (const SimTK::State &state) const
SimTK::Vector	**getTargetRegionalMeanProximity** (const SimTK::State &state) const
SimTK::Vector	**getCastingRegionalMeanProximity** (const SimTK::State &state) const
double	**getTargetTotalMaxProximity** (const SimTK::State &state) const
double	**getCastingTotalMaxProximity** (const SimTK::State &state) const
SimTK::Vector	**getTargetRegionalMaxProximity** (const SimTK::State &state) const
SimTK::Vector	**getCastingRegionalMaxProximity** (const SimTK::State &state) const
double	**getTargetTotalMeanPressure** (const SimTK::State &state) const
double	**getCastingTotalMeanPressure** (const SimTK::State &state) const
SimTK::Vector	**getTargetRegionalMeanPressure** (const SimTK::State &state) const
SimTK::Vector	**getCastingRegionalMeanPressure** (const SimTK::State &state) const
double	**getTargetTotalMaxPressure** (const SimTK::State &state) const
double	**getCastingTotalMaxPressure** (const SimTK::State &state) const
SimTK::Vector	**getTargetRegionalMaxPressure** (const SimTK::State &state) const
SimTK::Vector	**getCastingRegionalMaxPressure** (const SimTK::State &state) const
SimTK::Vec3	**getTargetTotalCenterOfProximity** (const SimTK::State &state) const
SimTK::Vec3	**getCastingTotalCenterOfProximity** (const SimTK::State &state) const
SimTK::Vector_< SimTK::Vec3 >	**getTargetRegionalCenterOfProximity** (const SimTK::State &state) const
SimTK::Vector_< SimTK::Vec3 >	**getCastingRegionalCenterOfProximity** (const SimTK::State &state) const
SimTK::Vec3	**getTargetTotalCenterOfPressure** (const SimTK::State &state) const
SimTK::Vec3	**getCastingTotalCenterOfPressure** (const SimTK::State &state) const
SimTK::Vector_< SimTK::Vec3 >	**getTargetRegionalCenterOfPressure** (const SimTK::State &state) const
SimTK::Vector_< SimTK::Vec3 >	**getCastingRegionalCenterOfPressure** (const SimTK::State &state) const
SimTK::Vec3	**getTargetTotalContactForce** (const SimTK::State &state) const
SimTK::Vec3	**getCastingTotalContactForce** (const SimTK::State &state) const
SimTK::Vector_< SimTK::Vec3 >	**getTargetRegionalContactForce** (const SimTK::State &state) const
SimTK::Vector_< SimTK::Vec3 >	**getCastingRegionalContactForce** (const SimTK::State &state) const
SimTK::Vec3	**getTargetTotalContactMoment** (const SimTK::State &state) const
SimTK::Vec3	**getCastingTotalContactMoment** (const SimTK::State &state) const
SimTK::Vector_< SimTK::Vec3 >	**getTargetRegionalContactMoment** (const SimTK::State &state) const
SimTK::Vector_< SimTK::Vec3 >	**getCastingRegionalContactMoment** (const SimTK::State &state) const
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	**computeForce** (const SimTK::State &state, 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...

OpenSim::Array< double >	**getRecordValues** (const SimTK::State &s) const Given SimTK::State object extract all the values necessary to report forces, application location frame, etc. More...
OpenSim::Array< std::string >	**getRecordLabels** () const Methods to query a **Force** for the value actually applied during simulation. More...

Property-related functions

const double &	**get_min_proximity** () const Get the value of the **min_proximity** property. More...
double &	**upd_min_proximity** () Get a writable reference to the **min_proximity** property. More...
void	**set_min_proximity** (const double &value) Set the value of the **min_proximity** property. More...
const double &	**get_max_proximity** () const Get the value of the **max_proximity** property. More...
double &	**upd_max_proximity** () Get a writable reference to the **max_proximity** property. More...
void	**set_max_proximity** (const double &value) Set the value of the **max_proximity** property. More...
const std::string &	**get_elastic_foundation_formulation** () const Get the value of the **elastic_foundation_formulation** property. More...
std::string &	**upd_elastic_foundation_formulation** () Get a writable reference to the **elastic_foundation_formulation** property. More...
void	**set_elastic_foundation_formulation** (const std::string &value) Set the value of the **elastic_foundation_formulation** property. More...
const bool &	**get_use_lumped_contact_model** () const Get the value of the **use_lumped_contact_model** property. More...
bool &	**upd_use_lumped_contact_model** () Get a writable reference to the **use_lumped_contact_model** property. More...
void	**set_use_lumped_contact_model** (const bool &value) Set the value of the **use_lumped_contact_model** property. More...

Socket-related functions

void	**connectSocket_target_mesh** (const **Object** &object) Connect the 'target_mesh' **Socket** to an object of type **Smith2018ContactMesh**. More...
void	**connectSocket_casting_mesh** (const **Object** &object) Connect the 'casting_mesh' **Socket** to an object of type **Smith2018ContactMesh**. 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 **Smith2018ArticularContactForce** *	**safeDownCast** (**OpenSim::Object** *obj) For use in MATLAB and Python to access the concrete class. More...
static const std::string &	**getClassName** () This returns "Smith2018ArticularContactForce". More...
Smith2018ArticularContactForce *	**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

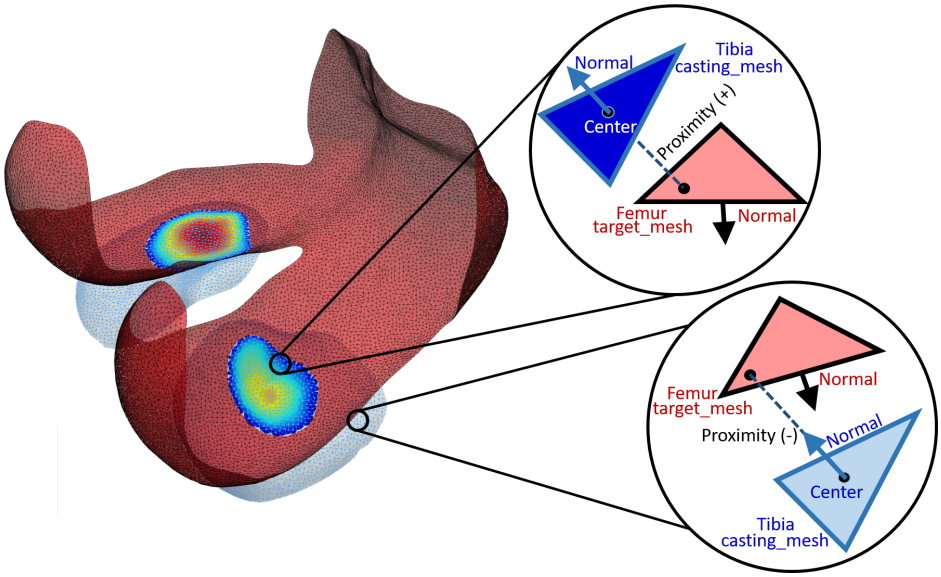
This **Force** component models the contact between a pair of triangulated surface meshes (.vtp, .stl, .obj).

It was orginially designed to represent articular contact between cartilage, mensici, or joint implants [1], but is generalizable so it could also represent foot-floor contact or a skin-device interfaces etc. The formulation of the contact model has previously been called an elastic foundation model [2] or discrete element analysis [3,4]. In this implementation, non-deforming triangulated meshes are allowed to interpenetrate and the proximity (distance between the meshes) is calculated for each triangle. For each triangle with a positive proximity (ie triangle interpenetrated the opposing mesh), the contact pressure on each triangle face is then calculated based on the proximity and material properties. This formulation is much faster than a finite element approach, but only calculates the contact pressures on the mesh surface and not the internal stresses. Additionally, it provides a simplifed representation of the contact as the meshes do not deform, but instead are geometrically rigid (ie the vertices within a mesh do not move relative to each other), but the contacting mesh pair are allowed to interpenetrate.

The while the force calculatation is similar to the **ElasticFoundationForce** component, the **Smith2018ArticularContactForce** provides a different parameterization of the relationship between the overlap depth and material properties to calculate the contact pressure. Additionally, it provides the proximity, pressure, and potential_energy for each mesh triangle as an output so that maps of these calculated values on the mesh surface can be visualized. Finally, it provides improved computational performance through a different collision detection method.

To calculate the pressure for each triangle, it is necessary to detect the mesh triangles that are interpenetrating (commonly called contact or collision detection in computer graphics literature) and calculate the proximity. This task is extremely slow if a brute force approach is applied to check every triangle in one mesh against every triangle in another mesh. Smith et al, CMBBE I&V, 2018 [1] introduced a method to efficiently detect contact between triangular meshes using Oriented Bounding Boxes (OBBs, a common approach in computer graphics) and several additional speed-ups that leverage the fact that changes in contact between timesteps are generally small. This approach has been implemented in the **Smith2018ArticularContactForce** component along with some additional features.

Two articulating triangular meshes are defined as **Smith2018ContactMesh** components (Sockets: casting_mesh and target_mesh). These meshes are fixed to bodies in the model, and thus their relative poses are determined by the model coordinates. To detect contact, a normal ray is cast from the center of each triangle in the casting mesh towards the overlapping target mesh. Ray intersection tests are then performed against an Oriented Bounding Box tree constructed around the target mesh. This algorithm is implemented in the **computeMeshProximity()** function with the OBB construction and ray intersection queries managed by the **Smith2018ContactMesh**.



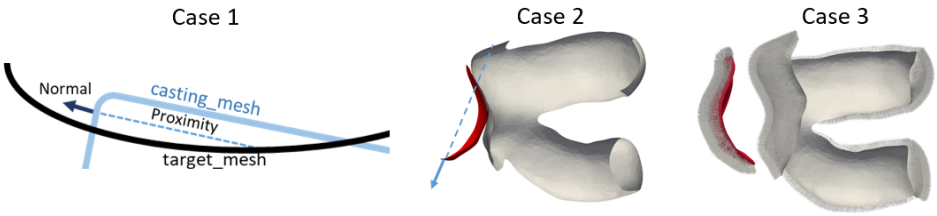
The major speed-up in the algorithm leverages the fact that changes in joint coordinates and thus contact patterns between time steps are generally small. Thus, after reposing the meshes, (i.e. realizePosition()) each triangle in the casting mesh is tested against the contacting target triangle from the previous pose. Additional speed-up can be gained by casting the normal ray in both directions (by setting min_proximity to a negative value), so even some of the out-of-contact triangles are "remembered". If the previous contacting triangle test fails, the casting ray is checked against the neighboring triangles (those that share a vertex) in the target mesh. Then if this test fails, the expensive casting ray–OBB test is performed. If the meshes were not in contact at the previous time step this does not cause an issue, just a slower solution, as here the ray-OBB tests will be peformed for every triangle in the casting_mesh.

Swapping the contact meshes changes the resulting forces

The ray casting is only performed from the casting_mesh towards the target_mesh. Thus, a pressure map is only generated for the casting_mesh. A force vector is computed for each triangle in the casting_mesh using **Force** = -normal*area*pressure, and the resultant force of all triangle forces is applied to the body to which the casting_mesh is attached. An equal and opposite force is applied to the body to which the target_mesh is attached. The important ramification of this is that if the casting_mesh and target_mesh are switched, the simulation results will NOT be exactly the same. For best performance, the casting_mesh should be set to the mesh that contains the smaller number of triangles.

As it is still useful to visualize the proximity and pressure maps for the target_mesh, and in most situations the resultant force from the target_mesh are very close to the mirrored casting_mesh resultant force, there is a ModelingOption named "flip_meshes" that will cause the ray casting to also be performed from the target_mesh to calculate triangle proximity and pressure values. Note the applied contact force in this case is still only that calculated for the casting_mesh.

Potential Pitfalls



Case 1 The casting_mesh mesh has significant curvature and the material properties are set in a manner that results in a compliment contact where large interpenetrations with curvature may occur. In this scenario, the distance along the normal ray from the casting_mesh to the target_mesh may be unrealistically large, resulting in high contact pressures. If only one mesh has high curvature, then this mesh can be defined as the target_mesh and the excessive proximity values are avoided. Another potential solution is to ensure that the initial positions of the meshes are set so that there is minimal or no contact, thus as the simulation progresses the increasing contact pressures will prevent significant interpentration of the meshes.

Case 2 Another pitfall of highly curved meshes is the potential for false contact detection as shown in the figure. Here, a triangle on the side of the patellar cartilage will be incorrectly identified as in contact with the femur. This problem can be avoided by setting min_proximity and max_proximity properties to values that limit the potential contact search area to feasible locations for your application. The min_proximity and max_proximity properties limit the search region for a contact triangle along the ray cast from the casting_mesh. The min_proximity can be set to a negative value if you would like proximity maps for the out of contact triangles. For example, if you are visualizing kinematics measured with fluoroscopy and only have meshes of the bones, using a negative min_proximity enables the distance between the subcondral surfaces to be calculated for each triangle in the casting_mesh.

Case 3 To reduce the number of triangles in the **Smith2018ContactMesh** and thus speed up the collision detection, the meshes do not need to be closed surfaces (water tight). However, this can cause issues if the initial positions of the meshes are set improperly or the positions of the meshes within a simulation progress to infeasible configurations. The figure depicts that the patella has been spun 180*, so if soft tissues pull the patella straight into the femur, no contact will be detected because the backside of the patella will collide with the femur. However, while the meshes do not need to be closed, they can be closed to avoid this issue for applications such as simulating the contact between a spinning ball bouncing on a surface.

The pressure-depth relationship

The relationship between overlap depth and pressure can be either linear or non-linear, depending on the value of the elastic_foundation_formulation property. The implemented equations are those proposed in Bei and Fregly, Med Eng Phys, 2004 [2]:

Linear:

$$P = E\frac{(1-\nu)}{(1+\nu)(1-2\nu)}\frac{d}{h}$$

Non-Linear:

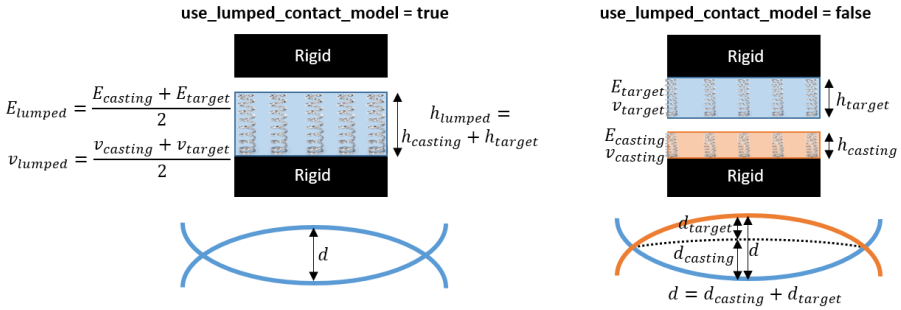
$$P = -E\frac{(1-\nu)}{(1+\nu)(1-2\nu)}\ln\left(1-\frac{d}{h}\right)$$

Where:

- P: pressure
- E: elastic modulus
- ν: Poisson's ratio
- d: depth of overlap
- h: height (i.e., thickness) of the elastic layer

Material/mesh properties

The original Bei and Fregly formulation assumes that a rigid object is contacting an object with a thin elastic layer. This assumption is appropriate when modeling joint replacements where a metal component (rigid) contacts a polyethylene component (elastic). To model cartilage-cartilage contact, this approach requires that the two cartilage layers are lumped together into one elastic layer, necessitating that a constant thickness, elastic modulus, and Poisson's ratio is assumed for each contacting triangle pair. Thus the total overlap depth is split equally between each triangle contact pair. As cartilage-cartilage contact often involves articulations between cartilage surfaces with varying thickness and material properties, the Bei and Fregly approach was extended to accommodate variable properties. The use_lumped_contact_model property controls whether the constant property or variable property formulation is used.



The variable property formulation is described in Zevenbergen et al, PLOS One, 2018 [5]. Here, the following system of four equations must be solved to obtain the local overlap depth (proximity) and pressure for the casting and target triangles.

$$
\begin{aligned}
P_{\text{casting}} &= f(E, \nu, h, d_{\text{casting}}) \\
P_{\text{target}} &= f(E, \nu, h, d_{\text{target}}) \\
P_{\text{casting}} &= P_{\text{target}} \\
d &= d_{\text{casting}} + d_{\text{target}}
\end{aligned}
$$

Here, the first two equations use the Bei and Fregly elastic foundation model (linear or non-linear) to define the relationship between the local mesh properties, local overlap depth and computed pressure. The third equation is a force equilibrium, assuming that the force applied to a pair of contacting triangles is equal and opposite. This formulation further assumes that the triangles in contact have the same area. The fourth equation states that the total overlap depth of the meshes (which is readily calculated) is the sum of the local overlap depths of the two elastic layers in contact.

This system of equations can be solved analytically if the linear pressure- depth relationship is used. If the non-linear relationship is used, the system of equations is solved using a numerical solver.

Outputs

This component has some outputs such as triangle_proximity, triangle_pressure, and triangle_potential_energy that return a SimTK::Vector (size = number of triangles) with a value corresponding to each triangle face in the respective target or casting mesh. There are also "summary" outputs that return values associated with the entire mesh such as contact area, mean/max proximity/pressure, center of proximity/pressure etc. Finally, there are regional summary outputs which return a SimTK::Vector (size = 6). Here, the entries in the vector reflect the summary metrics corresponding to subset of mesh triangles located in six specific regions. These regions are defined as the subset of mesh triangles whose center is located in the half space [+x, -x, +y, -y, +z, -z] in the local mesh coordinate system. If the mesh coordinate system is aligned with anatomical axes, then this enables simulation results to be more readily interpreted. For example, when performing simulations of the knee, if the z axis is aligned to the medial-lateral axis, points medially, and the origin is located between the femoral condyles, then the regional outputs corresponding to +z and -z will summarize the mesh triangles located on the medial and lateral condyles and thus enable comparisons of the loading in the medial and lateral compartments.

All outputs are reported in the local mesh reference frame (ie the reference frame of the mesh_file. The ContactForce and ContactMoment outputs are expressed in this frame and calculated at the origin of this frame.).

References

[1] Smith, C. R., Won Choi, K., Negrut, D., & Thelen, D. G. (2018). Efficient computation of cartilage contact pressures within dynamic simulations of movement. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, 6(5), 491-498.

[2] Bei, Y., & Fregly, B. J. (2004). Multibody dynamic simulation of knee contact mechanics. Medical engineering & physics, 26(9), 777-789.

[3] Volokh, K. Y., Chao, E. Y. S., & Armand, M. (2007). On foundations of discrete element analysis of contact in diarthrodial joints. Molecular & cellular biomechanics: MCB, 4(2), 67.

[4] Abraham, C. L., Maas, S. A., Weiss, J. A., Ellis, B. J., Peters, C. L., & Anderson, A. E. (2013). A new discrete element analysis method for predicting hip joint contact stresses. Journal of biomechanics, 46(6), 1121-1127.

[5] Zevenbergen, L., Smith, C. R., Van Rossom, S., Thelen, D. G., Famaey, N., Vander Sloten, J., & Jonkers, I. (2018). Cartilage defect location and stiffness predispose the tibiofemoral joint to aberrant loading conditions during stance phase of gait. PloS one, 13(10), e0205842.

Constructor & Destructor Documentation

◆ Smith2018ArticularContactForce() [1/2]

OpenSim::Smith2018ArticularContactForce::Smith2018ArticularContactForce ( )

◆ Smith2018ArticularContactForce() [2/2]

```
OpenSim::Smith2018ArticularContactForce::Smith2018ArticularContactForce ( const std::string &          name,
                                                                           Smith2018ContactMesh &     target_mesh,
                                                                           Smith2018ContactMesh &     casting_mesh
                                                                         )
```

Member Function Documentation

◆ clone()

```
Smith2018ArticularContactForce* OpenSim::Smith2018ArticularContactForce::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**.

◆ computeContactForceVector()

```
SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::computeContactForceVector ( double          pressure,
                                                                                  double          area,
                                                                                  SimTK::Vec3     normal
                                                                                ) const
```

protected

◆ computeContactMomentVector()

```
SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::computeContactMomentVector ( double          pressure,
                                                                                   double          area,
                                                                                   SimTK::Vec3     normal,
                                                                                   SimTK::Vec3     center
                                                                                 ) const
```

protected

◆ computeContactStats()

```
ContactStats
OpenSim::Smith2018ArticularContactForce::computeContactStats ( const Smith2018ContactMesh &     mesh,
                                                               const SimTK::Vector &            total_triangle_proximity,
                                                               const SimTK::Vector &            total_triangle_pressure,
                                                               const std::vector< int > &       triIndices
                                                             ) const
```

protected

◆ computeForce()

```
void
OpenSim::Smith2018ArticularContactForce::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**.

◆ computeMeshDynamics()

```
void
OpenSim::Smith2018ArticularContactForce::computeMeshDynamics ( const SimTK::State &                 state,
                                                               const Smith2018ContactMesh &         casting_mesh,
                                                               const Smith2018ContactMesh &         target_mesh,
                                                               SimTK::Vector_< SimTK::Vec3 > &      triangle_force,
                                                               SimTK::Vector &                      triangle_pressure,
                                                               SimTK::Vector &                      triangle_energy
                                                             ) const                                                  protected
```

◆ computeMeshProximity()

```
void
OpenSim::Smith2018ArticularContactForce::computeMeshProximity ( const SimTK::State &                 state,
                                                                const Smith2018ContactMesh &         casting_mesh,
                                                                const Smith2018ContactMesh &         target_mesh,
                                                                const std::string &                  cache_mesh_name,
                                                                SimTK::Vector &                      triangle_proximity
                                                              ) const                                                  protected
```

◆ computePotentialEnergy()

double OpenSim::Smith2018ArticularContactForce::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**.

◆ connectSocket_casting_mesh()

void OpenSim::Smith2018ArticularContactForce::connectSocket_casting_mesh (const Object & object) inline

Connect the 'casting_mesh' **Socket** to an object of type **Smith2018ContactMesh**.

Call **finalizeConnections()** afterwards to update the socket's connectee path property. The reference to the connectee set here takes precedence over the connectee path property.

◆ connectSocket_target_mesh()

void OpenSim::Smith2018ArticularContactForce::connectSocket_target_mesh (const Object & object) inline

Connect the 'target_mesh' **Socket** to an object of type **Smith2018ContactMesh**.

Call **finalizeConnections()** afterwards to update the socket's connectee path property. The reference to the connectee set here takes precedence over the connectee path property.

◆ extendAddToSystem()

```
void
OpenSim::Smith2018ArticularContactForce::extendAddToSystem ( SimTK::MultibodySystem & system ) const   override protected virtual
```

Default is to create a **ForceAdapter** which is a SimTK::Force::Custom as the underlying computational component.

Subclasses override to employ other SimTK::Forces; be sure to invoke **Force::extendAddToSystem()** at the beginning of the overriding method.

Reimplemented from **OpenSim::Force**.

◆ extendRealizeReport()

void OpenSim::Smith2018ArticularContactForce::extendRealizeReport (const SimTK::State & state) const override protected virtual

Perform computations that may depend on anything but are only used for reporting and cannot affect subsequent simulation behavior.

Reimplemented from **OpenSim::Component**.

◆ get_elastic_foundation_formulation()

const std::string& OpenSim::Smith2018ArticularContactForce::get_elastic_foundation_formulation ( ) const inline

Get the value of the **elastic_foundation_formulation** property.

◆ get_max_proximity()

const double& OpenSim::Smith2018ArticularContactForce::get_max_proximity ( ) const inline

Get the value of the **max_proximity** property.

◆ get_min_proximity()

const double& OpenSim::Smith2018ArticularContactForce::get_min_proximity ( ) const inline

Get the value of the **min_proximity** property.

◆ get_use_lumped_contact_model()

const bool& OpenSim::Smith2018ArticularContactForce::get_use_lumped_contact_model ( ) const inline

Get the value of the **use_lumped_contact_model** property.

◆ getCastingNumContactingTriangles()

int OpenSim::Smith2018ArticularContactForce::getCastingNumContactingTriangles (const SimTK::State & state) const inline

◆ getCastingRegionalCenterOfPressure()

SimTK::Vector_<SimTK::Vec3>
OpenSim::Smith2018ArticularContactForce::getCastingRegionalCenterOfPressure (const SimTK::State & state) const inline

◆ getCastingRegionalCenterOfProximity()

SimTK::Vector_<SimTK::Vec3>
OpenSim::Smith2018ArticularContactForce::getCastingRegionalCenterOfProximity (const SimTK::State & state) const inline

◆ getCastingRegionalContactArea()

SimTK::Vector OpenSim::Smith2018ArticularContactForce::getCastingRegionalContactArea (const SimTK::State & state) const inline

◆ getCastingRegionalContactForce()

```
SimTK::Vector_<SimTK::Vec3>
OpenSim::Smith2018ArticularContactForce::getCastingRegionalContactForce ( const SimTK::State & state ) const inline
```

◆ getCastingRegionalContactMoment()

```
SimTK::Vector_<SimTK::Vec3>
OpenSim::Smith2018ArticularContactForce::getCastingRegionalContactMoment ( const SimTK::State & state ) const inline
```

◆ getCastingRegionalMaxPressure()

```
SimTK::Vector OpenSim::Smith2018ArticularContactForce::getCastingRegionalMaxPressure ( const SimTK::State & state ) const inline
```

◆ getCastingRegionalMaxProximity()

```
SimTK::Vector
OpenSim::Smith2018ArticularContactForce::getCastingRegionalMaxProximity ( const SimTK::State & state ) const inline
```

◆ getCastingRegionalMeanPressure()

```
SimTK::Vector
OpenSim::Smith2018ArticularContactForce::getCastingRegionalMeanPressure ( const SimTK::State & state ) const inline
```

◆ getCastingRegionalMeanProximity()

```
SimTK::Vector
OpenSim::Smith2018ArticularContactForce::getCastingRegionalMeanProximity ( const SimTK::State & state ) const inline
```

◆ getCastingTotalCenterOfPressure()

```
SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::getCastingTotalCenterOfPressure ( const SimTK::State & state ) const inline
```

◆ getCastingTotalCenterOfProximity()

```
SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::getCastingTotalCenterOfProximity ( const SimTK::State & state ) const inline
```

◆ getCastingTotalContactArea()

```
double OpenSim::Smith2018ArticularContactForce::getCastingTotalContactArea ( const SimTK::State & state ) const inline
```

◆ getCastingTotalContactForce()

```
SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::getCastingTotalContactForce ( const SimTK::State & state ) const inline
```

◆ getCastingTotalContactMoment()

SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::getCastingTotalContactMoment (const SimTK::State & state) const inline

◆ getCastingTotalMaxPressure()

double OpenSim::Smith2018ArticularContactForce::getCastingTotalMaxPressure (const SimTK::State & state) const inline

◆ getCastingTotalMaxProximity()

double OpenSim::Smith2018ArticularContactForce::getCastingTotalMaxProximity (const SimTK::State & state) const inline

◆ getCastingTotalMeanPressure()

double OpenSim::Smith2018ArticularContactForce::getCastingTotalMeanPressure (const SimTK::State & state) const inline

◆ getCastingTotalMeanProximity()

double OpenSim::Smith2018ArticularContactForce::getCastingTotalMeanProximity (const SimTK::State & state) const inline

◆ getCastingTrianglePotentialEnergy()

SimTK::Vector OpenSim::Smith2018ArticularContactForce::getCastingTrianglePotentialEnergy (const SimTK::State & state) const inline

◆ getCastingTrianglePressure()

SimTK::Vector OpenSim::Smith2018ArticularContactForce::getCastingTrianglePressure (const SimTK::State & state) const inline

◆ getCastingTriangleProximity()

SimTK::Vector OpenSim::Smith2018ArticularContactForce::getCastingTriangleProximity (const SimTK::State & state) const inline

◆ getClassName()

static const std::string& OpenSim::Smith2018ArticularContactForce::getClassName ( ) inline static

This returns "Smith2018ArticularContactForce".

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

◆ getConcreteClassName()

const std::string& OpenSim::Smith2018ArticularContactForce::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**.

◆ getRecordLabels()

OpenSim::Array<std::string> OpenSim::Smith2018ArticularContactForce::getRecordLabels ( ) const 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::Smith2018ArticularContactForce::getRecordValues (const SimTK::State & state) const 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**.

◆ getTargetNumContactingTriangles()

int OpenSim::Smith2018ArticularContactForce::getTargetNumContactingTriangles (const SimTK::State & state) const inline

◆ getTargetRegionalCenterOfPressure()

SimTK::Vector_<SimTK::Vec3>
OpenSim::Smith2018ArticularContactForce::getTargetRegionalCenterOfPressure (const SimTK::State & state) const inline

◆ getTargetRegionalCenterOfProximity()

SimTK::Vector_<SimTK::Vec3>
OpenSim::Smith2018ArticularContactForce::getTargetRegionalCenterOfProximity (const SimTK::State & state) const inline

◆ getTargetRegionalContactArea()

SimTK::Vector OpenSim::Smith2018ArticularContactForce::getTargetRegionalContactArea (const SimTK::State & state) const inline

◆ getTargetRegionalContactForce()

SimTK::Vector_<SimTK::Vec3>
OpenSim::Smith2018ArticularContactForce::getTargetRegionalContactForce (const SimTK::State & state) const inline

◆ getTargetRegionalContactMoment()

```
SimTK::Vector_<SimTK::Vec3>
OpenSim::Smith2018ArticularContactForce::getTargetRegionalContactMoment ( const SimTK::State & state ) const inline
```

◆ getTargetRegionalMaxPressure()

```
SimTK::Vector OpenSim::Smith2018ArticularContactForce::getTargetRegionalMaxPressure ( const SimTK::State & state ) const inline
```

◆ getTargetRegionalMaxProximity()

```
SimTK::Vector OpenSim::Smith2018ArticularContactForce::getTargetRegionalMaxProximity ( const SimTK::State & state ) const inline
```

◆ getTargetRegionalMeanPressure()

```
SimTK::Vector OpenSim::Smith2018ArticularContactForce::getTargetRegionalMeanPressure ( const SimTK::State & state ) const inline
```

◆ getTargetRegionalMeanProximity()

```
SimTK::Vector
OpenSim::Smith2018ArticularContactForce::getTargetRegionalMeanProximity ( const SimTK::State & state ) const inline
```

◆ getTargetTotalCenterOfPressure()

```
SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::getTargetTotalCenterOfPressure ( const SimTK::State & state ) const inline
```

◆ getTargetTotalCenterOfProximity()

```
SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::getTargetTotalCenterOfProximity ( const SimTK::State & state ) const inline
```

◆ getTargetTotalContactArea()

```
double OpenSim::Smith2018ArticularContactForce::getTargetTotalContactArea ( const SimTK::State & state ) const inline
```

◆ getTargetTotalContactForce()

```
SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::getTargetTotalContactForce ( const SimTK::State & state ) const inline
```

◆ getTargetTotalContactMoment()

```
SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::getTargetTotalContactMoment ( const SimTK::State & state ) const inline
```

◆ getTargetTotalMaxPressure()

double OpenSim::Smith2018ArticularContactForce::getTargetTotalMaxPressure (const SimTK::State & state) const inline

◆ getTargetTotalMaxProximity()

double OpenSim::Smith2018ArticularContactForce::getTargetTotalMaxProximity (const SimTK::State & state) const inline

◆ getTargetTotalMeanPressure()

double OpenSim::Smith2018ArticularContactForce::getTargetTotalMeanPressure (const SimTK::State & state) const inline

◆ getTargetTotalMeanProximity()

double OpenSim::Smith2018ArticularContactForce::getTargetTotalMeanProximity (const SimTK::State & state) const inline

◆ getTargetTrianglePotentialEnergy()

SimTK::Vector
OpenSim::Smith2018ArticularContactForce::getTargetTrianglePotentialEnergy (const SimTK::State & state) const inline

◆ getTargetTrianglePressure()

SimTK::Vector OpenSim::Smith2018ArticularContactForce::getTargetTrianglePressure (const SimTK::State & state) const inline

◆ getTargetTriangleProximity()

SimTK::Vector OpenSim::Smith2018ArticularContactForce::getTargetTriangleProximity (const SimTK::State & state) const inline

◆ safeDownCast()

static Smith2018ArticularContactForce*
OpenSim::Smith2018ArticularContactForce::safeDownCast (OpenSim::Object * obj) inline static

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

Example: `cObj = Smith2018ArticularContactForce.safeDownCast(obj)`. This is equivalent to `dynamic_cast<Smith2018ArticularContactForce*>(obj)` in C++.

◆ set_elastic_foundation_formulation()

void OpenSim::Smith2018ArticularContactForce::set_elastic_foundation_formulation (const std::string & value) inline

Set the value of the **elastic_foundation_formulation** property.

◆ set_max_proximity()

void OpenSim::Smith2018ArticularContactForce::set_max_proximity (const double & value) inline

Set the value of the **max_proximity** property.

◆ set_min_proximity()

void OpenSim::Smith2018ArticularContactForce::set_min_proximity (const double & value) inline

Set the value of the **min_proximity** property.

◆ set_use_lumped_contact_model()

void OpenSim::Smith2018ArticularContactForce::set_use_lumped_contact_model (const bool & value) inline

Set the value of the **use_lumped_contact_model** property.

◆ upd_elastic_foundation_formulation()

std::string& OpenSim::Smith2018ArticularContactForce::upd_elastic_foundation_formulation ( ) inline

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

◆ upd_max_proximity()

double& OpenSim::Smith2018ArticularContactForce::upd_max_proximity ( ) inline

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

◆ upd_min_proximity()

double& OpenSim::Smith2018ArticularContactForce::upd_min_proximity ( ) inline

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

◆ upd_use_lumped_contact_model()

bool& OpenSim::Smith2018ArticularContactForce::upd_use_lumped_contact_model ( ) inline

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

OpenSim Property, Socket, Output, Input Documentation

◆ casting_mesh

Smith2018ContactMesh OpenSim::Smith2018ArticularContactForce::casting_mesh

"Ray casting mesh for collision detection."

In an XML file, you can set this **Socket**'s connectee path via the **<socket_casting_mesh >** element. This socket was generated with the **OpenSim_DECLARE_SOCKET** macro; see **AbstractSocket** for more information.

See also
connectSocket_casting_mesh()

◆ casting_num_contacting_triangles

int OpenSim::Smith2018ArticularContactForce::casting_num_contacting_triangles

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

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

◆ casting_regional_center_of_pressure

SimTK::Vector_<SimTK::Vec3> OpenSim::Smith2018ArticularContactForce::casting_regional_center_of_pressure

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

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

◆ casting_regional_center_of_proximity

SimTK::Vector_<SimTK::Vec3> OpenSim::Smith2018ArticularContactForce::casting_regional_center_of_proximity

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

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

◆ casting_regional_contact_area

SimTK::Vector OpenSim::Smith2018ArticularContactForce::casting_regional_contact_area

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

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

◆ casting_regional_contact_force

SimTK::Vector_<SimTK::Vec3> OpenSim::Smith2018ArticularContactForce::casting_regional_contact_force

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

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

◆ casting_regional_contact_moment

SimTK::Vector_<SimTK::Vec3> OpenSim::Smith2018ArticularContactForce::casting_regional_contact_moment

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

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

◆ casting_regional_max_pressure

SimTK::Vector OpenSim::Smith2018ArticularContactForce::casting_regional_max_pressure

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

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

◆ casting_regional_max_proximity

SimTK::Vector OpenSim::Smith2018ArticularContactForce::casting_regional_max_proximity

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

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

◆ casting_regional_mean_pressure

SimTK::Vector OpenSim::Smith2018ArticularContactForce::casting_regional_mean_pressure

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

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

◆ casting_regional_mean_proximity

SimTK::Vector OpenSim::Smith2018ArticularContactForce::casting_regional_mean_proximity

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

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

◆ casting_total_center_of_pressure

SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::casting_total_center_of_pressure

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

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

◆ casting_total_center_of_proximity

SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::casting_total_center_of_proximity

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

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

◆ casting_total_contact_area

double OpenSim::Smith2018ArticularContactForce::casting_total_contact_area

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

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

◆ casting_total_contact_force

SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::casting_total_contact_force

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

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

◆ casting_total_contact_moment

SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::casting_total_contact_moment

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

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

◆ casting_total_max_pressure

double OpenSim::Smith2018ArticularContactForce::casting_total_max_pressure

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

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

◆ casting_total_max_proximity

double OpenSim::Smith2018ArticularContactForce::casting_total_max_proximity

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

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

◆ casting_total_mean_pressure

double OpenSim::Smith2018ArticularContactForce::casting_total_mean_pressure

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

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

◆ casting_total_mean_proximity

double OpenSim::Smith2018ArticularContactForce::casting_total_mean_proximity

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

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

◆ casting_triangle_potential_energy

SimTK::Vector OpenSim::Smith2018ArticularContactForce::casting_triangle_potential_energy

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

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

◆ casting_triangle_pressure

SimTK::Vector OpenSim::Smith2018ArticularContactForce::casting_triangle_pressure

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

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

◆ casting_triangle_proximity

SimTK::Vector OpenSim::Smith2018ArticularContactForce::casting_triangle_proximity

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

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

◆ elastic_foundation_formulation

std::string OpenSim::Smith2018ArticularContactForce::elastic_foundation_formulation

"Formulation for depth-pressure relationship: " "'linear' or 'nonlinear'. " "Default value set to 'linear'."

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

See also
get_elastic_foundation_formulation(), **upd_elastic_foundation_formulation()**, **set_elastic_foundation_formulation()**

◆ max_proximity

double OpenSim::Smith2018ArticularContactForce::max_proximity

"The maximum proximity " "that is valid between contacting meshes to limit the search distance " "along the casting_mesh normal ray used for collision detection." "Default value set to 0.01 meters."

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

See also
get_max_proximity(), **upd_max_proximity()**, **set_max_proximity()**

◆ min_proximity

double OpenSim::Smith2018ArticularContactForce::min_proximity

"The minimum proximity " "that is valid between contacting meshes to limit the search distance " "along the casting_mesh normal ray used for collision detection. Note " "this can be negative if proximity maps should include triangles that " "are not in contact." "Default value set to 0.0 meters."

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

See also
get_min_proximity(), **upd_min_proximity()**, **set_min_proximity()**

◆ target_mesh

Smith2018ContactMesh OpenSim::Smith2018ArticularContactForce::target_mesh

"Target mesh for collision detection."

In an XML file, you can set this **Socket**'s connectee path via the **<socket_target_mesh >** element. This socket was generated with the **OpenSim_DECLARE_SOCKET** macro; see **AbstractSocket** for more information.

See also
connectSocket_target_mesh()

◆ target_num_contacting_triangles

int OpenSim::Smith2018ArticularContactForce::target_num_contacting_triangles

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

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

◆ target_regional_center_of_pressure

SimTK::Vector_<SimTK::Vec3> OpenSim::Smith2018ArticularContactForce::target_regional_center_of_pressure

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

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

◆ target_regional_center_of_proximity

SimTK::Vector_<SimTK::Vec3> OpenSim::Smith2018ArticularContactForce::target_regional_center_of_proximity

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

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

◆ target_regional_contact_area

SimTK::Vector OpenSim::Smith2018ArticularContactForce::target_regional_contact_area

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

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

◆ target_regional_contact_force

SimTK::Vector_<SimTK::Vec3> OpenSim::Smith2018ArticularContactForce::target_regional_contact_force

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

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

◆ target_regional_contact_moment

SimTK::Vector_<SimTK::Vec3> OpenSim::Smith2018ArticularContactForce::target_regional_contact_moment

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

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

◆ target_regional_max_pressure

SimTK::Vector OpenSim::Smith2018ArticularContactForce::target_regional_max_pressure

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

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

◆ target_regional_max_proximity

SimTK::Vector OpenSim::Smith2018ArticularContactForce::target_regional_max_proximity

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

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

◆ target_regional_mean_pressure

SimTK::Vector OpenSim::Smith2018ArticularContactForce::target_regional_mean_pressure

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

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

◆ target_regional_mean_proximity

SimTK::Vector OpenSim::Smith2018ArticularContactForce::target_regional_mean_proximity

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

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

◆ target_total_center_of_pressure

SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::target_total_center_of_pressure

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

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

◆ target_total_center_of_proximity

SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::target_total_center_of_proximity

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

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

◆ target_total_contact_area

double OpenSim::Smith2018ArticularContactForce::target_total_contact_area

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

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

◆ target_total_contact_force

SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::target_total_contact_force

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

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

◆ target_total_contact_moment

SimTK::Vec3 OpenSim::Smith2018ArticularContactForce::target_total_contact_moment

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

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

◆ target_total_max_pressure

double OpenSim::Smith2018ArticularContactForce::target_total_max_pressure

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

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

◆ target_total_max_proximity

double OpenSim::Smith2018ArticularContactForce::target_total_max_proximity

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

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

◆ target_total_mean_pressure

double OpenSim::Smith2018ArticularContactForce::target_total_mean_pressure

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

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

◆ target_total_mean_proximity

double OpenSim::Smith2018ArticularContactForce::target_total_mean_proximity

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

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

◆ target_triangle_potential_energy

SimTK::Vector OpenSim::Smith2018ArticularContactForce::target_triangle_potential_energy

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

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

◆ target_triangle_pressure

SimTK::Vector OpenSim::Smith2018ArticularContactForce::target_triangle_pressure

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

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

◆ target_triangle_proximity

SimTK::Vector OpenSim::Smith2018ArticularContactForce::target_triangle_proximity

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

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

◆ use_lumped_contact_model

bool OpenSim::Smith2018ArticularContactForce::use_lumped_contact_model

"Combine the thickness and the average material properties between " "the Smith2018ContactMeshes for both meshes and use Bei & Fregly 2003 " "lumped parameter Elastic Foundation model."

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

See also

get_use_lumped_contact_model(), **upd_use_lumped_contact_model()**, **set_use_lumped_contact_model()**

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

- OpenSim/Simulation/Model/Smith2018ArticularContactForce.h