Simulation of conforming contact in real-time multibody dynamics using a volumetric force model

David Vilela Freire

Ferrol, December 10th 2018

Index

- Motivation
- Multibody dynamics
- Contact model
- Collision detection
- Results
- Conclusions and future work

Motivation

Motivation

Simulation

- One of the most commonly used tools in the industry to research, create, test and operate machines and mechanisms
- Source of costs and time savings
- In the manufacturing line:
  - Fast design evaluation
  - Prototype testing stage reduction
  - Assembly line early error detection
- As training platform:
  - Operator physical risk minimization
  - Hardware costs and risk reduction for expensive machinery
  - Training courses automated evaluation
  - Simulation of hard or unusual working conditions

Virtual assembly

- Contact is a key factor to obtain accurate results in complex simulations
- Many machinery parts fit into each other, creating **conforming contacts:** the size of the contact footprint is not negligible compared to the size of the bodies
- Human-machine interaction imposes **real-time** execution requirements
- Contact model must be accurate and realistic

The problem with many contact models: realistic contacts and collisions

[Forza Horizon 4]

Motivation

DYNAMICS ENGINE
Real-time
Numerically robust
Small number of contacts
Not realistic

FEM METHODS
Accurate
Closer to a real pressure distribution
Many contacts
Slow

WHAT WE NEED
Real-time capable
Numerically robust
Accurate

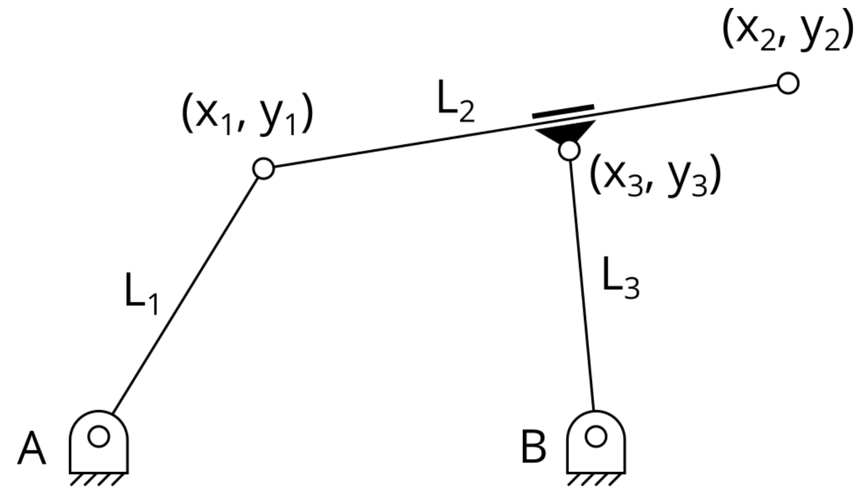
Multibody dynamics

Multibody dynamics

- Multibody dynamics enables the systematic computation of motion in mechanical systems
- The system is defined as an interconnected group of every element
- Robust, efficient, flexible and real-time capable
- A set of **coordinates**, a **formulation** and an **integrator** are combined to mathematically express mechanism dynamics

Natural coordinates

- Every element is defined independently
- Reference points are located on the joints and can be shared
- 4 entities needed: 1 point + 3 vectors
- Position and orientation are expressed easily
- Constraints are simple expressions



Index-3 Augmented Lagrangian formulation with projection of velocities and accelerations

DAE

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\lambda = \mathbf{Q}(\mathbf{q}, \dot{\mathbf{q}})$$
$$\mathbf{\Phi}(\mathbf{q}, t) = \mathbf{0}$$

ODE

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\lambda^{*} + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\alpha\mathbf{\Phi} = \mathbf{Q}(\mathbf{q}, \dot{\mathbf{q}})$$
$$\lambda_{i+1}^{*} = \lambda_{i}^{*} + \alpha\mathbf{\Phi}_{i+1},\ i = 0, 1, 2, \ldots$$

- Penalty at position level only
- Contraints are fullfilled but their derivatives ($\dot{\mathbf{\Phi}}, \ddot{\mathbf{\Phi}}$) are not
- Velocities and accelerations must be projected

$$\min V = \frac{1}{2}(\dot{\mathbf{q}} - \dot{\mathbf{q}}^{*})^{\mathrm{T}}\mathbf{M}(\dot{\mathbf{q}} - \dot{\mathbf{q}}^{*})$$
$$\text{subject to } \dot{\Phi}(\mathbf{q}, \dot{\mathbf{q}}, t) = \mathbf{0}$$

$$\min\ \mathrm{V} = \frac{1}{2}(\ddot{\mathbf{q}} - \ddot{\mathbf{q}}^{*})^{\mathrm{T}}\mathbf{M}(\ddot{\mathbf{q}} - \ddot{\mathbf{q}}^{*})$$
$$\text{subject to } \ddot{\Phi}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, t) = 0$$

Newmark integrators

- Implicit: I3AL formulation couples solving of the EOM with integration
- The system of equations is solved by the Newton-Raphson iteration

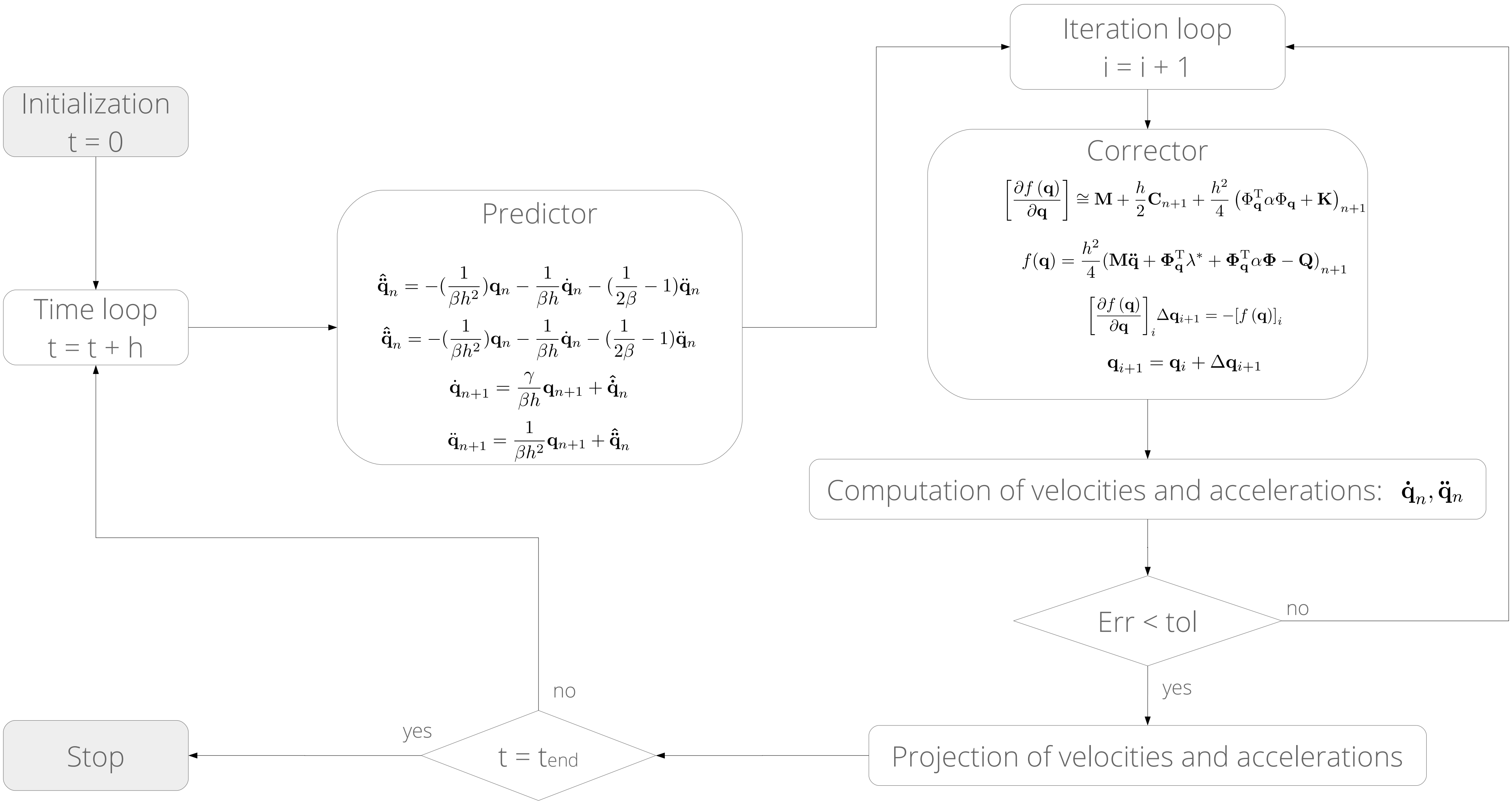
$$\left[\frac{\partial f(\mathbf{q})}{\partial \mathbf{q}}\right]_i \Delta\mathbf{q}_{i+1} = -[f(\mathbf{q})]_i$$

$$\mathbf{q}_{i+1} = \mathbf{q}_i + \Delta\mathbf{q}_{i+1}$$

- Terms from applied forces must be differenciated, and an approximation for the tangent matrix is used

$$f(\mathbf{q}) = \mathbf{M}\mathbf{q}_{n+1} + \frac{h^2}{4}\mathbf{\Phi}^{\mathrm{T}}_{\mathbf{q}_{n+1}}(\alpha\mathbf{\Phi}_{n+1} + \lambda_{n+1}) - \frac{h^2}{4}\mathbf{Q}_{n+1} + \frac{h^2}{4}\mathbf{M}\hat{\ddot{\mathbf{q}}}_n = \mathbf{0}$$

$$\left[\frac{\partial f(\mathbf{q})}{\partial \mathbf{q}}\right] \approx \mathbf{M} + \frac{h}{2}\mathbf{C} + \frac{h^2}{4}(\mathbf{\Phi}^{\mathrm{T}}_{\mathbf{q}}\alpha\mathbf{\Phi}_{\mathbf{q}} + \mathbf{K})$$



Iteration loop
i = i + 1
Initialization
t = 0
Corrector
$\left[\frac{\partial f(\mathbf{q})}{\partial \mathbf{q}}\right] \cong \mathbf{M} + \frac{h}{2}\mathbf{C}_{n+1} + \frac{h^2}{4}\left(\Phi_{\mathbf{q}}^{\mathrm{T}}\alpha\Phi_{\mathbf{q}} + \mathbf{K}\right)_{n+1}$
$f(\mathbf{q}) = \frac{h^2}{4}(\mathbf{M}\ddot{\mathbf{q}} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\lambda^* + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\alpha\boldsymbol{\Phi} - \mathbf{Q})_{n+1}$
$\left[\frac{\partial f(\mathbf{q})}{\partial \mathbf{q}}\right]_i \Delta\mathbf{q}_{i+1} = -[f(\mathbf{q})]_i$
$\mathbf{q}_{i+1} = \mathbf{q}_i + \Delta\mathbf{q}_{i+1}$
Predictor
$\hat{\ddot{\mathbf{q}}}_n = -(\frac{1}{\beta h^2})\mathbf{q}_n - \frac{1}{\beta h}\dot{\mathbf{q}}_n - (\frac{1}{2\beta} - 1)\ddot{\mathbf{q}}_n$
$\hat{\ddot{\mathbf{q}}}_n = -(\frac{1}{\beta h^2})\mathbf{q}_n - \frac{1}{\beta h}\dot{\mathbf{q}}_n - (\frac{1}{2\beta} - 1)\ddot{\mathbf{q}}_n$
$\dot{\mathbf{q}}_{n+1} = \frac{\gamma}{\beta h}\mathbf{q}_{n+1} + \hat{\dot{\mathbf{q}}}_n$
$\ddot{\mathbf{q}}_{n+1} = \frac{1}{\beta h^2}\mathbf{q}_{n+1} + \hat{\ddot{\mathbf{q}}}_n$
Time loop
t = t + h
Computation of velocities and accelerations: $\dot{\mathbf{q}}_n, \ddot{\mathbf{q}}_n$
Err < tol
no
yes
no
yes
Stop
t = t_{end}
Projection of velocities and accelerations

Contact model

Contact model

- There are different approaches to the contact implementation
- Contact complexity determines the contact model requirements
- Virtual assembly requirements:
  - **Accurate** in order to simulate machinery pieces that drive each other
  - **Real-time** to allow interactive frame-rates
  - **Conforming-contact capable** to enable contacts between arbitrary-size surfaces and fitting pieces

Contact model

Complementary methods

- Instantaneous impacts through instantaneous change in the balance momenta
- Imposed restrictions to handle long-duration contacts
- Not well-suited for neither I3AL formulation nor our friction model

Penalty methods

- Based on regularized-force models
- Forces are proportional to deformation to avoid penetration
- Usually based on Hertz contact theory:
  - Assume contact areas much smaller than the characteristic body dimensions
  - Not valid for conforming contact

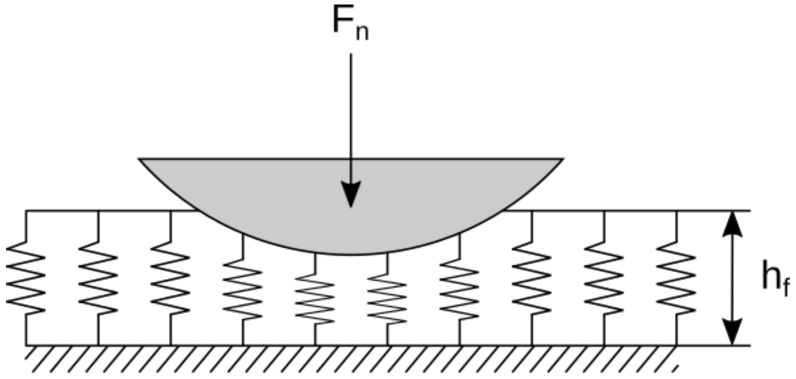
Contact model

Gonthier model

- Based on a modified Winkler elastic foundation
- Mimics a contact force distribution derived from the inter-penetration
- Based on the mesh intersection volumetric properties
- Includes spinning friction and rolling resistance
- Valid for any geometries with reasonably flat contact area

Normal model

$$\mathbf{F_n} = \frac{k_n}{h_n} V (1 + a v_n) \mathbf{n}$$



Tangential model

- Sliding friction:

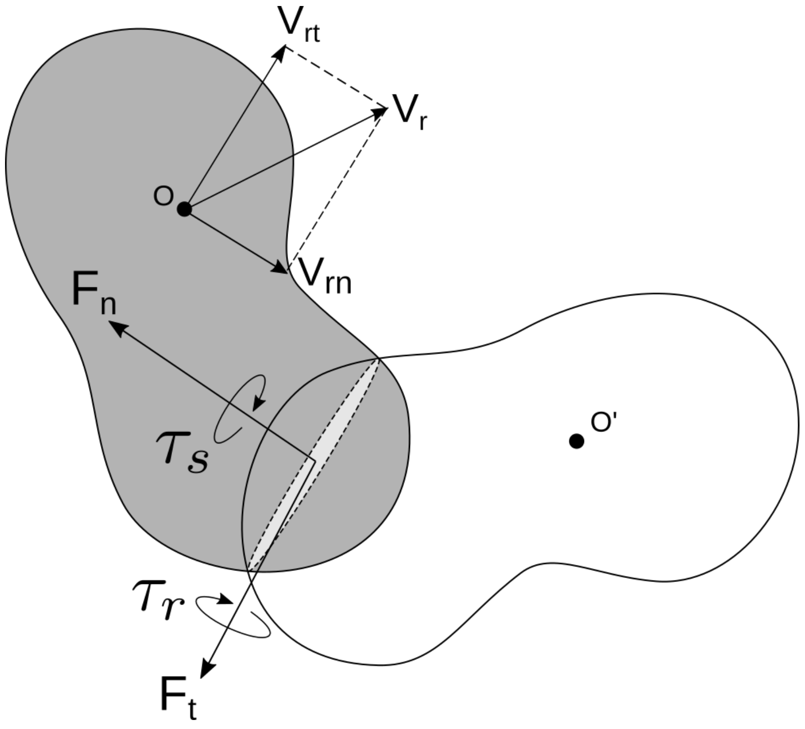
$$\mathbf{F_t} = F_n(\mathbf{v_t} + \omega_\mathbf{t} \times \rho_\mathbf{n}) \quad \Longrightarrow \quad \mathbf{F_t} = F_n \mathbf{v_t}$$

- Rolling resistance:

$$\tau_r = \frac{k_n a}{h_n} \mathbf{I_g} \omega_\mathrm{t}$$

- Spinning friction:

$$\tau_s = \frac{F_n}{V} \mathbf{I_g} \omega_\mathrm{n}$$

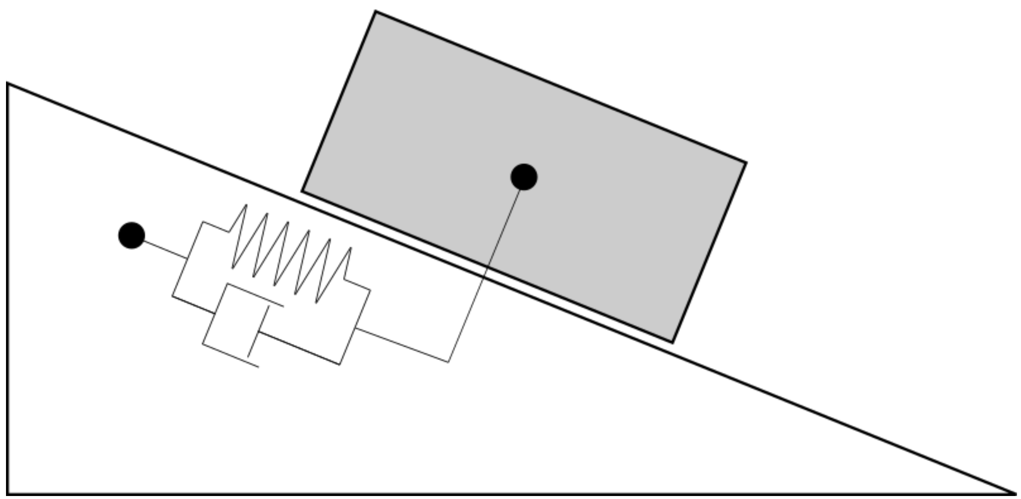


Bristle model [*Dopico10*]

- The stiction force is considered by means of viscoelastic elements acting between the colliding bodies, called bristles

$$\mathbf{F_t} = \kappa \mathbf{F_{stick}} + (1-\kappa)\mathbf{F_{slide}} - \mu_{visc}\mathbf{v_t}$$

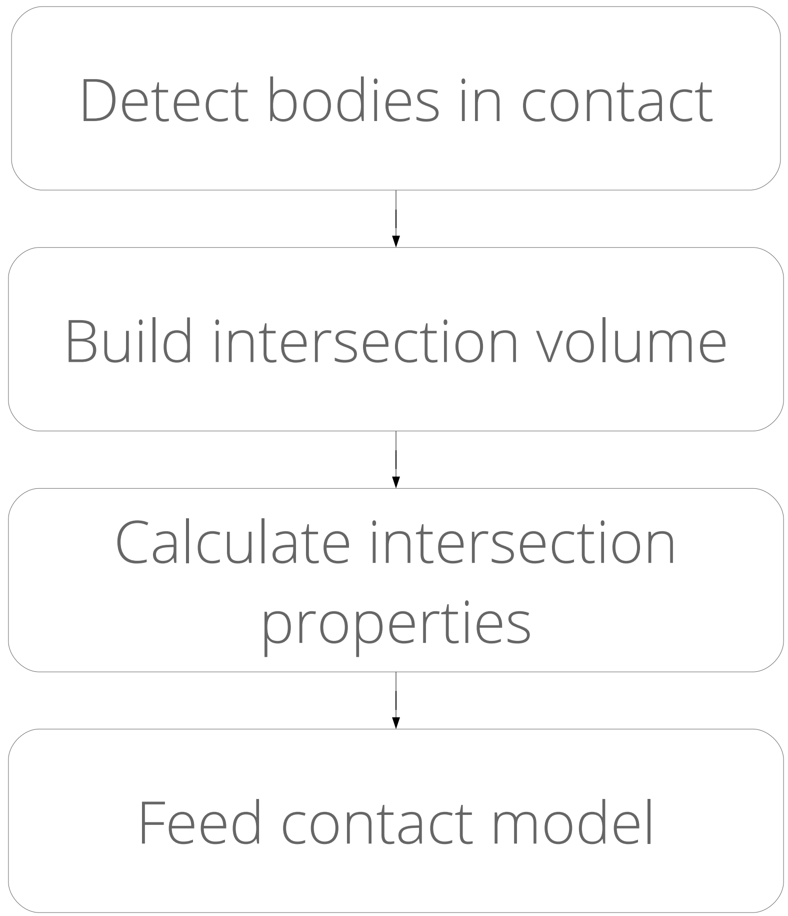
$$\kappa = \left\{ \begin{array}{ll} 0; & \|\mathbf{v}_t\| >> v_{stick} \\ 1; & \|\mathbf{v}_t\| = 0 \end{array} \right\} = \mathrm{e}^{-\left(\mathbf{v}_t^{\mathrm{T}}\mathbf{v}_t\right)/v_{stick}^2}$$



Collision detection

Collision detection stages

- Force model requires the computation of intersection volume, centroid and inertia tensor.

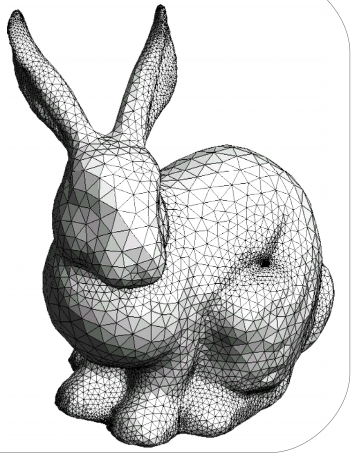


Geometric representation

- NURBS: free form surfaces. Flexible, non-real-time
- CSG: boolean operations. Non general
- Mesh discretization. General and real-time, approximate

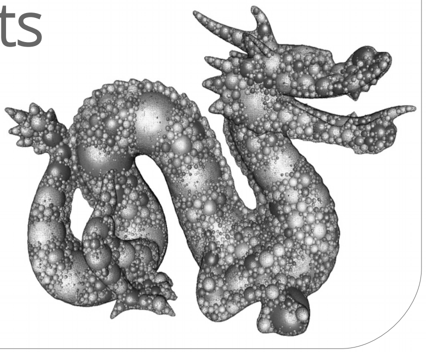
Method A:
Surface elements

- Triangle meshes



Method B:
Volume elements

- Sphere fillings



Triangle mesh method implementation

- LIMCODE: LIM Collision Detection
  - C++ language
  - Does not require a specific mesh format
  - Data type agnostic: float, double, multi-precision

Far detection stage

- Object level
- Simple tests to discard objects far apart
- Ignored if few objects

→

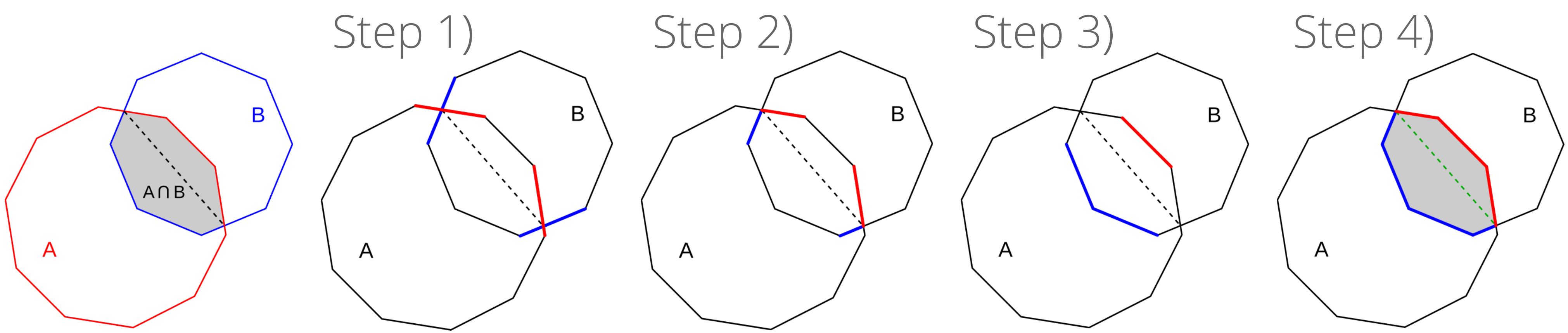
Near detection stage

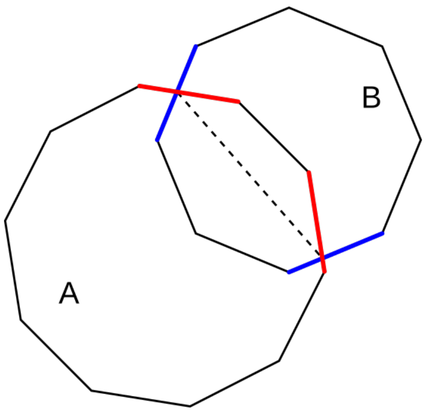
- Primitive level
- Precise intersections
- Hierarchical classification

Collision detection

Intersection volume computation: overview

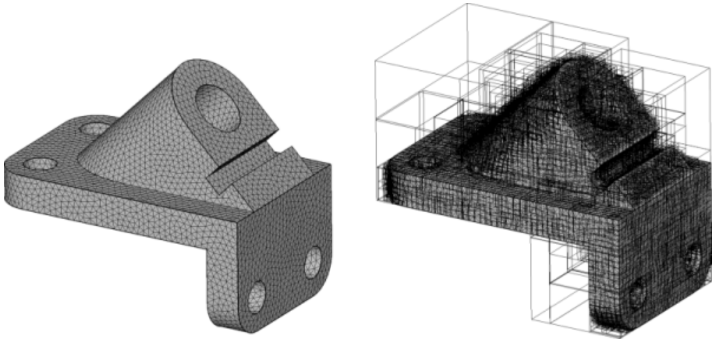
1. Intersecting faces detection
2. Clipping
3. Internal faces detection
4. Reconstruction

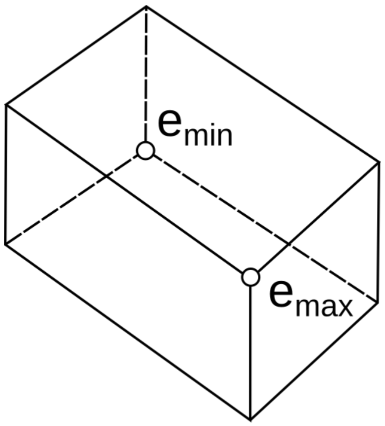




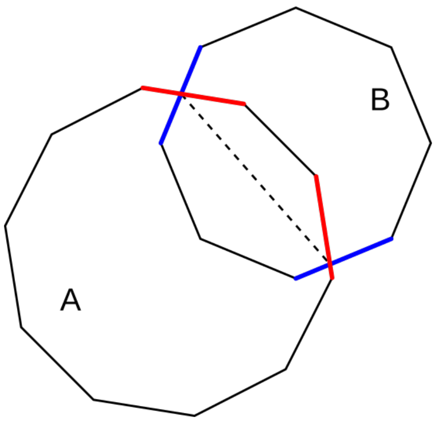
Step 1: AABB tree

- Hierarchical node structure
- Every node is a AABB volume
- Node AABBs are split by a plane perpendicular to their biggest dimension axis passing through the triangle group COM
- Collision check is performed AABB vs OOBB: $T'_B = T_A^{-1} T_B$
- $O(\log(n))$
- Traversal ends when two leaf nodes holding one triangle are tested

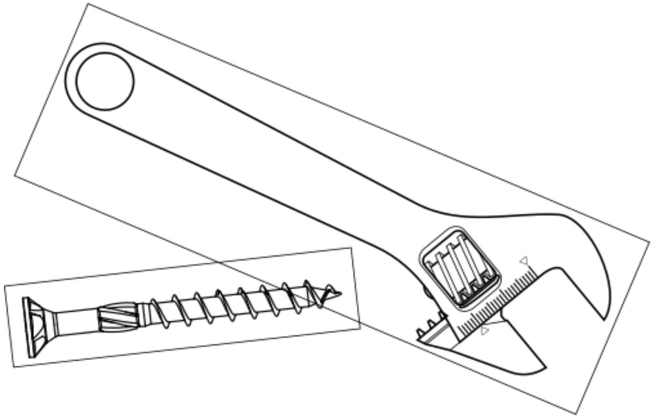


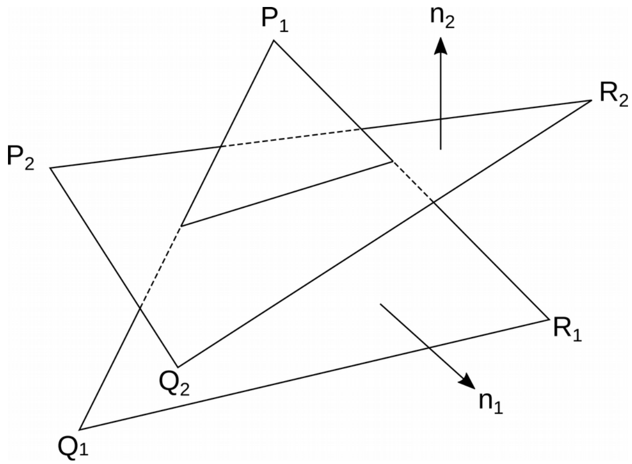


Step 1: Triangle intersection



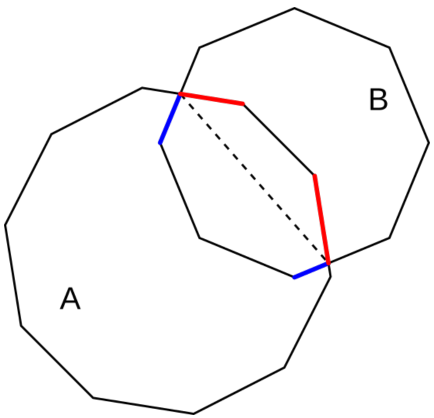
- AABB node test is not sufficient: box-box intersection does not guarantee mesh contact
- Devillers-Guigue method:
  - Combinatorial stage: vertex relative position detection and reordering
  - Evaluation stage: actual collision check

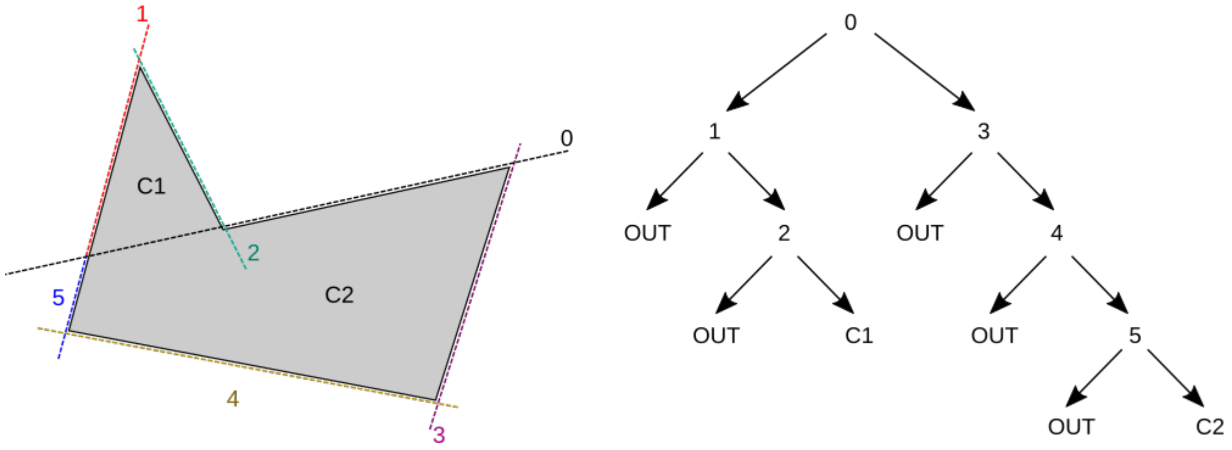




Step 2: Binary Space Partition Tree

- Needed to check if an entity lies in the inside/outside of the object
- Used for determining inner/outer points while clipping triangles
- Successively splits the mesh by a plane until each sub-mesh is a convex cell
- Object facets are used as splitting planes

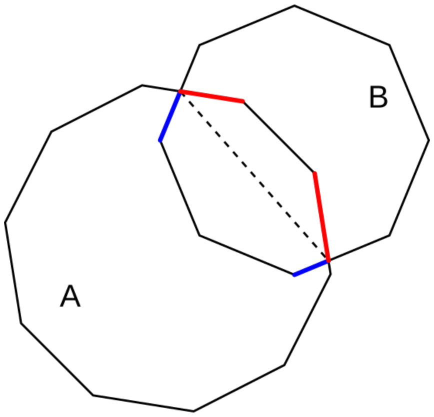


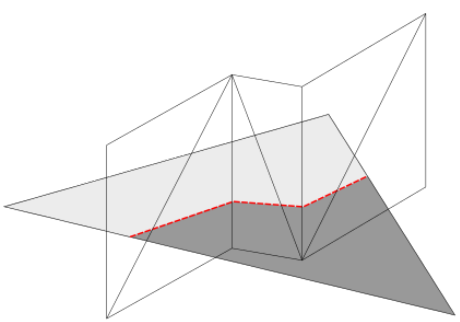


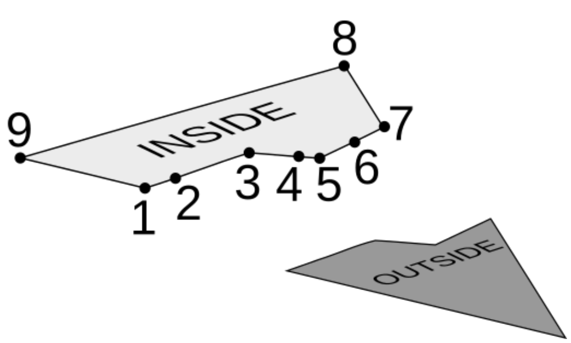
Collision detection

Step 2: Polygon clipping

- During the intersection stage, polygons must be cut against a plane
- Clipped polygons must be reconstructed
- Brittle numerical task if performed naively. Risk of inconsistent output
- Bernstein’s implementation:
  - Follow each vertex in order and classify it using predicates
  - Implemented as a Finite State Machine







IN	9	1	2	3	4	5	6	7	8
OUT		1	2	3	4	5	6	7	

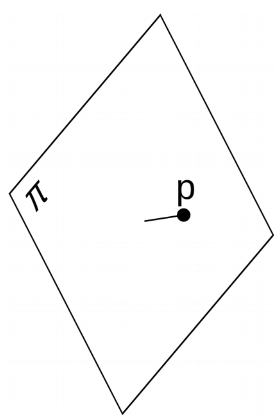
Geometrical predicates

- Classification operations on meshes are floating-point error prone, which can generate non-watertight meshes that lead to force inconsistencies
- Predicates are modularized expressions of common geometrical checks
- Geometrical inconsistencies are removed
- Common examples: point over/under plane, line-triangle intersection

Sugihara

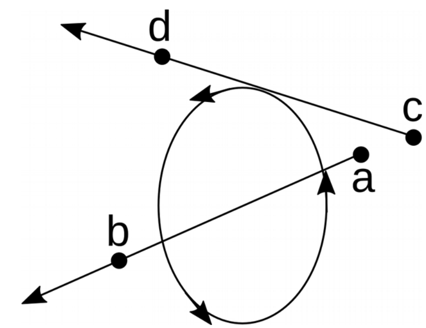
$$\mathbf{p} = [x, y, z]^T,\ \pi = ax + by + cz + d$$

$$\begin{vmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ a_\pi & b_\pi & c_\pi & d_\pi \end{vmatrix}$$

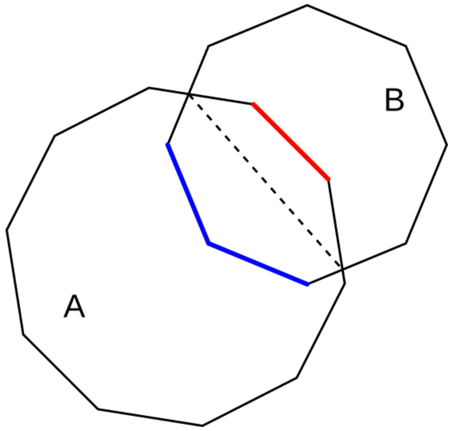


Devillers-Guige

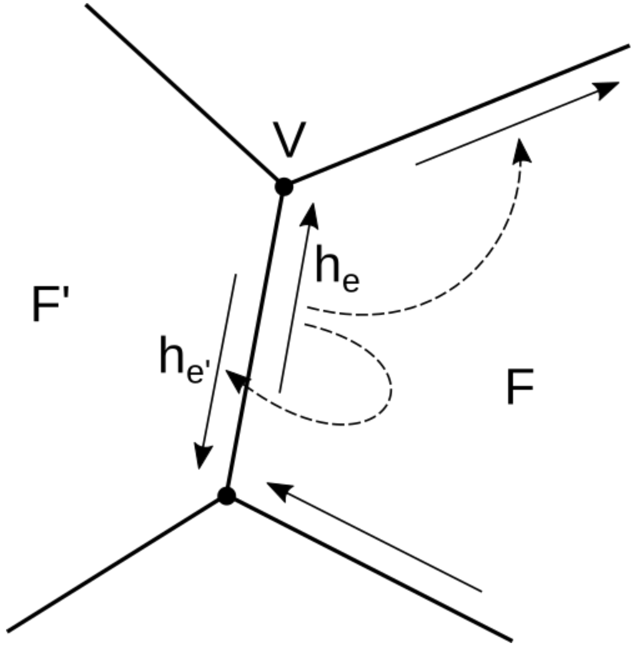
$$\begin{vmatrix} a_x & a_y & a_z & d_1 \\ b_x & b_y & b_z & d_1 \\ c_x & c_y & c_z & d_1 \\ d_x & d_y & d_z & d_1 \end{vmatrix}$$



Step 3: Half-edge structure [Lee79]

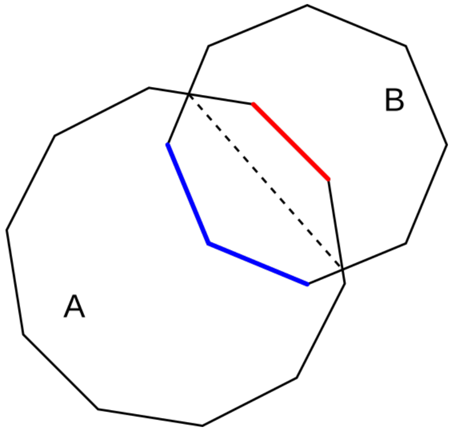


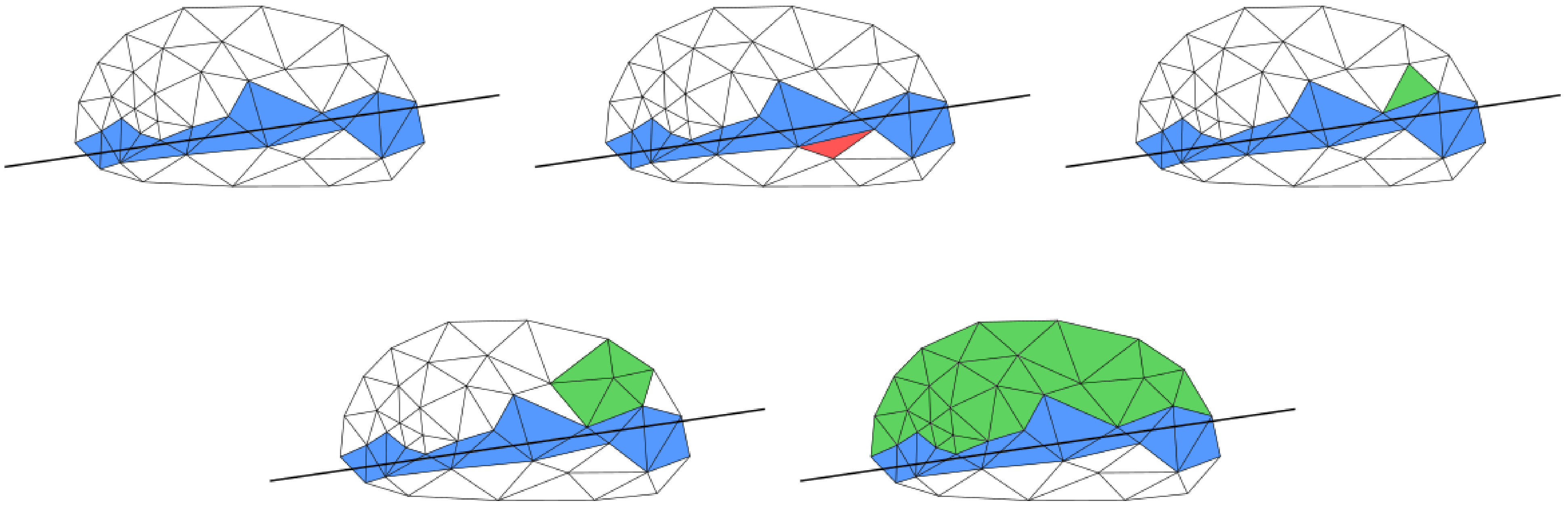
- Used to traverse a mesh surface over its faces, edges or nodes
- Facets connectivity info is precomputed
- Nodes hold a reference to:
  - Next vertex
  - Actual facet
  - Opposite half-edge node
  - Next half-edge node



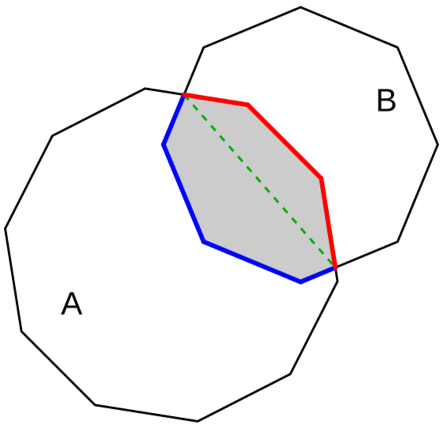
Step 3: Intersection hull closing algorithm

- Locate internal facet (BSP)
- Traverse HE structure and create neighbor list
- Add to list internal facets that are not on intersection facets list
- Continue until traversal cannot advance

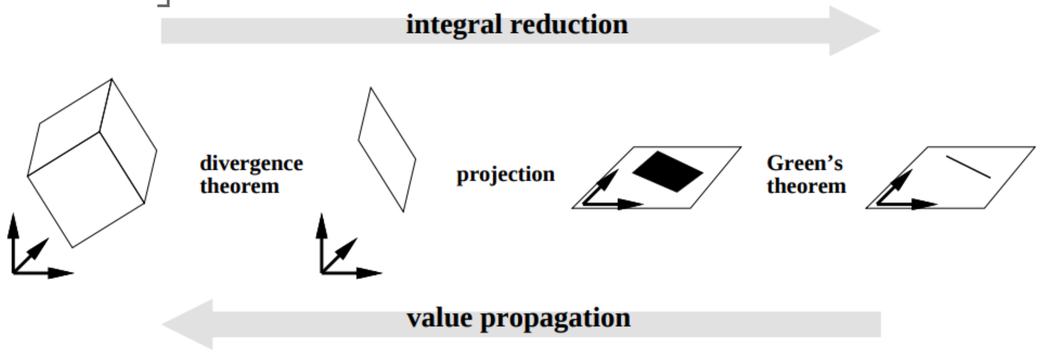




Step 4: Volume properties computation

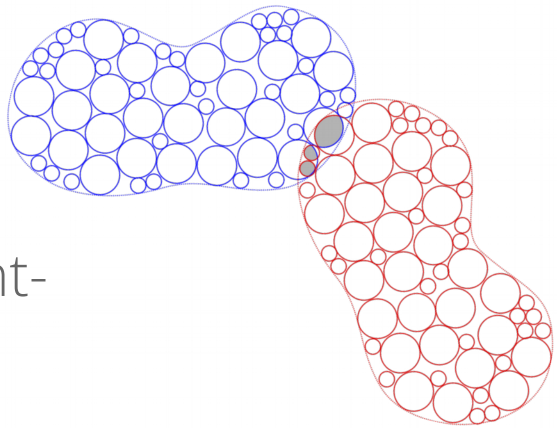


- Clipped intersecting triangles + internal triangles: manifold mesh
- Volume, center of mass and inertia tensor can be computed
- *"Fast and Accurate Computation of Polyhedral Mass Properties"* [Mirtich96]

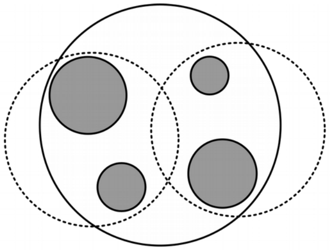


- Eberly further simplifies Mirtich's expressions for polyhedra with triangular faces

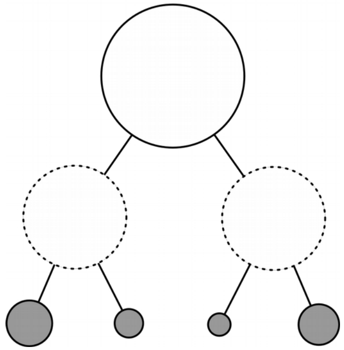
Inner Sphere Trees method implementation



- More general alternative than mesh detection
- Objects are approximated to non-overlapping, different-size sphere collections: sphere packings
- Computer Graphics Group U. Bremen: ProtoSphere, CollDet

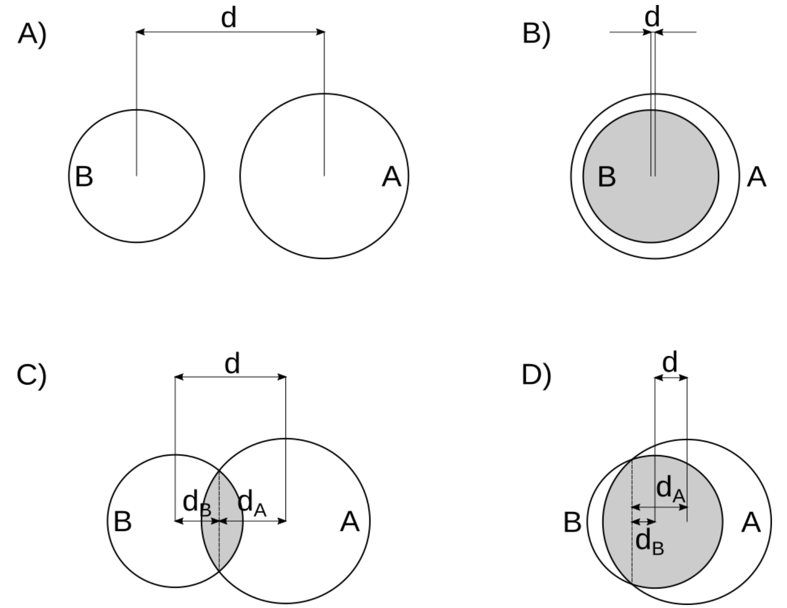


- Inner Sphere Tree hierarchy:
  - Nodes are also spheres
  - A node covers all its leaves but not all its direct children
  - ~10k spheres: rough approximation (85%), ~100 contacts
  - IST - IST intersection: sphere pairs list. **Multiple contact forces:** one for every colliding pair



Collision detection

Sphere - sphere intersection



- Distance between two given spheres ($R_A > R_B$)

$$d = \sqrt{(\mathbf{p}_B - \mathbf{p}_A)(\mathbf{p}_B - \mathbf{p}_A)}$$

$$d_A = \frac{R_A^2 + d^2 - R_B^2}{2d} \quad d_B = \frac{R_B^2 + d^2 - R_A^2}{2d}$$

$$d = d_A + d_B$$

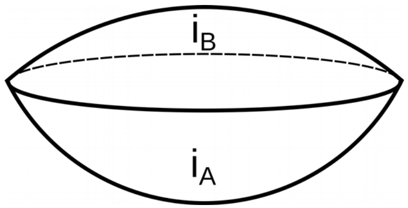
A) $d \geq R_A + R_B$: No contact

B) $R_A \geq d + R_B$: Sphere B inside A

C) $d \geq d_A$: Less than half sphere B inside A

D) $d < d_A$: More than half sphere B inside A

- Mass properties are simple expressions: spherical caps
- Intersection properties are added-up to get the total values

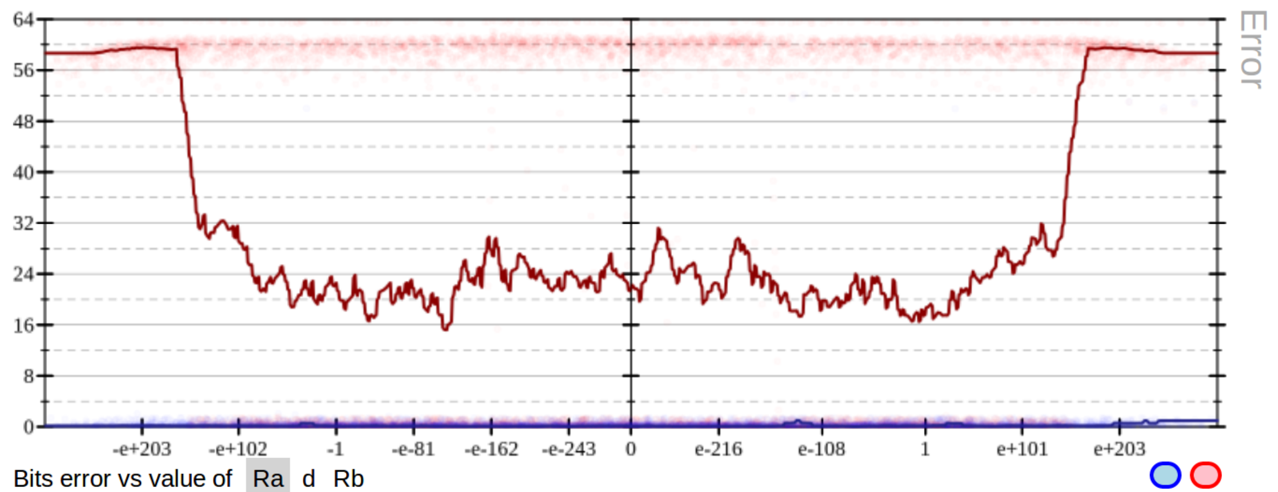


Numerical error optimization

- Previous expressions are bad numerically conditioned due to magnitude ratio between coordinates and indentations
- Truncation and loss of significance can lead to division-by-zero among other effects
- HERBIE tool: identifies errors known to operators and rewrites expressions to minimize floating-point error [Panchekha15]

$$d_A = \frac{R_A^2 + d^2 - R_B^2}{2d}$$

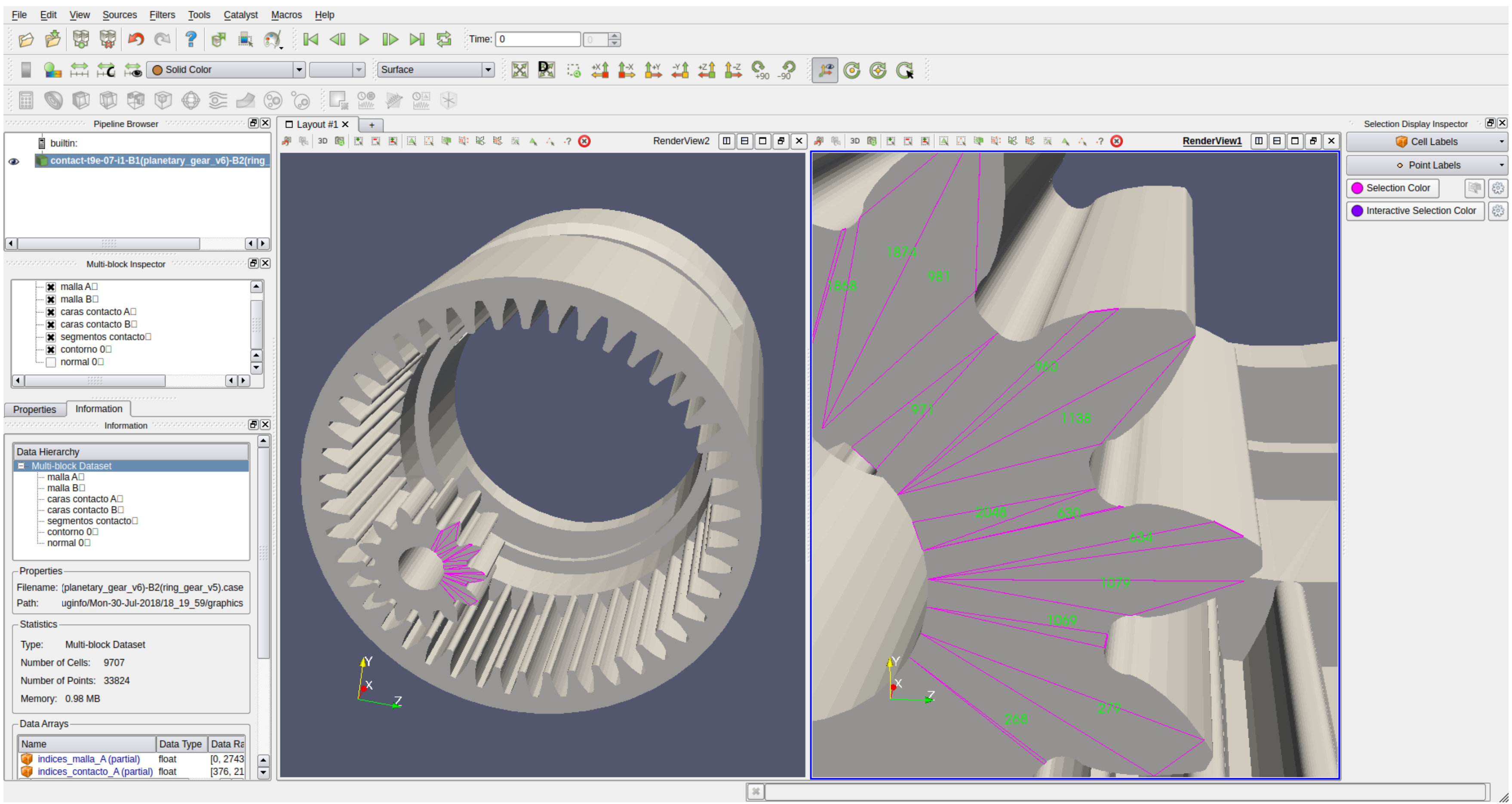
$$d_A = 0.5\frac{R_A}{\frac{d}{R_A}} + 0.5d - 0.5\frac{R_B}{\frac{d}{R_B}}$$



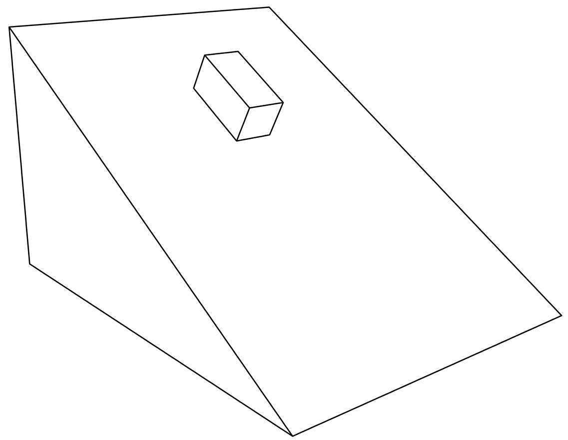
Collision detection

MBSDEBUG: data export and visualization tool

- Collision data is exported to Paraview: face/vertex indices, colliding triangles, clipped triangles, inner triangles, collision contours, normals



Results

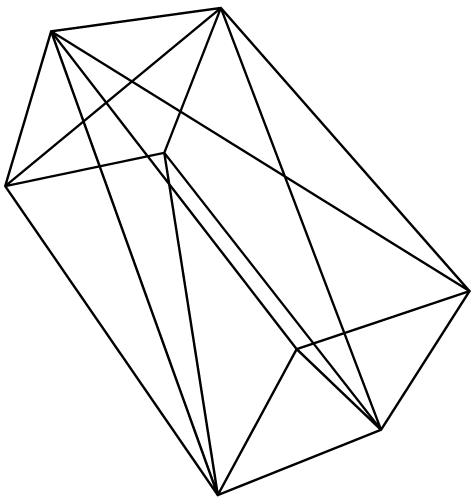


Test 1: block sliding on plane

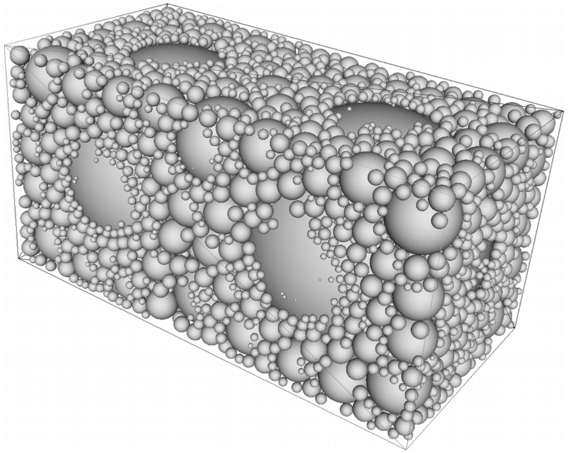
- Designed to validate the normal and tangential models
- m = 1 kg
- Dimensions 1m x 0.5m x 0.5m
- Compare critical angle vs theoretical one $\theta = atan(\mu)$
- 4-second simulation at 30º: trajectory, velocity, angular deviation (roll, pitch, yaw) errors

Test 1: mesh model

- Simplest triangular mesh possible for block and floor
- Started sliding at 26º (97.87% of the theoretical value)
- Effects of the initial non-penetration position: small transition phase
- Trajectory, velocity and angular error plots show minimal deviation from expected values



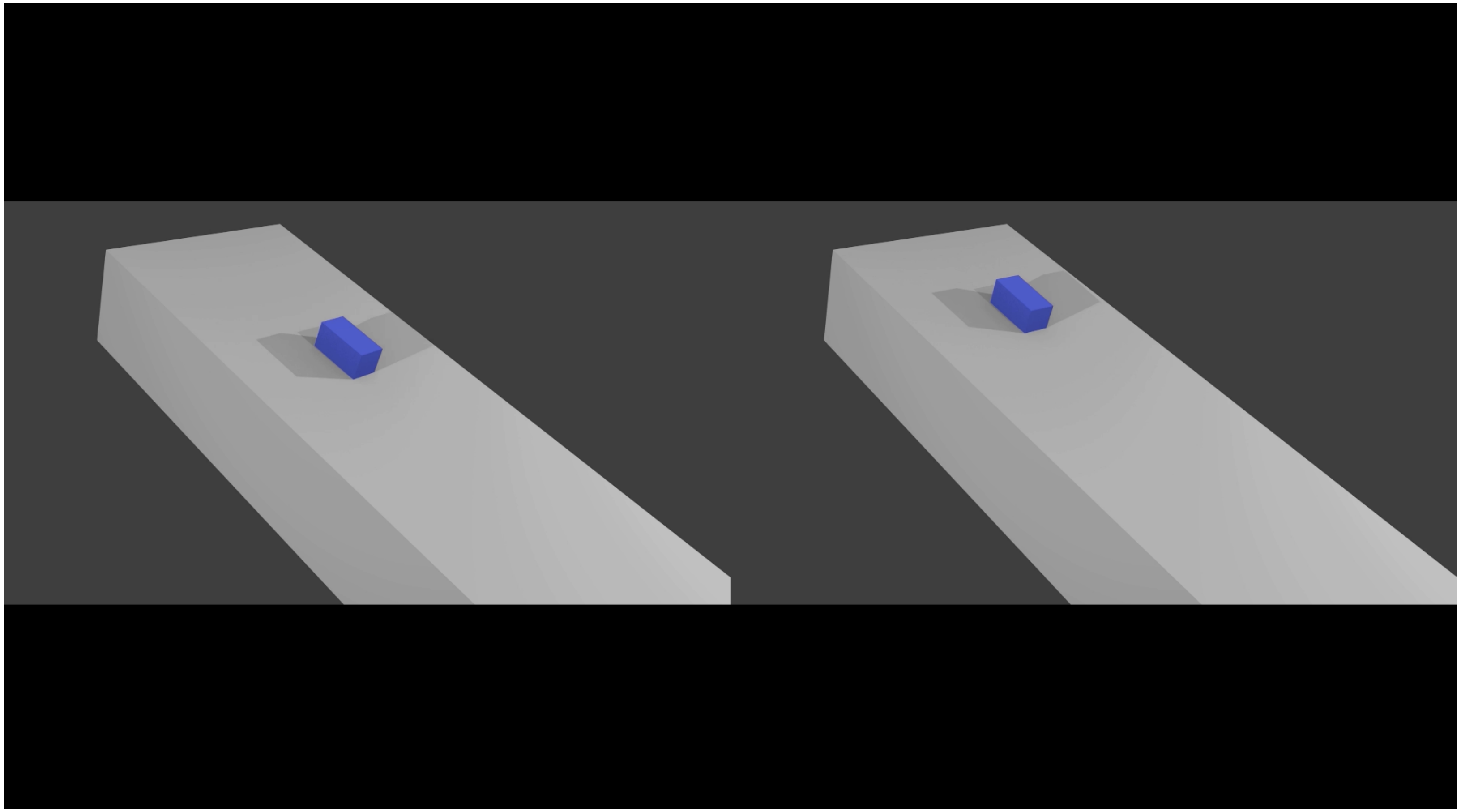
Test 1: Inner Sphere Tree model



- ~12k (block) and ~15k spheres (floor)
- Multiple forces are applied
- Bristle model does not stabilize, block is not stopped
- Trajectory, velocity and angular errors are significant

Angle (º)	Residual velocity (m/s)
10	1×10^{-5}
22	1×10^{-4}
23	1×10^{-3}
25	1×10^{-2}

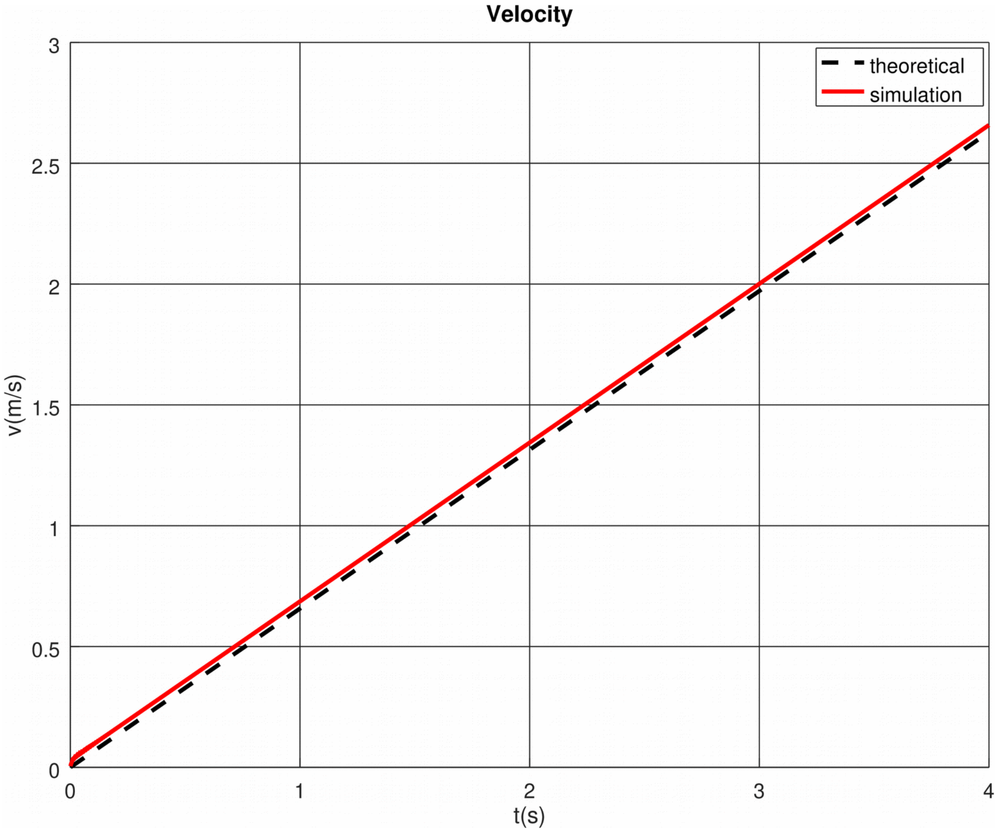
Test 1: video comparison



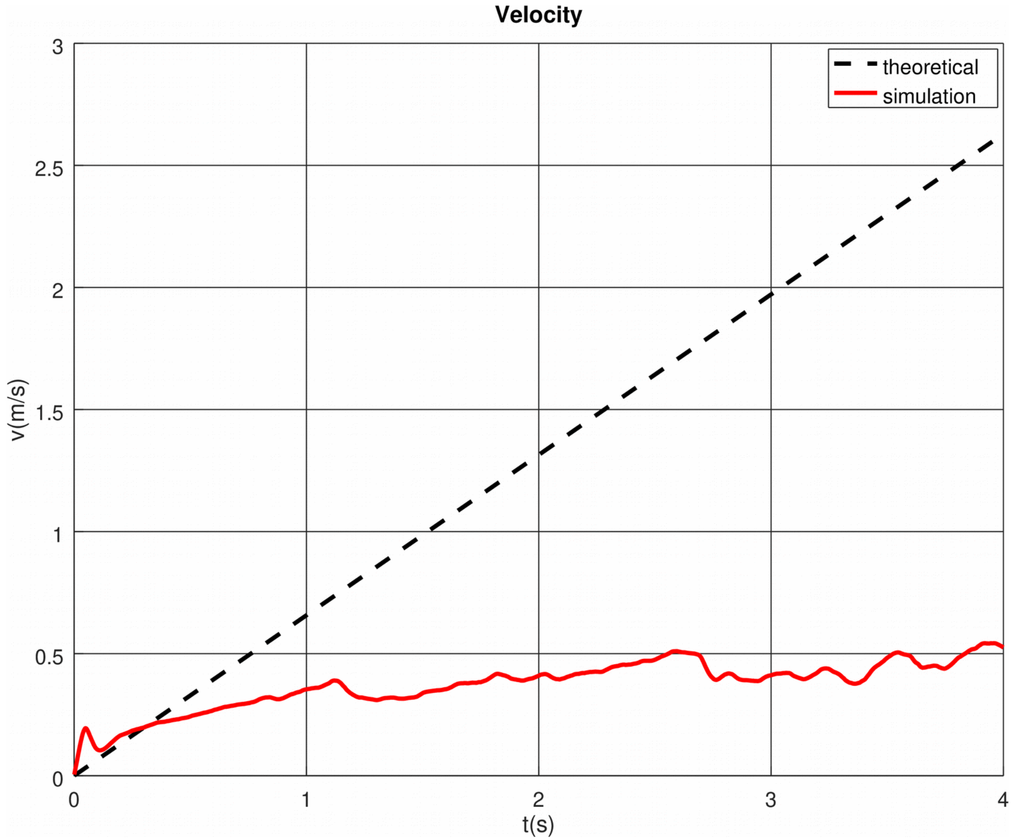
Triangle mesh model

Inner Sphere Tree model

Test 1: velocity profiles

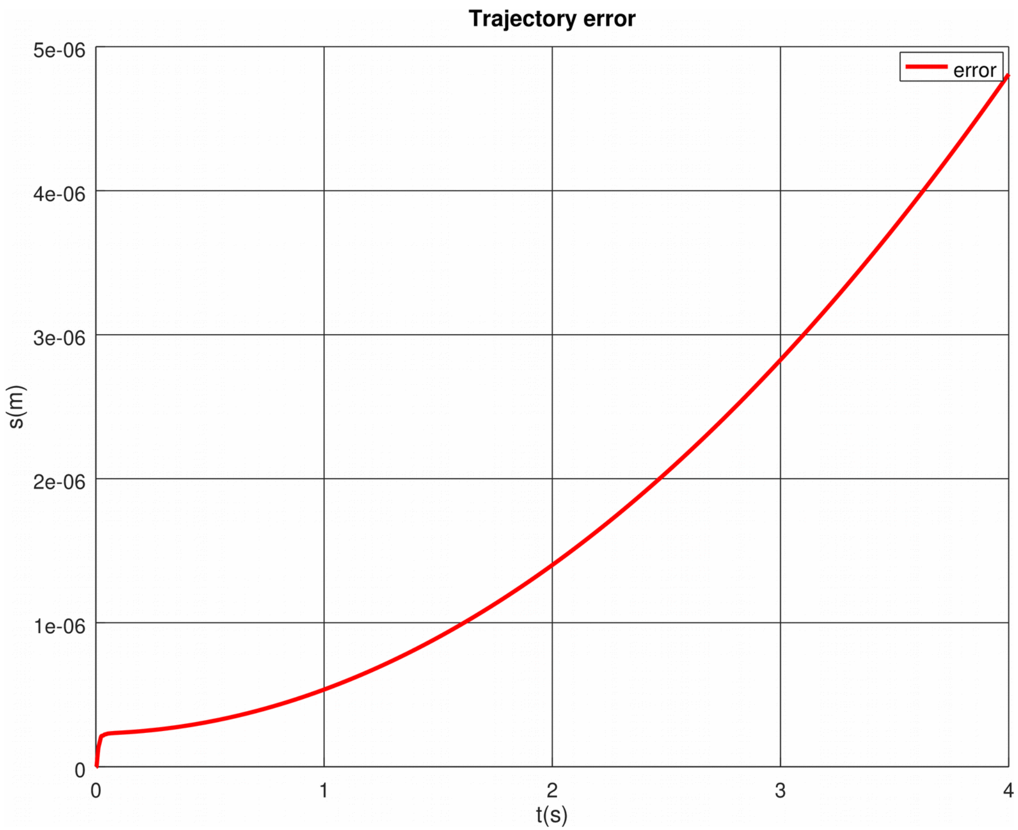


Triangle mesh model

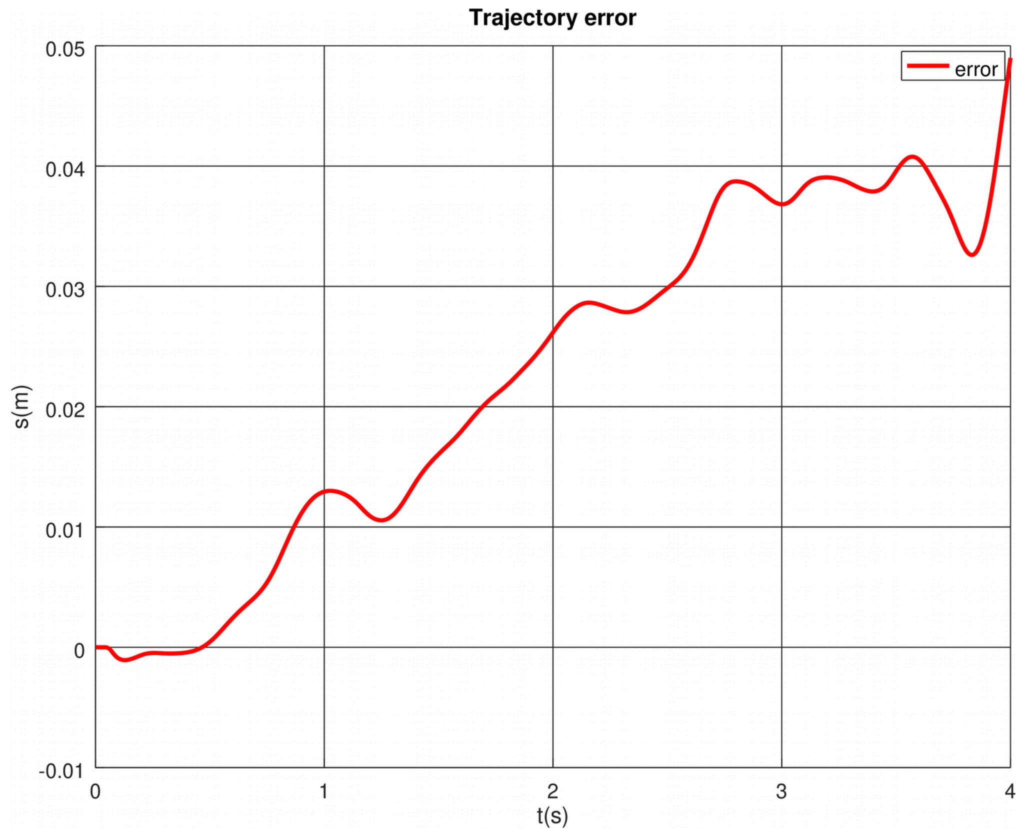


Inner Sphere Tree model

Test 1: trajectory profiles



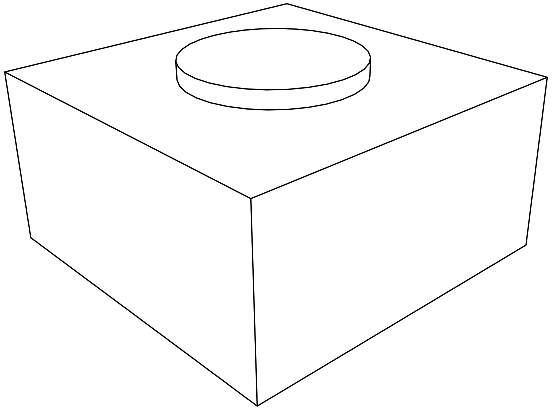
Triangle mesh model



Inner Sphere Tree model

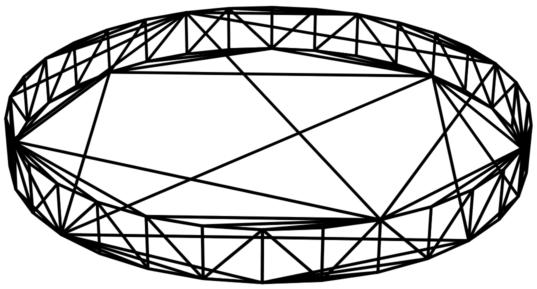
Test 2: disk rotating on a plane

- Designed to validate the spinning friction model
- m = 1 kg
- R = 0.25 m H = 0.05 m
- $\omega_0 = 5\pi$ rad/s
- Trajectory, velocity, angular velocity and angular deviation
- Theoretical stop at **t = 0.5 s**



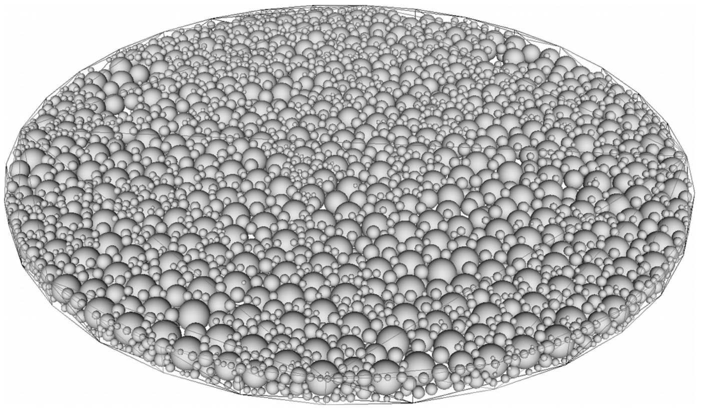
Test 2: mesh model

- Friction not constant: depends on the angular velocity
- Logarithmic decrease instead of linear
- Braking response smooth but braking time inaccurate (1s)
- Trajectory and angular error very low ($\sim 1 \times 10^{-4}$)

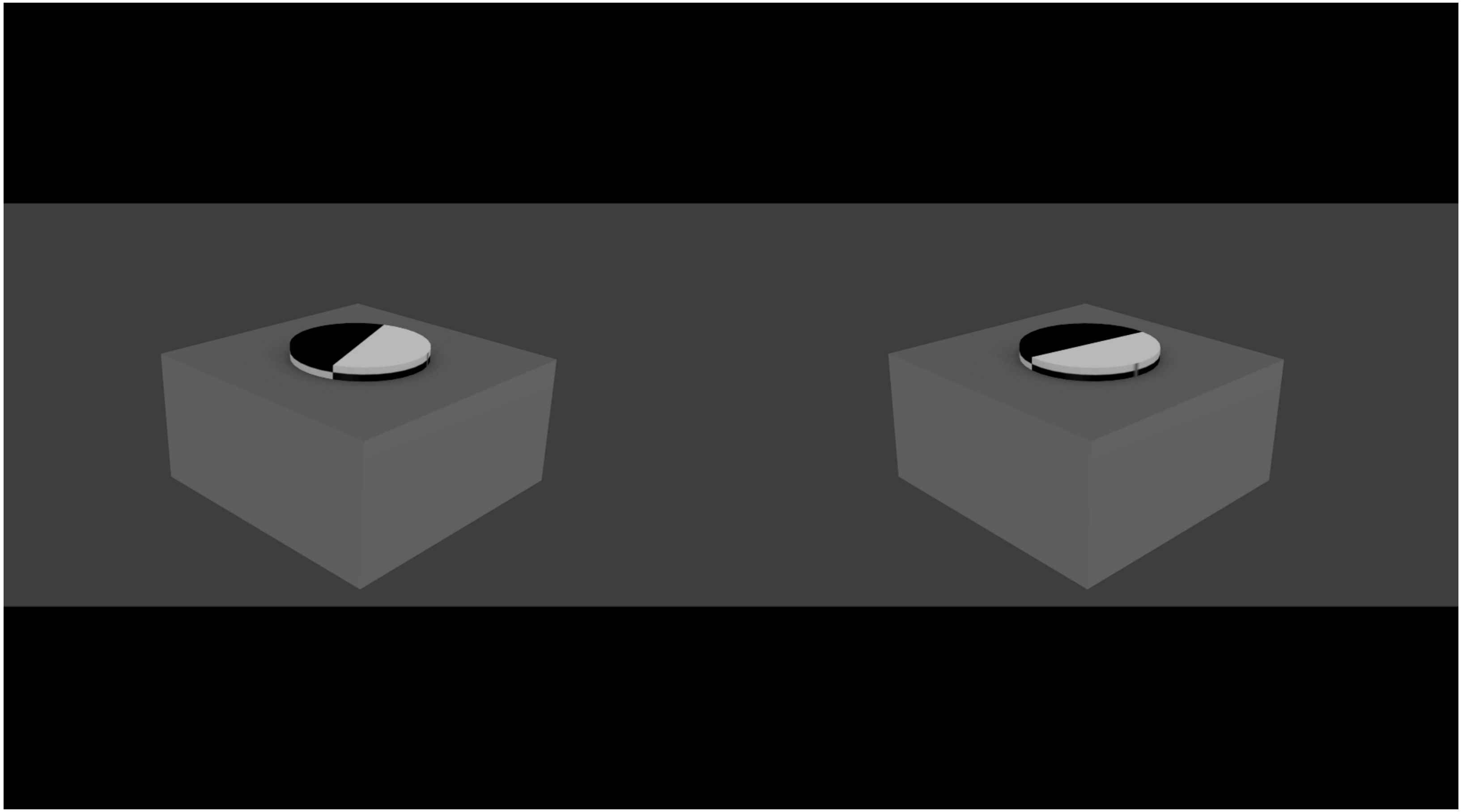


Results

- Test 2: Inner Sphere Tree model
- ~11k spheres (disk) and ~21k spheres (floor)
- Almost linear braking response
- Stop time close to theoretical one
- Higher trajectory, velocity and angular errors
- Rough response with vibrations



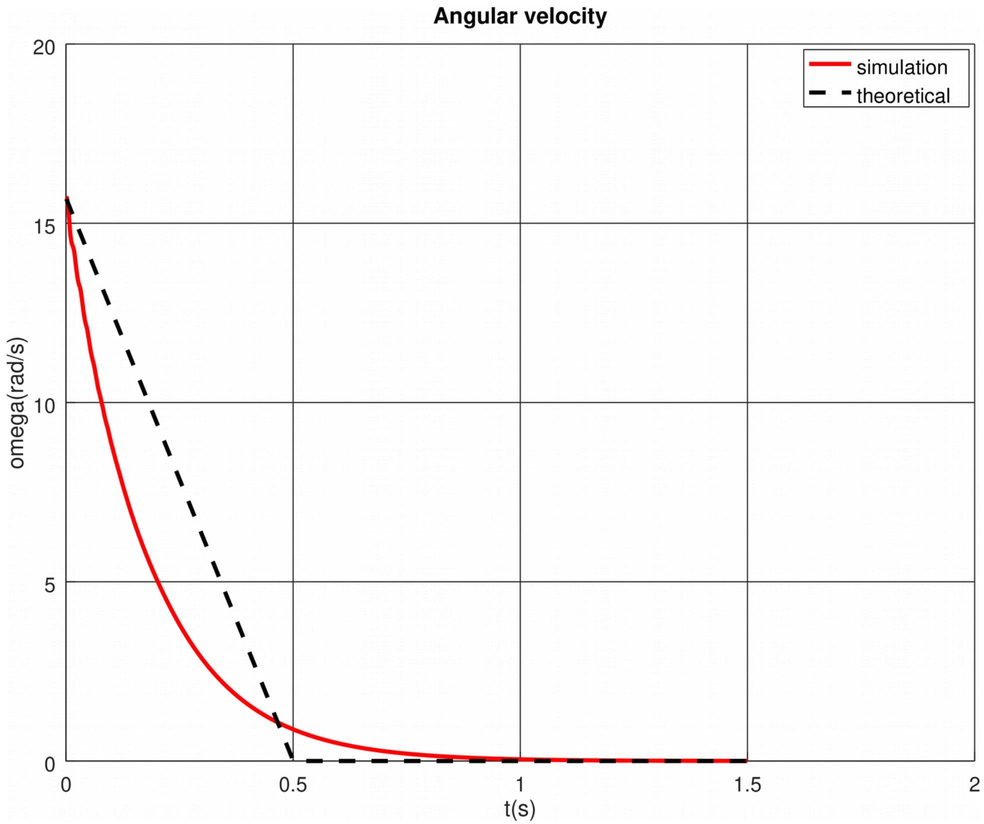
Test 2: video comparison



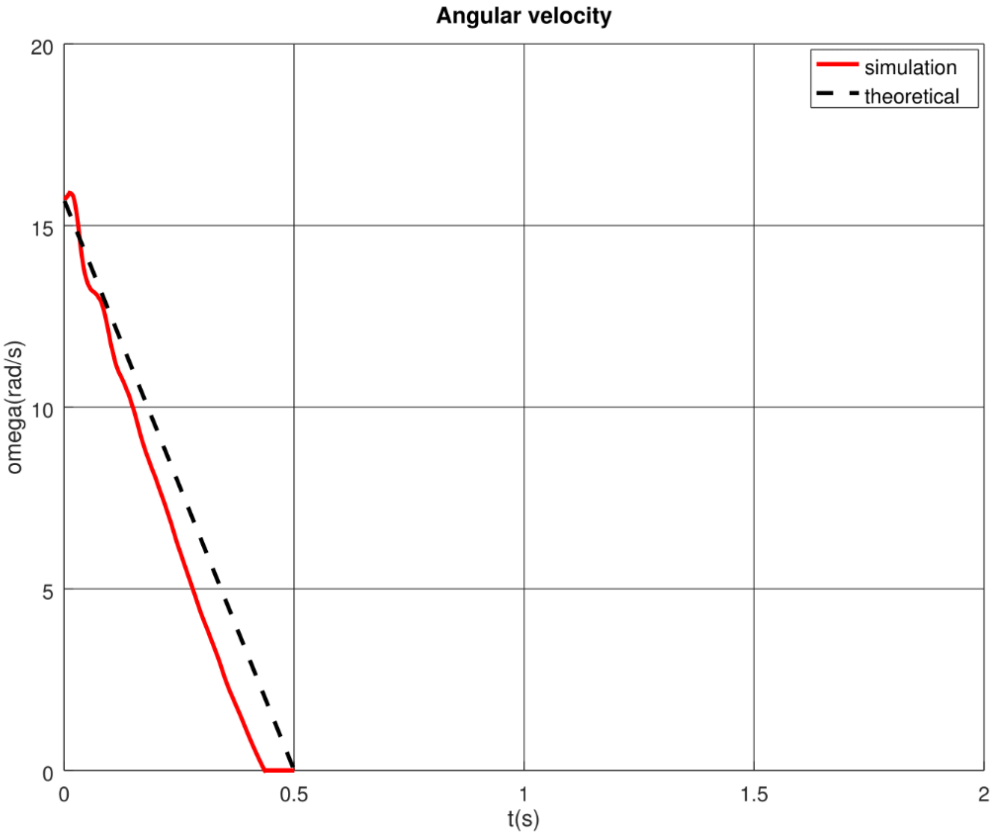
Triangle mesh model

Inner Sphere Tree model

Test 2: angular velocity profiles



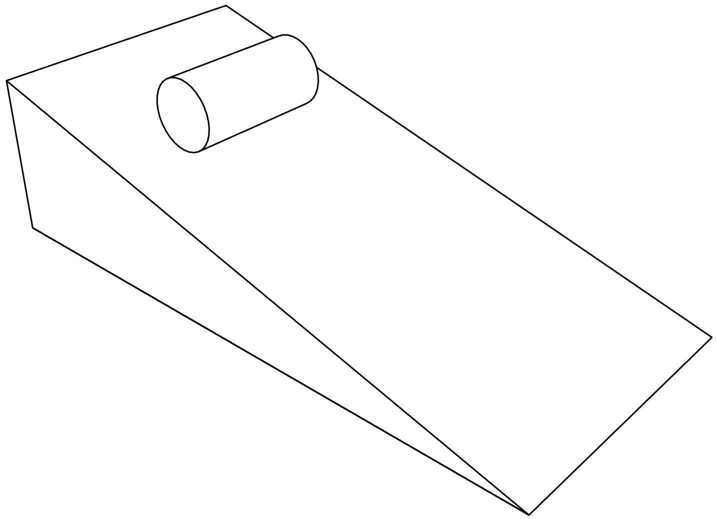
Triangle mesh model

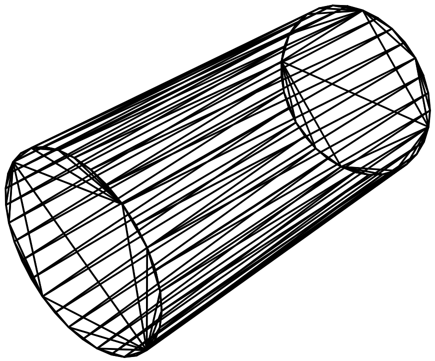


Inner Sphere Tree model

Test 3: cylinder rolling on a plane

- Designed to check the quality of the rolling resistance model
- m = 1 kg   R = 0.25 m   H = 1 m
- 15º inclination plane
- Trajectory, velocity and angular errors





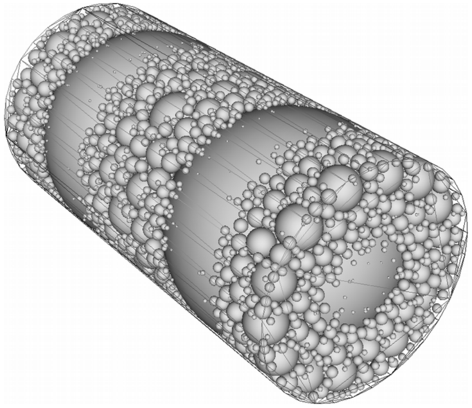
Test 3: mesh model

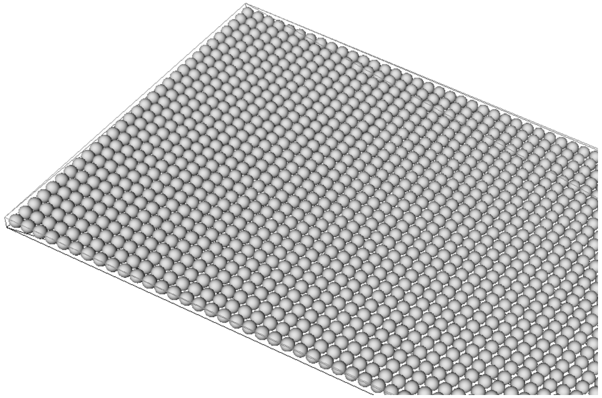
- Velocity profile matches closely the theoretical solution
- Minimal trajectory and angular errors
- Trajectory error shows quadratic evolution
- Great plane inclinations induce increasing frequency noises in the graphs

Results

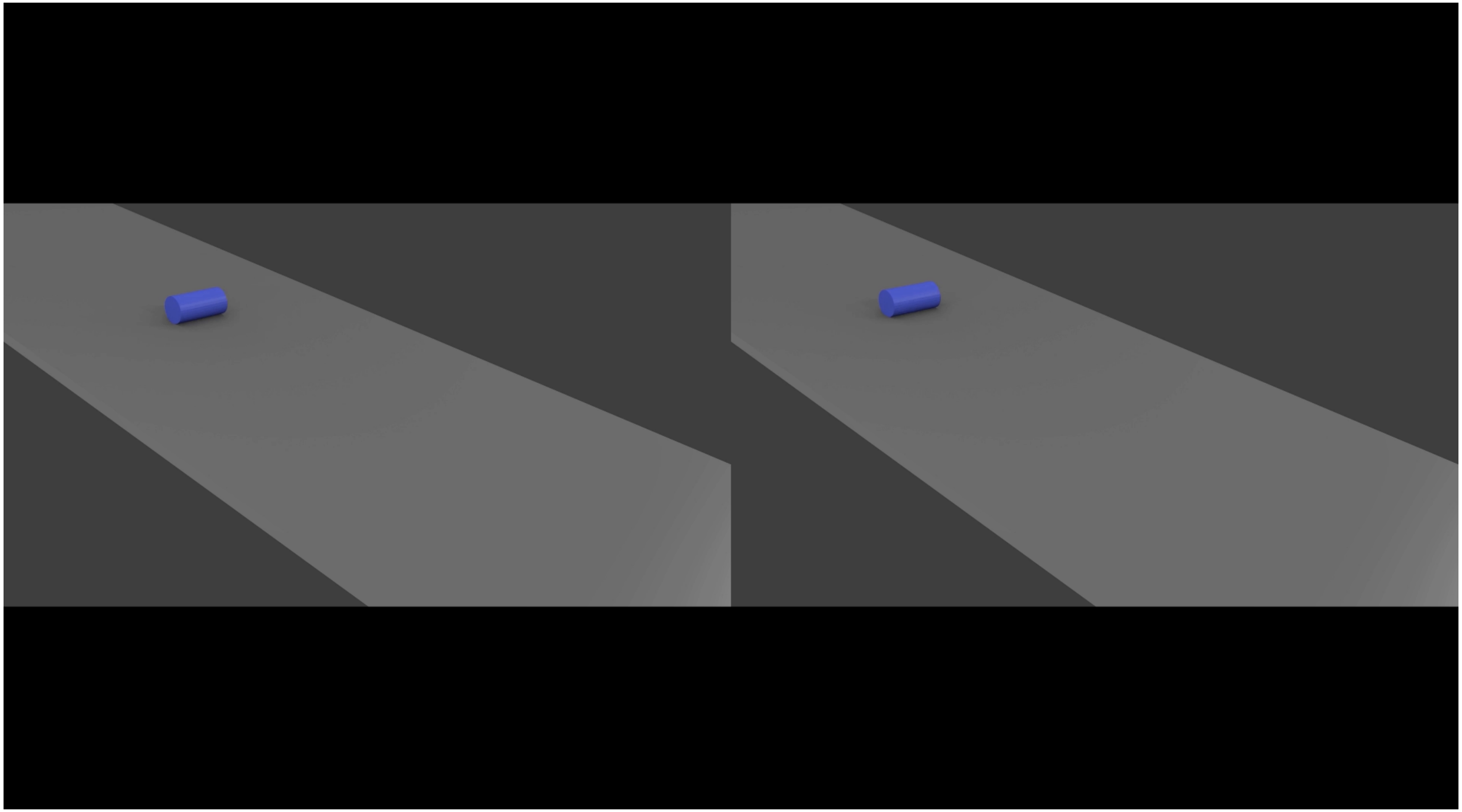
Test 3: Inner Sphere Tree model

- ~9k spheres (cylinder) and ~6k spheres (ground)
- Ground homogeneous sphere distribution was needed
- Velocity profile a little lower than theoretical
- Sphere collisions perceptible as “steps”
- Small trajectory and angular deviations





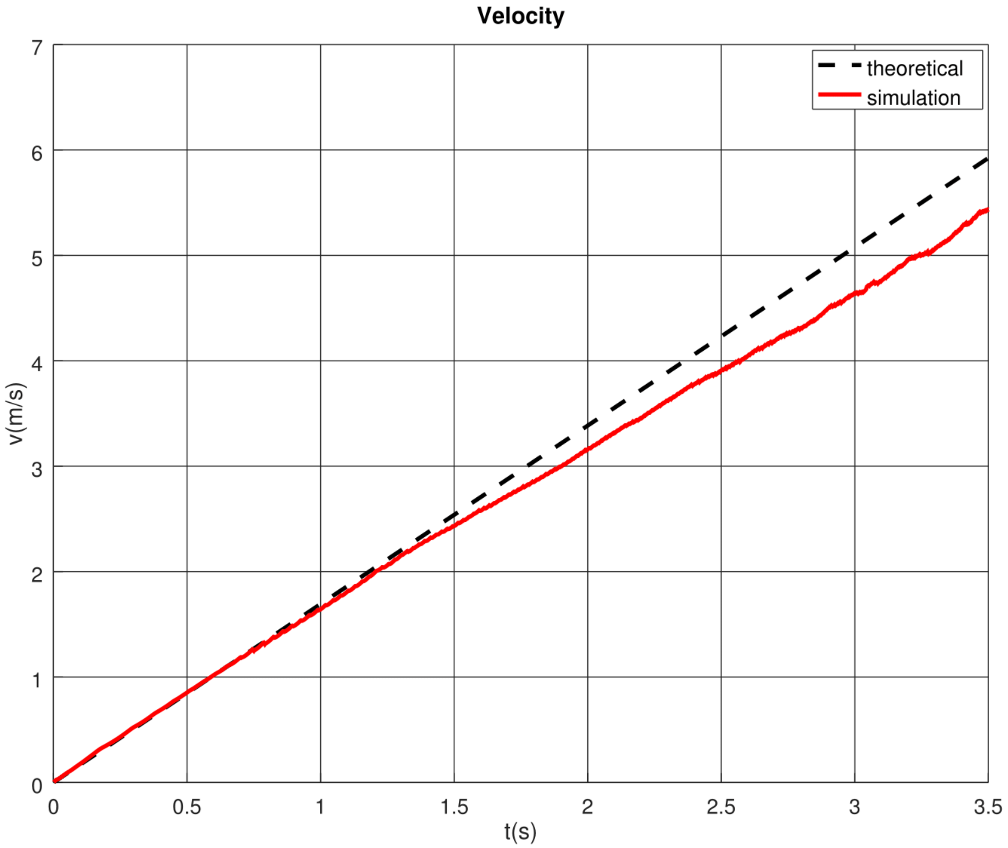
Test 3: video comparison



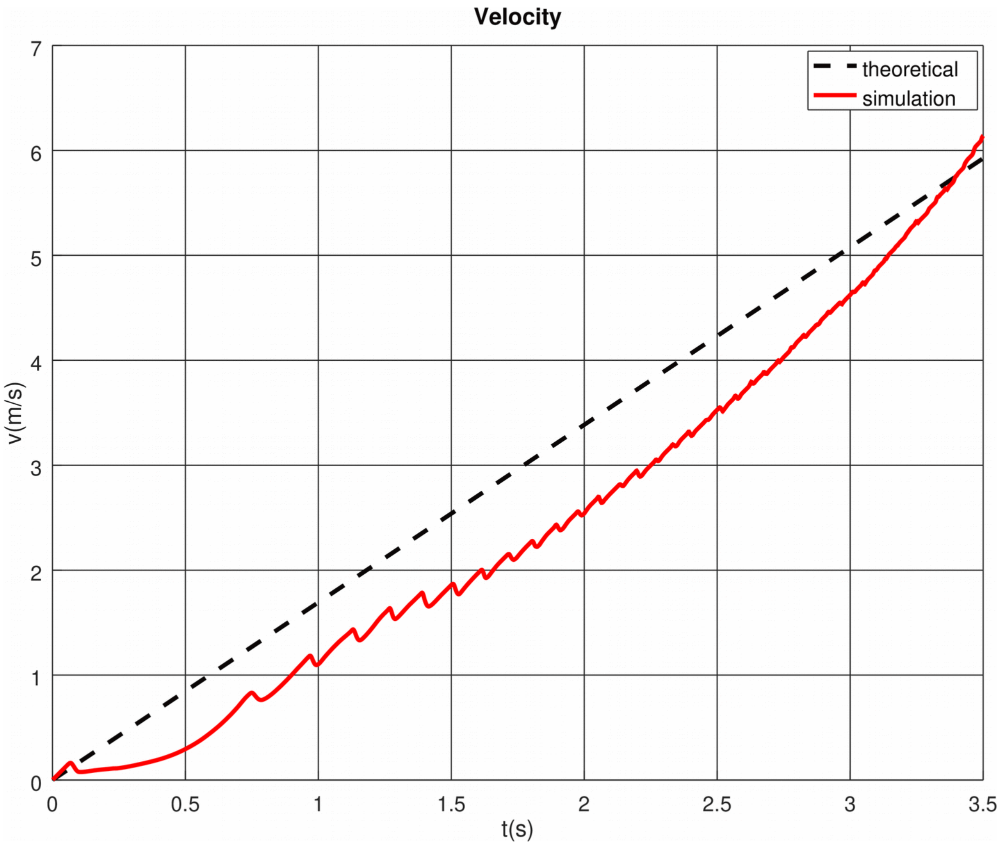
Triangle mesh model

Inner Sphere Tree model

Test 3: velocity profiles



Triangle mesh model



Inner Sphere Tree model

Performance comparison

- Graphic output disabled
- Full simulation including object loading, preprocess and output

Test	Model	Simulated time	Execution time	Real Time ratio
Sliding plane	Mesh	4	1.362	0.340
	Spheres	4	25.754	6.438
Rotating disk	Mesh	1.5	1.973	1.315
	Spheres	0.5	12.397	24.794
Rolling cylinder	Mesh	3.5	1.893	0.540
	Spheres	3.5	12.503	3.572

Conclusions and future work

Conclusions and future work

Contributions

- A Gonthier volumetric contact model was implemented to simulate conforming contacts in real-time. A collision detection library (LIMCODE) and data a export and visualization library (MBSDEBUG) were also created. A volumetric properties calculation algorithm was developed on top of CollDet library.

- Two different collision detection algorithms (meshes and spheres) were employed.
  - Triangle mesh model
  - Inner Sphere Tree model (developed at CGVR U. Bremen)

- Three different tests were designed to validate different aspects of the force models: sliding/sticking, spinning and rolling.

Conclusions

- Mesh model:
  - Real-time
  - Accurate
  - Less general
  - Restricted to planar contours
- IST model:
  - Not real-time
  - Less realistic
  - More general
  - Dependent on object dimensions ratio
- Mesh model ran ~20 times faster yielding more realistic simulations

Future work

- Collision generalization: more tests, multiple objects, complex contacts
- Optimizations: larger time-steps, calling collision detection once per time-step
- Parallelization
- Time critical IST collision detection
- Multiple contact stiction
- Arbitrary shape decomposition for non-planar contour collisions

Simulation of conforming contact in real-time multibody dynamics using a volumetric force model

Ferrol, December 10th 2018