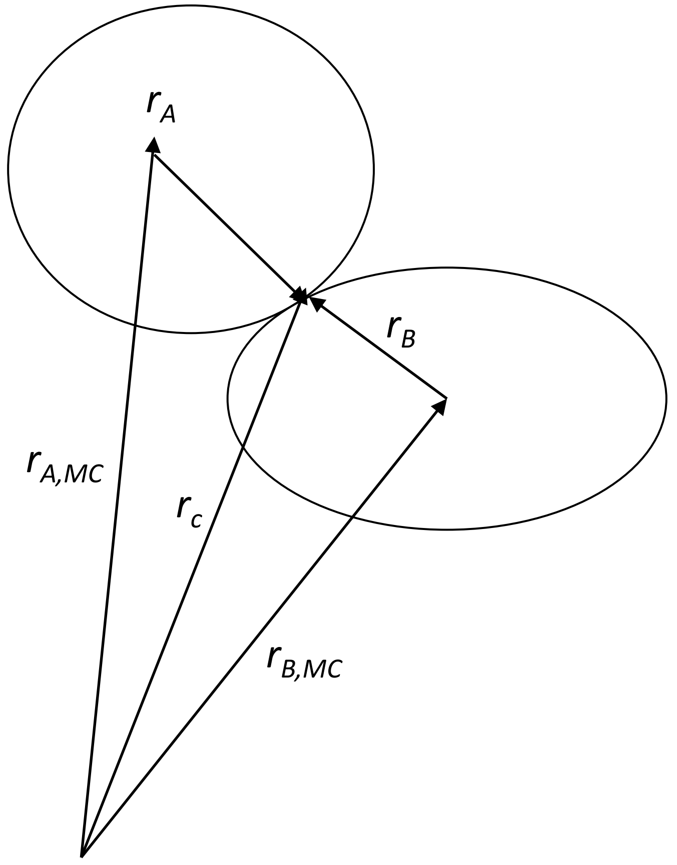
Visual Interactive Simulation

- Newton Coulomb impulses for rigid body contacts

- Outline
- The ”collision matrix”
- The Newton Coulomb impulse for rigid bodies
- N-C updates of translational and rotational velocities
- Multiple contact points
- Constraint stabilization – SPOOK

Visual Interactive Simulation, Spring 2010

- Two rigid bodies intersect, and thus have a contact point $\boldsymbol{r}_c$, a penetration depth, and a contact normal.



Newton Coulomb Impulses and the Collision Matrix

A (yet to be determined) collision impulse $\boldsymbol{J}$ will change the center of mass velocities as,

$$\boldsymbol{v}'_{A,B} = \boldsymbol{v}_{A,B} \;+/-\; \boldsymbol{J}/m_{A,B}$$

where m are the body masses.

It will also change the angular velocities,

$$\omega'_{A,B} = \omega_{A,B} +/- \boldsymbol{I}^{-1}{}_{A,B}(\boldsymbol{r}_{A,B} \times \boldsymbol{J})$$

where $\boldsymbol{I}$ are the body inertia tensors (in world coordinates).

The new velocities at the collision points thus are,

$$\begin{aligned}\boldsymbol{u}'_{A,B} &= \boldsymbol{u}_{A,B} +/- (\boldsymbol{J}/m_{A,B} + (\boldsymbol{I}^{-1}{}_{A,B}(\boldsymbol{r}_{A,B} \times \boldsymbol{J})) \times \boldsymbol{r}_{A,B}) \\ &= \boldsymbol{u}_{A,B} +/- (\boldsymbol{1}/m_{A,B} - \boldsymbol{r}_{A,B}{}^{\times} \boldsymbol{I}^{-1}{}_{A,B}\, \boldsymbol{r}_{A,B}{}^{\times})\, \boldsymbol{J}\end{aligned}$$

The collision matrix

Thus, the *change* in the relative collision velocity is,

$$\Delta \boldsymbol{u} = \boldsymbol{u}' - \boldsymbol{u} = (\boldsymbol{u}'_A - \boldsymbol{u}'_B) - (\boldsymbol{u}_B - \boldsymbol{u}_A)$$

$$\Delta \boldsymbol{u} = (\boldsymbol{1}/m_A + \boldsymbol{1}/m_B - (\boldsymbol{r}_A^{\times} \boldsymbol{I}^{-1}_A \boldsymbol{r}_A^{\times} + \boldsymbol{r}_B^{\times} \boldsymbol{I}^{-1}_B \boldsymbol{r}_B^{\times})) \boldsymbol{J}$$

which can be written as,

$$\Delta \boldsymbol{u} = \boldsymbol{K} \boldsymbol{J}$$

where ***K*** is called the *collision matrix*.

$$\boldsymbol{K} = (\boldsymbol{1}/m_A + \boldsymbol{1}/m_B - (\boldsymbol{r}_A^{\times} \boldsymbol{I}^{-1}_A \boldsymbol{r}_A^{\times} + \boldsymbol{r}_B^{\times} \boldsymbol{I}^{-1}_B \boldsymbol{r}_B^{\times}))$$

The collision matrix can be interpreted as the effective inverse inertia that the collision impulse acts on. This inertia is a combination of the inertia from changing the translational velocity and the angular velocity, respectively.

Newton Coulomb impulses for rigid bodies

First we assume sticking, with zero tangential velocity after the impact, that is,

$$\mathbf{u}'_t = 0 \qquad \mathbf{u}'_n = -e\,\mathbf{u}_n$$

Given that $\Delta\mathbf{u} = \mathbf{u}' - \mathbf{u}$, we can write the impulse as,

$$\mathbf{J} = \mathbf{K}^{-1}(\mathbf{u}' - \mathbf{u})$$

We can compute normal and tangential components of $\mathbf{J}$

$$\mathbf{J}_n = (\mathbf{J}\cdot\mathbf{n})\,\mathbf{n} \qquad \mathbf{J}_t = \mathbf{J} - \mathbf{J}_n$$

The tangential friction is reactive and depends on the normal force. Sticking requires that,

$$|\mathbf{J}_t| \le \mu|\mathbf{J}_n|$$

Otherwise we must consider sliding friction.
(identical with the ”max(...)” clamping law in previous lectures for the simple particle case – look it up and compare!).

Newton Coulomb impulses for rigid bodies

For sliding, we first determine the direction of sliding,

$\boldsymbol{u}_n = (\boldsymbol{u} \cdot \boldsymbol{n})\,\boldsymbol{n}$ $\qquad$ $\boldsymbol{u}_t = \boldsymbol{u} - \boldsymbol{u}_n$

Now we can compute the sliding unit vector,

$\boldsymbol{t} = \boldsymbol{u}_t \,/\, |\boldsymbol{u}_t|$

Next, we define the sliding impulse to be,

$\boldsymbol{J} = j\,\boldsymbol{n} - \mu j\,\boldsymbol{t}$ $\qquad$ (i.e. $\boldsymbol{J}_n = j\,\boldsymbol{n}$)

$0 \le \mu \le 1$ is the friction coefficient.

Newton Coulomb impulses for rigid bodies

Thus, we know

- an expression for the impulse
- how this impulse changes the collision velocity
- how the normal velocity changes due to Newton's impact law

Take the dot product of $\Delta \boldsymbol{u} = \boldsymbol{u}' - \boldsymbol{u} = \boldsymbol{K}\,\boldsymbol{J}$ with the collision normal $\boldsymbol{n}$,

$$\boldsymbol{u}'_n = \boldsymbol{u}_n + \boldsymbol{n}^T \boldsymbol{K}\,\boldsymbol{J}$$

Use Newton's impact law $\boldsymbol{u}'_n = -e\,\boldsymbol{u}_n$

$$-\,e\,\boldsymbol{u}_n = \boldsymbol{u}_n + \boldsymbol{n}^T \boldsymbol{K}\,\boldsymbol{J}$$

Newton Coulomb impulses for rigid bodies

Rearrange and insert the expression for $\boldsymbol{J}$,

$$-(1+e)\boldsymbol{u}_n = \boldsymbol{n}^T \boldsymbol{K}(j\boldsymbol{n} - \mu\, j\, \boldsymbol{t})$$

Finally we solve for *j*, to compute the unknown impulse.

$$-(1+e)\boldsymbol{u}_n = \boldsymbol{n}^T \boldsymbol{K}(\boldsymbol{n} - \mu\, \boldsymbol{t})\, j$$

$$\boxed{j = -(1+e)\boldsymbol{u}_n / \boldsymbol{n}^T \boldsymbol{K}(\boldsymbol{n} - \mu\, \boldsymbol{t})}$$

Knowing *j*, we can *compute the impulse*, and thus the *new post-collision translational and rotational velocities of body A and B*, respectively!

Review – NC Rigid Body Collisions

So far we have...

- (Broad phase collision detection)
- Narrow phase collision detection
  - Contact pairs, points, penetrations, normals
- Compute collision velocity
- Check if colliding, resting or separating
- If colliding, compute the NC impulse
- Update translational (CoM) and rotational velocity

We're still missing....

... add external forces, handle resting contacts, time integrate, render, handle multiple contacts, ...

Iterative pairwise solver

Basic idea:

- During each timestep loop through the collision set and solve collisions pair-wise, and ”hope” that for each pass/iteration, there will be fewer and/or smaller collisions, until they are eventually all resolved... (i.e. all colliding velocities are turned into separating velocities).

Many tricks (and problems):

- How to choose the order of when you loop through the collision set?! (e.g. Gauss-Seidel vs Gauss-Southwell vs Jacobi vs SOR...)
- How to avoid cycles (i.e. non-convergence)?!
- What if the solution doesn’t exist?
- In general good convergence for the first ~10 iterations, while high precision solves have 1/N convergence and require hundreds or thousands of iterations (ineffecient)

- Note: The pairwise iterative solver indeed corresponds to solving a system of equations with constraints, i.e. a matrix equation. This is not at all obvious in the current formalism we’re using, but will be in forthcoming lectures.

Contact stabilization

Collision impulses:

- do not guarantee constraint satisfaction
- are entirely reactive, so they do not stabilize contacts when velocities are low, i.e for resting contacts!

Contact stabilization must be handled in some other way, typically so that it produces stable resting contacts and compensates for:

- overlaps
- colliding velocities (or too small separating velocities)
- external forces

Basic idea:

Find contact impulses (Lagrange multipliers) that take the system to a non-violating state over discrete time. These contact impulses should compensate constraint overlap, constraint velocity and external forces.

Remember: If we formulate this so that the penetration should be exactly resolved at the end of the timestep, the velocity will overshoot, and we get an unstable simulation. So, this has to be systematically designed for stability in the discrete mechanical stepper.

Contact stabilization – SPOOK scheme

- Attach a damped spring at the contact point
- 1D case with a single contact:

$$\left(\frac{1}{m_i}+\frac{1}{m_j}\right)\lambda = -\frac{a}{h}q_{ij} - bv_{ij} - h\left(\frac{f_i}{m_i}-\frac{f_j}{m_j}\right)$$

$$v_{i,j}^{'} = v_{i,j} \pm \frac{\lambda}{m_{i,j}}$$

where *h* is the timestep, *q* the overlap (negative), *v* the constraint velocity - the relative normal velocity at the contact point.

The forces *f* on the two bodies, contain all external forces (in particular gravity).

Note that this 1D example applies to a debug case in your lab project, with a single sphere on a plane!

If we have several contacts, they will be coupled and we get several coupled equations.

In an iterative approach, λ is initially zero for all contacts, but takes a value according to the above.

$\tilde{\epsilon}$

Contact stabilization – SPOOK scheme

- Much faster convergence can be achieved with a certain relaxation of λ, which can be interpreted as a *regularized mass,*

$$\left(\frac{1}{m_i} + \frac{1}{m_j} + \tilde{\epsilon}\right)\lambda = -\frac{a}{h} q_{ij} - b v_{ij} - h\left(\frac{f_i}{m_i} - \frac{f_j}{m_j}\right)$$

$$v_{i,j}^{'} = v_{i,j} \pm \frac{\lambda}{m_{i,j}}$$

- The regularization can potentially be reduced during the iterations, but typically it is choosen constant so that the computed λ becomes slightly smaller than with no regularization.
- The regularization allows small constraint overlaps, which is bad (but realistic), but it substantially reduces jitter due to over shooting, and it also improves convergence rate since it improves the condition number of the linear system. The regularization has physical origin and improves the model!
- How should the parameters *a,b,* ε be choosen?? Either call this ”trick of the trade” or derive parameters from a principle. We prefer the latter and will do this later.

Contact stabilization – SPOOK scheme

In SPOOK, the parameters are choosen so that the stabilization corresponds to a critically damped spring with a certain relaxation time.

$$\left(\frac{1}{m_i} + \frac{1}{m_j} + \tilde{\epsilon}\right)\lambda = -\frac{a}{h}\,q_{ij} - bv_{ij} - h\left(\frac{f_i}{m_i} - \frac{f_j}{m_j}\right)$$

$$v_{i,j}^{'} = v_{i,j} \pm \frac{\lambda}{m_{i,j}}$$

$$\tilde{\epsilon} = \frac{4/k}{h^2(1+4d)}$$

$$a = \frac{4/h}{1+4d}$$

$$b = \frac{4d}{1+4d}$$

where k is a spring constant, and d is the number of timesteps over which the stabilization should restore the constraint.

Contact stabilization – SPOOK scheme

The generalized expression for rigid bodies is:

$$(GM^{-1}G^{T} + \Sigma)\,\lambda = -a\,G\,q - b\,G\,W - h\,G\,M^{-1}f$$

$$v_{i,j}^{'} = v_{i,j} \pm \frac{\lambda}{m_{i,j}}$$

$$\tilde{\epsilon} = \frac{4/k}{h^2(1+4d)}$$

$$a = \frac{4/h}{1+4d}$$

$$b = \frac{4d}{1+4d}$$

where k is a spring constant, and d is the number of timesteps over which the stabilization should restore the constraint.

Contact stabilization – SPOOK scheme - iterations

Extending this pairwise model to a Gauss-Seidel iterative model for multiple contacts is straightforward, since Gauss-Seidel iterations are about solving one equation at a time!

When solving for l we might run into negative values (attractive impulses). These are not allowed. We just clamp l to zero in these cases. This corresponds to a ”projected Gauss-Seidel method for solving a mixed linear complementarity problem”.