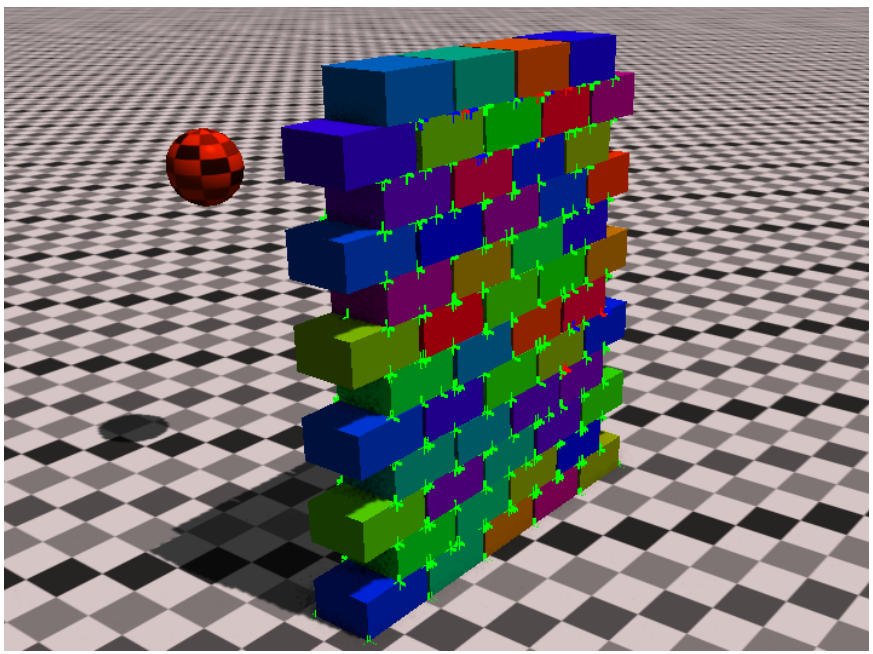
Sequential Impulse Solver for Rigid Body Dynamics

Intermediate graduation project presentation

Marijn Tamis



April 12, 2015

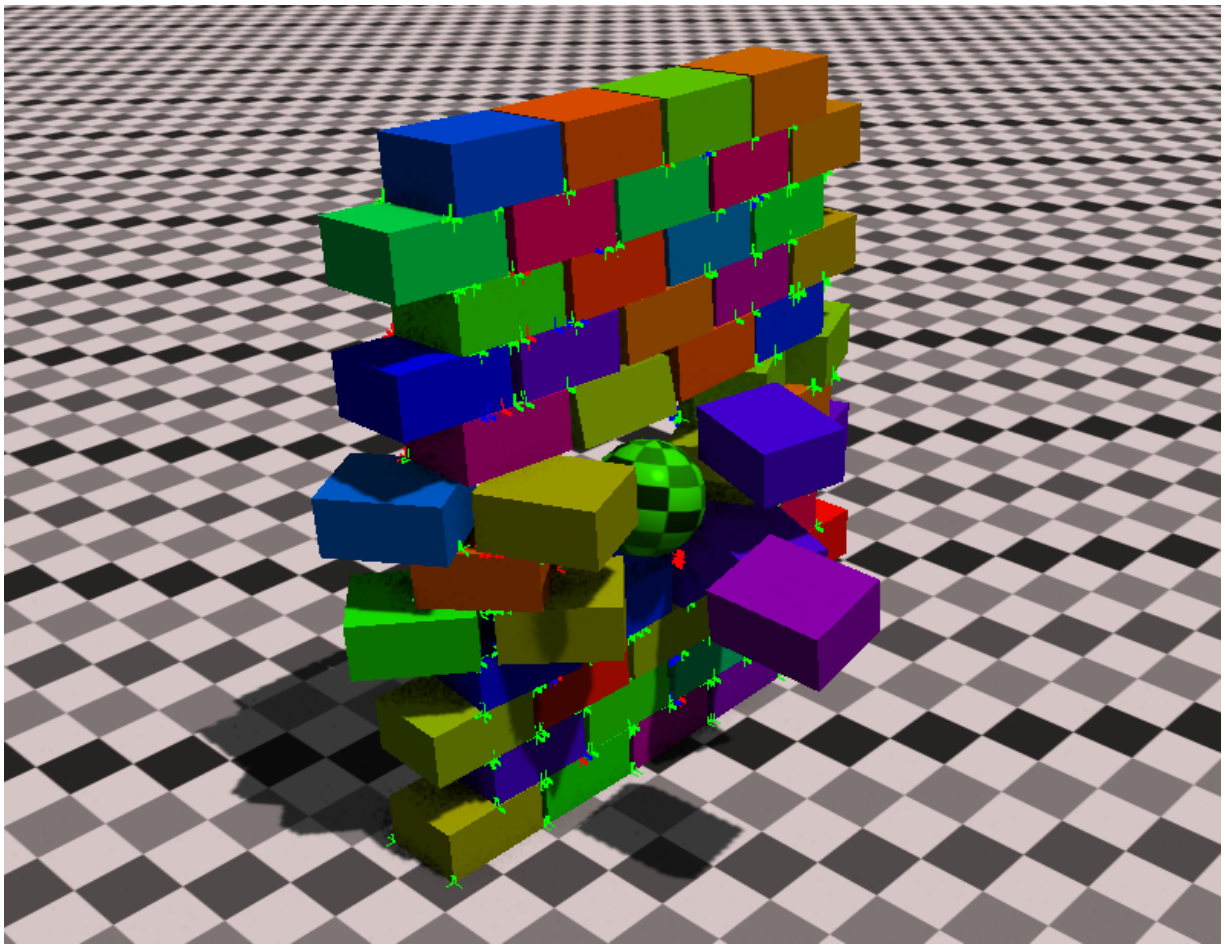
Table of contents

What I've been doing

Impulse solver

- Separate experimental impulse solver.
- Integrating specific test joints with the Crystal Engine.

Demonstration



Algorithm overview

What is a constraint solver?

- A constraint solver is a *black box* that calculates forces to *correct* the current state of the simulation.
- You provide it with contact points/constraints and get back their reaction forces.

Algorithm overview

What is a constraint solver?

- A constraint solver is a *black box* that calculates forces to *correct* the current state of the simulation.
- You provide it with contact points/constraints and get back their reaction forces.

What isn't a constraint solver?

- Collision detection.
- Contact point generation.

Other methods

Projected Gauss-Seidel solver

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathsf{T}}\boldsymbol{\lambda} = \frac{1}{\Delta t}\boldsymbol{\zeta} - \mathbf{J}\left(\frac{1}{\Delta t}\mathbf{v}_1 + \mathbf{M}^{-1}\mathbf{f}_{\text{ext}}\right)$$

Other methods

Projected Gauss-Seidel solver

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathsf{T}}\boldsymbol{\lambda} = \frac{1}{\Delta t}\boldsymbol{\zeta} - \mathbf{J}\left(\frac{1}{\Delta t}\mathbf{v}_1 + \mathbf{M}^{-1}\mathbf{f}_{\text{ext}}\right)$$

Eberly formula

$$\frac{-\left(\epsilon+1\right)\left(\overline{\mathbf{v}}_r\right)\mathbf{n}}{m_b^{-1} + m_a^{-1} + \left(\left(\mathbf{I}_b^{-1}\left(\mathbf{r}_b \times \mathbf{n}\right)\right) \times \mathbf{r}_b + \left(\mathbf{I}_a^{-1}\left(\mathbf{r}_a \times \mathbf{n}\right)\right) \times \mathbf{r}_a\right)\mathbf{n}} = p_c$$

Impulses

Different approach

- Only worry about a single contact at a time.

Impulses

Different approach

- Only worry about a single contact at a time.
- Supports stable stacking, friction, and constraints.

Impulses

Different approach

- Only worry about a single contact at a time.
- Supports stable stacking, friction, and constraints.
- Naive implementation is simple and fast.

Impulses

Different approach

- Only worry about a single contact at a time.
- Supports stable stacking, friction, and constraints.
- Naive implementation is simple and fast.
- "The math is almost understandable" - Erin Catto

Constraint function

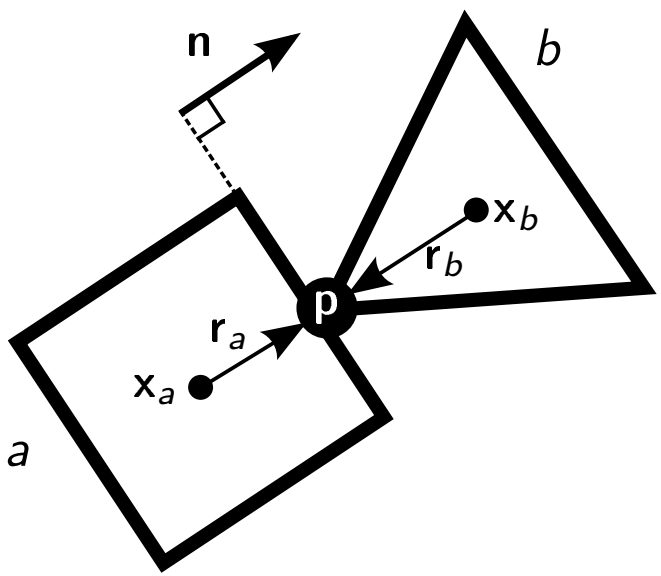
Position constraint

$$c\left(\mathbf{x}_a, \mathbf{q}_a, \mathbf{x}_b, \mathbf{q}_b\right) = 0$$

Constraint function

Example

$$c(\mathbf{x}_a, \mathbf{q}_a, \mathbf{x}_b, \mathbf{q}_b) = (\mathbf{p}_b - \mathbf{p}_a) \cdot \mathbf{n} = 0$$



Constraint function

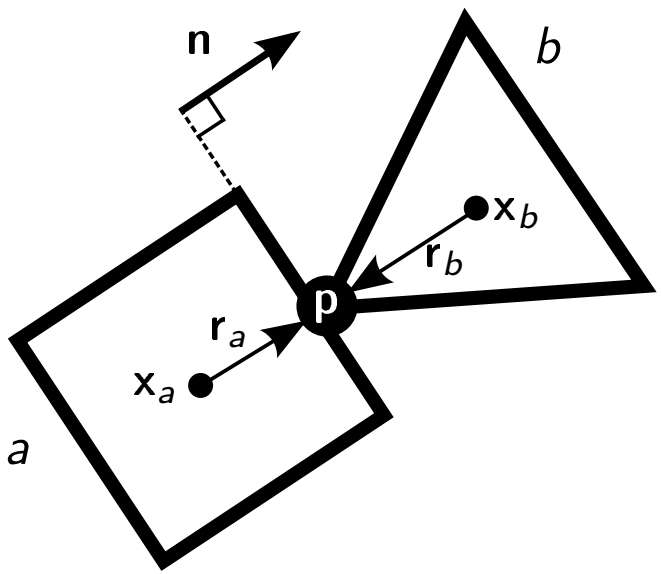
Velocity constraint

$$\frac{d}{dt}\,\mathrm{c}\,(\ldots) = \dot{\mathrm{c}}\,(\ldots) = 0$$

Constraint function

Example

$$\dot{c}(\ldots) = (\frac{d}{dt}\mathbf{p}_b - \frac{d}{dt}\mathbf{p}_a) \cdot \mathbf{n} = 0$$



Constraint function

Split of the velocity

$$\dot{c}(\ldots) = \mathbf{jv}$$

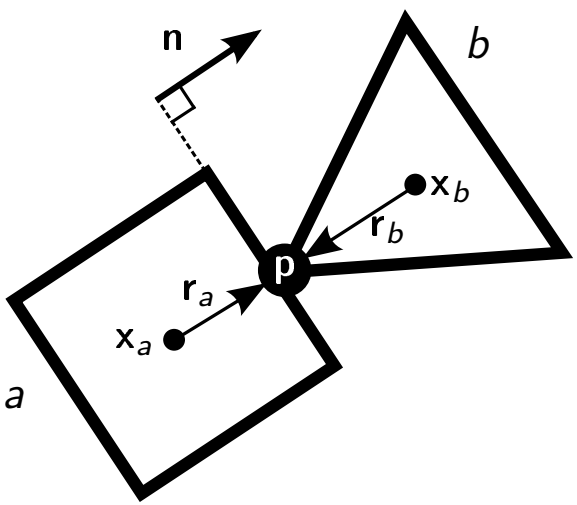
Where $\mathbf{j}$ contains the "constraint axes" and $\mathbf{v}$ body velocities.

Constraint function

Example

jv can be worked out from algebraic reordering of $\dot{c}(\ldots)$.

$$\mathbf{j} = \begin{bmatrix} -\mathbf{n} \\ -\mathbf{r}_a \times \mathbf{n} \\ \mathbf{n} \\ \mathbf{r}_a \times \mathbf{n} \end{bmatrix}^T \qquad \mathbf{v} = \begin{bmatrix} \mathbf{v}_a \\ \boldsymbol{\omega}_a \\ \mathbf{v}_b \\ \boldsymbol{\omega}_b \end{bmatrix}$$



Body variables

Mass

Combine the masses and inertia for both bodies:

$$\mathbf{M}^{-1} = \begin{bmatrix} [m_a^{-1}\mathbf{D}] & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_a^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & [m_b^{-1}\mathbf{D}] & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_b^{-1} \end{bmatrix}$$

Body variables

Mass

Combine the masses and inertia for both bodies:

$$\mathbf{M}^{-1}\mathbf{v} = \begin{bmatrix} [m_a^{-1}\mathbf{D}] & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_a^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & [m_b^{-1}\mathbf{D}] & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_b^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{v}_a \\ \boldsymbol{\omega}_a \\ \mathbf{v}_b \\ \boldsymbol{\omega}_b \end{bmatrix} = \begin{bmatrix} \mathbf{v}_a m_a \\ \boldsymbol{\omega}_a \mathbf{I}_a^{-1} \\ \mathbf{v}_b m_b \\ \boldsymbol{\omega}_b \mathbf{I}_b^{-1} \end{bmatrix}$$

Impulse

So we can use impulses:

$$\boldsymbol{\rho}_c = \begin{bmatrix} \boldsymbol{\rho}_a \\ \boldsymbol{\ell}_a \\ \boldsymbol{\rho}_b \\ \boldsymbol{\ell}_b \end{bmatrix} = \mathbf{M}^{-1}\Delta\mathbf{v}$$

Calculating the constraint impulse

Equations of motion

$$\mathbf{v}_2 = \mathbf{v}_1 + \mathbf{M}^{-1}\boldsymbol{\rho}_c = \mathbf{M}^{-1}\dot{\mathbf{v}}\Delta t \quad (1)$$

$$0 = \mathbf{j}\mathbf{v}_2 + b \quad (2)$$

$$\boldsymbol{\rho}_c = \mathbf{j}^{\mathsf{T}}\lambda\Delta t \quad (3)$$

Calculating the constraint impulse

Equations of motion

$$\mathbf{v}_2 = \mathbf{v}_1 + \mathbf{M}^{-1}\rho_c = \mathbf{M}^{-1}\dot{\mathbf{v}}\Delta t \quad (1)$$

$$0 = \mathbf{j}\mathbf{v}_2 + b \quad (2)$$

$$\rho_c = \mathbf{j}^{\mathsf{T}}\lambda\Delta t \quad (3)$$

Solving for λ

Substituting (1) into (2):

$$0 = \mathbf{j}\left(\mathbf{v}_1 + \mathbf{M}^{-1}\rho_c\right) + b$$

$$0 = \mathbf{j}\mathbf{v}_1 + \mathbf{j}\mathbf{M}^{-1}\rho_c + b$$

$$0 = \mathbf{j}\mathbf{v}_1 + \mathbf{j}\mathbf{M}^{-1}\rho_c + b$$

$$-\mathbf{j}\mathbf{M}^{-1}\rho_c = \mathbf{j}\mathbf{v}_1 + b$$

Calculating the constraint impulse

Equations of motion

$$\mathbf{v}_2 = \mathbf{v}_1 + \mathbf{M}^{-1}\boldsymbol{\rho}_c = \mathbf{M}^{-1}\dot{\mathbf{v}}\Delta t \quad (1)$$

$$0 = \mathbf{j}\mathbf{v}_2 + b \quad (2)$$

$$\boldsymbol{\rho}_c = \mathbf{j}^{\mathsf{T}}\lambda\Delta t \quad (3)$$

Solving for λ

Now we substitute (3):

$$-\mathbf{j}\mathbf{M}^{-1}\boldsymbol{\rho}_c = \mathbf{j}\mathbf{v}_1 + b$$

$$-\mathbf{j}\mathbf{M}^{-1}\mathbf{j}^{\mathsf{T}}\lambda\Delta t = \mathbf{j}\mathbf{v}_1 + b$$

Calculating the constraint impulse

Equations of motion

$$\mathbf{v}_2 = \mathbf{v}_1 + \mathbf{M}^{-1}\rho_c = \mathbf{M}^{-1}\dot{\mathbf{v}}\Delta t \quad (1)$$

$$0 = \mathbf{j}\mathbf{v}_2 + b \quad (2)$$

$$\rho_c = \mathbf{j}^{\mathsf{T}}\lambda\Delta t \quad (3)$$

Solving for λ

Solve for λ:

$$-\mathbf{j}\mathbf{M}^{-1}\mathbf{j}^{\mathsf{T}}\lambda\Delta t = \mathbf{j}\mathbf{v}_1 + b$$

$$\lambda = \frac{\mathbf{j}\mathbf{v}_1 + b}{-\mathbf{j}\mathbf{M}^{-1}\mathbf{j}^{\mathsf{T}}\Delta t}$$

Calculating the constraint impulse

Equations of motion

$$\mathbf{v}_2 = \mathbf{v}_1 + \mathbf{M}^{-1}\rho_c = \mathbf{M}^{-1}\dot{\mathbf{v}}\Delta t \quad (1)$$

$$0 = \mathbf{j}\mathbf{v}_2 + b \quad (2)$$

$$\rho_c = \mathbf{j}^{\mathsf{T}}\lambda\Delta t \quad (3)$$

Solving for λ

And use (3) to get the impulse again:

$$\lambda = \frac{\mathbf{j}\mathbf{v}_1 + b}{-\mathbf{j}\mathbf{M}^{-1}\mathbf{j}^{\mathsf{T}}\Delta t}$$

$$\rho_c = \mathbf{j}^{\mathsf{T}}\left(\frac{\mathbf{j}\mathbf{v}_1 + b}{-\mathbf{j}\mathbf{M}^{-1}\mathbf{j}^{\mathsf{T}}}\right)$$

Calculating the constraint impulse

That's it! We've just derived the Eberly formula for $\epsilon = 0$!

Eberly formula

$$\frac{-(\epsilon+1)(\overline{\mathbf{v}}_r)\,\mathbf{n}}{m_b^{-1} + m_a^{-1} + \left(\left(\mathbf{I}_b^{-1}(\mathbf{r}_b \times \mathbf{n})\right) \times \mathbf{r}_b + \left(\mathbf{I}_a^{-1}(\mathbf{r}_a \times \mathbf{n})\right) \times \mathbf{r}_a\right)\mathbf{n}} = \rho_c$$

Global solution

Sequential impulses

- Now we sequentially calculate and apply ρ_c for the constraints.
- We also need to clamp the impulses for inequality constraints (eg. $c(\ldots) \geq 0$).

Global solution

Sequential impulses

- Now we sequentially calculate and apply ρ_c for the constraints.
- We also need to clamp the impulses for inequality constraints (eg. $c(\ldots) \geq 0$).
- Don't clamp the corrective impulse.
- Clamp the total accumulated impulse of the constraint.

The algorithm

foreach *Constraint* **do**
  // λ_{total} is tracked separately for each constraint
  Initialize λ_{total} to 0;
end
while *Not satisfied with global solution* **do**
  foreach *Constraint* **do**
    Calculate $\lambda_{\text{corrective}}$;
    $\lambda_{\text{old}} = \lambda_{\text{total}}$;
    $\lambda_{\text{total}} += \lambda_{\text{corrective}}$;
    Clamp λ_{total} to allowed interval;
    Update body velocities using the impulse calculated from the difference $\lambda_{\text{total}} - \lambda_{\text{old}}$;
  end
end

Things not mentioned

We don't have all day

- The full derivative of the position constraint.
- The b in the impulse equation. It can be used for error correction.

Other improvements

- Contact caching for stable stacking.
- Slop, and many other tricks to improve the simulation.

Resources

Good resources

- Erin Catto's Iterative Dynamics with Temporal Coherence
- Erin Catto's GDC presentations.
- Bullet physics forums.
- My specialization paper (about the PGS solver).

Questions

Questions?

www.mft-spirit.nl

mftspirit@gmail.com