# Building a physics engine - part 3: collision response

Dr. Giuseppe Maggiore

NHTV University of Applied Sciences
Breda, Netherlands

# Table of contents

1. Collision response
2. Constrained dynamics
3. Setting up the constraints
4. Equations of motion
5. Iterative solution of a system of equations
6. Contact caching
7. Assignment

# Collision response

## Collision response system

- Solving physical constraints in addition to the equations of motion
- Constraints are mostly contact constraints, but also
  - Friction constraints
  - Distance constraints
  - Joint angle constraints
  - ...
- The ideal collision response system deals with all of these

# Collision response

## Naïve take

- Apply the constraints to the objects in pairs
- Use the laws of *conservation of motion* for each collision; $P_0$ is the point of collision, $x_A$ and $x_B$ are the centres of mass of the objects, $v^{-1}$ is the pre-impact velocity of the objects, $v^+$ is the post-impact velocity of the objects
  - $f = \frac{-(1+\epsilon)(N_0 \cdot (v_A^{-1} - v_B^{-1})) + (\omega_A^- \cdot (r_A \times N_0) - \omega_B^- \cdot (r_B \times N_0)))}{1/m_A + 1/m_B + (r_A \times N_0)^T J_A^{-1} (r_A \times N_0) + (r_B \times N_0)^T J_B^{-1} (r_B \times N_0)}$
  - $r_A = P_0 - x_A$, $r_B = P_0 - x_B$
  - $v_A^+ = v_A^- + \frac{fN_0}{m_A}$
  - $\omega_A^+ = \omega_A^- + J_A^-(r_A \times (fN_0))$
- Push away from interpenetration as long as interpenetration exists

# Collision response

## Naïve take

- Jitters a lot, and does not support stacking
- May be acceptable in very sparse scenarios (space/flight simulator)

# Collision response

## Naïve take number 2

- Apply the constraints to the objects in pairs
- Apply again during the same tick
- Average/combine the various impulses
- Push apart objects so they do not penetrate

# Collision response

## Naïve take number 2

- Apply the constraints to the objects in pairs
- Apply again during the same tick
- Average/combine the various impulses
- Push apart objects so they do not penetrate
- Constraints are still broken
- *Hack-y method*, gives no guarantees
- Still does not support stacking

# Constrained dynamics

## Unconstrained kinematics

- A rigid body is characterised by
- $\dot{x} = v$
- $\dot{q} = \frac{1}{2} wq$

# Constrained dynamics

## Unconstrained kinematics

- For a system of $N$ bodies, we can define the system derivative as

$$V = \begin{bmatrix} v_1 \\ \omega_1 \\ \vdots \\ v_N \\ \omega_N \end{bmatrix}$$

# Constrained dynamics

## Constraints

- Our system allows *pairwise* constraints between bodies
- The $k$-th constraint, between bodies $i$ and $j$, has the form $C_k(x_i, q_i, x_j, q_j) = 0$
- The vector $C$ holds all the constraints. $C = 0$, or $C(x) = 0$, is a function of the state vector, so by the chain rule $\dot{C} = JV = 0$

# Constrained dynamics

## Constraint forces

- Each constraint causes a reaction force $f_c$ and a reaction torque $\tau_c$
- The vector of all reaction forces is

$$F_c = \begin{bmatrix} f_{c1} \\ \tau_{c1} \\ \vdots \\ f_{cN} \\ \tau_{cN} \end{bmatrix}$$

# Constrained dynamics

## Constraint forces

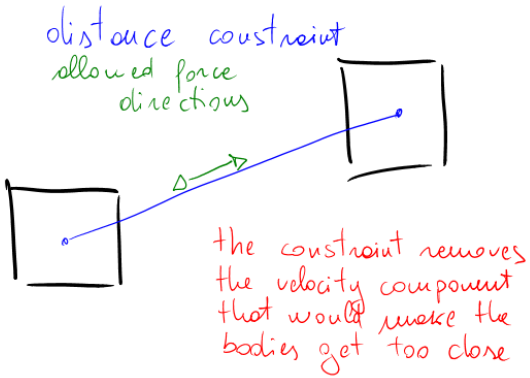
- We know that $\dot{C} = JV = 0$
- $JV = \begin{bmatrix} J_1 \cdot V \\ \vdots \\ J_M \cdot V \end{bmatrix} = 0$
- This means that $V$ is orthogonal to each row of $J$

# Constrained dynamics

## Constraint forces

- Constraint forces perform no work, so $F_c \cdot V = 0$

# Principle of virtual work



# Constrained dynamics

## Constraint forces

- We can use $F_c = J^T\lambda$ for some vector $\lambda$ of undetermined force multipliers
- $F_c \cdot V = J^T\lambda \cdot V = (\sum_i J_i\lambda_i) \cdot V = \sum_i J_i \cdot V\lambda_i = \sum_i 0\lambda_i = 0$

# Constrained dynamics

## Constraint forces

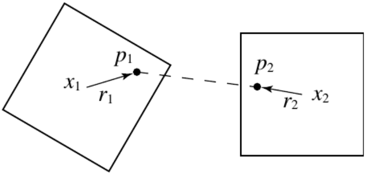
- We will thus compute the matrix $J$ of constraints from the collision system
- We then solve for $\lambda$, compute $F_c$, and finally obtain $V$

# Constrained dynamics

## Distance constraints

- The simplest constraint is a distance constraint
- Two points of two bodies must remain at a given distance
- $C(x_i, q_i, x_j, q_j) = \frac{1}{2}(|p_j - p_i|^2 - L^2) = 0$
- If we derive this, we get
  $\dot{C}(x_i, q_i, x_j, q_j) = \underbrace{(p_j - p_i)}_{d}(v_j + \omega_j \times r_j - v_i - \omega_i \times r_i)$
- We split this into a row for $J$ and a part of $V$:
- $\dot{C}(x_i, q_i, x_j, q_j) =$
  $\underbrace{\left[-d^T \;\; -(r_i \times d)^T \;\; d^T \;\; (r_j \times d)^T\right]}_{\text{a row of } J} \underbrace{[v_i \; \omega_i \; v_j \; \omega_j]^T}_{\text{some columns of } V}$

# Distance constraint



# Constrained dynamics

## Distance constraints

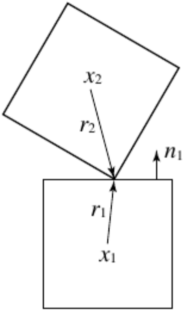
- We are abusing the notation; the "row of $J$" also contains many zeroes $(6 \times (N_{\text{bodies}} - 2))$
- The only columns that are not zeroed are those corresponding to the bodies $i$ and $j$
- $\dot{C}(x_i, q_i, x_j, q_j) =$
$$\underbrace{\begin{bmatrix} \dots & 0 & -d^T & -(r_i \times d)^T & 0 & \dots & 0 & d^T & (r_j \times d)^T & \dots & 0 \end{bmatrix}}_{\text{a row of } J} V$$

# Constrained dynamics

## Contact constraints

- We may also model contact constraints
- The contact constraint measures the object separation; it is negative in case of overlap
- $C(x_i, q_i, x_j, q_j) = (x_j + r_j - x_i - r_i) \cdot n_i = 0$
- $\dot{C}(x_i, q_i, x_j, q_j) =$ $(v_j + \omega_j \times r_j - v_i - \omega_i \times r_i) \cdot n_i + (x_j + r_j - x_i - r_i) \cdot \omega_i \times n_i$
- We assume that both penetration and angular velocity are small, so we ignore the second term
- $\dot{C}(x_i, q_i, x_j, q_j) \approx (v_j + \omega_j \times r_j - v_i - \omega_i \times r_i) \cdot n_i$

# Contact constraint



# Constrained dynamics

## Contact constraints

- We can now separate $\dot{C}$ into $J$ and $V$:
- $\dot{C}(x_i, q_i, x_j, q_j) = (v_j + \omega_j \times r_j - v_i - \omega_i \times r_i) \cdot n_i = \left[-n_i^T \;\; -(r_i \times n_i)^T \;\; n_i^T \;\; (r_j \times n_i)^T\;\right] [v_i \; \omega_i \; v_j \; \omega_j]^T$

# Constrained dynamics

## Contact constraints

- Notice that the force between bodies in contact can push them apart, but not pull them together
- This means that $0 \leq \lambda_k \leq +\infty$, where $k$ is the constraint index for a contact constraint

# Constrained dynamics

## Contact constraints

- In some cases penetration might happen anyway
- Numerical errors or issues with discrete steps
- We allow the velocity to be augmented with a *pushing factor* which is proportional to the penetration
- This means that for contact constraints $J_i V = -\beta C_i$, for $\beta \leq \frac{1}{\Delta t}$

# Constrained dynamics

## Friction constraints

- Friction constraints are very similar to contact constraints
- Friction happens along the tangent plane, so we have two constraints (one for $u_i = T$ and one for $u_j = B$)
  - $\dot{C}_{u_i}(x_i, q_i, x_j, q_j) = (v_j + \omega_j \times r_j - v_i - \omega_i \times r_i) \cdot u_i = \left[-u_i^T \ \ -(r_i \times u_i)^T \ \ u_i^T \ \ (r_j \times u_i)^T\ \right] [v_i \ \omega_i \ v_j \ \omega_j]^T$
  - $\dot{C}_{u_j}(x_i, q_i, x_j, q_j) = (v_j + \omega_j \times r_j - v_i - \omega_i \times r_i) \cdot u_j = \left[-u_j^T \ \ -(r_i \times u_j)^T \ \ u_j^T \ \ (r_j \times u_j)^T\ \right] [v_i \ \omega_i \ v_j \ \omega_j]^T$

# Constrained dynamics

## Friction constraints

- We must also bound the friction value (this is an approximation) to take the friction coefficient into account
- $-\mu m_c g \leq \lambda_{u_1} \leq \mu m_c g$ and $-\mu m_c g \leq \lambda_{u_2} \leq \mu m_c g$, where $m_c$ is the mass assigned to the contact point

# Equations of motion

## Equations of motion

- We now integrate our constraint system with the equations of motion
- We know the *Newton* – *Euler* equations of motion are

$$\begin{aligned} m\dot{v} &= F &= f_c + f_{ext} \\ I\dot{w} &= \tau &= \tau_c + \tau_{ext} \end{aligned}$$

# Equations of motion

## Equations of motion

- We can define a single, big matrix for all the bodies

$$M = \begin{pmatrix} m_1 E_{3\times 3} & 0 & \dots & 0 & 0 \\ 0 & I_1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & m_n E_{3\times 3} & 0 \\ 0 & \dots & 0 & 0 & I_n \end{pmatrix}$$

- $E_{3\times 3}$ is just the identity matrix

# Equations of motion

## Equations of motion

- We can easily invert this matrix

$$M^{-1} = \begin{pmatrix} (m_1 E_{3\times 3})^{-1} & 0 & \dots & 0 & 0 \\ 0 & I_1^{-1} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & (m_n E_{3\times 3})^{-1} & 0 \\ 0 & \dots & 0 & 0 & I_n^{-1} \end{pmatrix}$$

# Equations of motion

## Equations of motion

- We can define a single, big vector for all the external forces

$$F_{\text{ext}} = \begin{bmatrix} f_{\text{ext}1} \\ \tau_{\text{ext}1} \\ \vdots \\ f_{\text{ext}N} \\ \tau_{\text{ext}N} \end{bmatrix}$$

# Equations of motion

## Equations of motion

- Since we know that $F_C = J^T\lambda$, we can rewrite the equations of motion for $n$ bodies as $\begin{cases} M\dot{V} & = & J^T\lambda + F_{\text{ext}} \\ JV & = & \epsilon \end{cases}$
- $\epsilon$ is the vector of force offsets which allows contact forces to perform work
- We have too many unknowns: $V$, $\dot{V}$, and $\lambda$

# Equations of motion

## Equations of motion

- We approximate $\dot{V} \approx \frac{V_2 - V_1}{\Delta t}$
- We replace $\dot{V}$ $\begin{cases} M\frac{V_2 - V_1}{\Delta t} & = & J^T\lambda + F_{\text{ext}} \\ JV_2 & = & \epsilon \end{cases}$
- We solve for $V_2$ $\begin{cases} V_2 & = & \Delta t M^{-1}(J^T\lambda + F_{\text{ext}}) + V_1 \\ V_2 & = & J^T\epsilon \end{cases}$

# Equations of motion

## Equations of motion

- We can now finish solving for $\lambda$
- $J^T\epsilon = \Delta t M^{-1}(J^T\lambda + F_{\text{ext}}) + V_1$
- $J^T\epsilon - V_1 - \Delta t M^{-1}F_{\text{ext}} = \Delta t M^{-1}(J^T\lambda)$
- $\frac{\epsilon}{\Delta t} - JV_1 - \Delta t J M^{-1}F_{\text{ext}} = JM^{-1}J^T\lambda$

# Equations of motion

## Equations of motion

- The equation $\frac{\epsilon}{\Delta t} - JV_1 - \Delta t J M^{-1} F_{\text{ext}} = J M^{-1} J^T \lambda$ admits infinite solutions; this is due to redundant constraints, such as a table with more than three legs
- The force combinations that solve the system are usually infinite

# Equations of motion

## Equations of motion

- Once $\lambda$ is computed, we can determine $F_c$, $F$, and then $V_2$
- A regular integration step is then performed with the new velocities $V_2$

# Iterative solution

## System of equations

- The equation $\frac{\epsilon}{\Delta t} - JV_1 - \Delta t J M^{-1} F_{\text{ext}} = J M^{-1} J^T \lambda$ can be restated in simpler form
- $\underbrace{\frac{\epsilon}{\Delta t} - JV_1 - \Delta t J M^{-1} F_{\text{ext}}}_{b} = \underbrace{J M^{-1} J^T}_{A} \underbrace{\lambda}_{x}$
- $Ax = b$ for some $A, b$
- These systems can be solved iteratively with a method such as Projected Gauss-Seidel (PGS)

# (Projected) Gauss-Seidel

```
while not converged
  Δx_i = (b_i − Σ_j a_ij x_j) / A_ii
  x_i = clamp(x_i + Δx_i, min_i, max_i)
```

$$\Delta x_i = \left(b_i - \sum_j a_{ij} x_j\right) / A_{ii}$$
$$x_i = \mathtt{clamp}(x_i + \Delta x_i, min_i, max_i)$$

# Iterative solution

## Sparse matrices

- Remember that $A$ is going to be very sparse
- You may optimize the summation $\Delta x_i = (b_i - \sum_j a_{ij} x_j)/A_{ii}$ a lot by ignoring the zero entries of $A$

# Contact caching

## Contact caching

- PGS is faster the closer the initial $x$ vector is to the solution
- If we store previous contact points and their $\lambda_i$ values, PGS converges sooner
- Just be aware of this
  - Also be aware that it is rather hard to build in practice
  - If you attempt it, chances of success may be low

# Assignment

## Assignment

- Before the end of next week
- Group-work archive/video on Natschool or uploaded somewhere else and linked in your report
- Individual report by each of you on Natschool
- Build a collision response system that supports collisions between multiple objects

# That's it

Thank you!