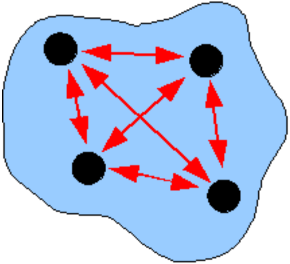
From scene graph to equations

François Faure[1]

[1] Grenoble Universities
INRIA - Evasion

September 2007

A physical body



State vectors: x, v, f, a, aux, ...
Influenced by:

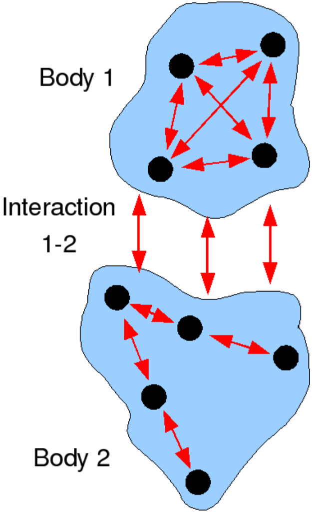
- Force $f(x, v)$ and stiffness $K = \frac{df}{dx}$
- Mass M
- Constraints $c(x)$, C

Two bodies interacting

- Body 1:
  - $x_1, \ v_1, \ f_1$
  - $f_1(x_1, v_1), \ K_{11}(x_1)$
  - M_1
  - $c_1(x), \ C_1$
- Interaction 1-2:

  $1 \rightarrow 2$ $f_{12}(x_1, v_1, x_2, v_2)$, $K_{12}(x_1, v_1, x_2, v_2)$

  $2 \rightarrow 1$ $f_{21}(x_1, v_1, x_2, v_2)$, $K_{21}(x_1, v_1, x_2, v_2)$
- Body 2



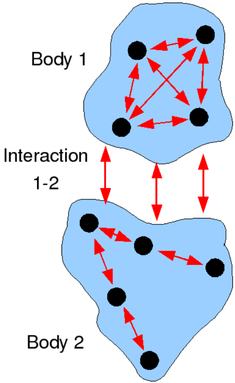
Implicit Euler

Solve $C(M + h^2 K)C\Delta v = hC(f + hKv)$

- C models constraints as filters

Apply conjugate gradient solution

- Does not addess the entries of the matrix
- Performs only matrix-vector products and vector products
- Products can be performed blockwise, in any order

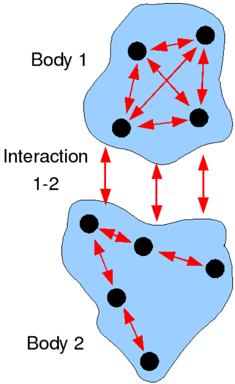


State vectors

- positions $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$
- velocities $v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$
- auxiliary vectors $a = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$,

  $aux = \begin{pmatrix} aux_1 \\ aux_2 \end{pmatrix} \ldots$
- force

  $f = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} f_1(x_1, v_1) + f_{12}(x_1, v_1, x_2, v_2) \\ f_2(x_2, v_2) + f_{21}(x_1, v_1, x_2, v_2) \end{pmatrix}$



Vector operations

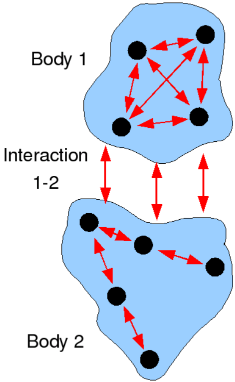
- Sums are computed in parallel:

$$x + ay = \begin{pmatrix} x_1 + ay_1 \\ x_2 + ay_2 \end{pmatrix}$$

- Dot products require to sum over all objects:

$$x^T y = x_1^T y_1 + x_2^T y_2$$

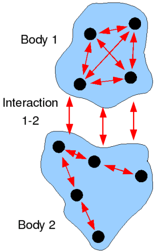
- State vectors can be stored and processed in parallel in each body
- They can even have different types in different bodies, *e.g.* particles and a rigid body



System matrices

Block structure:

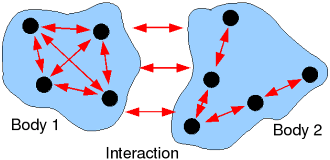
- $M = \begin{pmatrix} M_1 & \\ & M_2 \end{pmatrix}$ Mass matrix, block-diagonal
- $K = \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix}$ Stiffness matrix, generally sparse
- $C = \begin{pmatrix} C_1 & \\ & C_2 \end{pmatrix}$ Filter matrix, block-diagonal



Conjugate gradient solution:

- We do not need to address the entries of the matrices
- We need to compute their products with vectors

Matrix-vector product



- Without constraints:

$$(M + h^2 K)y = \begin{pmatrix} M_{11}y_1 + h^2 K_{11}y_1 + h^2 K_{12}y_2 \\ M_{22}y_2 + h^2 K_{22}y_2 + h^2 K_{21}y_1 \end{pmatrix}$$

- With constraints:

$$C(M+h^2 K)Cy = \begin{pmatrix} C_1 M_{11} C_1 y_1 + h^2 C_1 K_{11} C_1 y_1 + h^2 C_1 K_{12} C_2 y_2 \\ C_2 M_{22} C_2 y_2 + h^2 C_2 K_{22} C_2 y_2 + h^2 C_2 K_{21} C_1 y_1 \end{pmatrix}$$

Scene structure and elementary operations

System of bodies

- Body 1
  - State vectors x, v, f, aux1, ...
  - Mass: $M_1*, \quad M_1^{-1}*$
  - Force(s): $+= f_1(x_1, v_1), \quad += K_1*$
  - Filter(s): $c(x_1), \quad C_1*$
- Body 2
- Interaction 1-2
  - $+= f_{12}(), \quad += f_{21}(), \quad += K_{12}*, \quad += K_{21}*$

Right-hand term of the implicit integration

- vector to compute:

$$b = hC(f(x, v) + hK(x)v)$$

- operations:

	propagateState(x,v)
$b_i = 0$	b.clear()
$b_i += K_{ii} v_i$	AccumulateDForceAction(b,v)
$b_i += K_{ij} v_j$	
$b_i *= h$	v_teq(b,h)
$b_i += f_i(x_i, v_i)$	AccumulateForceAction(b)
$b_i *= h$	v_teq(b,h)
$b_i = C_i b_i$	ApplyConstraintsAction(b)

- PropagateStateAction updates the force and stiffness operators

Computation of a matrix-vector product

- In each body i:

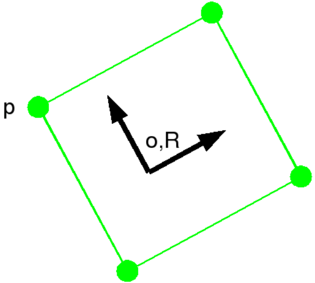
$$f_i = (C(M+h^2K)Cy)_i = C_i(M_{ii}+h^2K_{ii})C_iy_i + C_ih^2\sum_{j\neq i} K_{ij}C_jy_j$$

- Operations:

$f_i = 0$	f.clear()
$y_i = C_iy_i$	ApplyConstraintsAction(y)
$f_i+ = M_iy_i$	ApplyMassAction(f,y)
$f_i+ = K_{ii}y_i$	AccumulateDForceAction(f,y)
$f_i+ = K_{ij}y_j$	
$f_i = C_if_i$	ApplyConstraintsAction(f)

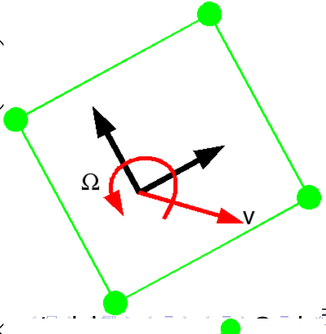
Mapped points

example: points attached to a rigid body



position: $p = o + R(op)$ velocity:
$v = v_o + po \times \Omega$

$$\begin{pmatrix} \Omega \\ v_p \end{pmatrix} = \begin{pmatrix} \Omega \\ v_o + po \times \Omega \end{pmatrix} = \begin{pmatrix} I & po\times \\ & I \end{pmatrix}$$



force in p:
$$\begin{pmatrix} f_p \\ \tau_p \end{pmatrix} = \begin{pmatrix} f_o = f_p \\ \tau_o = \tau_p + op \times f_o \end{pmatrix} \text{ and}$$

$$\begin{pmatrix} f_o \\ \tau_o \end{pmatrix} = \begin{pmatrix} f_p = f_o \\ \tau_p = \tau_o + op \times f_o \end{pmatrix} = \begin{pmatrix} I \\ op\times \end{pmatrix} = J^T$$

Mapping child forces to the parent DOF

Given $v_c = Jv_p$ and force f_c applied to the child, the equivalent parent force is:

$$f_p = J^T f_c$$

Proof: To be equivalent, f_c and f_p must have the sma virtual power:

$$\begin{aligned} f_p^T v_p &= f_c^T v_c \text{for any} v_p \\ &= f_c^T J v_p \text{for any} v_p \\ f_p^T &= f_c^T J \\ f_p &= J^T f_c \end{aligned}$$