SIGGRAPH ASIA Course

Practical Rigid Body Physics for Games

Abstract

This course concerns real-time rigid body simulation and its application in video games. The means of achieving the twin aims of high-speed and stable simulations under the limits imposed by a high frame rate of 60 FPS will be examined from a practical standpoint. Methods examined will be the Constraint Based Method (LCP) and the Impulse Based Method, commonly used in both commercial and open-source engines.

As iterative solvers are used as the basic technology for modern physics engines, stability and speed are essentially two sides of the same coin. Focus will therefore be placed on achieving stability in the simulation, using as few iterations as possible. After introducing the latest knowledge garnered from SIGGRAPH and GDC, I will introduce and explain some effective reform measures we have developed.

In the latest game platforms, simulation parallelization has become essential technology. The application of parallelization technology developed in the field of high performance computing to game engine development will be explored through reference to practical examples. We will perform real-time demonstrations of our in-house developed physics simulator to examine the effectiveness of both existing methods and our own refined methods.

Biography

Jumpei Tsuda, Senior Expert, Koei Co., Ltd.
Began his career as a solid modeling system development engineer. Joined Koei, where he first worked in crowd AI, before moving on to research and development roles in game development, covering such fields as collision detection, physics simulation and motion control.

Syllabus

Background [25 minutes]

- Constraint Based Method
- Impulse Based Method
- Iterative Solver

Acceleration and Stabilization Techniques [35 minutes]

- Slop, Permutation, Warm Start
- Shock Propagation
- Weight Amplification (our method)
- Aggressive Sleep (our method)
- Demo

Parallelization Techniques [35 minutes]

- Multi Color Ordering (Argebraic and Graphic)
- Cell-like Ordering (our method)
- Demo

Q&A [10 minutes]

Practical Rigid Body Physics for Games

KOEI Co., Ltd.

Technical Development Division

Jumpei Tsuda

The Physics of Rigid Bodies (with Constraints)

Motion Equation

Linear:

$$\mathbf{F} = m\dot{\mathbf{v}}$$

(**F**:force, m:mass, $\dot{\mathbf{v}}$:acceleration)

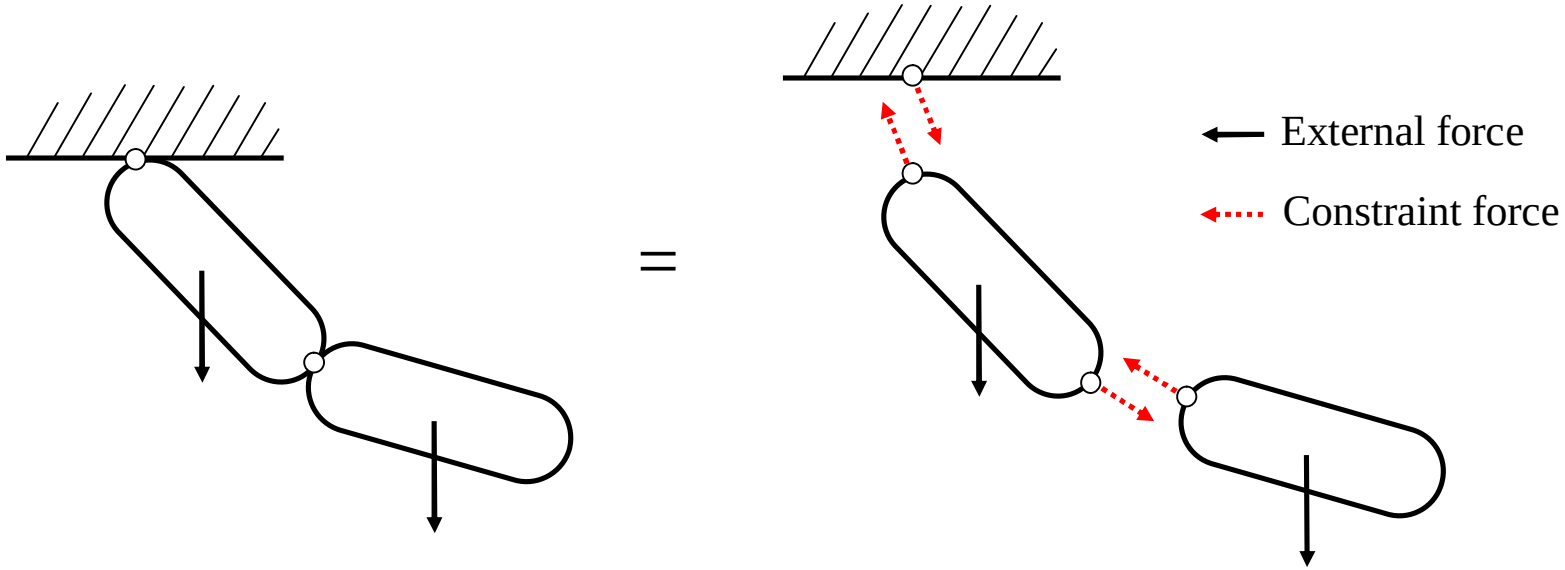
Rotational:

$$\boldsymbol{\tau} = \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}$$

($\boldsymbol{\tau}$:torque, **I**:inertia tensor, $\boldsymbol{\omega}$:angular velocity)

Constraint Based Method (CBM)

- [Erleb05], [Baraff89]
- Calculate the force that the constraint exerts on objects.
- Insert the constraint force into the motion equation's **F** and **τ**.
- Then solve the same way as if there was no constraint.



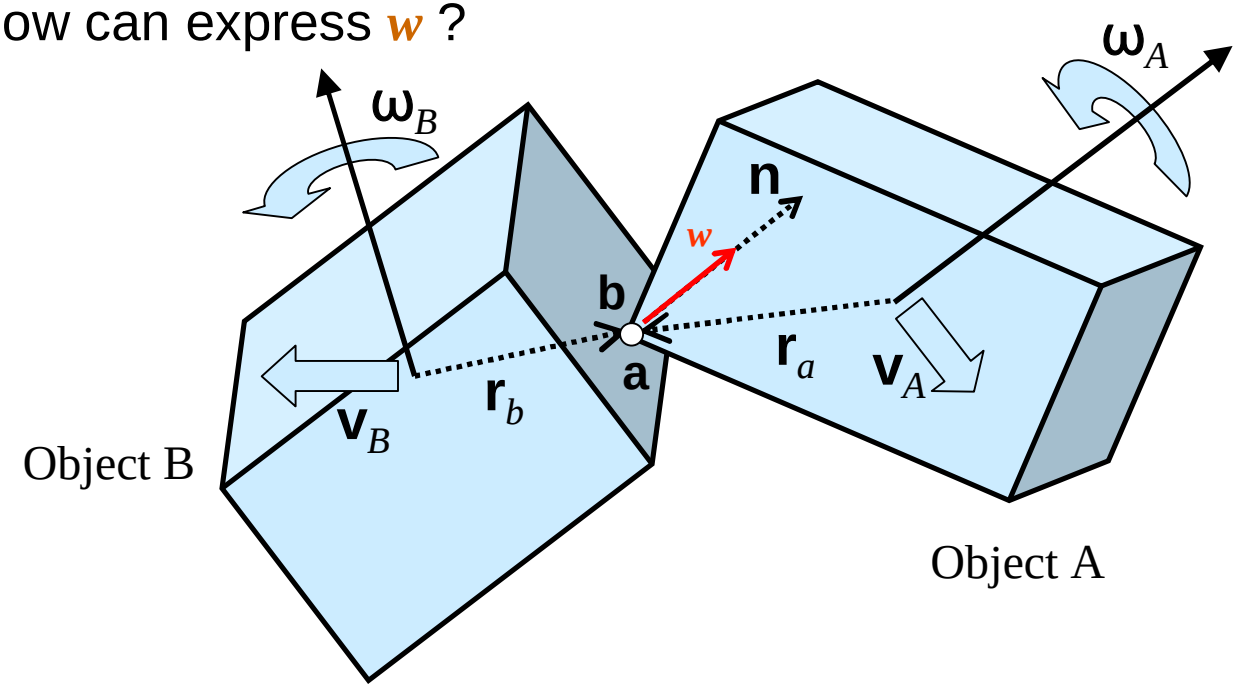
Relative Velocity

Object A is colliding with object B.

Contact points **a** and **b** belong to objects A and B, respectively.

w denotes the velocity of point **a** relative to point **b** along the contact normal **n**.

How can express w ?



$\mathbf{v}_A$ = COG velocity of object A

$\mathbf{v}_B$ = COG velocity of object B

$\boldsymbol{\omega}_A$ = angular velocity of object A

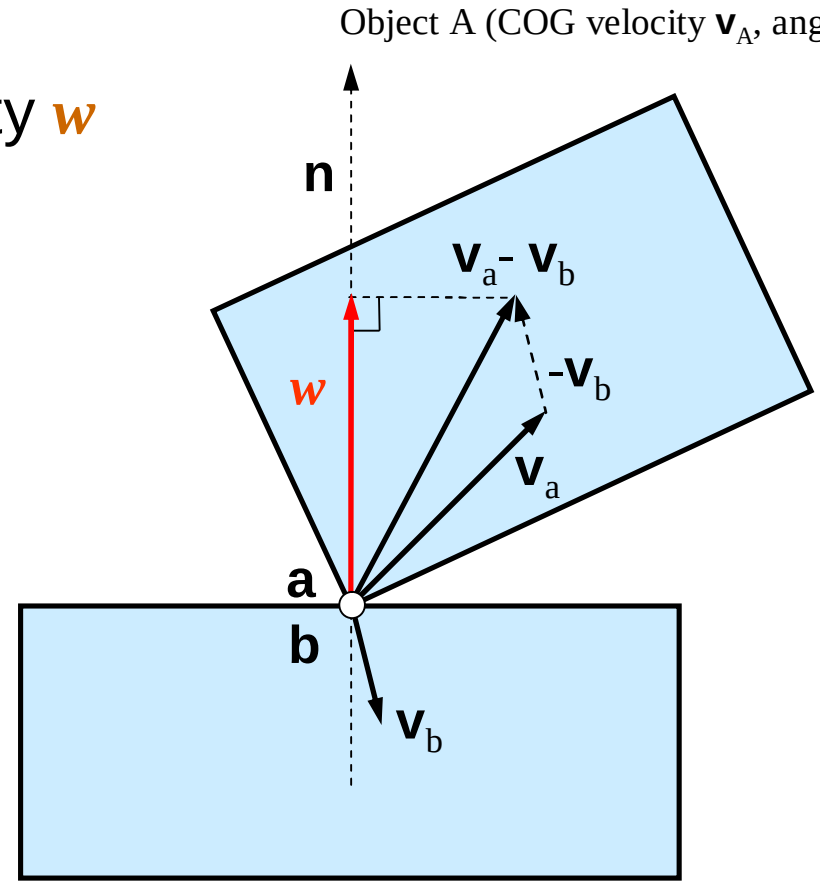
$\boldsymbol{\omega}_B$ = angular velocity of object B

$\mathbf{r}_a$ = COG of object A → point **a**

$\mathbf{r}_b$ = COG of object B → point **b**

Relative Velocity

Represent the relative velocity w along the contact normal **n** with $\mathbf{v}_A, \mathbf{v}_B, \omega_A$, and ω_B



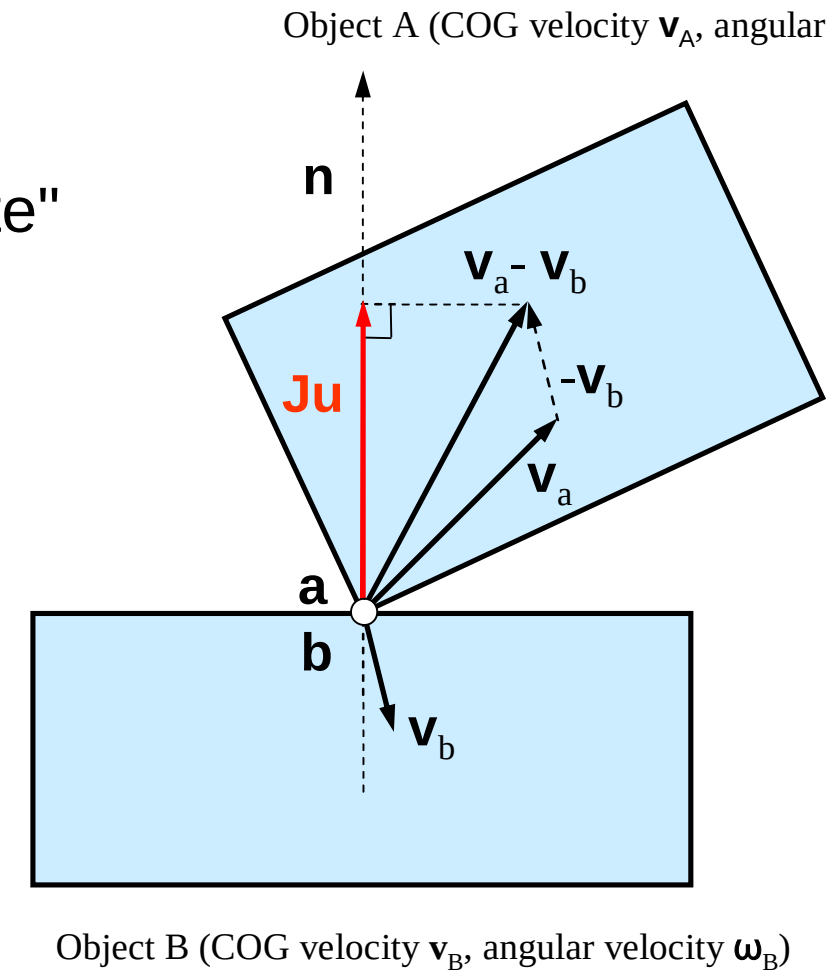
Relative Velocity

$$
\begin{aligned}
w &= \mathbf{n} \cdot (\mathbf{v}_a - \mathbf{v}_b) \\
&= \mathbf{n} \cdot (\mathbf{v}_A + \boldsymbol{\omega}_A \times \mathbf{r}_a - \mathbf{v}_B - \boldsymbol{\omega}_B \times \mathbf{r}_b) \\
&= \mathbf{n} \cdot \mathbf{v}_A + \mathbf{r}_a \times \mathbf{n} \cdot \boldsymbol{\omega}_A - \mathbf{n} \cdot \mathbf{v}_B - \mathbf{r}_b \times \mathbf{n} \cdot \boldsymbol{\omega}_B \\
&= (\mathbf{n}^{\mathsf{T}}, (\mathbf{r}_a \times \mathbf{n})^{\mathsf{T}}, -\mathbf{n}^{\mathsf{T}}, -(\mathbf{r}_b \times \mathbf{n})^{\mathsf{T}}) \begin{pmatrix} \mathbf{v}_A \\ \boldsymbol{\omega}_A \\ \mathbf{v}_B \\ \boldsymbol{\omega}_B \end{pmatrix} \\
&= (\mathbf{J}_A, \mathbf{J}_B) \begin{pmatrix} \mathbf{u}_A \\ \mathbf{u}_B \end{pmatrix} \qquad [\, \mathbf{J}_A \in \mathrm{R}^{1x6}, \mathbf{J}_B \in \mathrm{R}^{1x6}, \mathbf{u}_A \in \mathrm{R}^{6x1}, \mathbf{u}_B \in \mathrm{R}^{6x1} \,] \\
&= \mathbf{J}\mathbf{u} \qquad [\, \mathbf{J} \in \mathrm{R}^{1x12}, \mathbf{u} \in \mathrm{R}^{12x1} \,]
\end{aligned}
$$

Constraint Condition

Therefore, the condition
"a collision does not penetrate"
is given by

$$\mathbf{Ju} \geqq 0$$



Constraint Condition

In general, the constraint condition can be described by

$$\mathbf{Ju} \geqq 0 \quad \text{(no penetration, movement range limitation, etc.)}$$

or

$$\mathbf{Ju} = 0 \quad \text{(joints, etc.)}$$

Constraint Force Direction

Suppose that **J** has been determined, then

Direction of constraint force = $\mathbf{J}^T$

Why is this?

- **J** represents the direction in which the object cannot move any further.
- There is no "resistance" in any direction in which the object can move, so no other forces arise.
- The force only arises in the direction in which the object cannot move.

For detailed analysis, see [Catt08]

Mass Matrix

Handle mass and Inertia tensor together.

$$\mathbf{M} = \begin{pmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & I_{xy} & I_{xz} \\ 0 & 0 & 0 & I_{yx} & I_{yy} & I_{yz} \\ 0 & 0 & 0 & I_{zx} & I_{zy} & I_{zz} \end{pmatrix}$$

Introducing the Force

Velocity after one timestep is given by

$$\mathbf{u}' = \mathbf{u} + \dot{\mathbf{u}} \times \Delta t$$

$$= \mathbf{u} + \mathbf{M}^{-1}(\mathbf{F}_{external} + \mathbf{F}_{constraint})\Delta t$$

If the constraint still exists after one timestep

$$\mathbf{Ju}' = \mathbf{Ju} + \mathbf{JM}^{-1}(\mathbf{F}_{external} + \mathbf{F}_{constraint})\Delta t \geqq 0$$

Suppose the constraint force is of magnitude λ in the direction $\mathbf{J}^T$

$$\mathbf{Ju}' = \mathbf{Ju} + \mathbf{JM}^{-1}(\mathbf{F}_{external} + \mathbf{J}^T\lambda)\Delta t \geqq 0$$

$$= \mathbf{JM}^{-1}\mathbf{J}^T\lambda\Delta t + \mathbf{J}(\mathbf{u} + \mathbf{M}^{-1}\mathbf{F}_{external}\Delta t) \geqq 0$$

Introducing the Force

For simplicity we incorporate Δt into the λ term

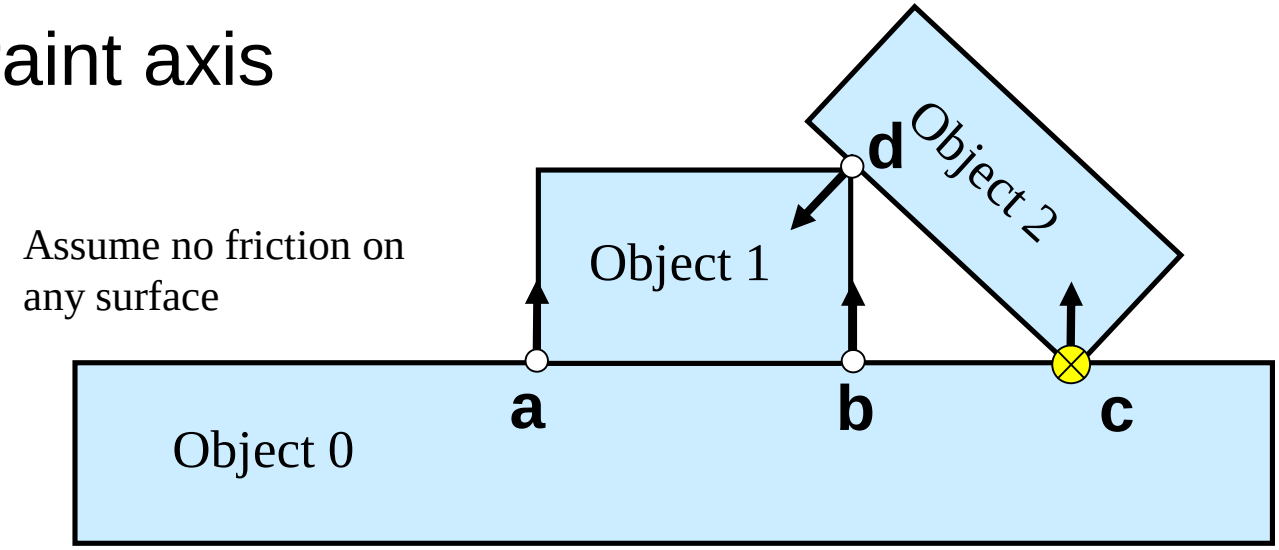
$$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T}\lambda + \mathbf{b} \geqq 0$$ (no penetration, movement range limitation, etc.)

or

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T}\lambda + \mathbf{b} = 0$$ (joints, etc.)

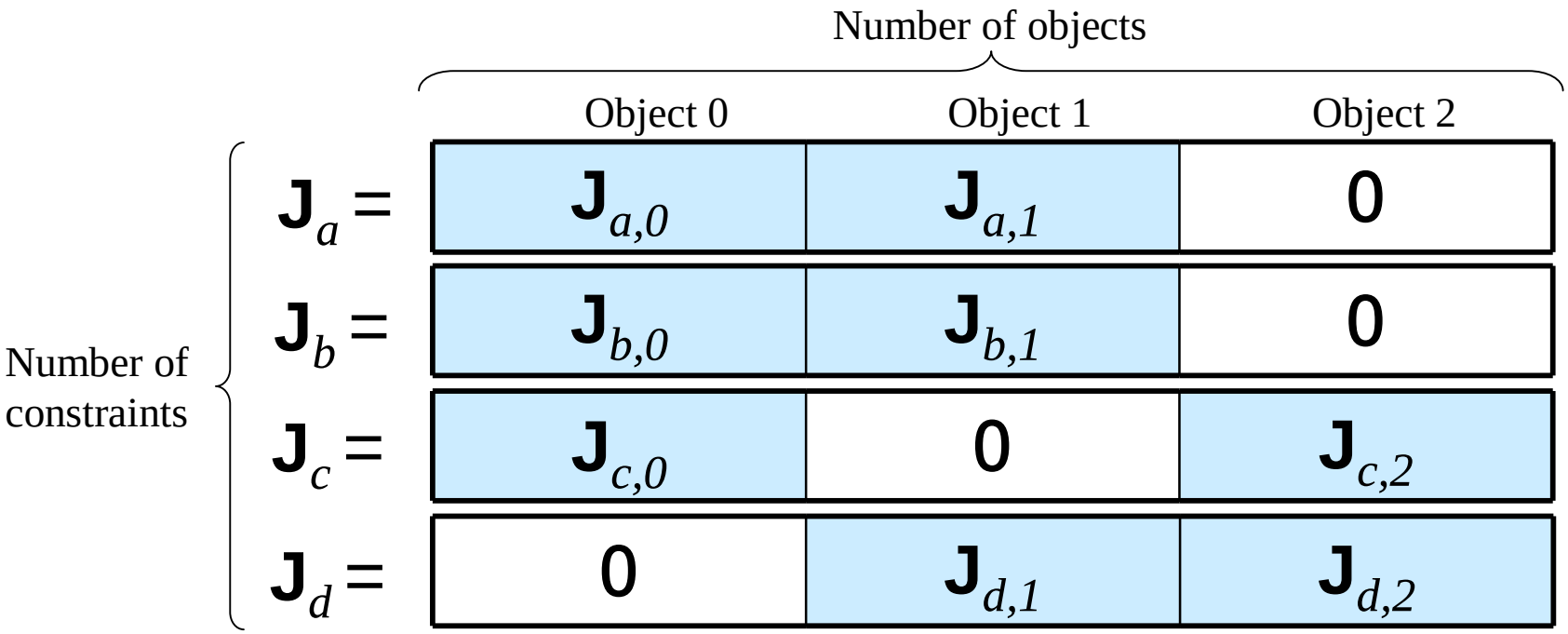
Multibody Physics

- On-face constraint = bilateral constraint
- Contact point = unilateral constraint
- Constraint axis



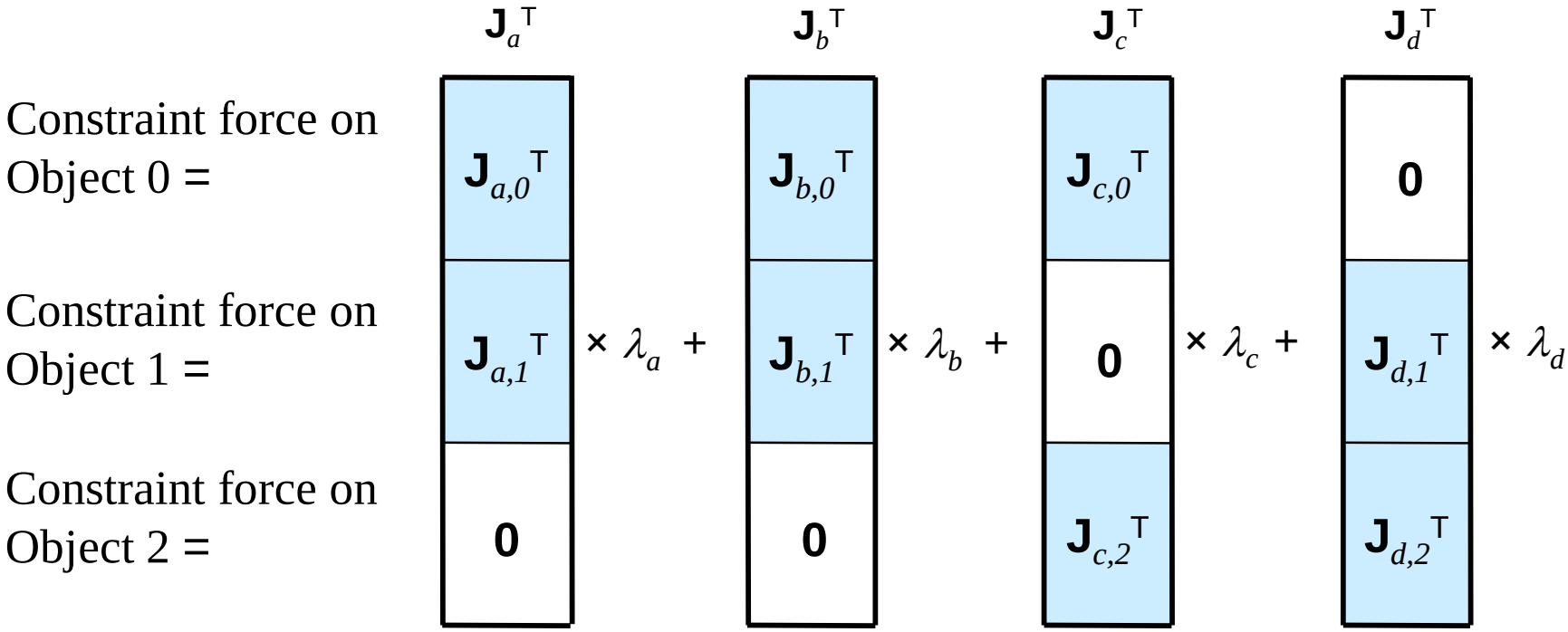
Constraint Matrix

Create a matrix representing the entire system of constraints.

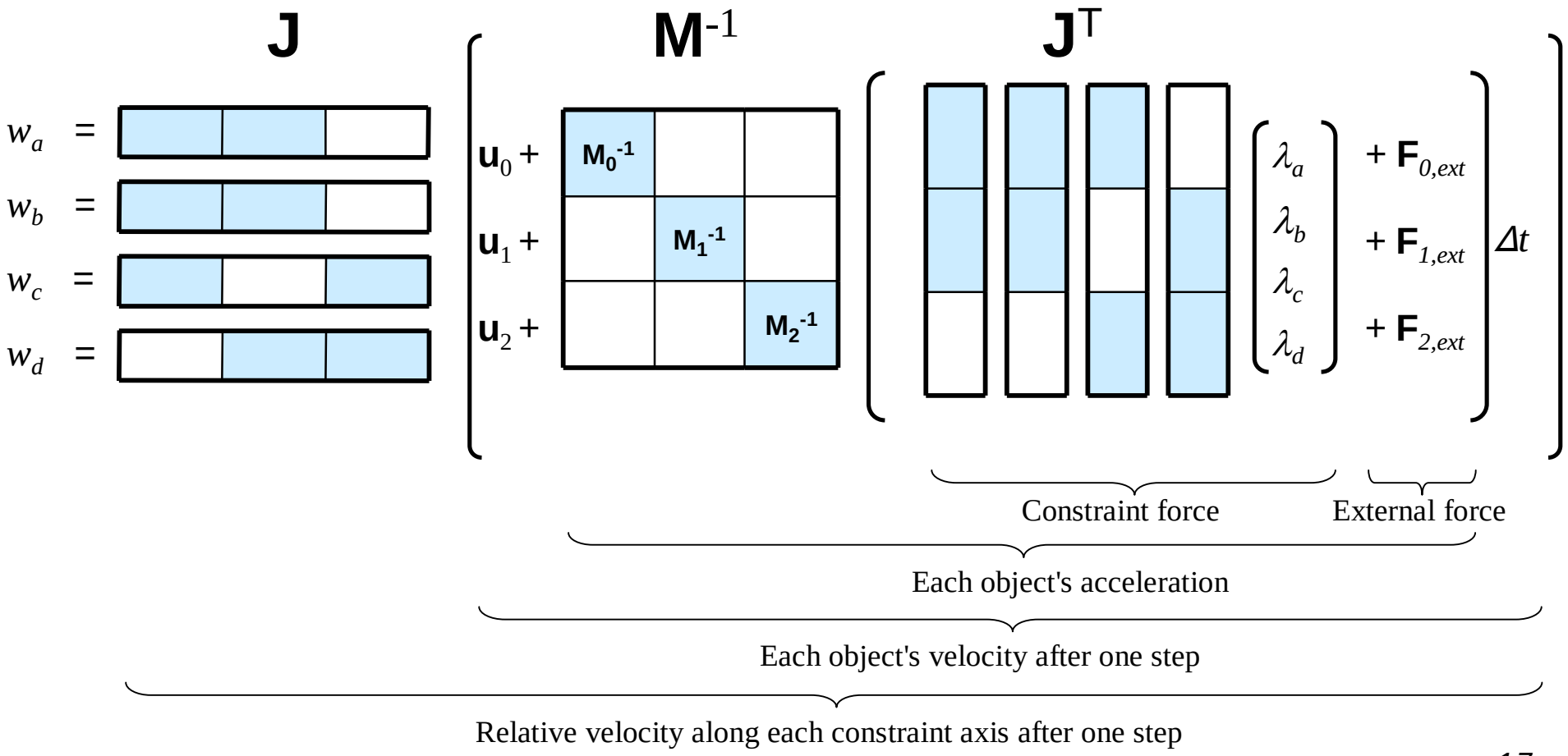


Only 12 elements are effective in each row

Constraint Force



Constraint Formulation [Erleb05]



Constraint Formulation

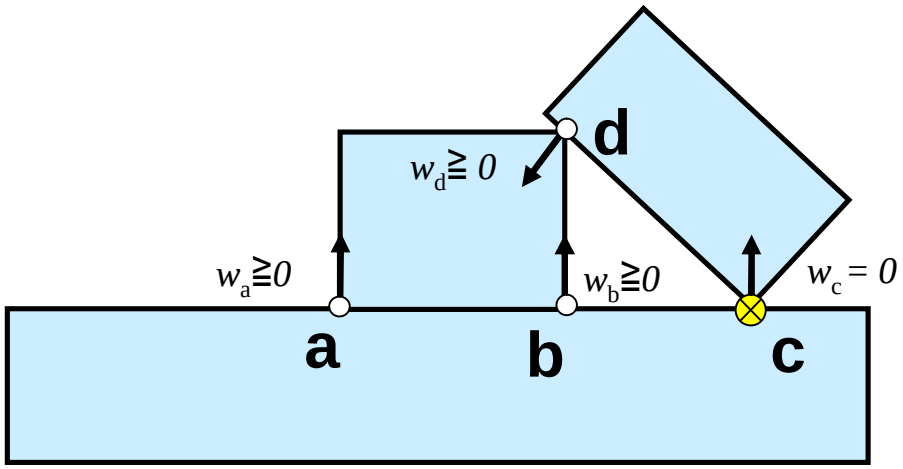
$$\begin{pmatrix} w_a \\ w_b \\ w_c \\ w_d \end{pmatrix} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathsf{T}}\boldsymbol{\lambda}\Delta t + \mathbf{J}(\mathbf{u} + \mathbf{M}^{-1}\mathbf{F}_{ext}\Delta t) = \mathbf{A}\boldsymbol{\lambda} + \mathbf{b}$$

(For simplicity, Δt is incorporated into $\boldsymbol{\lambda}$ on the right-hand side)

Constraint Formulation

Now we have equations...

$$\mathbf{A}\lambda + \mathbf{b} = \begin{matrix} w_a & \geqq & 0 \\ w_b & \geqq & 0 \\ w_c & = & 0 \\ w_d & \geqq & 0 \end{matrix}$$



with mixed equalities and inequalities!

LCP(Linear Complementary Problem)

Take **w = Aλ + b .**

Solve for **λ** that

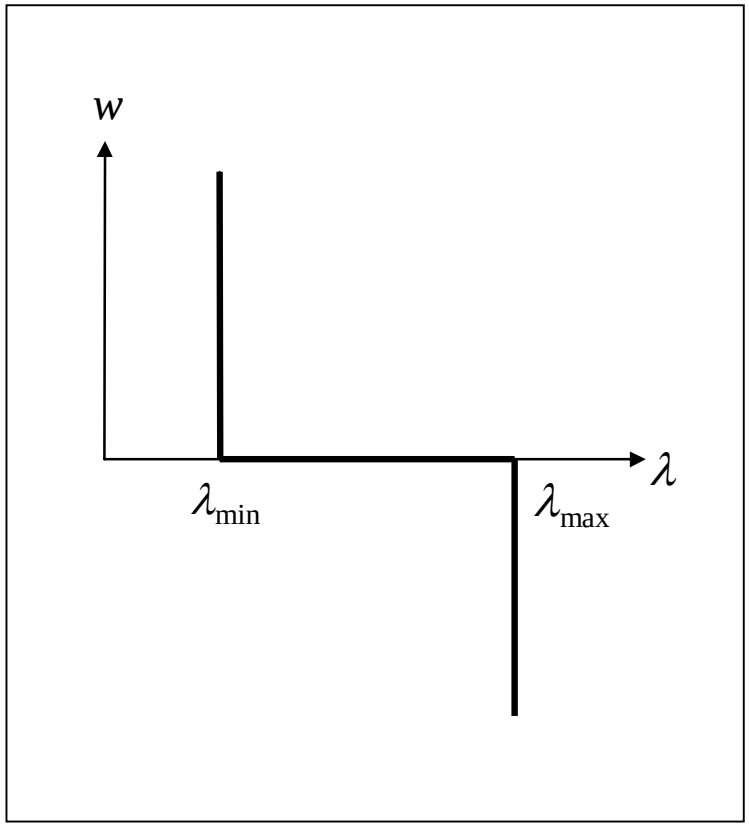
satisfies the following:

$\lambda = \lambda_{min}$ then $w > 0$

$\lambda_{min} < \lambda < \lambda_{max}$ then $w = 0$

$\lambda = \lambda_{max}$ then $w < 0$

For more details, see [AIMMS07]



LCP

With the LCP we can describe many different constraints in a unified fashion.(Only red items are valid)

Non-penetration constraint ($\lambda_{min} = 0$, $\lambda_{max} = \infty$)

$\lambda = 0$	then $w > 0$
$0 < \lambda < \infty$	then $w = 0$
$\lambda = \infty$	then $w < 0$

Bilateral constraint ($\lambda_{min} = -\infty$, $\lambda_{max} = \infty$)

$\lambda = -\infty$	then $w > 0$
$-\infty < \lambda < \infty$	then $w = 0$
$\lambda = \infty$	then $w < 0$

Gauss-Seidel Method (GSM)

Transform a linear equation as follows

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix}$$

$$\begin{pmatrix} -a_{00} & 0 & 0 \\ 0 & -a_{11} & 0 \\ 0 & 0 & -a_{22} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & a_{01} & a_{02} \\ a_{10} & 0 & a_{12} \\ a_{20} & a_{21} & 0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix}$$

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & -a_{01}/a_{00} & -a_{02}/a_{00} \\ -a_{10}/a_{11} & 0 & -a_{12}/a_{11} \\ -a_{20}/a_{22} & -a_{21}/a_{22} & 0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_0/a_{00} \\ b_1/a_{11} \\ b_2/a_{22} \end{pmatrix}$$

Gauss-Seidel Method

Consider the transformed equation to be an "update expression."

The k th update is

$$\begin{pmatrix} x_0^{k+1} \\ x_1^{k+1} \\ x_2^{k+1} \end{pmatrix} = \mathbf{A} \begin{pmatrix} x_0^{k} \\ x_1^{k} \\ x_2^{k} \end{pmatrix} + \mathbf{b}$$

Suppose $\mathbf{x}^{k+1} = \mathbf{x}^{k} = \mathbf{x}_{conv}$ after some updates,

$\mathbf{x}_{conv}$ is the solution to the original equation.

(This is trivially shown by tracing the transformation back.)

Gauss-Seidel Method

Use the results of previous rows to update the current row immediately.

$$\boxed{x_0^{k+1}} = a_{00}\,x_0^k + a_{01}\,x_1^k + a_{02}\,x_2^k + b_0$$

$$\boxed{x_1^{k+1}} = a_{10}\,\boxed{x_0^{k+1}} + a_{11}\,x_1^k + a_{12}\,x_2^k + b_1$$

$$x_2^{k+1} = a_{20}\,\boxed{x_0^{k+1}} + a_{21}\,\boxed{x_1^{k+1}} + a_{22}\,x_2^k + b_2$$

Projected Gauss-Seidel Method

An effective means for solving LCP [Cottle92]

Clamp λ at every iteration.

$$\lambda = \text{clamp}(\lambda, \lambda_{min}, \lambda_{max})$$

That's all.

Projected Gauss-Seidel Method

Why can this solve for the LCP?

Suppose that **A** = **JM**$^{-1}$**J**^{T}, then diagonal elements a_{kk} of **A** are positive.

k' th row of expression **w = Aλ + b** is

$$w_k = a_{k1}\lambda_1 + a_{k2}\lambda_2 + \ldots + a_{kk}\lambda_k + \ldots + a_{kn-1}\lambda_{n-1} + a_{kn}\lambda_n + b_k$$

When $\lambda_{min} < \lambda_k < \lambda_{max}$

w_k is the k th row of equation **Aλ + b = 0 ,** so it will converge to 0

When $\lambda_k < \lambda_{min}$, if we let $\lambda_k \leftarrow \lambda_{min}$ then

w_k becomes a larger value than before the clamp. i.e. it will converge to $w_k > 0$

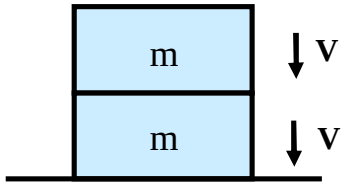
When $\lambda_k > \lambda_{max}$, if we let $\lambda_k \leftarrow \lambda_{max}$ then

w_k becomes a smaller value than before the clamp. i.e. it will converge to $w_k < 0$

Impulse Based Method (IBM)

- [Guendelman03], [Mirtich95]
- Assume a constraint is a series of collisions.
- With every collision, a small impulse is applied.
- Many types of constraints are expressible.
  - collision, non-penetration, joint, etc.

Impulse Based Method

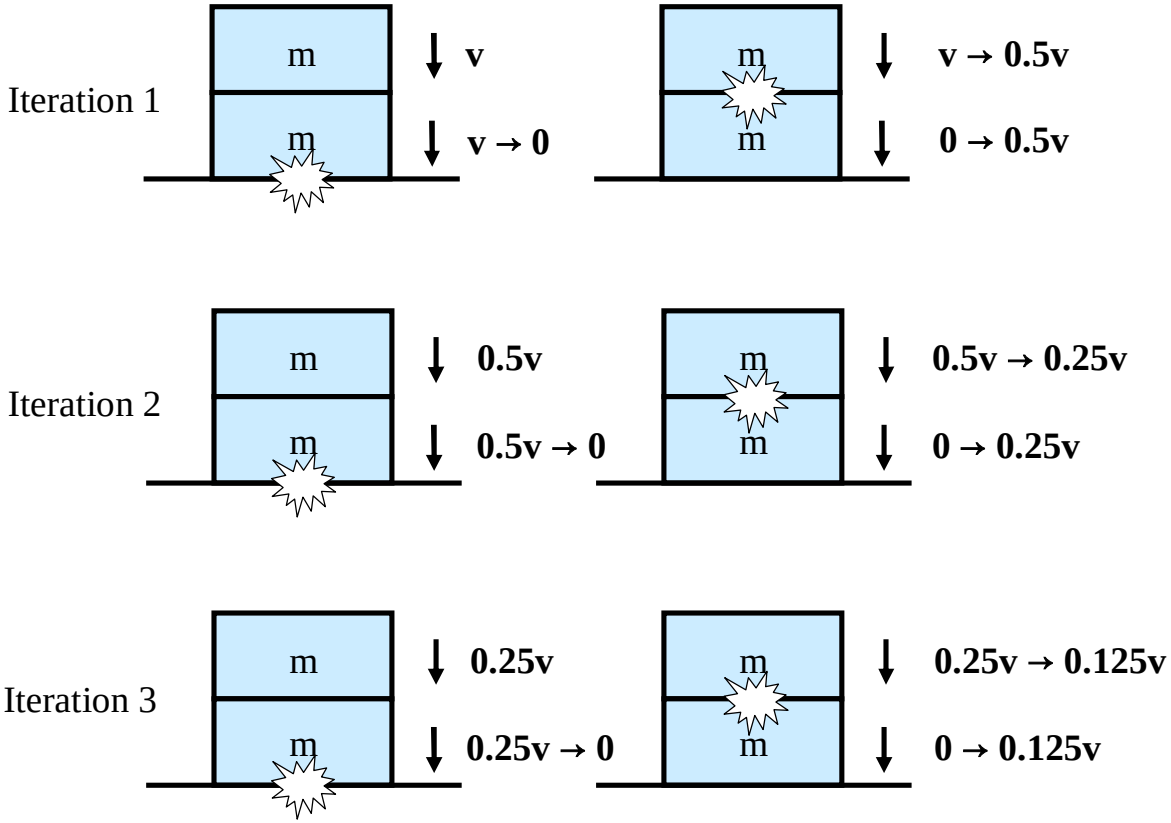


Example of rest contact

Conservation of momentum

$m_0\mathbf{v'}_0 + m_1\mathbf{v'}_1 = m_0\mathbf{v}_0 + m_1\mathbf{v}_1$

Collide two objects iteratively and propagate the force.



Coefficients of restitution set to 0

CBM vs. IBM

Recall the constraint condition.

$$\mathbf{A\lambda + b = Ju'}$$

The i th row of this expression is

$$\Sigma_k A_{i,k}\lambda_k + b_i = \Sigma_k J_{i,k}u'_k$$

The i th row of **J** includes only 12 effective elements corresponding to two objects interacting with each other.

CBM vs. IBM

Therefore, when using GSM, processing one row is equal to calculating the λ generated by the interaction of these two objects.

λ has been multiplied by Δt , so this is equal to an impulse.

Moreover, the λ s (=Impulses) of previous rows are used to calculate the current row's λ immediately. This means that the new velocity which has been updated by other contact or joint forces is used to calculate the current λ (=Impulse) within an iteration.

CBM vs. IBM

Essentially,

CBM with GSM-like iterative solving = IBM.

So, the following discussion can be applied to both methods.

Stabilization and Performance

Stabilization and Performance

- What is stabilization?
  - When the forces are balanced the object will be at rest or moving with constant velocity.
- Stabilization and performance optimization are one and the same
  - The faster a system is stabilized, the fewer iterations the solver has to process.
- Sleep functions do not contribute to stability
  - Once the system is stabilized, then you can sleep it.

Situations requiring Stability

- Stacking
  - Poor stability leads to fall-apart.
- Complex joint interactions (ragdoll phys.,etc.)
  - Poor stability means the object won't ever "die."

Stabilization Techniques

- Slop (tolerance of penetration)
- Permutation (exchange of rows)
- Warm Start (initialize with previous step)
- Shock Propagation (remedy for stacking)

Stabilization Techniques

- Weight Amplification (our method)
  - an improved version of Shock Propagation
- Aggressive Sleep (our method)
  - an improved version of Sleep

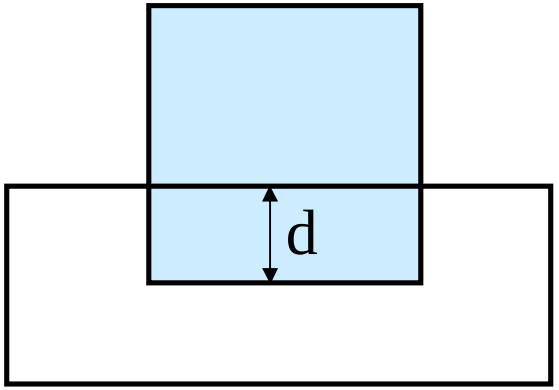
Resolving Penetration

When an object penetrates, add velocity such as to cancel out the penetration.

The depth of penetration is d,

and the relative velocity after one step is **u'.**

Then the constraint condition is



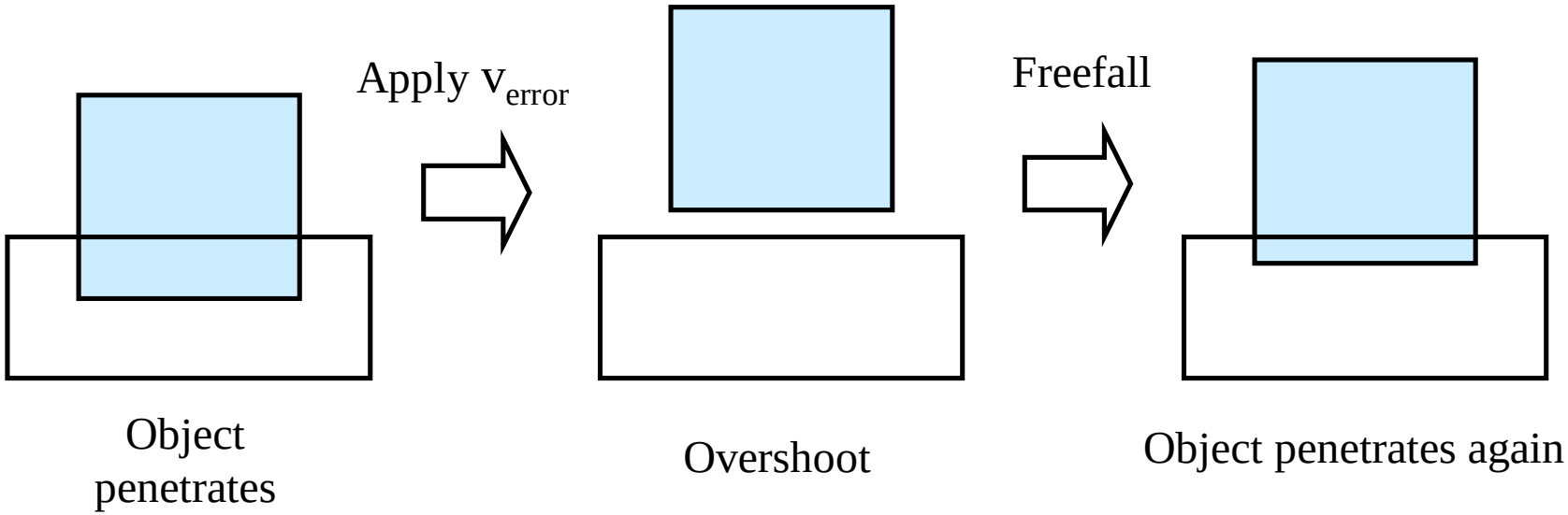
$$\mathbf{Ju'} \geqq v_{error}$$

where

$v_{error} = kd/\Delta t$ (k is a coefficient of reduction less than 1)

Slop (Tolerance of Penetration)

If we merely insert the penetration-canceling velocity v_{error} into the equations then oscillation occurs. [Catt08]

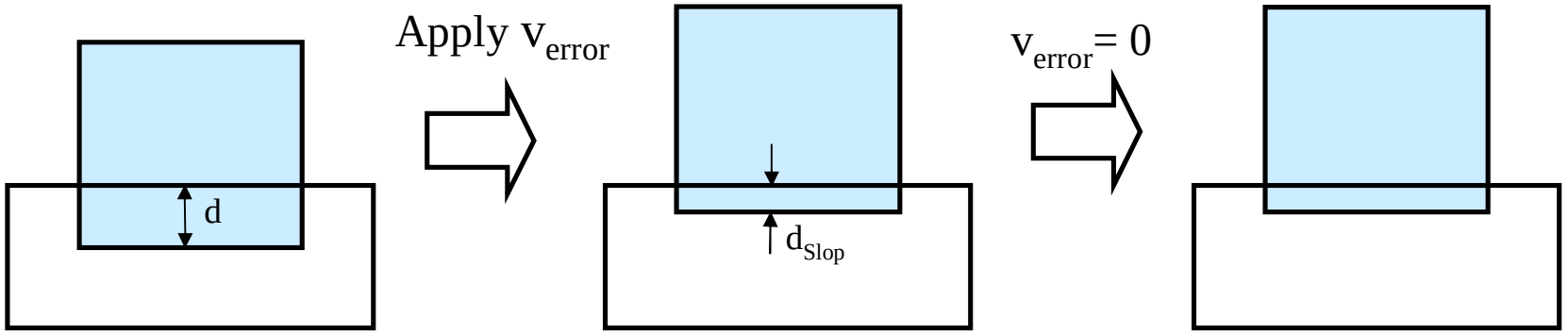


Slop (Tolerance of Penetration)

We mustn't completely cancel out penetration.

Allow penetration down to a depth d_{slop} → produce a stable support force

$v_{error} = (d - d_{slop})/\Delta t$



Object penetrates

Allow penetration down to d_{Slop}

Stabilizes with small penetration

Permutation

- Every row of the LCP corresponds to a constraint.
- The stacking order is often reflected strongly in the constraint matrix.
- If you solve in the stacking order then errors accumulate.
  - The simulation becomes unstable.
- Randomly rearrange rows of the matrix before starting the solver.

Warm Start

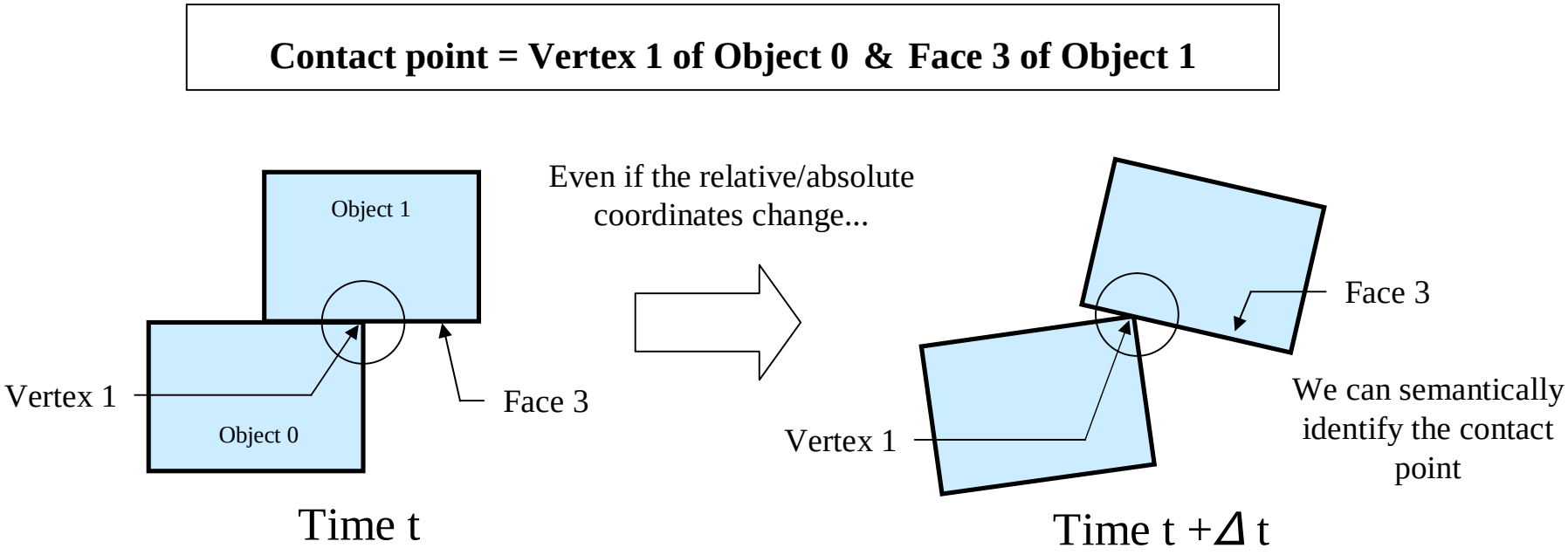
- [Catt05], [Erleb05]
- The Gauss-Seidel method is an iterative convergence calculation.
- The closer it starts to the solution, the faster it converges.
- Take the previous step's results as your new initial values.
- You will see a huge improvement in stability/performance.

Warm Start

- The previous step's solution is the constraint force λ
- In the next step, where will we apply λ ?
- Joints are persistent
  - No searching necessary
- Contact points are not so
  - Must search to find the identical point

Warm Start

- Contact points will change somewhat between steps.
- Apply a semantic ID to each contact point [Moravan04]

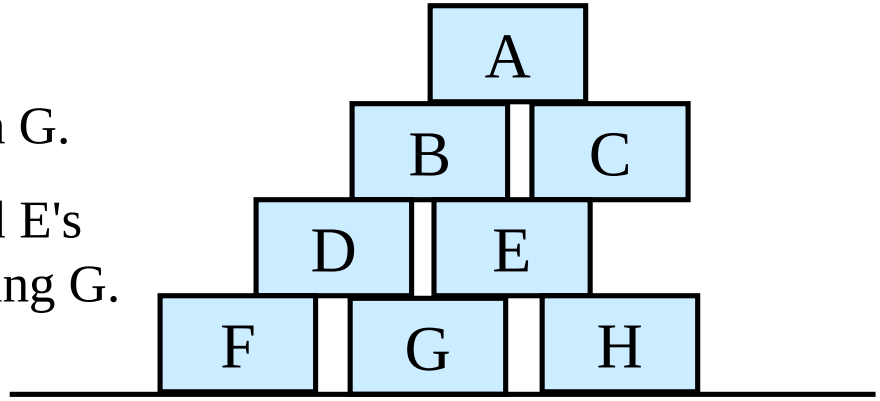


Stacking

- Why is stacking difficult?
  - Must simulate force propagation on multiple levels
  - The more levels, the more iterations are needed

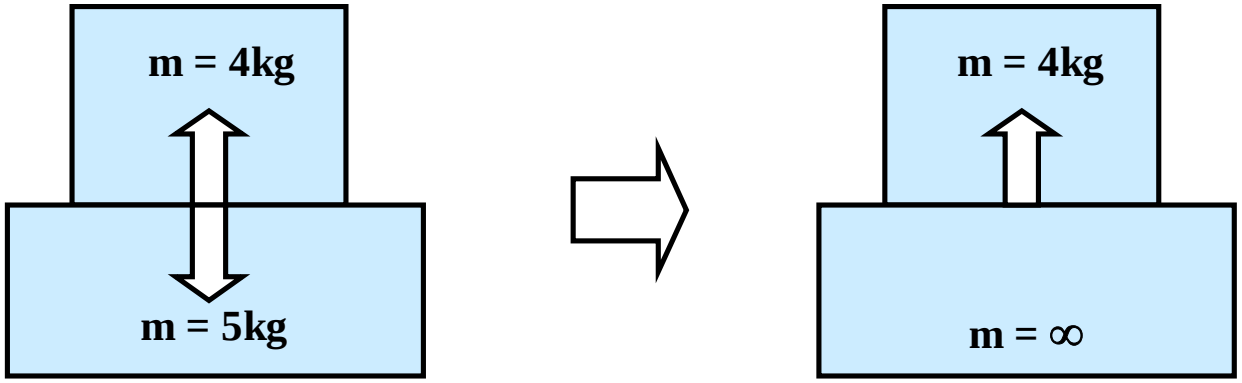
A, B, C, D, E all exert a force on G.

F and H support some of D's and E's weight, thereby indirectly affecting G.



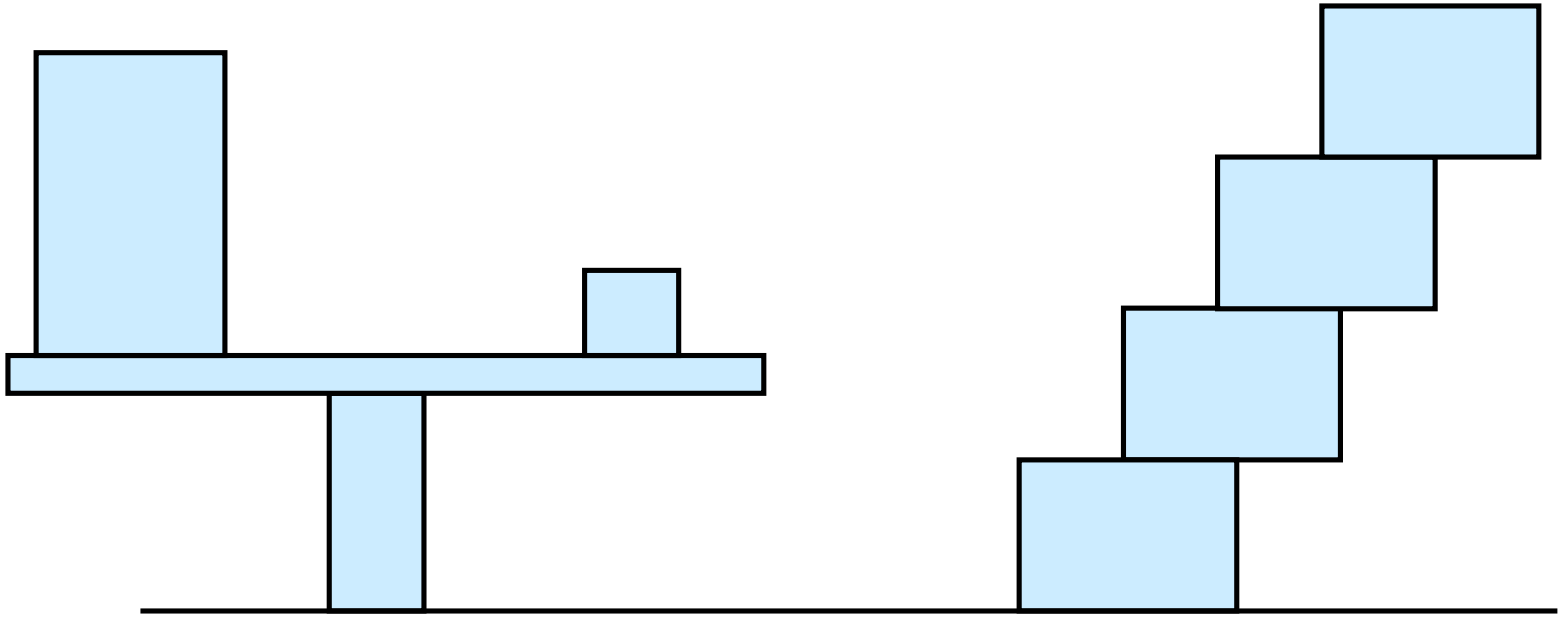
Shock Propagation

- [Guendelman03]
- A powerful solution to the stacking problem.
- A kind of "fake", but an execellnt idea for real-time simulation.
- Take the "bottom" object's mass to be ∞.
- Restrict the direction of "force" propagation to "down-to-up".



The Weight Feeling Problem

- This is a side effect of Shock Propagation.
- The system becomes "too stable".



The Weight Feeling Problem

A quick solution:

λ_0 = the solution when the masses are not changed

λ_1 = the solution when the "bottom" is set to $m = \infty$

Take a weighted average of the two solutions:

$$\lambda = (1 - c)\lambda_0 + c\lambda_1$$

But...

Cost of Shock Propagation

- When applied to the LCP you must run the simulation twice.
- The performance cost of the solver is doubled.

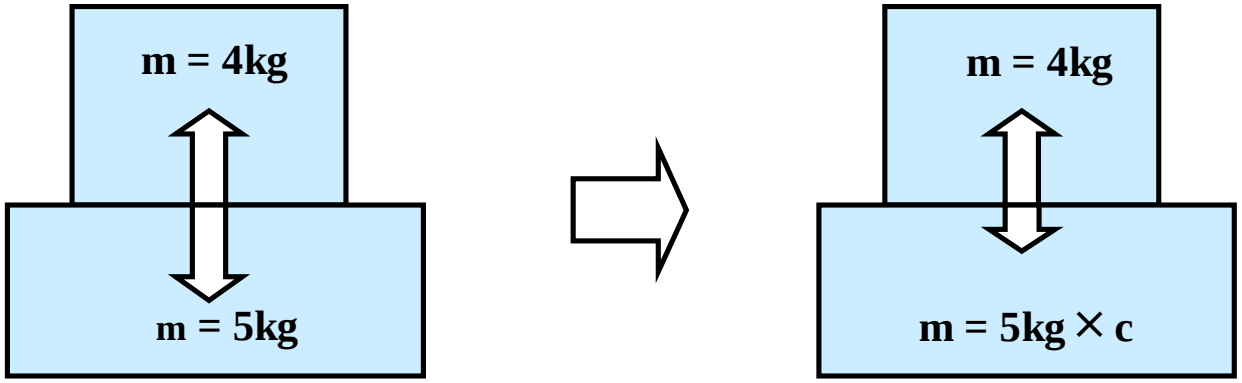
Improvement of Shock Propagation

- "First, heat the bathwater to 100℃"
- "That's too hot, so add cold water until it cools down to 40℃"
- Is there no room for improvement with this picture?

Why not heat it to just 40℃ to begin with?

Weight Amplification (our method)

- Apply an amplification coefficient c to the "bottom" object.
- $c = 1.4$ gives very good results.
- This works for stacks of up to tens of objects.
- This fixes the Weight Feeling problem at no additional cost.

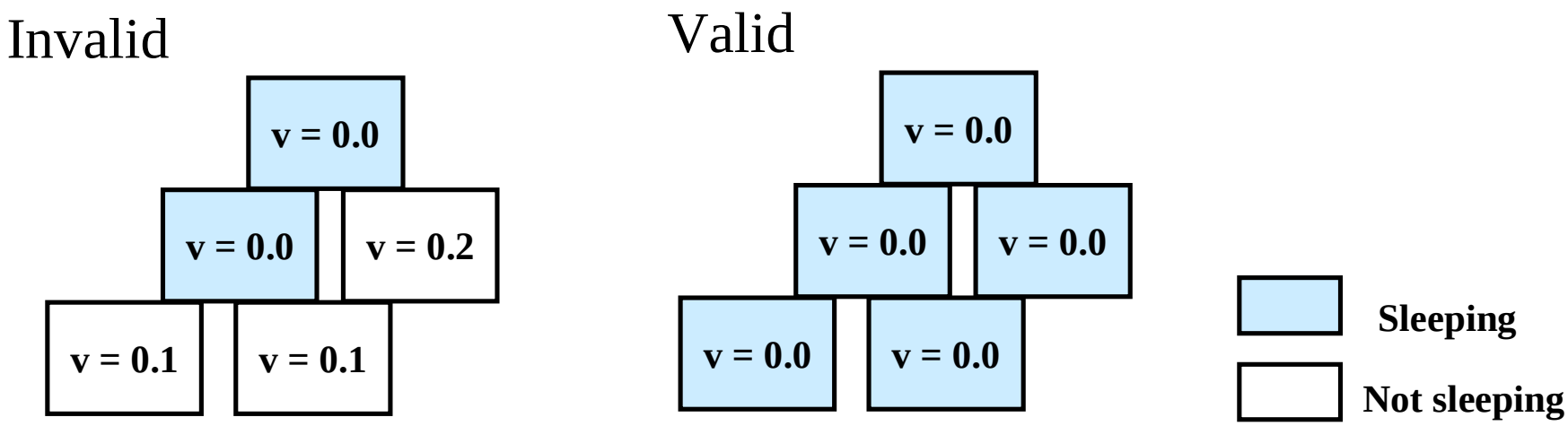


Sleep

- If an object's velocity is close to zero, exclude it from the simulation.
- Reduce unnecessary simulation costs.

Sleep

- Apply to each "simulation island."
- Can't sleep just individual objects.
- Sleep does not contribute to the simulation's stability.



Aggressive Sleep (our method)

Slop

Contact point creation is stabilized

↓

The structure of the LCP to be solved is also stabilized

Aggressive Sleep

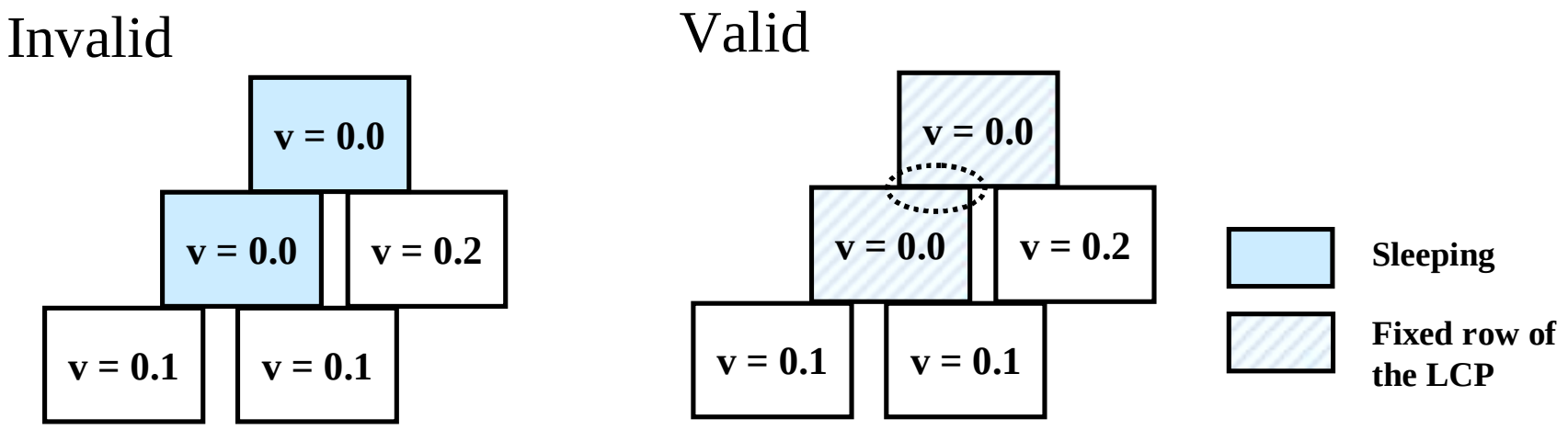
The rows of the LCP corresponding to the resting objects are fixed (using the previous values)

↓

The structure of the LCP to be solved is also stabilized

Aggressive Sleep (our method)

- The fixed row of the LCP uses the previous step's values (helps performance, too).
- Can be applied to individual objects.
- Don't have to be careful in determining whether an object is at rest (easily adjusted)
- The simulation still runs, so objects don't mistakenly stop.



Result (Stabilization)

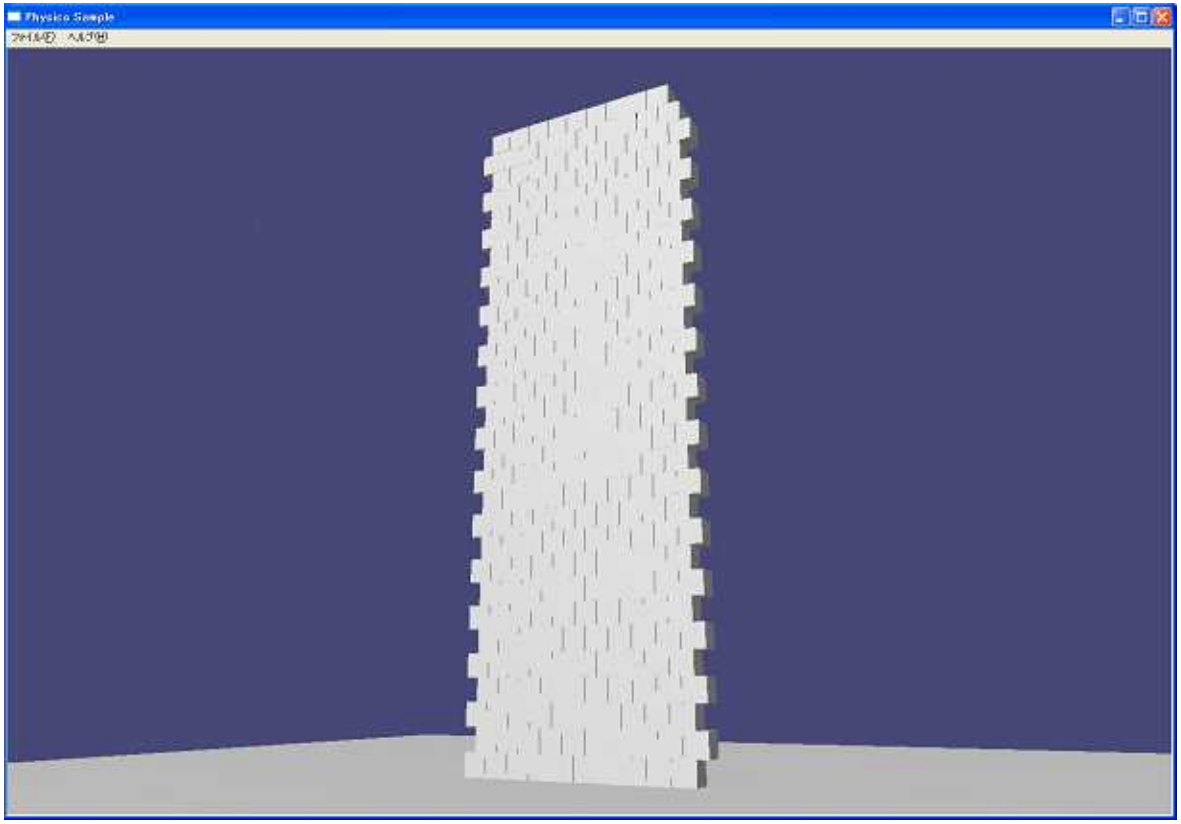


Figure 1: 300 boxes stacking (30 stacking stages)

Result (Stabilization)

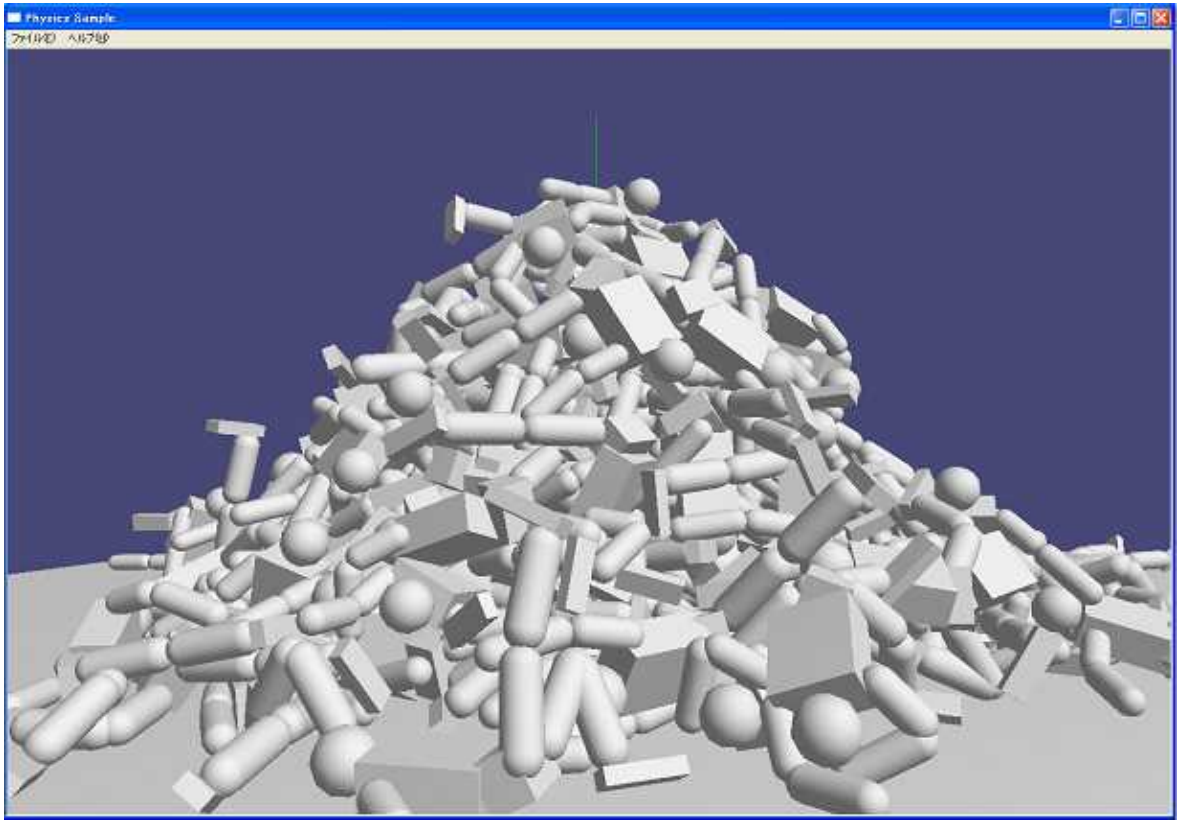


Figure 2: 150 ragdolls stacking (13 objects / ragdoll)

Result (Stabilization)

Combination of stabilization methods	Necessary iterations for resting
Slop Permuation Warm Start	90 times (at least)
Slop Permutation Warm Start Weight Amplification	15 times

Result of Figure1

Combination of stabilization methods	Necessary iterations for resting
Slop Permutation Warm Start Weight Amplification	50 times (at least)
Slop Permutation Warm Start Weight Amplification Aggressive Sleep	10 times

Result of Figure2

Parallelization

Difficulty of Parallelization

Stage of simulation	Difficulty
Broad Phase	difficult
Narrow Phase	easy
Constraint Building	easy
Constraint Solving	very difficult
Integration	easy

Constraint Solving

- Constraint forces of bodies are mutually dependent.
  - When using GSM, rows of the coefficient matrix are mutually dependent.
- Need to find independent parts of process.
- Simulation island?
  - Practically efficiency is very low.
  - Huge simulation islands appear very often.

Finding Independent parts

- Multi Color Ordering
- Red-Black Ordering
- Cell-like Ordering (our method)

Multi Color Ordering

- # of colors = # of dependent groups.
- # of colors = # of synchronizations.
- Within a group constraints are independent.
- When moving from group(=color) to group, synchronization is needed.

Multi Color Ordering (Algebraic)

If an element (i, j) of a coefficient matrix is non-zero, assign different colors to unknowns corresponding to i and j respectively.

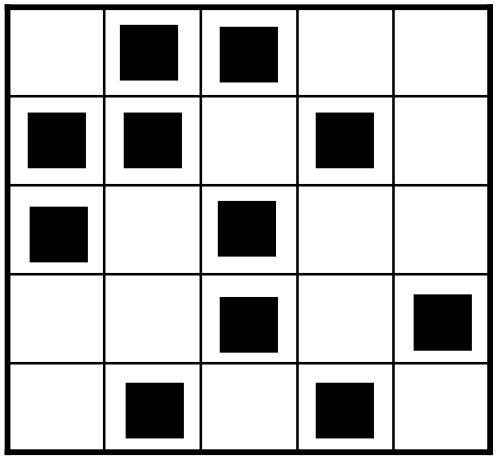
[Mifune05] proposed the following code.

```
float A[N][N];   //coefficient matrix.
int color[N];    //colors of rows.

for (i = 0; i < N; ++i) color[i] = -1;

for (i = 0; i < N; ++i) {
        m = 0;
        (*)
        for (j = 0; j < i; j++) {
                if (A[i][j] != 0 && color[j] == m) {
                        ++m;
                        goto (*);
                }
        }
        color[i] = m;
}
```

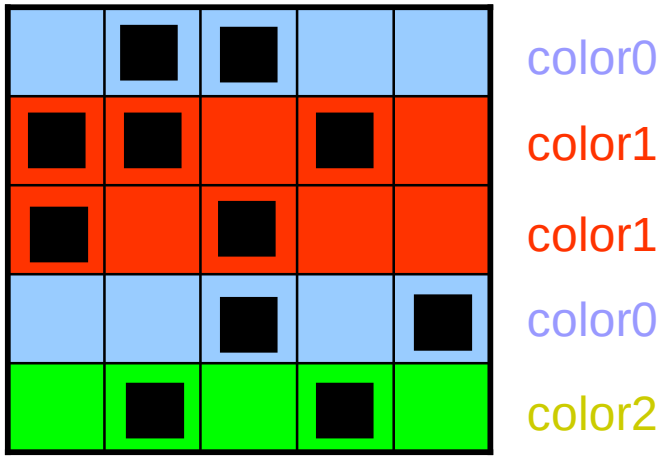
Multi Color Ordering (Algebraic)

- By using previous code...

Coefficient matrix A



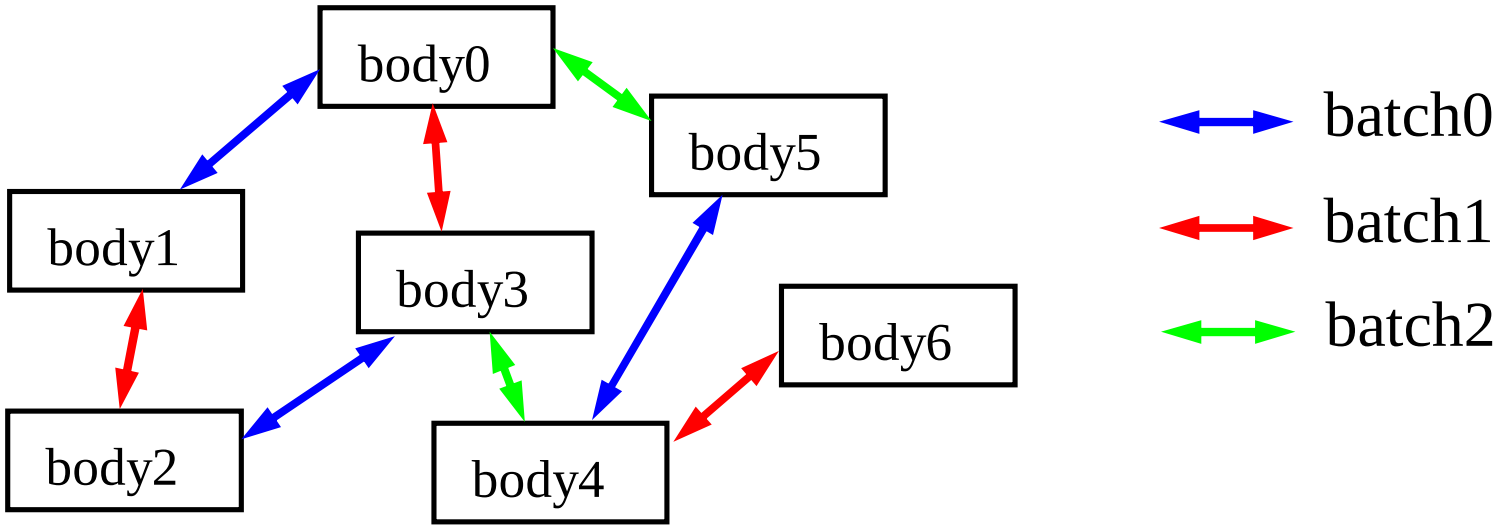
Non-zero element



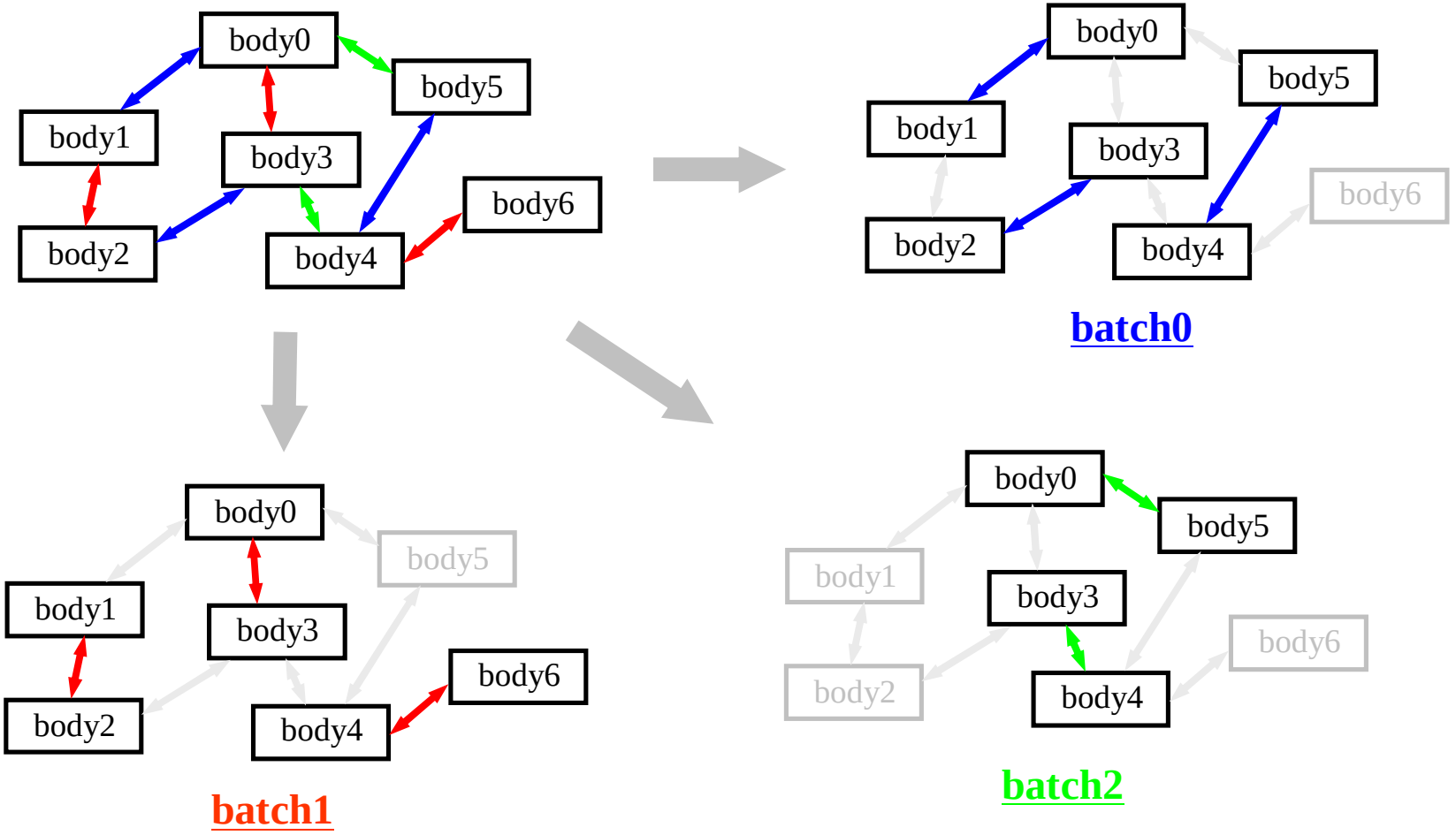
Rows of the same color are independent of each other.
So the solver can handle them in parallel.

Multi Color Ordering (Geometric)

- "Link" means joint or contact.
- Classify links into "batches". [Chen05], [Keogh07]
- Within a batch each link is independent.



Multi Color Ordering (Geometric)



Multi Color Ordering

- Algorithm is relatively simple.
- Synchronization cost tends to increase.
  - If a body contacts with n bodies, there would be at least n synchronizations.

Cell-like Ordering (our method)

- A variant of block multi color ordering [Yosui07].
- Exploit contact graph [Hahn88].

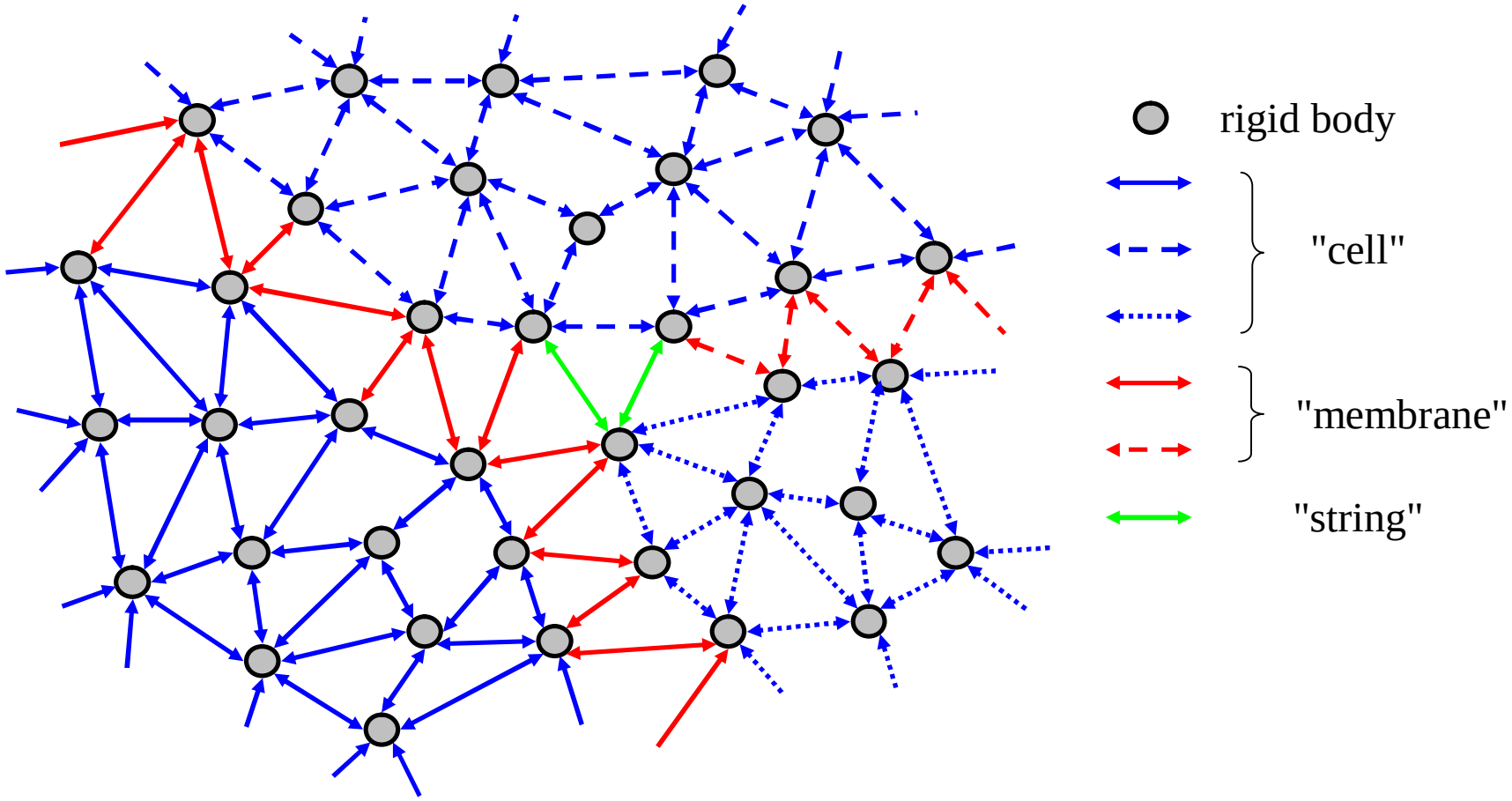
step 1. Breadth-first search for links connected to a body.

step 2. Add the links to the current block.

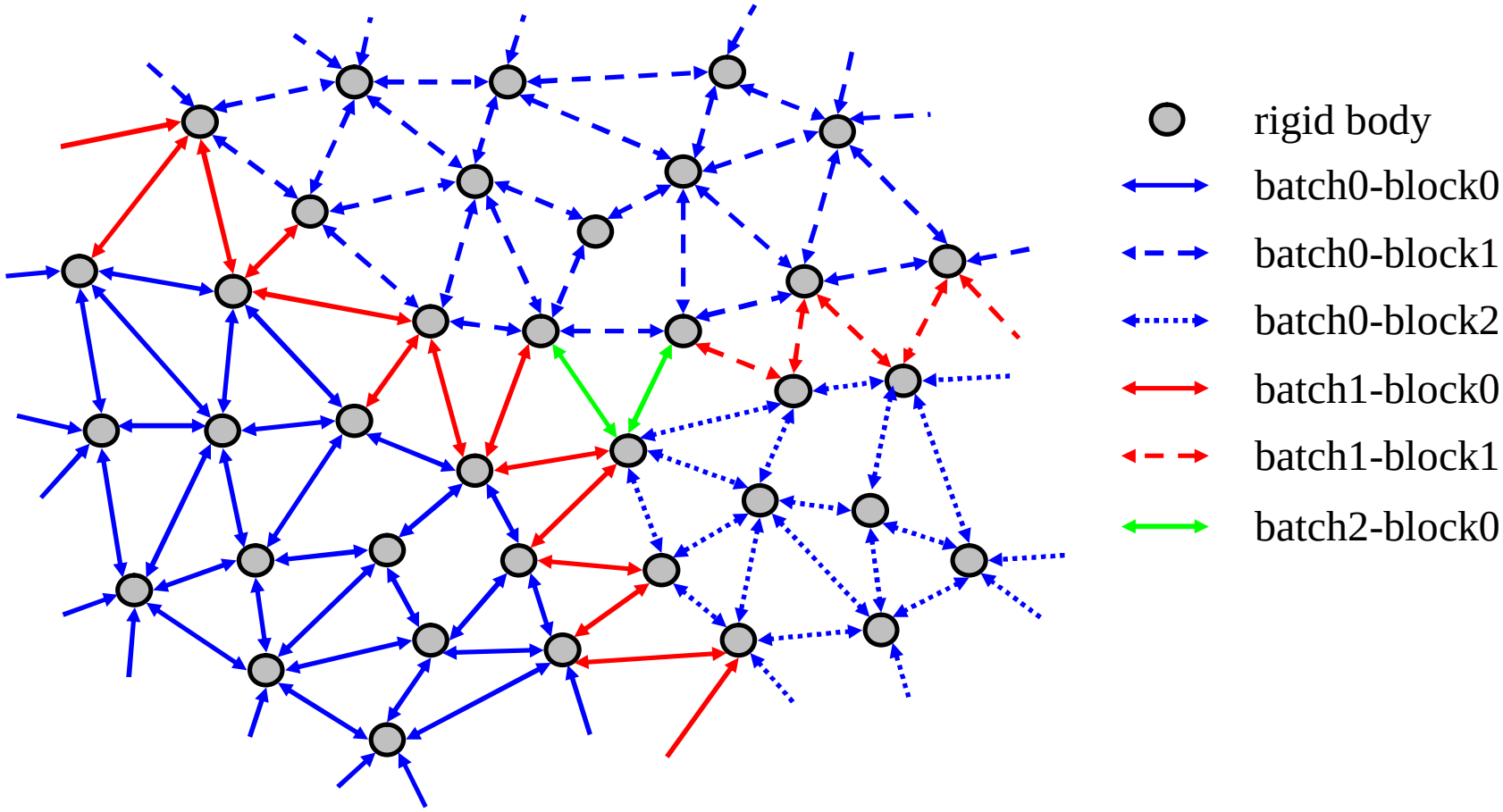
step 3. If the # of links reaches the block granularity, move to the next block.

step 4. goto step 1.

Cell-like Ordering (Analogy)



Cell-like Ordering

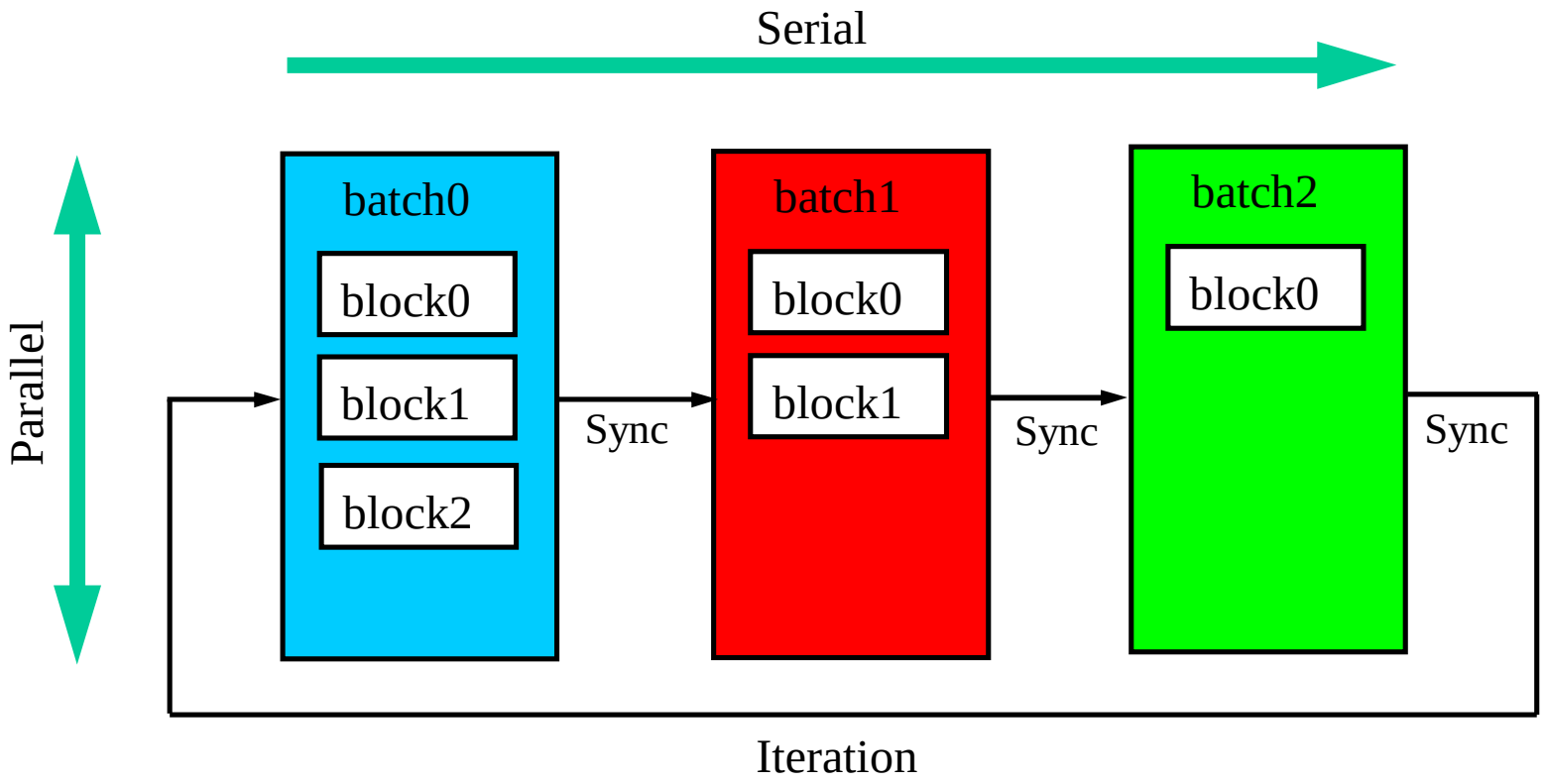


Cell-like Ordering

- Less synchronization cost.
- Logically 4 synchronizations per iteration.
  - "volume" → "face" → "line" → "point"
  - In case of thousands of objects.
- Practically 3 synchronizations per iteration.
  - "volume(cell)" → "face(membrane)" → "line(string)"
  - In case of hundreds of objects.

Cell-like Ordering

Within a batch each block is processed in parallel.



Cell-like Ordering

- Within a batch each block is independent.
  - i.e. parallelizable.
- Block Granularity is flexible.
  - On recent game platforms, this is important.
- Only graph search is necessary.
  - Geometric calculation not necessary.

Cell-like Ordering (Code)

```
l          :link (joint or contact).
b, b0, b1  :body (rigid body).
L          :set of links.
B          :set of bodies.
Q          :set of bodies which works as a queue.
Lij        :sets of links subscripts i and j(batch and block indexes, respectively).
Bij        :sets of bodies subscripts i and j(batch and block indexes, respectively).
g          :granularity.

CellLikeOrdering()
begin
     i ← 0; j ← 0;
     L ← all links in a scene;
     while L ≠ ϕ
          B ← all bodies connected to L;
          while B ≠ ϕ
               b ← an element of B;
               B ← B - b;
               Lij ← ϕ; Bij ← ϕ;
               SearchAroundBody(b);
               j ← j + 1;
          end while
          i ← i + 1;
          j ← 0;
     end while
end
```

Cell-like Ordering (Code)

```
SearchAroundBody(b)
begin
     g ← 0;
     while g < block granularity
          for each l connected to b
               if l ∈ L
                    b0 ← the body on one side of l;
                    b1 ← the body on the other side of l;
                    if ¬({b0, b1} ∩ Bpq ≠ ϕ s.t. p ≠ i ∨ q ≠ j)
                         Lij ← Lij + l;
                         Bij ← Bij + {b0, b1};
                         L ← L - l;
                         B ← B - {b0, b1};
                         g ← g + 1;
                         append {b0, b1} to the tale of Q;
                    end if
               end if
          end for
          b ← the first element of Q;
          Q ← Q - b;
     end while
     return;
end
```

Result (Parallelization)

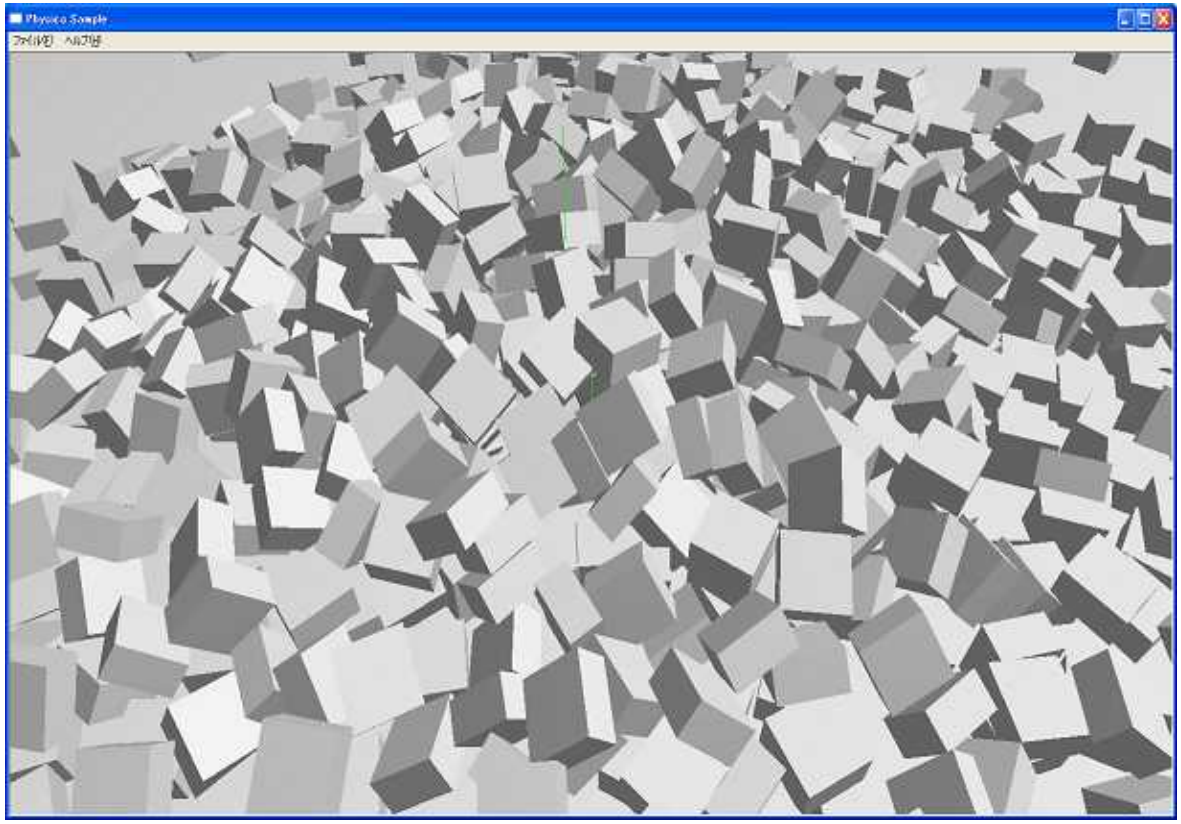


Figure 3: 720 boxes stacking.

Result (Parallelization)

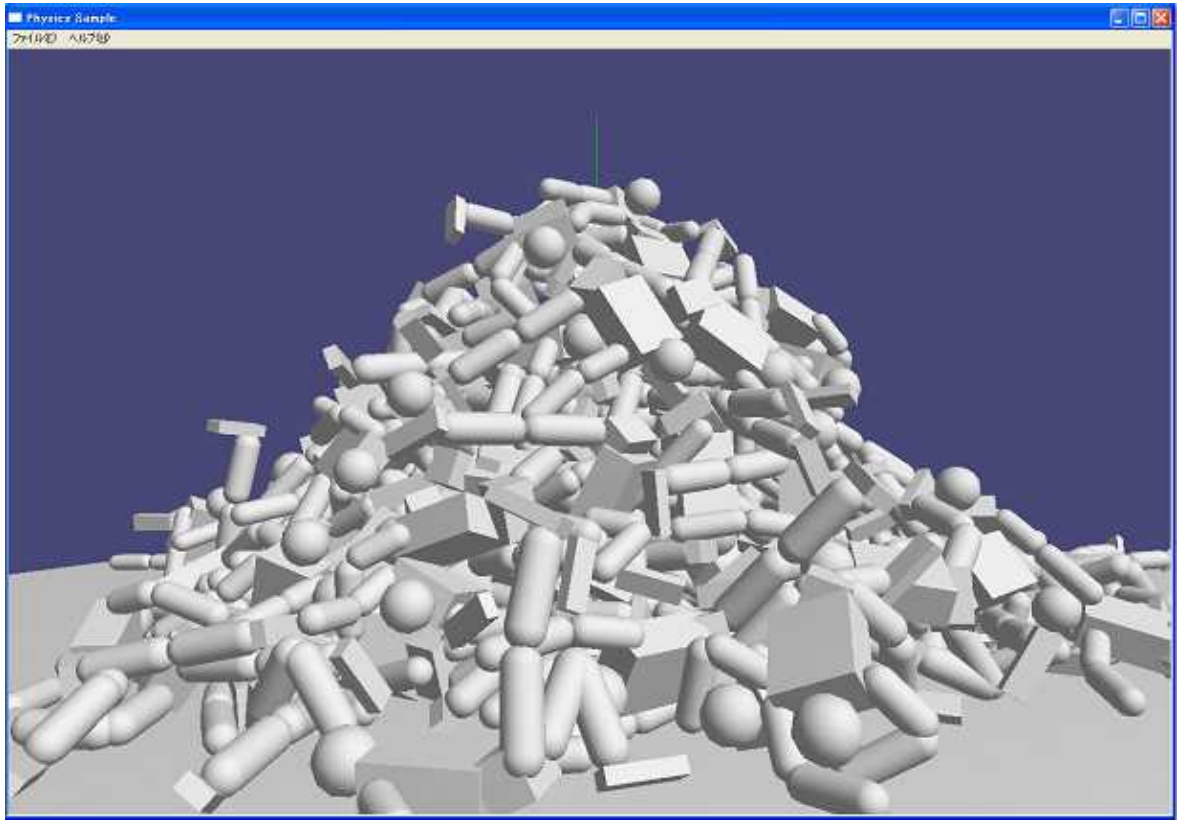


Figure 4: 1950 objects (= 150 ragdolls) stacking.

Result (Parallelization)

Batch #		0	1	2	3	4	5	6	7	8	9	10	11
Number of λs / Batch	Multi Color Ordering	3759	1762	1227	795	474	180	93	18	9	-	-	-
	Cell-like Ordering	7113	1116	93	-	-	-	-	-	-	-	-	-

Result of Figure 3: In case of Multi Color Ordering, there are 9 syncronizations, while Cell-like Ordering needs only 3 syncronizations.

Batch #		0	1	2	3	4	5	6	7	8	9	10	11
Number of λs / Batch	Multi Color Ordering	4191	3714	3084	2451	1866	1071	669	342	195	78	48	6
	Cell-like Ordering	12015	5172	525	3	-	-	-	-	-	-	-	-

Result of Figure 4: In case of Multi Color Ordering, there are 12 syncronizations, while Cell-like Ordering needs only 4 syncronizations.

References

[AIMMS07] AIMMS Language Reference. *http://www.aimms.com/aimms/download/manuals/AIMMS3LR_MixedComplementarity.pdf*

[Baraff 89] David Baraff. Analytical method for dynamics simulation of nonpenetrating bodies. *Computer Graphics Vol. 23, No. 3, 223-232, 1989.*

[Catt08] Erin Catto. Modeling and Solving constraints. *GDC2008 Tutorial Note.*

[Catt05] Erin Catt. Iterative Dynamics with Temporal Coherence. *http://www.continuousphysics.com/ftp/pub/test/physics/papers/IterativeDynamics.pdf*

[Chen07] Yen-Kuang Chen et al. High-Performance Physical Simulations on Next-Generation Architecture with Many Cores. *http://www.intel.com/technology/itj/2007/v11i3/8-simulations/4-methodology.htm*

[Cottle92] R.W.Cottle et al. The Linear Complementarity Problem. *Academic Press.*

[Erleb05] Kenny Erleben. Stable, robust, and versatile multibody dynamics animation. *PhD. thesis, Department of Computer Science, University of Copenhagen, Denmark, 2005.*

References

[Guendelman03] Eran Guendelman et al. Nonconvex rigid bodies with stacking. *ACM Tansaction on Graphics, Vol. 22 Issue 3, July 2003.*

[Hahn88] James K. Hahn. Realistic Animation of Rigid Bodies. *Computer Graphics, Vol. 22 Number 4, August 1988.*

[Iwashita01] Takeshi Iwashita et al. Algebraic Multi-Color Ordering Method for Parallelized ICCG Solver in Unstructured Finite Element Analysis. *http://www.tokyo.rist.or.jp/sss2001/Abst/P_Iwashita_Takeshi.pdf*

[Keogh07] Chris Keogh. Physics in Games. *http://boombox.ucs.ed.ac.uk/physicspodcasts/genint/2007/resources/ChrisKeogh.pps*

[Mifune05] T. Mifune et al. A Parallel Algebraic Multigrid Preconditioner Using Algebraic Multicolor Ordering for Magnetic Finite Element Analyses. *http://www.fz-juelich.de/nic-series/volume33/237.pdf*

[Mirtich95] Brian Mirtich. Impulse-based simulation of rigid bodies. *Proceedings of the 1995 symposium on Interactive 3D graphics.*

References

[Moravan04] Adam Moravanszky et al. Fast Contact Reduction for Dynamics Simulation. *Game Programming Gems 4, Charles River Media.*

[Yosui07] Kuniaki Yosui et al. A Parellel Multigrid Solver for High Frequency Electromagnetic Field Analyses with Small-scale PC Cluster. *Electronics and Communication in Japan, Vol. 127 Issue 8, 2007.*