Resolving systems of contacts for computer animation

Steve Marschner
CS 5643 Spring 2025
Cornell University

In previous lectures we worked out how to resolve isolated collisions between particles or between rigid bodies. These methods are great as long as contacts stay separated in time. But once we have simultaneous contacts, either because they are so close in time that it is too expensive to keep resolving them sequentially or because we have resting contacts that persist over time, we need to be smarter about how contacts interact with one another.

Spherical particles without friction

Let's start where we started with contact resolution, with contacts between pairs of spherical particles. Recall that we derived a collision impulse for a pair of colliding spheres A and B, using the simple heuristic (the "restitution hypothesis") that the post-contact normal velocity is opposite the pre-contact normal velocity but scaled down to account for energy loss in the collision:

$$v_1^+ = -c_r v_1^- \text{ where } v_1 = \mathbf{n}_1 \cdot (\mathbf{v}_a - \mathbf{v}_b)$$

I am calling this collision "collision 1" and numbered the velocity and normal appropriately, since there will soon be more collisions to worry about. We resolved the collision by applying impulses $\gamma_1 \hat{\mathbf{n}}_1$ to particle A and $-\gamma_1 \hat{\mathbf{n}}_1$ to particle B, which led to

$$\begin{aligned}\mathbf{v}_a^+ &= \mathbf{v}_a^- + m_a^{-1}\gamma_1 \hat{\mathbf{n}}_1 \\ \mathbf{v}_b^+ &= \mathbf{v}_b^- - m_b^{-1}\gamma_1 \hat{\mathbf{n}}_1\end{aligned}$$

and

$$\begin{aligned} v_1^+ = -c_r v_n^- &= v_1^- + \left(m_a^{-1} + m_b^{-1}\right)\gamma_1 \\ \left(m_a^{-1} + m_b^{-1}\right)\gamma_1 &= -(1+c_r)v_1^- \qquad (1) \\ \gamma_1 &= -(1+c_r)m_{\text{eff}} v_1^- \end{aligned}$$

where $m_{\text{eff}} = \left(m_a^{-1} + m_b^{-1}\right)^{-1}$. But what if these particles also are involved in other simultaneous collisions? Then their velocities are also affected by the impulses related to these other collisions and this derivation doesn't account for this so the results will be wrong.

Resolving a collision in the context of a second collision

Let's start with one other collision. Suppose particle B also is in contact with particle C, and let's further imagine someone has told us the impulse that is applied to B by this collision (call it collision 2). We just have to work out the impulse for collision 1 between A and B.

The impulse $\gamma_2 \hat{\mathbf{n}}_2$ contributes to the final velocity of B:

$$\mathbf{v}_b^+ = \mathbf{v}_b^- - m_b^{-1}\gamma_1\hat{\mathbf{n}}_1 + m_b^{-1}\gamma_2\hat{\mathbf{n}}_2$$

and if we propagate this change through the other equations above we get

$$\begin{aligned}
\mathbf{v}_1^+ &= \mathbf{v}_a^+ - \mathbf{v}_b^+ = \mathbf{v}_1^- + m_a^{-1}\gamma_1\hat{\mathbf{n}}_1 + m_b^{-1}\gamma_1\hat{\mathbf{n}} - m_b^{-1}\gamma_2\hat{\mathbf{n}}_2 \\
v_1^+ &= -c_r v_n^- = \hat{\mathbf{n}}_1 \cdot \mathbf{v}_1^+ = v_1^- + \left(m_a^{-1} + m_b^{-1}\right)\gamma_1 - m_b^{-1}\gamma_2\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2 \\
\left(m_a^{-1} + m_b^{-1}\right)\gamma_1 &= -(1 + c_r)v_1^- + m_b^{-1}\gamma_2\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2 \\
\gamma_1 &= m_{\text{eff}}\left(-(1 + c_r)v_1^- + m_b^{-1}\gamma_2\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2\right)
\end{aligned}$$

That is, the impulse calculation is the same except we add a correction for the effect of the other impulse on B's velocity in the $\hat{\mathbf{n}}_1$ direction.

If there were many other collisions involving B, we could add them to get $\boldsymbol{\gamma}_b$, and we could do the same for A, and use the equation

$$\gamma_1 = m_{\text{eff}}\left(-(1 + c_r)v_1^- - m_a^{-1}\hat{\mathbf{n}}_1 \cdot \boldsymbol{\gamma}_{1a} + m_b^{-1}\hat{\mathbf{n}}_1 \cdot \boldsymbol{\gamma}_{1b}\right)$$

$$\text{where } \boldsymbol{\gamma}_{1a} = \sum_{i \neq 1} s_{ia}\gamma_i\hat{\mathbf{n}}_i \text{ and } \boldsymbol{\gamma}_{1b} = \sum_{i \neq 1} s_{ib}\gamma_i\hat{\mathbf{n}}_i$$

where s_{ix} is $+1$ if X is the first body in collision i (so that $\mathbf{n}_i$ points towards A) and -1 if X is the second body in collision i.

This is great if we know all the other impulses, but of course in practice we start of not knowing any of the impulses. A simple idea that turns out to work is to iterate this process:

```
compute prospective velocities of all objects for this time step
find all collisions, initialize all γs to zero
while not converged:
    for i in range(number of collisions):
        compute γ_i using the current values for all other γ_j, clamping to γ_i ≥ 0
apply all impulses to update velocities of all objects
```

One problem is that this collision response scheme can't recover from an overlap between stationary objects. It will dutifully compute an impulse just sufficient to bring the normal velocity to zero, but that leaves the overlap the same as it was. Small errors can accumulate over time steps and lead to objects sinking slowly into other objects. To fix this, you can add a nudge by adding a little velocity to the goal that is proportional to the overlap:

$$\gamma_1 = m_{\text{eff}}\left(-(1 + c_r)v_1^- + c_d d_1 - m_a^{-1}\hat{\mathbf{n}}_1 \cdot \boldsymbol{\gamma}_{1a} + m_b^{-1}\hat{\mathbf{n}}_1 \cdot \boldsymbol{\gamma}_{1b}\right)$$

where d_1 is the overlap depth of collision 1 and c_d is a coefficient that controls the strength of this correction; setting it too high can cause instability. This will tend to lead to exponential decay towards zero overlap.

Why does it work?

This iterative method to resolve simultaneous collisions seems plausible, and it works pretty well in practice, but can we say something more reassuring about this algorithm — say a proof that it converges?

It turns out that we can package exactly what we have been doing into a problem about matrices. It's a little more abstract but it's worth the effort to understand it!

If we stand back from the process we have been using, it looks like this:

1. Write the new and old normal velocities as a function of the new and old object velocities.

2. Write the objects' new velocities as a function of their old velocities and the collision impulses.

3. Use the restitution hypothesis to write an equation that can be solved for the collision impulses.

So far we have been writing down these equations one object at a time, but using matrix-vector arithmetic we can write them all together as one big system.

For step 1, If we look at the equation that computes the relative velocity in the direction of the collision normal for collision 1, called v_1 above, from the particle velocities it looks like

$$v_1 = \hat{\mathbf{n}}_1 \cdot \mathbf{v}_a - \hat{\mathbf{n}}_1 \cdot \mathbf{v}_b = \begin{bmatrix} \cdots & \hat{\mathbf{n}}_1^T & \cdots & -\hat{\mathbf{n}}_1^T & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ \mathbf{v}_a \\ \vdots \\ \mathbf{v}_b \\ \vdots \end{bmatrix} = \mathbf{J}_1 \mathbf{v}$$

Here both the row vector $\mathbf{J}_1$ and the column vector $\mathbf{v}$ are of length $2N$ (for a 2D system) where N is the total number of particles. The hidden entries of $\mathbf{J}_1$ are zero, and $\mathbf{v}$ is simply a list of all the velocities in the system. The $\hat{\mathbf{n}}_1$ and $-\hat{\mathbf{n}}_1$ are positioned in $\mathbf{J}_1$ so that they multiply with $\mathbf{v}_a$ and $\mathbf{v}_b$ to produce the formula we want. If we have several collisions to worry about, we can compute them all at once by stacking similar vectors for several collisions into a $k \times N$ matrix $\mathbf{J}$ that has one row for each of the k collisions in the system. Then $\mathbf{v}_n = \mathbf{J}\mathbf{v}$ where $\mathbf{v}_n$ is a vector gathering the normal velocities for all the collisions.

The name $\mathbf{J}$ stands for "Jacobian" — it is the derivative of the mapping from the particle velocities we keep in the system state to the normal-direction relative velocities we need to reason about collisions.

This formula works both before and after the collision:

$$\mathbf{v}_n^- = \mathbf{J}\mathbf{v}^-$$
$$\mathbf{v}_n^+ = \mathbf{J}\mathbf{v}^+$$

For step 2, the formulas we used to compute the post-collision velocities were

$$\mathbf{v}_a^+ = \mathbf{v}_a^- + m_a^{-1}\gamma_1\hat{\mathbf{n}}_1$$
$$\mathbf{v}_b^+ = \mathbf{v}_b^- - m_b^{-1}\gamma_1\hat{\mathbf{n}}_1$$

If you look at this for a minute you'll see the same vector $\mathbf{J}_1$ hidden in there. If we write down the effect of impulse 1 on all the velocities in the system we get

$$\begin{bmatrix} \vdots \\ \mathbf{v}_a^+ \\ \vdots \\ \mathbf{v}_b^+ \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \mathbf{v}_a^- \\ \vdots \\ \mathbf{v}_b^- \\ \vdots \end{bmatrix} + \begin{bmatrix} \vdots \\ m_a^{-1}\hat{\mathbf{n}}_1 \\ \vdots \\ -m_b^{-1}\hat{\mathbf{n}}_1 \\ \vdots \end{bmatrix} \gamma_1$$

$$\mathbf{v}^+ = \mathbf{v}^- + \mathbf{M}^{-1}\mathbf{J}_1^T\gamma_1$$

where $\mathbf{M}$ is the mass matrix of the system, which just has the masses of the particles on the diagonal:

$$\mathbf{M} = \begin{bmatrix} m_1 & 0 & \cdots & 0 & 0 \\ 0 & m_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & m_N & 0 \\ 0 & 0 & \cdots & 0 & m_N \end{bmatrix}$$

Again we can use the matrix multiplication machinery to handle all the collisions at once:

$$\begin{aligned} \mathbf{v}^+ &= \mathbf{v}^- + \mathbf{M}^{-1}\mathbf{J}_1^T\gamma_1 + \cdots + \mathbf{M}^{-1}\mathbf{J}_k^T\gamma_k \\ &= \mathbf{v}^- + \mathbf{M}^{-1}\mathbf{J}^T\boldsymbol{\gamma} \end{aligned}$$

Here k is the number of collisions and $\boldsymbol{\gamma}$ is a k-vector that has the magnitudes of the collision impulses in its entries. So that summarizes the effect of all the simultaneous collision impulses on all the velocities in the system.

For step 3, it's now a simple algebraic substitution to write the pre-and post-collision normal velocities:

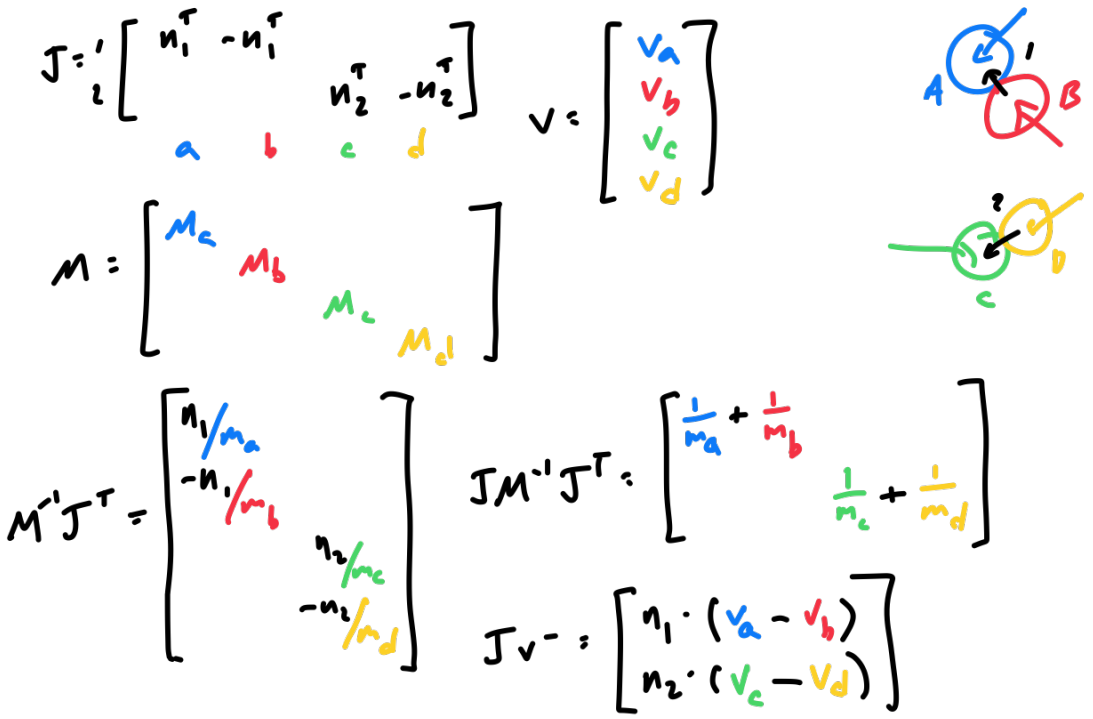
$$\mathbf{v}_n^- = \mathbf{J}\mathbf{v}^-$$
$$\mathbf{v}_n^+ = \mathbf{J}\mathbf{v}^+ = \mathbf{J}\mathbf{v}^- + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\boldsymbol{\gamma}$$

Our restitution hypothesis applies to all the collisions so it reads $\mathbf{v}_n^+ = -c_r\mathbf{v}_n^-$, which expands to

$$\begin{aligned} \mathbf{J}\mathbf{v}^- + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\boldsymbol{\gamma} &= -c_r\mathbf{J}\mathbf{v}^- \\ \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\boldsymbol{\gamma} &= -(1+c_r)\mathbf{J}\mathbf{v}^- \qquad (2) \\ \mathbf{A}\boldsymbol{\gamma} &= \mathbf{b} \end{aligned}$$

This is a square $k \times k$ linear system that can be solved for $\boldsymbol{\gamma}$! This is great because it implies we have a single, easy-to-find solution, as long as this matrix does not turn out to be singular. (Spoiler alert: it will turn out not to be quite this simple.)

Let's try this out for a case where collisions are independent. Suppose we have a situation with two pairs of particles colliding separately. Then there are four velocities in the system and $\mathbf{J}$ is a 2×8 matrix, which I'll write in four columns containing 2D vectors.



Since $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ is diagonal, the solution to this system (2) is easy to compute:

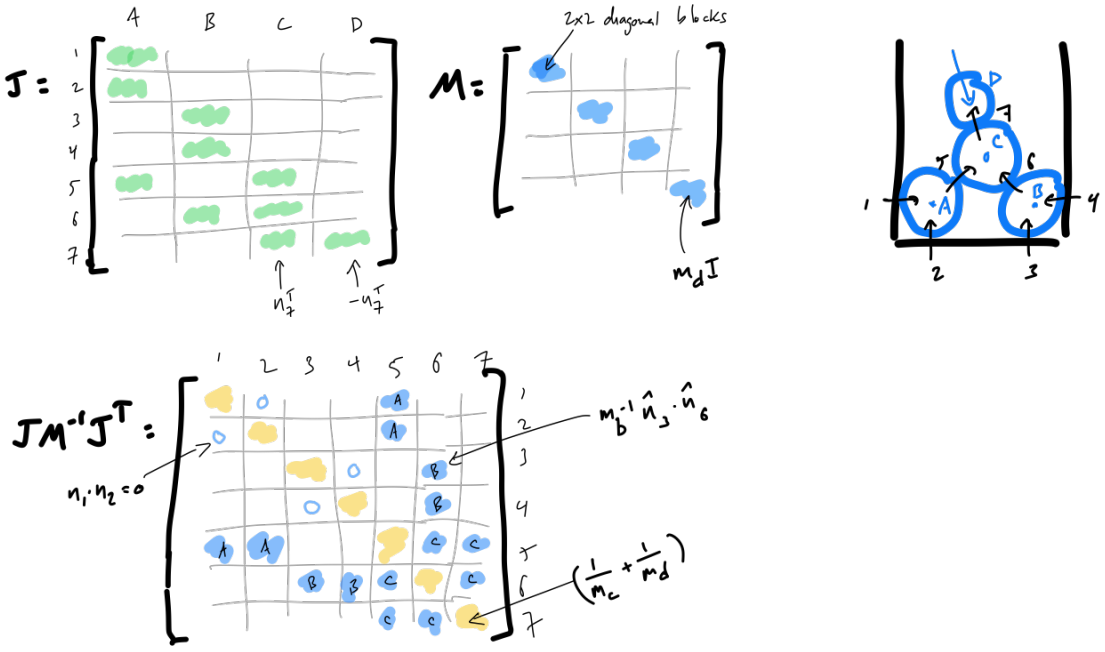
$$\gamma_1 = -(1+c_r)\,\hat{\mathbf{n}}_1 \cdot (\mathbf{v}_a^- - \mathbf{v}_b^-) \Big/ \left(m_a^{-1} + m_b^{-1}\right)$$

$$\gamma_2 = -(1+c_r)\,\hat{\mathbf{n}}_2 \cdot (\mathbf{v}_c^- - \mathbf{v}_d^-) \Big/ \left(m_c^{-1} + m_d^{-1}\right)$$

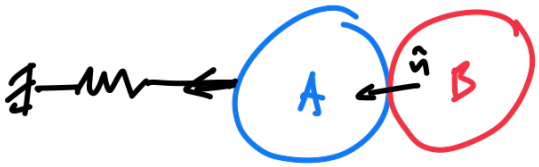
This is no surprise, since it matches what we earlier derived in (1).

The diagonal structure of $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ reflects the fact that the two contacts in this example involve non-overlapping pairs of objects. If some objects are in contact with multiple other objects, then off-diagonal entries will appear in this matrix. Since it takes a little larger example to illustrate this, I'll switch to mainly illustrating only the sparsity structure of these matrices (i.e. which entries are nonzero). Below is an example with 7 objects plus some fixed walls. Note that an object that is involved in collisions i and j is responsible for contributions to the (i, j) entry of $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$. Objects involved in only one collision contribute only to the diagonal entries, and an object involved in n collision contributes to n^2 entries in this matrix. In this example, the boundaries are treated as collisions involving only one object. Since $\mathbf{n}_i \cdot \mathbf{n}_j = \mathbf{n}_j \cdot \mathbf{n}_i$, the system matrix is symmetric.

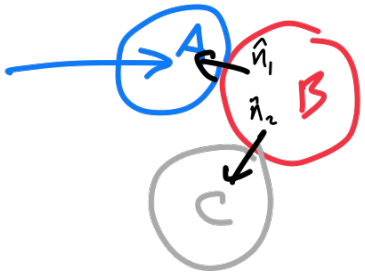
To solve this system we could use any linear solver, though we will want to choose one that is efficient for relatively large, sparse matrices; this tends to push us towards iterative methods for solving the system.



There is one problem with this formulation so far. Suppose we have a situation like the one shown here:



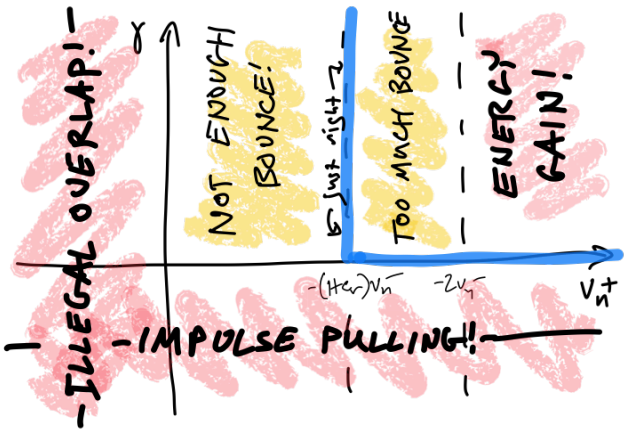
There are two bodies at rest and in contact, and A experiences a force that pulls it away from B. If we go ahead and solve for the collision impulse we will get a value $\gamma < 0$, which dutifully ensures that the relative normal velocity remains zero but inappropriately pulls B along with A. For an isolated collision like this we can solve the problem by first computing γ and then clamping it to ensure $\gamma \geq 0$. But in a more complicated setting this will not work. Here is an example with two collisions:



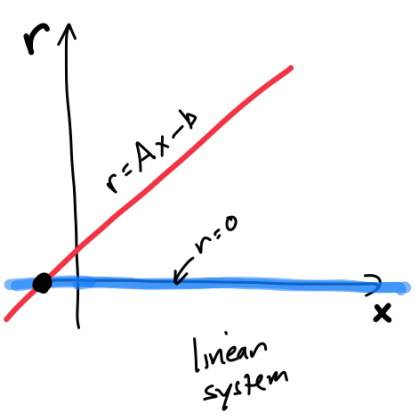
If we compute γ_1 and γ_2, then set $\gamma_2 = 0$, γ_1 by itself will no longer produce the correct relative velocity for collision 1. This $\gamma > 0$ constraint needs to be part of the system, not something we impose after solving it. So to restate the problem we actually want to solve: we want to find $\boldsymbol{\gamma}$ such that $\mathbf{A}\boldsymbol{\gamma} = \mathbf{b}$; each row of this equation says that the normal velocity for one collision has the value given by the restitution hypothesis. But actually it's OK to for one of the velocities v_i^+ to be too positive, as long as the corresponding impulse γ_i is zero. We can state this formally by saying we want to solve two complementary inequalities:

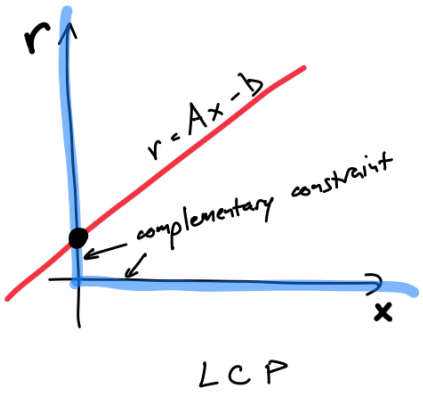
$$\mathbf{A}\boldsymbol{\gamma} - \mathbf{b} \geq 0 \quad \text{and} \quad \boldsymbol{\gamma} \geq 0 \quad \text{and} \quad (\mathbf{A}\boldsymbol{\gamma} - \mathbf{b}) \cdot \boldsymbol{\gamma} = 0$$

That is, the excess normal velocity must be nonnegative; the impulse must be nonnegative; but they can't both be positive, at least one has to be zero. This means each contact has two "modes:" when $\gamma_i > 0$ then the whole system of impulses has to ensure that $v_i^+ = -(1 + c_r)v_i^-$; on the other hand, when $v_i^+ > -(1 + c_r)v_i^-$ then $\gamma_i = 0$. It's not allowed for both to be positive at once, because then impulses could push arbitrarily hard, producing too much energy. It's not allowed for impulses to be negative because contact forces can't pull objects together. It's not allowed for post-collision velocities to be below the target because that would lose energy that is supposed to be kept, or if the post-collision velocity is negative that means objects will become (more) overlapping.



This type of problem is called a "linear complementarity problem" or LCP because it has two quantities that are related linearly, and the solution is defined by complementary constraints. This idea is both very intuitive and at the same time frustratingly tricky to express intuitively; LCPs are also both simple to formulate and at the same time sometimes tricky to solve numerically. The following picture can be helpful:





A regular linear system $\mathbf{Ax} = \mathbf{b}$ could be thought of as a pair of constraints on the unknown vector $\mathbf{x}$ and the residual $\mathbf{r}$. One is $\mathbf{r} = \mathbf{Ax} - \mathbf{b}$ and the other is $\mathbf{r} = \mathbf{0}$. In the LCP the second constraint is illustrated by the L-shaped set in the figure. Even though it's formally written with inequalities this is not an inequality constraint; it's just an odd-shaped equality constraint.

Now we need a method for solving this problem. There are a number of methods available, and one of the simplest is a small tweak to the Gauss-Seidel algorithm, which is a basic iterative method for solving linear equations.

The Gauss-Seidel algorithm considers the equations, or rows, of a linear system in sequence. It solves the *i*th equation for the *i*th variable, acting as if the other variables are known:

$$\sum_{j=0}^{N} a_{ij} x_j = b_i$$

$$\sum_{j=0}^{i-1} a_{ij} x_j + a_{ii} x_i + \sum_{j=i+1}^{N} a_{ij} x_j = b_i$$

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=0}^{i-1} a_{ij} x_j - \sum_{j=i+1}^{N} a_{ij} x_j \right)$$

So one step of Gauss-Seidel uses this formula, using the most recently updated values of x_j. One iteration of G-S is a full pass through all the variables in sequence, with each one computed using the newest values of all the previous ones. The G-S algorithm is guaranteed to converge for symmetric, positive-definite matrices, which ours is if there are no redundant collisions (and the algorithm tends to tolerate redundancies well).

When we apply this method with $\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$, $\mathbf{b} = -(1 - c_r)\mathbf{J}^T\mathbf{v}^-$, and $\mathbf{x} = \boldsymbol{\gamma}$, it expands to:

$$\gamma_i = m_{\text{eff}} \left(-(1 - c_r) v_j^- - m_a^{-1} \sum_{k \neq i} s_{ak} \gamma_k \hat{\mathbf{n}}_k \cdot \hat{\mathbf{n}}_j + m_b^{-1} \sum_{k \neq i} s_{bk} \gamma_k \hat{\mathbf{n}}_k \cdot \hat{\mathbf{n}}_j \right)$$

where we are naming the two objects in collision i as A and B, and as before, s_{xk} is +1 if X is the first object in collision k, −1 if X is the second object in collision k, and 0 if X is not involved in collision k. This is exactly the same procedure we came up with above by reasoning about the velocities and impulses affecting the two objects in a collision, which is reassuring!

To solve an LCP instead of a normal linear system, you can use a variant called "Projected Gauss-Seidel" which can be used to solve linear systems with constraints by simply clamping each newly computed value to the nearest feasible value. In our setting this simply means clamping impulses to zero at each step.

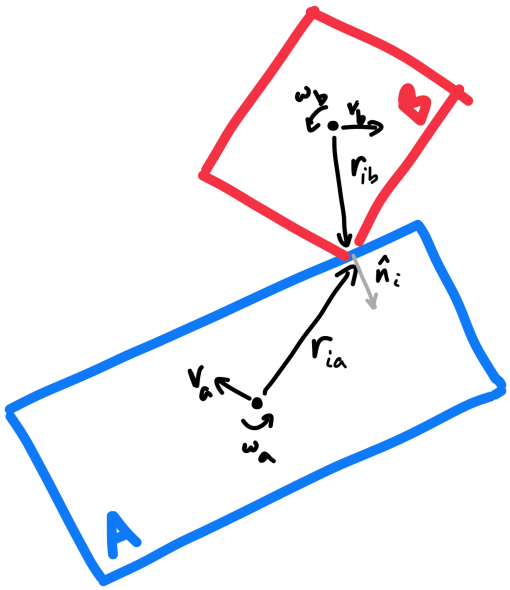
So what was the point of doing all this with the matrices when we arrived at the same algorithm before by just thinking about the problem?

1. It gives us the theoretical results about the Projected Gauss-Seidel algorithm to stand on in believing that our method actually converges to a meaningful solution.

2. It is maybe a bit less error-prone since it encodes a lot of the complexity of the formulas into the machinery of matrix multiplication, and this will be even more helpful when we get to the rigid body case.

3. It lets you read papers about contact processing, which are usually written in this language.

Rigid bodies



For rigid bodies, we derived the equations for a pair of colliding bodies back in the rigid body lecture notes. The steps were writing the collision's normal velocity in terms of the body velocities and collision geometry:

$$v_i = \hat{\mathbf{n}}_i \cdot \mathbf{v}_{\text{rel}} = \hat{\mathbf{n}} \cdot \left(\mathbf{v}_a - \mathbf{v}_b + \omega_a \times \mathbf{r}_a - \omega_b \times \mathbf{r}_b\right),$$

writing the objects' new velocities in terms of the collision impulse:

$$\Delta\mathbf{v}_a = m_a^{-1}\gamma_i\hat{\mathbf{n}}_i \qquad \Delta\omega_a = I_a^{-1}\mathbf{r}_{ia} \times \gamma_i\hat{\mathbf{n}}_i$$

$$\Delta\mathbf{v}_b = -m_b^{-1}\gamma_i\hat{\mathbf{n}}_i \qquad \Delta\omega_b = -I_b^{-1}\mathbf{r}_{ib} \times \gamma_i\hat{\mathbf{n}}_i,$$

and solving for the impulse magnitude:

$$\gamma_i = -(1 + c_r)m_{\text{eff},i}v_i^-$$

$$m_{\text{eff},i} = \left(m_a^{-1} + m_b^{-1} + I_a^{-1}\hat{\mathbf{n}} \cdot (\mathbf{r}_{ia} \times \hat{\mathbf{n}}_i) \times \mathbf{r}_{ia} + I_b^{-1}\hat{\mathbf{n}} \cdot (\mathbf{r}_{ib} \times \hat{\mathbf{n}}_i) \times \mathbf{r}_{ib}\right)^{-1}.$$

Here I have introduced the index i for all the quantities pertaining to the collision between A and B, in preparation for generalizing things to account for other collisions. It's still implied that A and B are the names of the objects involved in collision i, but we only have to write down the details of one collision at a time so this will be OK.

If A and B are also involved in other collisions, and we assume the impulses for those other collisions are already known, then we just have to modify the change-of-velocity formulas to include these other impulses:

$$\Delta\mathbf{v}_a = m_a^{-1}\gamma_i\hat{\mathbf{n}}_i + m_a^{-1}\sum_{j\neq i} s_{ja}\gamma_j\hat{\mathbf{n}}_j \qquad \Delta\omega_a = I_a^{-1}\mathbf{r}_{ia} \times \gamma_i\hat{\mathbf{n}}_i + I_a^{-1}\sum_{j\neq i} s_{ja}\mathbf{r}_{ja} \times \gamma_j\hat{\mathbf{n}}_j$$

$$\Delta\mathbf{v}_b = -m_b^{-1}\gamma_i\hat{\mathbf{n}}_i + m_b^{-1}\sum_{j\neq i} s_{jb}\gamma_j\hat{\mathbf{n}}_j \qquad \Delta\omega_b = -I_b^{-1}\mathbf{r}_{ib} \times \gamma_i\hat{\mathbf{n}}_i + I_b^{-1}\sum_{j\neq i} s_{jb}\mathbf{r}_{jb} \times \gamma_j\hat{\mathbf{n}}_j$$

The effect of these changes on the relative velocity looks like

$$\begin{aligned}\mathbf{v}_{\text{rel}}^{+} &= \mathbf{v}_{\text{rel}}^{-} + \left(\Delta\mathbf{v}_a + \Delta\omega_a \times \mathbf{r}_{ia}\right) - \left(\Delta\mathbf{v}_b + \Delta\omega_b \times \mathbf{r}_{ib}\right) \\ &= \mathbf{v}_{\text{rel}}^{-} + \left(m_a^{-1}\hat{\mathbf{n}}_i + m_b^{-1}\hat{\mathbf{n}}_i + I_a^{-1}(\mathbf{r}_{ia} \times \hat{\mathbf{n}}_i) \times \mathbf{r}_{ia} + I_b^{-1}(\mathbf{r}_{ib} \times \hat{\mathbf{n}}_i) \times \mathbf{r}_{ib}\right)\gamma_i + \\ &\qquad \Delta\mathbf{v}_a^{\text{other}} - \Delta\mathbf{v}_b^{\text{other}} + \Delta\omega_a^{\text{other}} \times \mathbf{r}_{ia} - \Delta\omega_b^{\text{other}} \times \mathbf{r}_{ib}\end{aligned}$$

where $\Delta\mathbf{v}_a^{\text{other}} = m_a^{-1} \sum_{j \neq i} s_{ja}\gamma_j\hat{\mathbf{n}}_j$ and $\Delta\omega_a^{\text{other}} = I_a^{-1} \sum_{j \neq i} s_{ja}\mathbf{r}_{ja} \times \gamma_j\hat{\mathbf{n}}_j$ and likewise for B.

When we propagate these on into the computation of the post-collision normal velocity we get

$$\begin{aligned}v_i^{+} &= \hat{\mathbf{n}}_i \cdot \mathbf{v}_{\text{rel}}^{+} \\ &= v_i^{-} + m_{\text{eff},i}^{-1}\gamma_i + \hat{\mathbf{n}} \cdot \left(\Delta\mathbf{v}_a^{\text{other}} - \Delta\mathbf{v}_b^{\text{other}} + \Delta\omega_a^{\text{other}} \times \mathbf{r}_{ia} - \Delta\omega_b^{\text{other}} \times \mathbf{r}_{ib}\right)\end{aligned}$$

and solving for γ_i (remembering that all the γ_j involved in the last term are assumed known):

$$\gamma_i = -\, m_{\text{eff},i}\left[(1 + c_r)v_i^{-} + \hat{\mathbf{n}} \cdot \left(\Delta\mathbf{v}_a^{\text{other}} - \Delta\mathbf{v}_b^{\text{other}} + \Delta\omega_a^{\text{other}} \times \mathbf{r}_{ia} - \Delta\omega_b^{\text{other}} \times \mathbf{r}_{ib}\right)\right]$$

This is more complex than in the particle case, but it's quite analogous. We can readily chase down all the terms in this formula into things we can easily evaluate if we know all the collisions in the system.

This equation is to resolve collision i assuming we have already resolved all collisions j where $j \neq i$. But the iteration we used for particles also applies in this case. Generalizing it a bit we get:

> compute prospective velocities and angular velocities of all bodies for this time step
> find all collisions, set all γ_i to zero
> while not converged:
> for i in range(number of collisions):
> compute γ_i using the current values for all other γ_j, clamping to $\gamma_i \geq 0$
> apply all impulses to update velocities and angular velocities of all bodies

As with particles we will want an overlap-repair impulse that pushes overlapping objects gently apart, so that errors don't persist and accumulate over time:

$$\gamma_i = -\, m_{\text{eff}}\left[(1 + c_r)v_i^{-} - c_d d_i + \hat{\mathbf{n}} \cdot \left(\Delta\mathbf{v}_a^{\text{other}} - \Delta\mathbf{v}_b^{\text{other}} + \Delta\omega_a^{\text{other}} \times \mathbf{r}_{ia} - \Delta\omega_b^{\text{other}} \times \mathbf{r}_{ib}\right)\right]$$

where d_i is the penetration depth of collision i, measured so that overlapping objects produce positive depths.

As with the particles, this iteration can be interpreted as a Projected Gauss-Seidel iteration applied to solve a linear complementarity problem, and as such it is guaranteed to converge, in theory, under suitable conditions. In practice this iteration will work pretty well for reasonably friendly collision situations. Some things that count as unfriendly include heavy objects stacked on top of light ones and tall stacks or deep piles of objects. The characteristic artifact of non-convergence is objects sinking into one another fast enough that the overlap-repair impulse can't keep them well separated.

Many refinements to this method exist and methods in this general family are the basis of collision response in most production rigid body simulators.

A bit about implementation

When implementing iterative collision response, there are a number of considerations that come up and some implementation choices can make things simpler and faster.

One issue is with computing the sums over related collisions that are required to form $\boldsymbol{\gamma}_x$ for particles or $\Delta\mathbf{v}_x^{\text{other}}$ and $\Delta\omega_x^{\text{other}}$ for rigid bodies. Computing these for collision i requires finding all the collisions that involve the objects involved in collision i, and if you do this by searching all collisions then the collision response will be quadratic in the number of collisions which will be too slow when objects pile up. One option is to build a data structure that lets you find the related collisions; you can think of the collision system as a graph with objects as nodes and collisions as edges and use your favorite data structure to navigate this graph. But this turns out to be unnecessary, because the effects on each object are quite related.

If we look at the quantity that needs to be computed for particles, it looks like

$$m_a^{-1}\boldsymbol{\gamma}_{ia} - m_b^{-1}\boldsymbol{\gamma}_{ib} \text{ where } \boldsymbol{\gamma}_{ix} = \sum_{j\neq i} s_{jx}\gamma_j\hat{\mathbf{n}}_j$$

But the $\boldsymbol{\gamma}$s for the same object in different collisions only differ by which collision is left out of the sum. We could rewrite it as

$$\begin{aligned}\boldsymbol{\gamma}_{ix} &= \sum_j s_{jx}\gamma_j\hat{\mathbf{n}}_j - s_{ix}\gamma_i\hat{\mathbf{n}}_i \\ &= \boldsymbol{\gamma}_x - s_{ix}\gamma_i\hat{\mathbf{n}}_i\end{aligned}$$

Expanding the difference above noting that $s_{ia} = 1$ and $s_{ib} = -1$,

$$m_a^{-1}\boldsymbol{\gamma}_{ia} - m_b^{-1}\boldsymbol{\gamma}_{ib} = m_a^{-1}\boldsymbol{\gamma}_a - m_b^{-1}\boldsymbol{\gamma}_b - (m_a^{-1} + m_b^{-1})\gamma_i\hat{\mathbf{n}}_i$$

Now that $\boldsymbol{\gamma}_x$ is a sum is over all collisions we can compute it ahead of time and just store $\boldsymbol{\gamma}_x$ for each object X. When it is time to compute the impulse for collision i we need to subtract off the current impulse for collision i:

$$\begin{aligned}\gamma_i^{\text{new}} &= m_{\text{eff}}\left(-(1+c_r)v_i^- - \hat{\mathbf{n}}_i\cdot(m_a^{-1}\boldsymbol{\gamma}_{ia} + m_b^{-1}\boldsymbol{\gamma}_{ib})\right) \\ &= m_{\text{eff}}\left(-(1+c_r)v_i^- - \hat{\mathbf{n}}_i\cdot(m_a^{-1}\boldsymbol{\gamma}_a - m_b^{-1}\boldsymbol{\gamma}_b - (m_a^{-1}+m_b^{-1})\gamma_i^{\text{old}}\hat{\mathbf{n}}_i)\right) \\ &= m_{\text{eff}}\left(-(1+c_r)v_i^- - \hat{\mathbf{n}}_i\cdot(m_a^{-1}\boldsymbol{\gamma}_a - m_b^{-1}\boldsymbol{\gamma}_b) + m_{\text{eff}}^{-1}\gamma_i^{\text{old}}\right) \\ &= \gamma_i^{\text{old}} + m_{\text{eff}}\left(-(1+c_r)v_i^- - \hat{\mathbf{n}}_i\cdot(m_a^{-1}\boldsymbol{\gamma}_a - m_b^{-1}\boldsymbol{\gamma}_b)\right) \\ \Delta\gamma_i &= m_{\text{eff}}\left(-(1+c_r)v_i^- - \hat{\mathbf{n}}_i\cdot(m_a^{-1}\boldsymbol{\gamma}_a - m_b^{-1}\boldsymbol{\gamma}_b)\right)\end{aligned}$$

So this boils down to: go ahead and let the impulse for collision i be included as an "other collision" when computing collision i, and then what you get is an update to add to γ_i.

For the G-S iteration to work, after computing the update $\Delta\lambda_i$ and adding it to λ_i it's important to remember to update $\boldsymbol{\gamma}_a$ and $\boldsymbol{\gamma}_b$ for the objects A and B involved in a collision by adding $s_{ix}\Delta\gamma_i\hat{\mathbf{n}}_i$ so that future impulse computations will be using the the new value of γ_i.