Constrained Dynamics

Andrew Witkin

PIXAR
Animation Studios
Beyond Points and Springs

• You can make just about anything out of point masses and springs, *in principle*.

• In practice, you can make anything you want as long as it’s jello.

• Constraints will buy us:
  – Rigid links instead of goopy springs.
  – Ways to make interesting contraptions.
A bead on a wire

• Desired Behavior:
  – The bead can slide freely along the circle.
  – It can never come off, however hard we pull.

• Question:
  – How does the bead move under applied forces?
Penalty Constraints

- Why not use a spring to hold the bead on the wire?

- Problem:
  - Weak springs ⇒ goopy constraints
  - Strong springs ⇒ neptune express!

- A classic stiff system.
The basic trick ($f = mv$ version)

- 1st order world.
- **Legal velocity**: tangent to circle ($N \cdot v = 0$).
- *Project* applied force $f$ onto tangent: $f' = f + f_c$
- Added normal-direction force $f_c$: *constraint force*.
- No tug-of-war, no stiffness.

\[
f_c = -\frac{f \cdot N}{N \cdot N} N \quad f' = f + f_c
\]
\[ f = ma \]

- Same idea, but...
- *Curvature* (\( \kappa \)) has to match.
- \( \kappa \) depends on both \( a \) and \( v \):
  - the faster you’re going, the faster you have to turn.
- Calculate \( f_c \) to yield a legal *combination* of \( a \) and \( v \).
- Blechh!
Now for the Algebra ...

- Fortunately, there’s a general recipe for calculating the constraint force.
- First, a single constrained particle.
- Then, generalize to constrained particle systems.
Representing Constraints

I. Implicit:
\[ C(x) = |x| - r = 0 \]

II. Parametric:
\[ x = r[\cos \theta, \sin \theta] \]
Maintaining Constraints Differentially

- Start with legal position and velocity.
- Use constraint forces to ensure legal curvature.

\[ C = 0 \quad \text{legal position} \]
\[ \dot{C} = 0 \quad \text{legal velocity} \]
\[ \ddot{C} = 0 \quad \text{legal curvature} \]
Constraint Gradient

\[ N = \frac{\partial C}{\partial x} \]

**Implicit:**

\[ C(x) = |x| - r = 0 \]

Differentiating \( C \) gives a normal vector. This is the direction our constraint force will point in.
Constraint Forces

Constraint force: gradient vector times a scalar, $\lambda$.

Just one unknown to solve for.

Assumption: constraint is passive—no energy gain or loss.

$$f_c = \lambda N$$
Constraint Force Derivation

\[ C(x(t)) \]

\[ \dot{C} = N \cdot \dot{x} \]

\[ \ddot{C} = \frac{\partial}{\partial t} \left[ N \cdot \dot{x} \right] \]

\[ = \dot{N} \cdot \dot{x} + N \dddot{x} \]

\[ f_c = \lambda N \]

\[ \dddot{x} = \frac{\dot{f} + f_c}{m} \]

Set \( \ddot{C} = 0 \), solve for \( \lambda \):

\[ \lambda = -m \frac{\dot{N} \cdot \dot{x} - N \cdot f}{N \cdot N} \]

Constraint force is \( \lambda N \).
Example: Point-on-circle

Write down the constraint equation.

Take the derivatives.

Substitute into generic template, simplify.

\[ C = \left| \mathbf{x} \right| - r \]
\[ \mathbf{N} = \frac{\partial C}{\partial \mathbf{x}} = \frac{\mathbf{x}}{\left| \mathbf{x} \right|} \]
\[ \dot{\mathbf{N}} = \frac{\partial^2 C}{\partial \mathbf{x} \partial t} = \frac{1}{\mathbf{x}} \left[ \dot{\mathbf{x}} - \frac{\mathbf{x} \cdot \dot{\mathbf{x}}}{\mathbf{x} \cdot \mathbf{x}} \right] \]
\[ \mathbf{\lambda} = -m \frac{\dot{\mathbf{N}} \cdot \dot{\mathbf{x}}}{\mathbf{N} \cdot \mathbf{N}} - \frac{\mathbf{N} \cdot \mathbf{f}}{\mathbf{N} \cdot \mathbf{N}} = \left[ m \frac{\left( \mathbf{x} \cdot \dot{\mathbf{x}} \right)^2}{\mathbf{x} \cdot \mathbf{x}} - m \left( \mathbf{x} \cdot \dot{\mathbf{x}} \right) - \mathbf{x} \cdot \mathbf{f} \right] \frac{1}{\left| \mathbf{x} \right|} \]
Drift and Feedback

- In principle, clamping $C$ at zero is enough.

- Two problems:
  - Constraints might not be met initially.
  - Numerical errors can accumulate.

- A feedback term handles both problems:

  $\ddot{C} = -\alpha C - \beta \dot{C}$, instead of $\dot{C} = 0$

  $\alpha$ and $\beta$ are magic constants.
Now we know how to simulate a bead on a wire.

Next: a constrained particle system.
  - E.g. constrain particle/particle distance to make rigid links.

Same idea, but...
Constrained particle systems

- Particle system: a point in state space.
- Multiple constraints:
  - each is a function $C_i(x_1, x_2, \ldots)$
  - Legal state: $C_i = 0$, $\forall$ i.
  - Simultaneous projection.
  - Constraint force: linear combination of constraint gradients.
- Matrix equation.
Compact Particle System Notation

\[ \dot{q} = WQ \]

**q:** 3n-long *state vector.*

**Q:** 3n-long *force vector.*

**M:** 3n x 3n diagonal *mass matrix.*

**W:** M-inverse (element-wise reciprocal)

\[ q = [x_1, x_2, \ldots, x_n] \]

\[ Q = [f_1, f_2, \ldots, f_n] \]

\[ M = \begin{bmatrix} m_1 & & \\ & m_1 & \\ & & m_n \end{bmatrix} \]

\[ W = M^{-1} \]
Particle System Constraint Equations

Matrix equation for $\lambda$

$$ [JWJ^T]\lambda = -\dot{J}\dot{q} - [JW]Q $$

Constrained Acceleration

$$ \ddot{q} = W[Q + J^T\lambda] $$

More Notation

- $C = [C_1, C_2, \ldots, C_m]$
- $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_m]$
- $J = \frac{\partial C}{\partial q}$
- $\dot{J} = \frac{\partial^2 C}{\partial q \partial t}$

Derivation: just like bead-on-wire.
How do you implement all this?

• We have a global matrix equation.
• We want to build models on the fly, just like masses and springs.
• Approach:
  – Each constraint adds its own piece to the equation.
Matrix Block Structure

- Each constraint contributes one or more blocks to the matrix.
- Sparsity: many empty blocks.
- Modularity: let each constraint compute its own blocks.
- Constraint and particle indices determine block locations.
Global Stuff

Global and Local

Constraint
Each constraint must know how to compute these:

$$C = \|\mathbf{x}_1 - \mathbf{x}_2\| - r$$

Distance Constraint

Constraint Structure

$$\begin{align*}
\frac{\partial C}{\partial \mathbf{x}_1} & \quad \frac{\partial C}{\partial \mathbf{x}_2} \\
\frac{\partial^2 C}{\partial \mathbf{x}_1 \partial t} & \quad \frac{\partial^2 C}{\partial \mathbf{x}_2 \partial t}
\end{align*}$$
Constrained Particle Systems

particles \( n \) time forces \( n\text{forces} \) consts \( n\text{consts} \)

\( \begin{align*}
\mathbf{x} & \quad \mathbf{x} \\
\mathbf{v} & \quad \mathbf{v} \\
\mathbf{f} & \quad \mathbf{f} \\
\mathbf{m} & \quad \mathbf{m}
\end{align*} \)
Modified Deriv Eval Loop

1. Clear Force Accumulators
2. Apply forces
3. Compute and apply Constraint Forces
4. Return to solver

Added Step

Computing and applying forces is an added step in the modified derivative evaluation loop.
**Constraint Force Eval**

- After computing ordinary forces:
  - Loop over constraints, assemble global matrices and vectors.
  - Call matrix solver to get $\lambda$, multiply by $J^T$ to get constraint force.
  - Add constraint force to particle force accumulators.
Impress your Friends

- The requirement that constraints not add or remove energy is called the \textit{Principle of Virtual Work}.

- The $\lambda$’s are called \textit{Lagrange Multipliers}.

- The derivative matrix, $J$, is called the \textit{Jacobian Matrix}. 

A whole other way to do it.

**I. Implicit:**

\[ C(x) = |x| - r = 0 \]

**II. Parametric:**

\[ x = r \left[ \cos \theta, \sin \theta \right] \]
Parametric Constraints

\[ x = r \begin{bmatrix} \cos \theta, \sin \theta \end{bmatrix} \]

- Constraint is always met exactly.
- One DOF: \( \theta \).
- Solve for \( \ddot{\theta} \).
Parametric bead-on-wire ($f = mv$)

- $x$ is not an independent variable.
- First step—get rid of it:
  \[
  \dot{x} = \frac{f + f_c}{m} \quad f = mv \text{ (constrained)}
  \]
  \[
  \dot{x} = T \dot{\theta} \quad \text{chain rule}
  \]
  \[
  T \dot{\theta} = \frac{f + f_c}{m} \quad \text{combine}
  \]

\[
T = \frac{\partial x}{\partial \theta}
\]
For our next trick...

As before, assume \( f_c \) points in the normal direction, so

\[
T \cdot f_c = 0
\]

We can nuke \( f_c \) by dotting \( T \) into both sides:

\[
T \dot{\theta} = \frac{f + f_c}{m}
\]

from last slide

\[
T \cdot T \dot{\theta} = \frac{T \cdot f + T \cdot f_c}{m}
\]

blam!

\[
\theta = \frac{1}{m} \frac{T \cdot f}{T \cdot T}
\]

take

from last slide

rearrange.
Parametric Constraints: Summary

- Generalizations: \( f = ma \), particle systems
  - Like implicit case (see notes.)
- Big advantages:
  - Fewer DOF’s.
  - Constraints are always met.
- Big disadvantages:
  - Hard to formulate constraints.
  - No easy way to combine constraints.
- Official name: Lagrangian dynamics.
Things to try at home:

- A bead on a wire (implicit, parametric)
- A double pendulum.
- A triple pendulum.
- Simple interactive tinkertoys.