Symplectic Integration

and

Cosntraints

Adrien Treuille

Symplectic

- **Consider the system:**

$$\dot{\mathrm{x}} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathrm{x}$$

- **or equivalently:**

$$\dot{x} = -y$$
$$\dot{y} = x$$

- **We want to solve x *explicitly* and y *implicitly*:**

$$x_{i+1} = x_i - h \cdot y_i \qquad \textit{(explicit)}$$
$$y_{i+1} = y_i + h \cdot x_{i+1} \qquad \textit{(implicit)}$$

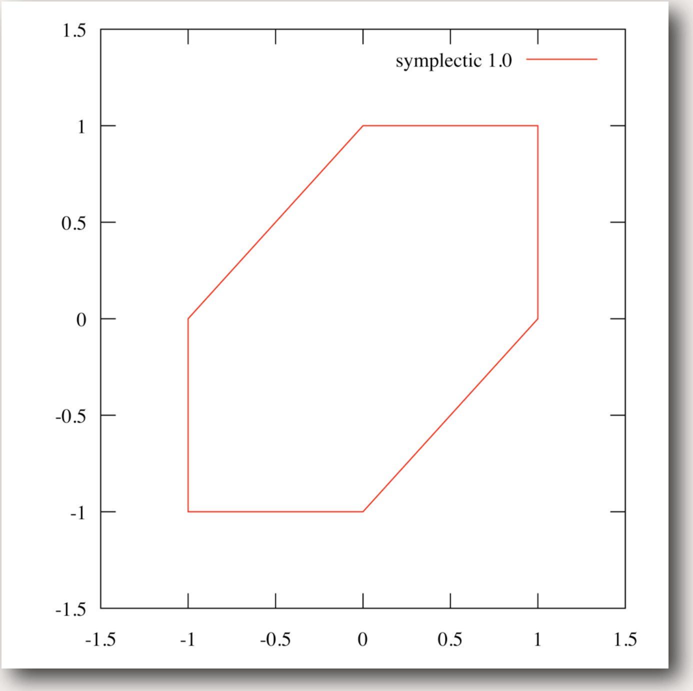
Long Term Evolution

(h = 1.0)

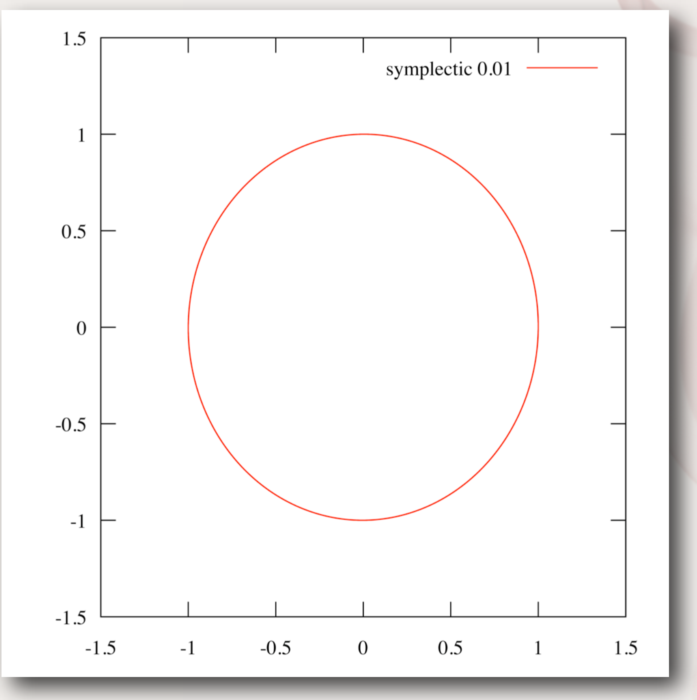
```
x =  1.0;
y =  0.0;
while(true):
  print x, y;
  x -= y;
  y += x;
```

```
 1.0  0.0
 1.0  1.0
 0.0  1.0
-1.0  0.0
-1.0 -1.0
 0.0 -1.0
 1.0  0.0
 1.0  1.0
 0.0  1.0
-1.0  0.0
-1.0 -1.0
 0.0 -1.0
 1.0  0.0
```

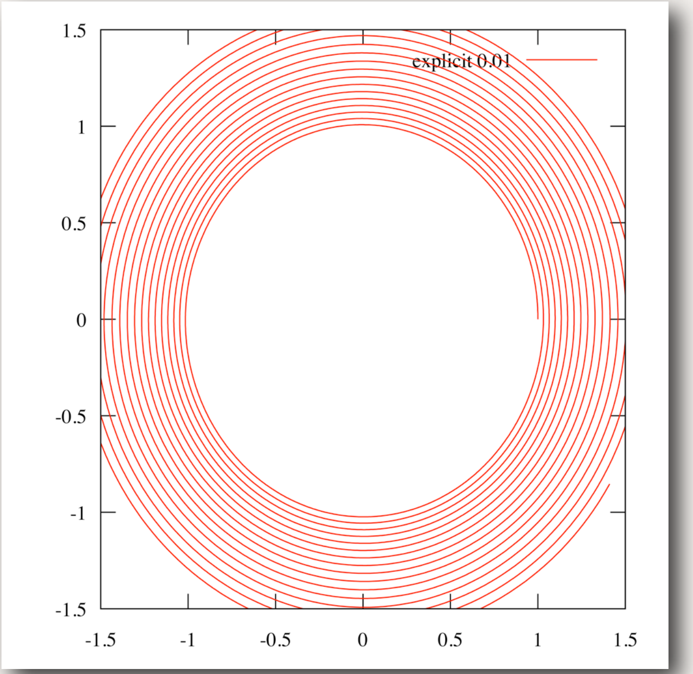
Long Term Evolution



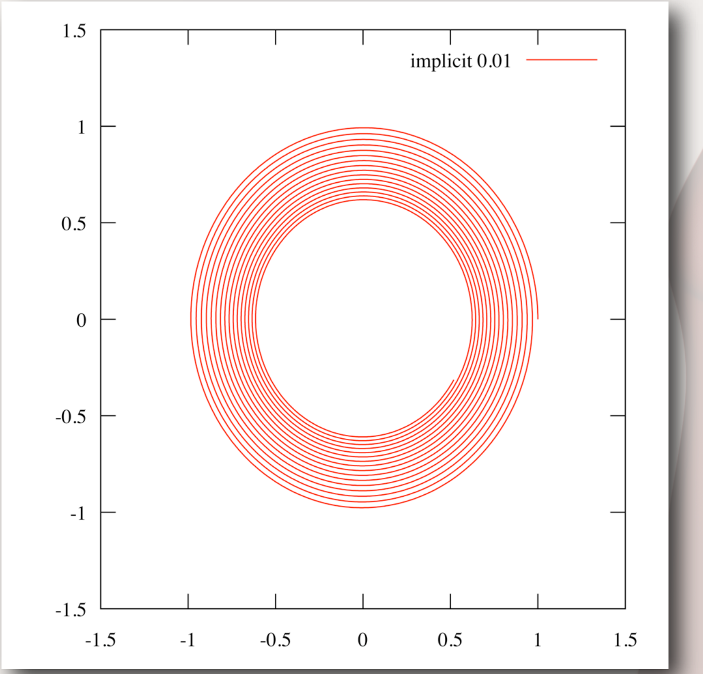
Symplectic 1.0



Symplectic 0.01



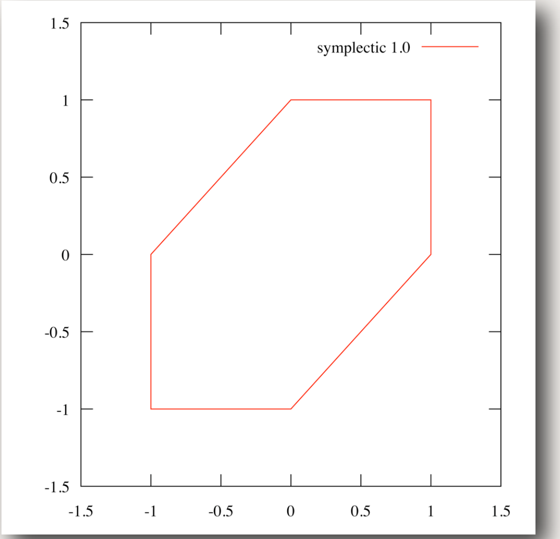
Explicit 0.01



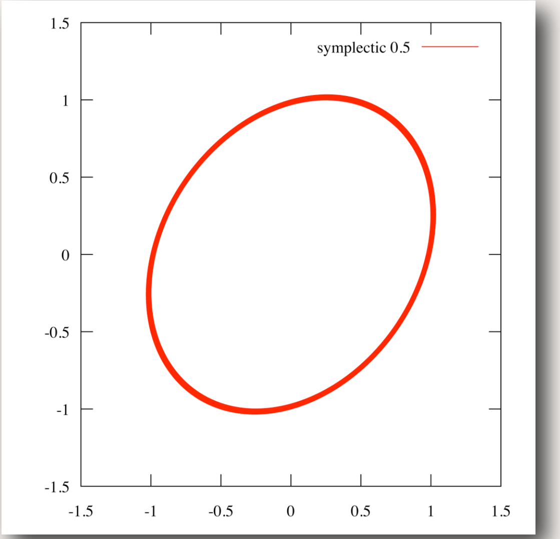
Implicit 0.01

Long Term Evolution

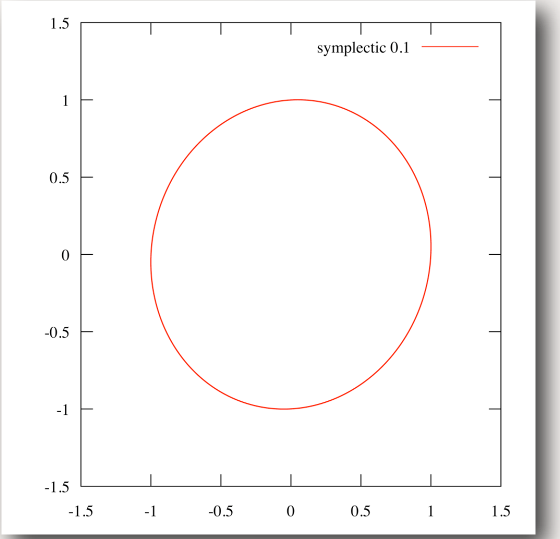
Decreasing Timestep:



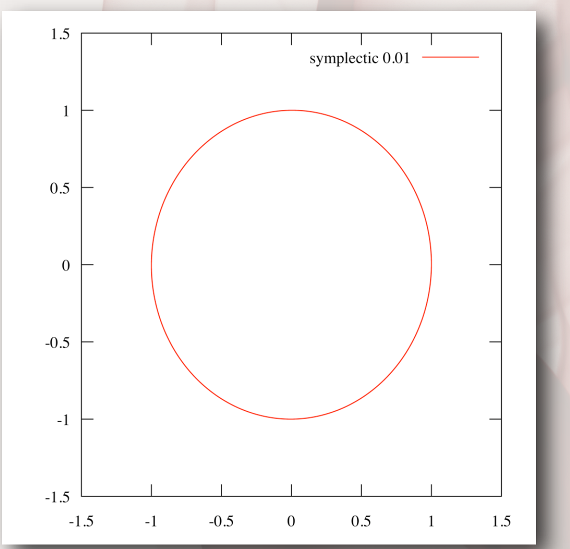
1.0



0.5

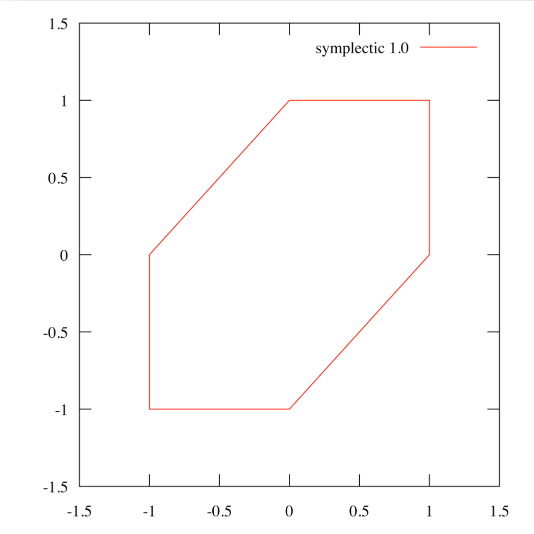


0.1

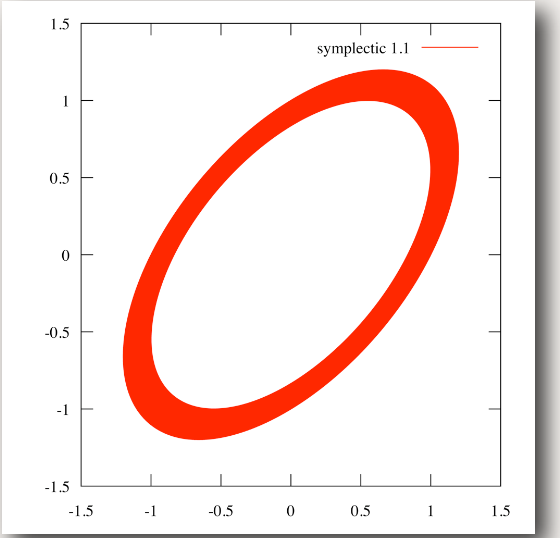


0.01

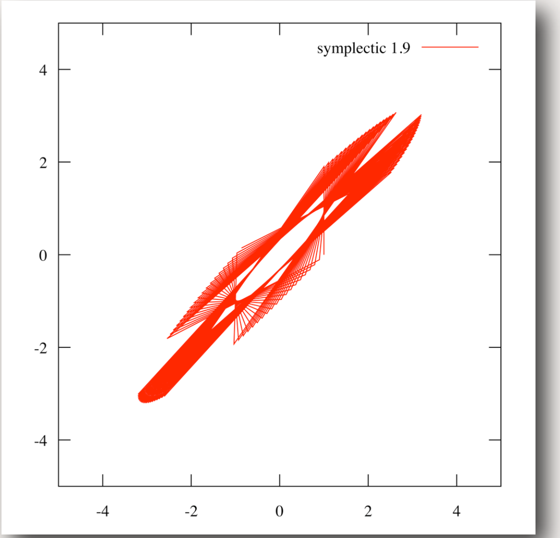
Increasing Timestep:



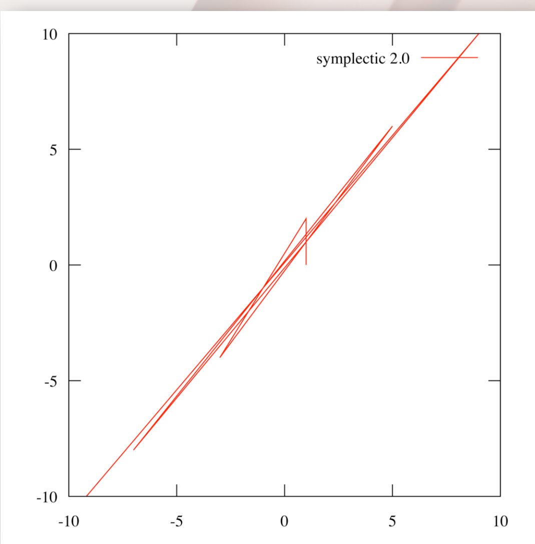
1.0



1.1



1.9



2.0

Why?

- **The symplectic integrator:**

$$\begin{aligned} x_{i+1} &= x_i - hy_i \\ y_{i+1} &= y_i + hx_{i+1} \end{aligned}$$

- **Can be rewritten:**

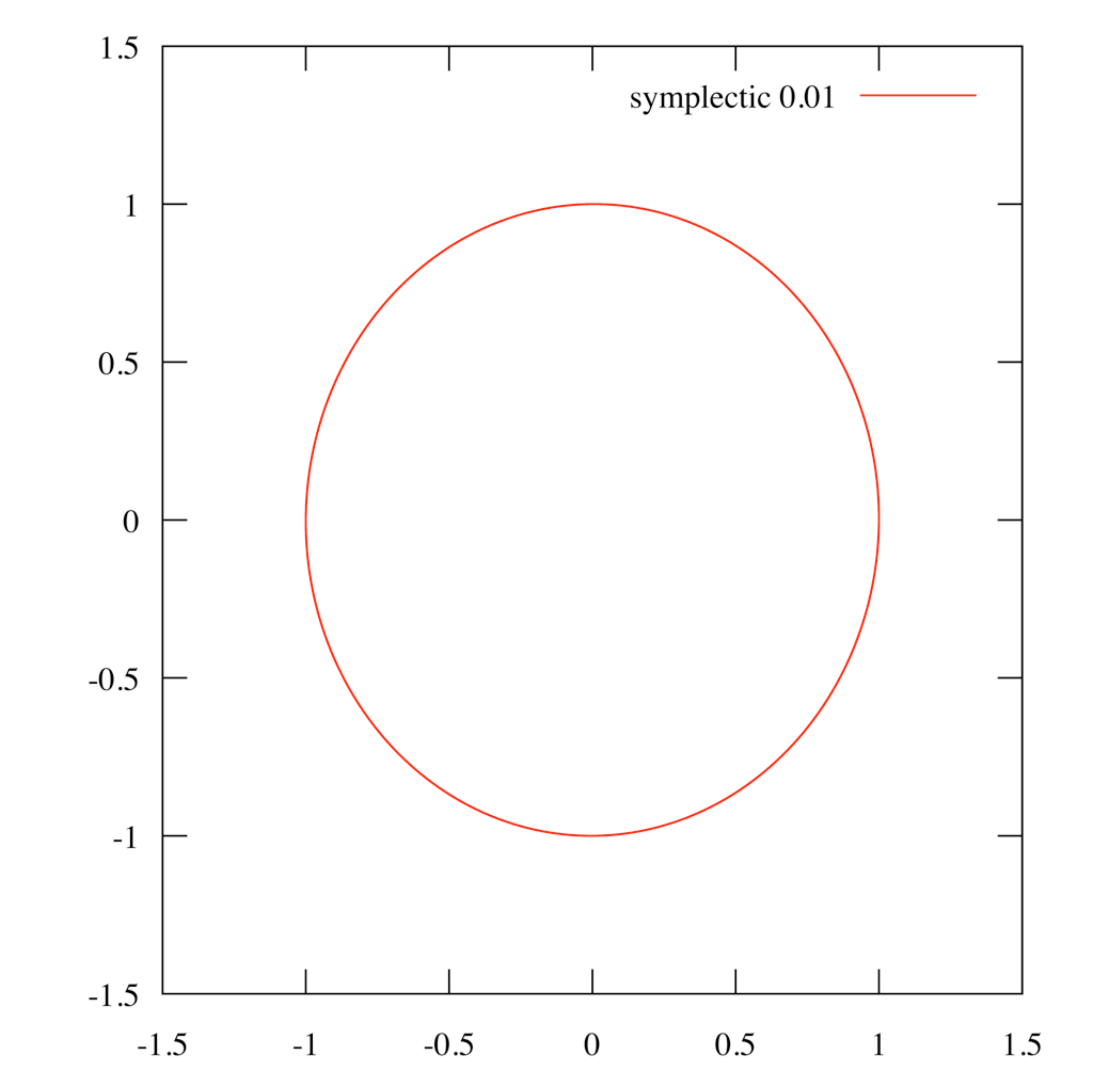
$$\mathbf{x}_{i+1} = \left[\begin{array}{cc} 1 & -h \\ h & 1-h^2 \end{array} \right] \mathbf{x}_i$$

- **Which implies:**

$$\mathbf{x}_i = \left[\begin{array}{cc} 1 & -h \\ h & 1-h^2 \end{array} \right]^i \mathbf{x}_0$$

- **But:**

$$\left\| \mathrm{eig}\left(\left[\begin{array}{cc} 1 & -h \\ h & 1-h^2 \end{array} \right] \right) \right\| = 1 \qquad \text{if } h < 2$$


symplectic 0.01
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5

Symplectic

- This is not general:
  - Hamiltonian systems
  - Preserves Area
- Why did we learn this?
- Numerical Integration is subtle!
  - Small changes can have profound long-term effects.

Differential Constraints

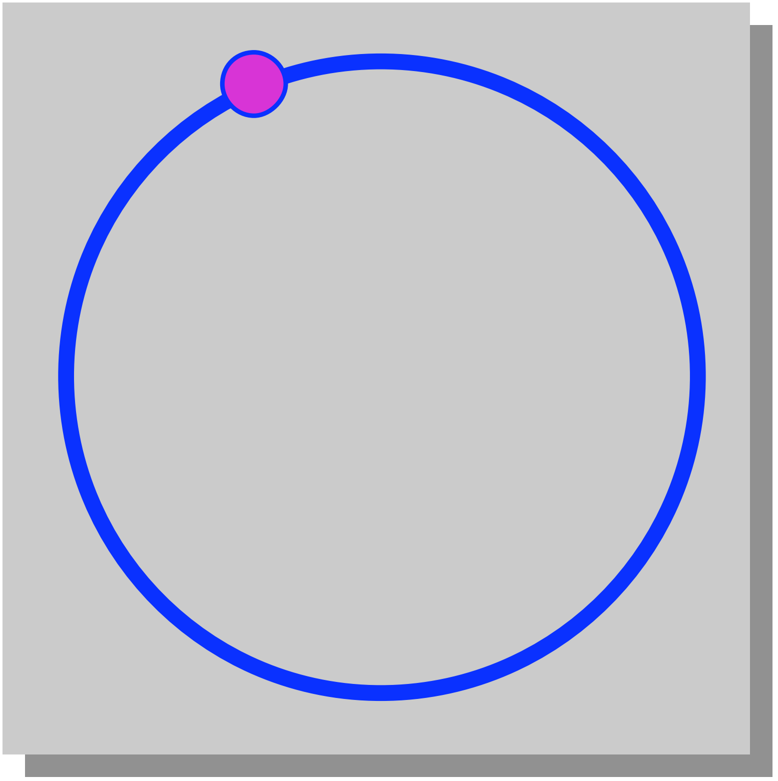
thanks to Adrew Witkin and Zoran Popivić

Differential Constraints

Beyond Points and Springs

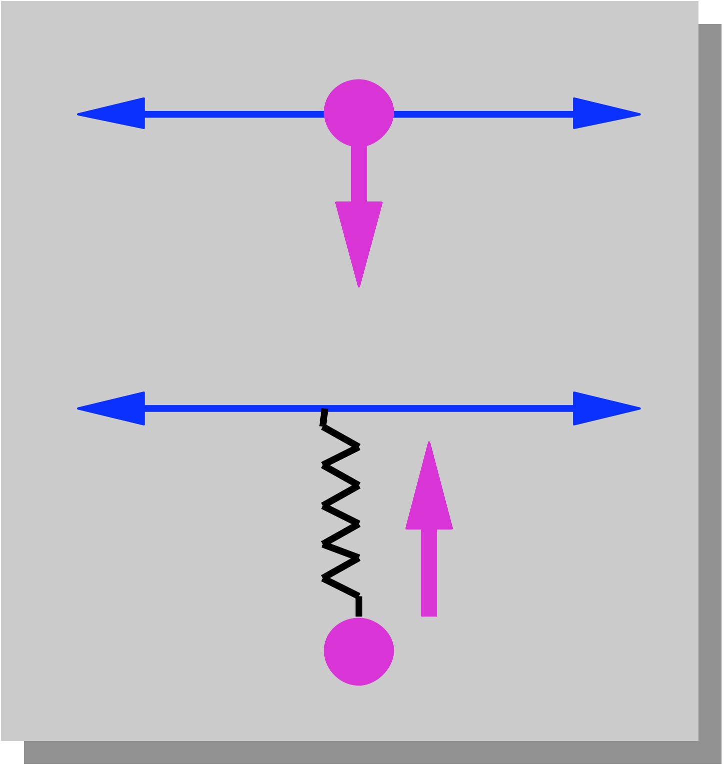
- **You can make just about anything out of point masses and springs, *in principle***

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

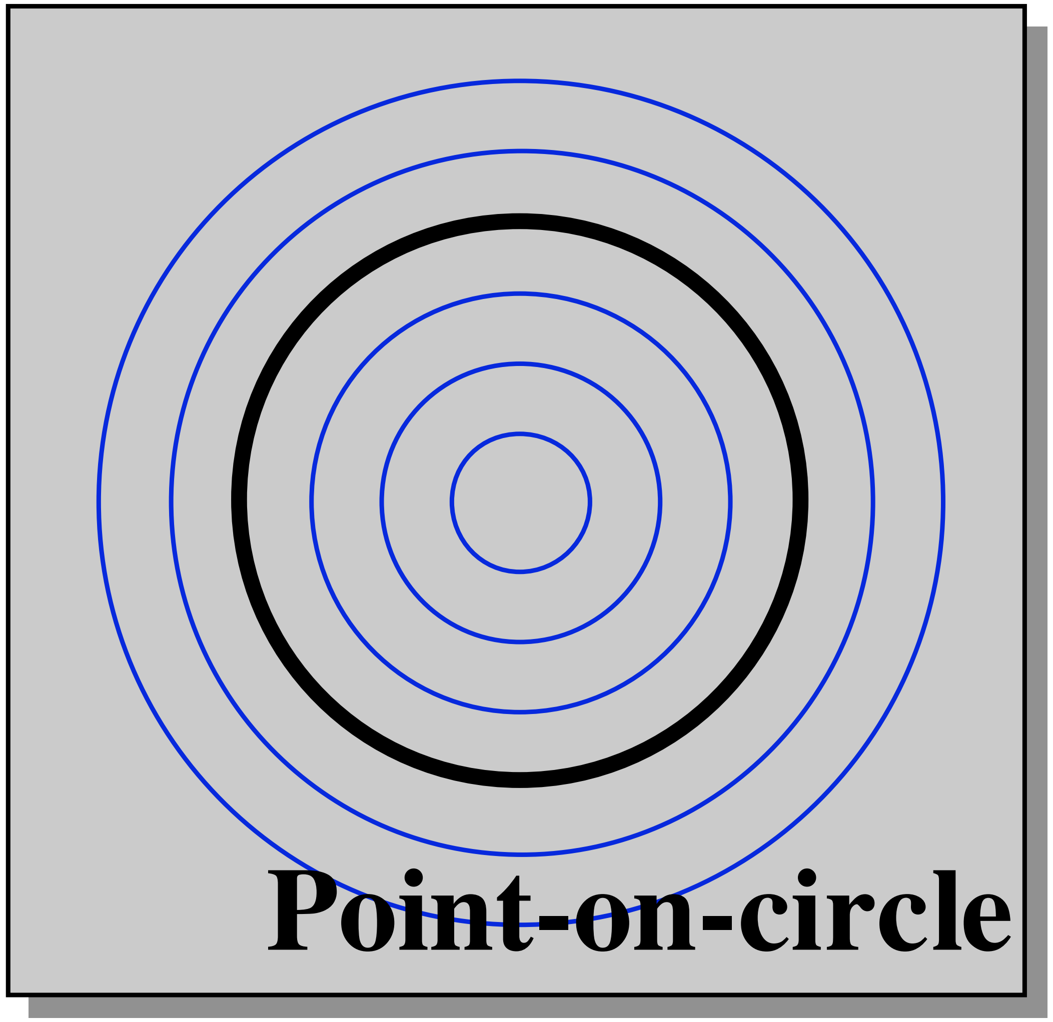


- Why not use a spring to hold the bead on the wire?
- Problem:
  - Weak springs ⇒ goopy constraints
  - Strong springs ⇒ neptune express!
- A classic *stiff system*

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



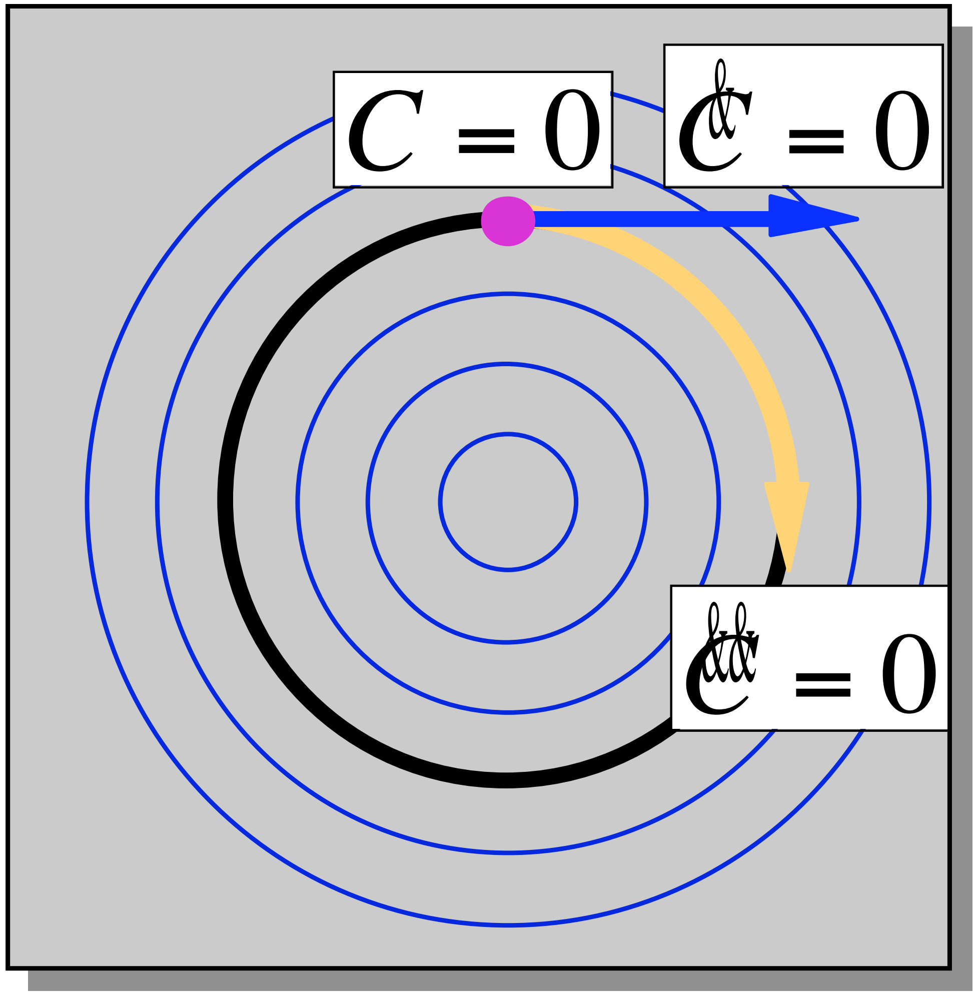
I. Implicit:

$$C(\mathbf{x}) = |\mathbf{x}| - r = 0$$

~~***II. Parametric:***~~

$$\xcancel{\mathbf{x} = r\left[\cos\theta, \sin\theta\right]}$$

Maintaining Constraints Differentially



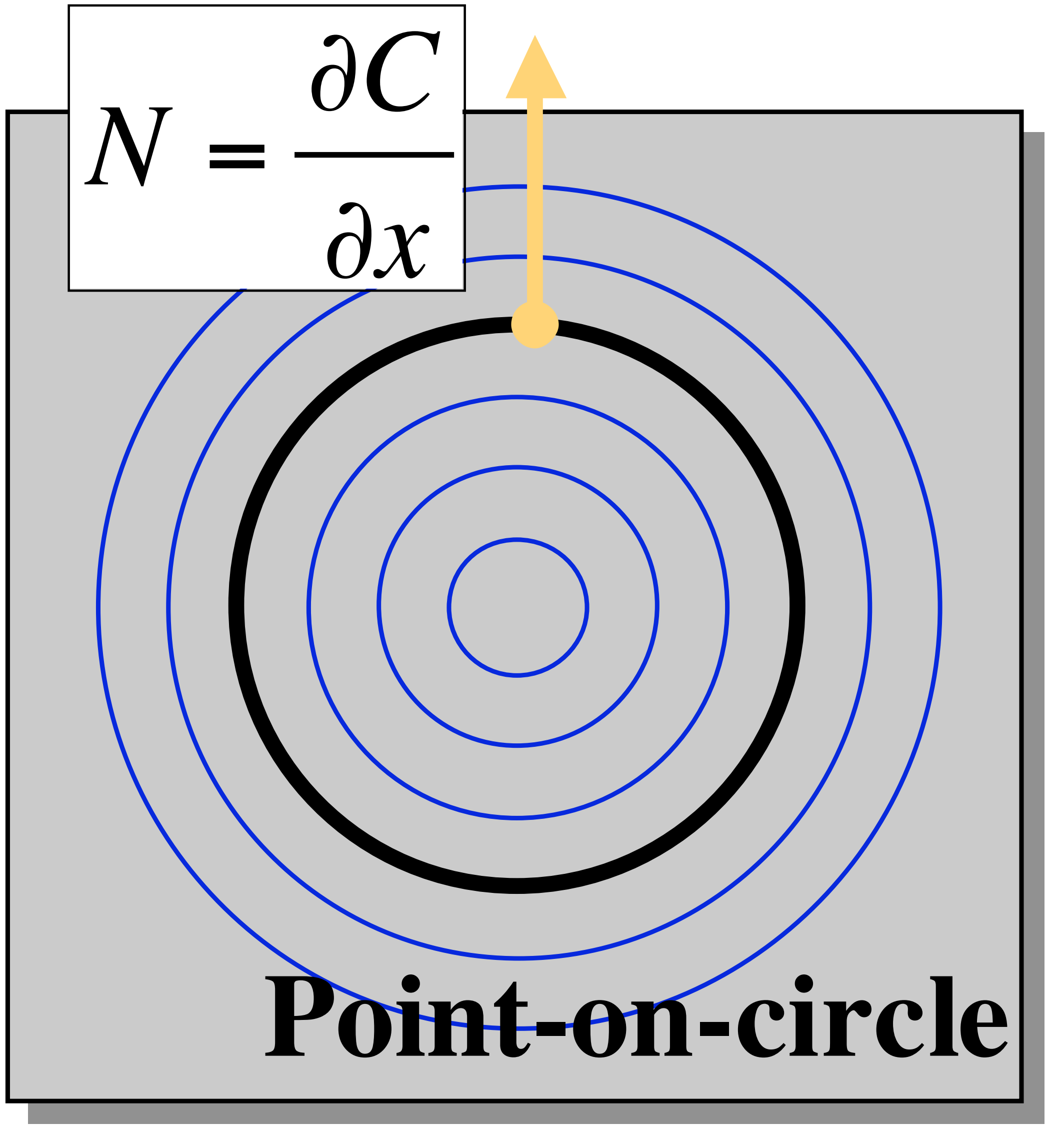
- **Start with legal position and velocity.**
- **Use constraint forces to ensure legal curvature.**

$C = 0$ legal position

$\dot{C} = 0$ legal velocity

$\ddot{C} = 0$ legal curvature

Constraint Gradient



$$N = \frac{\partial C}{\partial x}$$

Point-on-circle

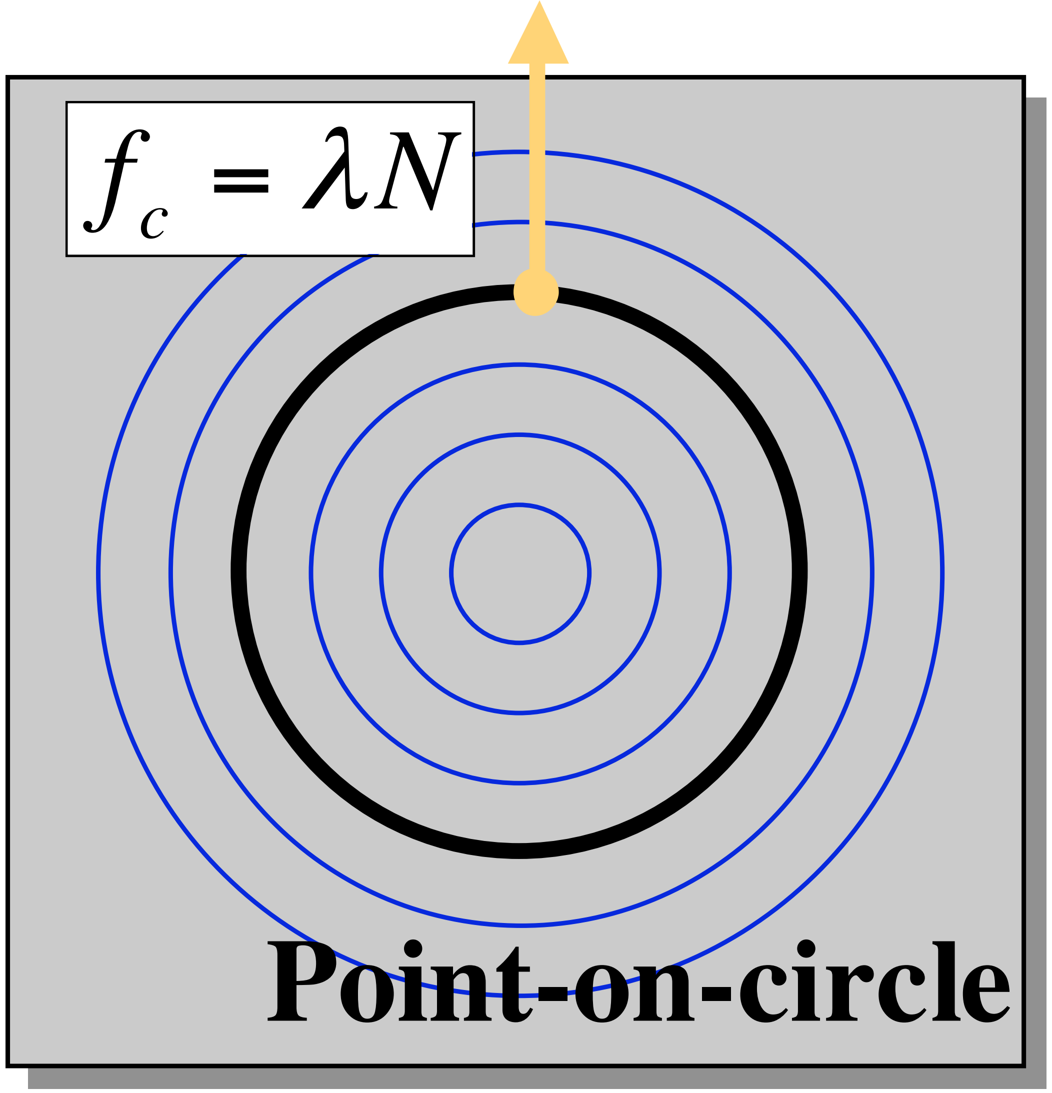
Implicit:

$$C(\mathbf{x}) = |\mathbf{x}| - r = 0$$

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 λ

Just one unknown to solve for

Assumption: constraint is passive—no energy gain or loss

Constraint Force Derivation

$$\boxed{C\left(x(t)\right)}$$

$$\dot{C} = N\cdot\dot{x}$$

$$\ddot{C} = \frac{\partial}{\partial t}\left(N\cdot x\right)$$

$$= \dot{N}\cdot\dot{x} + N\cdot\ddot{x}$$

$$\ddot{x} = \frac{f + f_c}{m}$$

$$f_c = \lambda N$$

Set $\ddot{C} = 0$, solve for λ:

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

Constraint force is λN.

Notation: $N = \dfrac{\partial C}{\partial x}, \dot{N} = \dfrac{\partial^2 C}{\partial x\partial t}$

Example: Point-on-circle

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

$$\mathbf{N} = \frac{\partial C}{\partial \mathbf{x}} = \frac{\mathbf{x}}{|\mathbf{x}|}$$

$$\dot{\mathbf{N}} = \frac{\partial^2 C}{\partial \mathbf{x} \partial t} = \frac{1}{|\mathbf{x}|}\left[\dot{\mathbf{x}} - \frac{\mathbf{x}\cdot\dot{\mathbf{x}}}{\mathbf{x}\cdot\mathbf{x}}\mathbf{x}\right]$$

Write down the constraint equation.

Take the derivatives.

Substitute into generic template, simplify.

$$\lambda = -m\frac{\dot{\mathbf{N}}\cdot\dot{\mathbf{x}}}{\mathbf{N}\cdot\mathbf{N}} - \frac{\mathbf{N}\cdot\mathbf{f}}{\mathbf{N}\cdot\mathbf{N}} = \left[m\frac{(\mathbf{x}\cdot\dot{\mathbf{x}})^2}{\mathbf{x}\cdot\mathbf{x}} - m(\dot{\mathbf{x}}\cdot\dot{\mathbf{x}}) - \mathbf{x}\cdot\mathbf{f}\right]\frac{1}{|\mathbf{x}|}$$

Tinkertoys

- 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…

Compact Particle System Notation

$$\ddot{\mathbf{q}} = \mathbf{W}\mathbf{Q}$$

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

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

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

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

$$\mathbf{q} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$$
$$\mathbf{Q} = [\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_n]$$
$$\mathbf{M} = \begin{bmatrix} m_1 & & & & & & \\ & m_1 & & & & & \\ & & m_1 & & & & \\ & & & \ddots & & & \\ & & & & m_n & & \\ & & & & & m_n & \\ & & & & & & m_n \end{bmatrix}$$
$$\mathbf{W} = \mathbf{M}^{-1}$$

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

Solving for the Constraint Force

$$\ddot{x} = \frac{1}{m}(f + \hat{f})$$

$$C(x) = x\cdot x - 1 = 0$$

$$\dot{C}(x) = 2x\cdot\dot{x} = 0$$

$$\ddot{C}(x) = 2(x\cdot\ddot{x} + \dot{x}\cdot\dot{x})$$

$$\dot{x}\cdot\dot{x} + x\cdot\left(\frac{1}{m}(f + \hat{f})\right) = 0$$

$$x\cdot\hat{f} = -m\dot{x}\cdot\dot{x} - xf$$

Bead on a Wire

$$\ddot{q} = W(Q + \hat{Q})$$

$$C(q) = 0$$

$$\dot{C} = \frac{\partial C}{\partial q}\dot{q} = J\dot{q}$$

$$\ddot{C} = \dot{J}\dot{q} + J\ddot{q}$$

$$\dot{J}\dot{q} + JW(Q + \hat{Q}) = 0$$

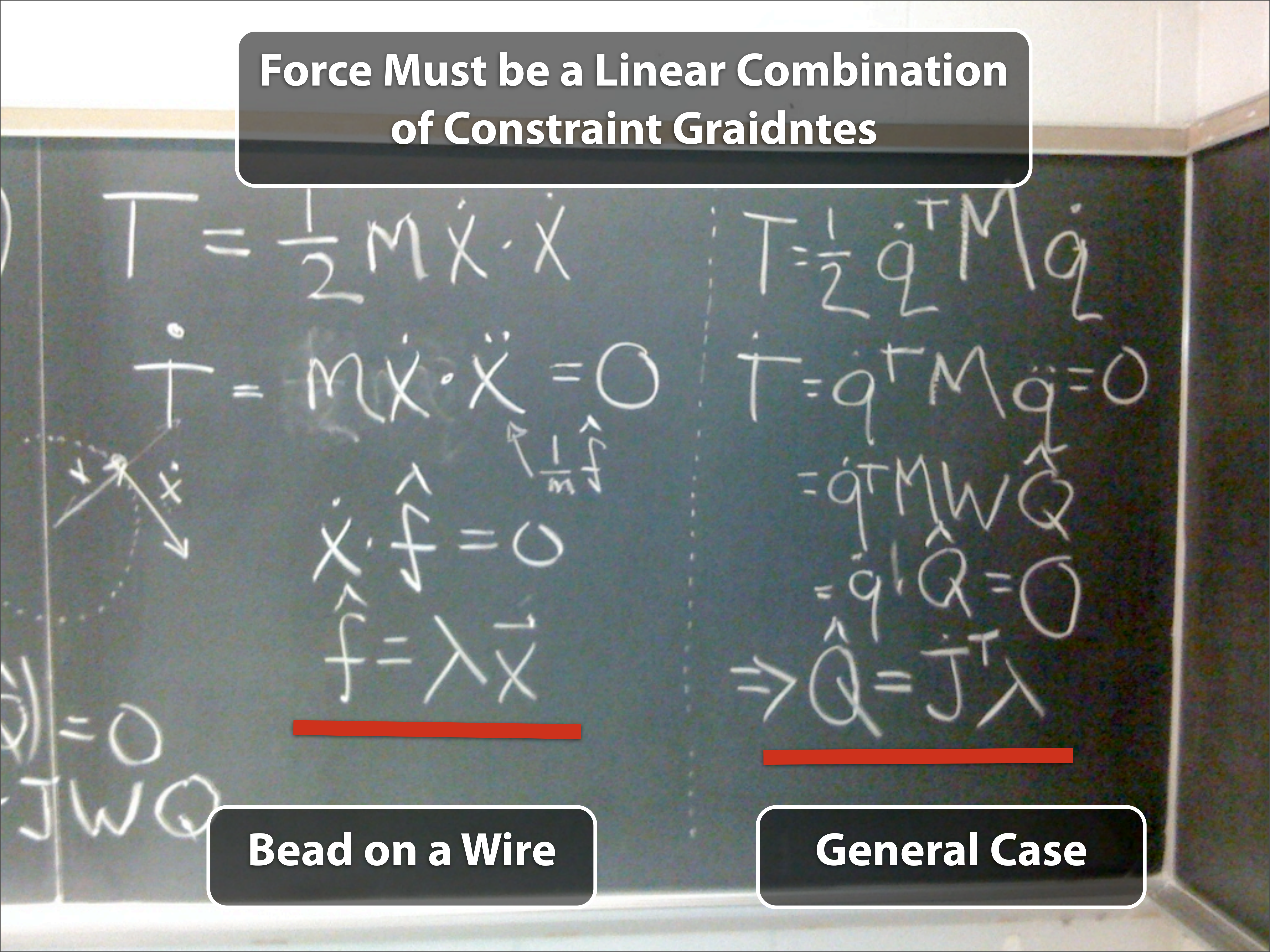
$$JW\hat{Q} = -\dot{J}\dot{q} - JWQ$$

General Case

T

$\dot{T}$

Force Must be a Linear Combination of Constraint Graidntes



$$T = \frac{1}{2} m \dot{x} \cdot \dot{x}$$

$$\dot{T} = m \dot{x} \cdot \ddot{x} = 0 \quad \frac{1}{m}\hat{f}$$

$$\dot{x} \cdot \hat{f} = 0$$

$$\hat{f} = \lambda \vec{x}$$

$$T = \frac{1}{2} \dot{q}^T M \dot{q}$$

$$\dot{T} = \dot{q}^T M \ddot{q} = 0$$

$$= \dot{q}^T M W \hat{Q}$$

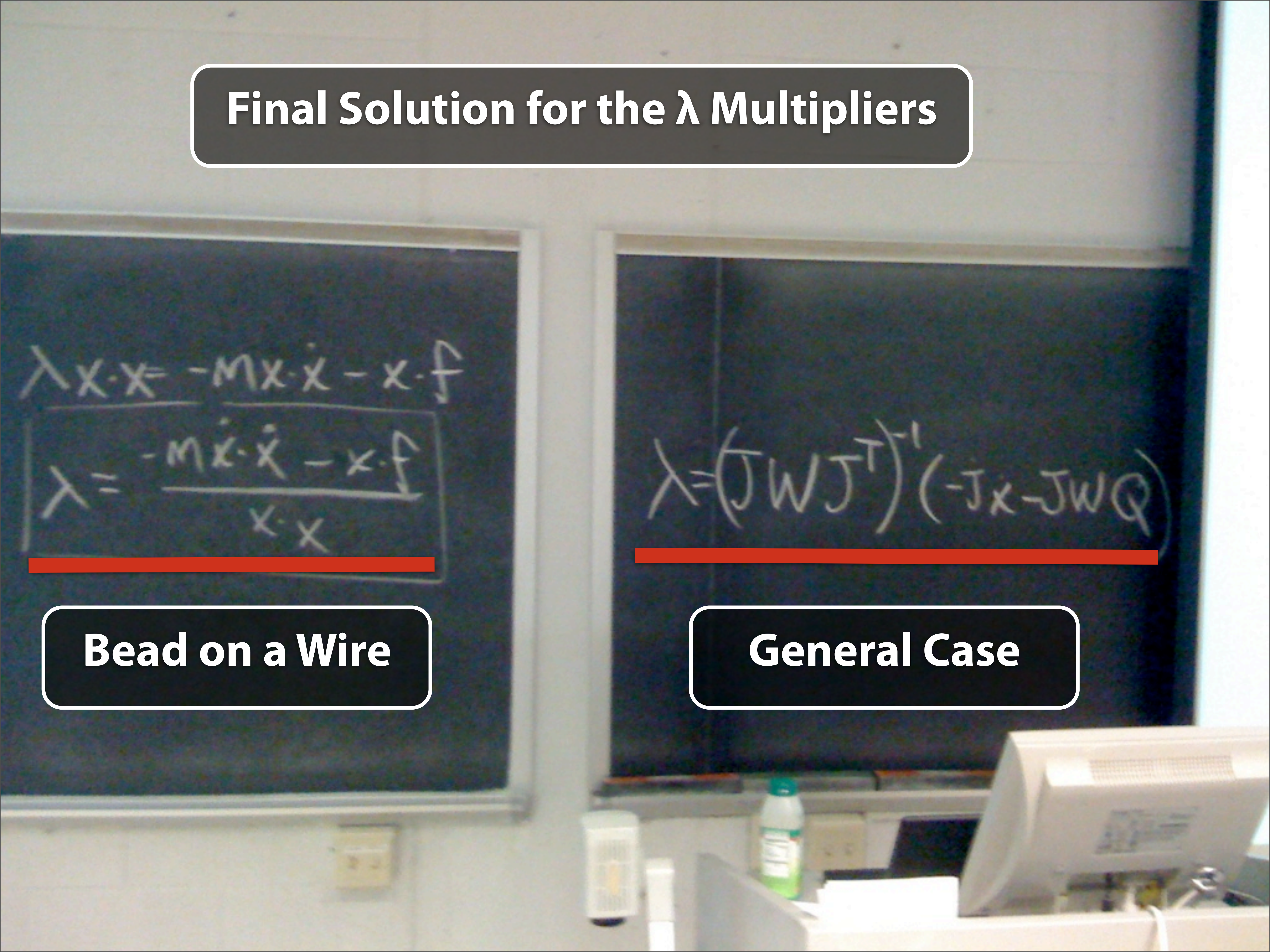
$$= \dot{q}^T \hat{Q} = 0$$

$$\Rightarrow \hat{Q} = J^T \lambda$$

Bead on a Wire

General Case

Final Solution for the λ Multipliers



$$\lambda x\cdot x = -m x\cdot\dot{x} - x\cdot f$$

$$\boxed{\lambda = \frac{-m\dot{x}\cdot\dot{x} - x\cdot f}{x\cdot x}}$$

Bead on a Wire

$$\lambda = (JWJ^T)^{-1}(-\dot{J}\dot{x} - JWQ)$$

General Case

Particle System Constraint Equations

Matrix equation for λ

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

Constrained Acceleration

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

Derivation: just like bead-on-wire.

More Notation

$$\mathbf{C} = \left[C_1, C_2, \cdots, C_m\right]$$

$$\lambda = \left[\lambda_1, \lambda_2, \cdots, \lambda_m\right]$$

$$\mathbf{J} = \frac{\partial \mathbf{C}}{\partial \mathbf{q}}$$

$$\dot{\mathbf{J}} = \frac{\partial^2 \mathbf{C}}{\partial \mathbf{q}\,\partial t}$$

Drift and Feedback

- In principle, clamping $\ddot{C}$ at zero is enough
- Two problems:
  - Constraints might not be met initially
  - Numerical errors can accumulate
- A feedback term handles both problems:

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

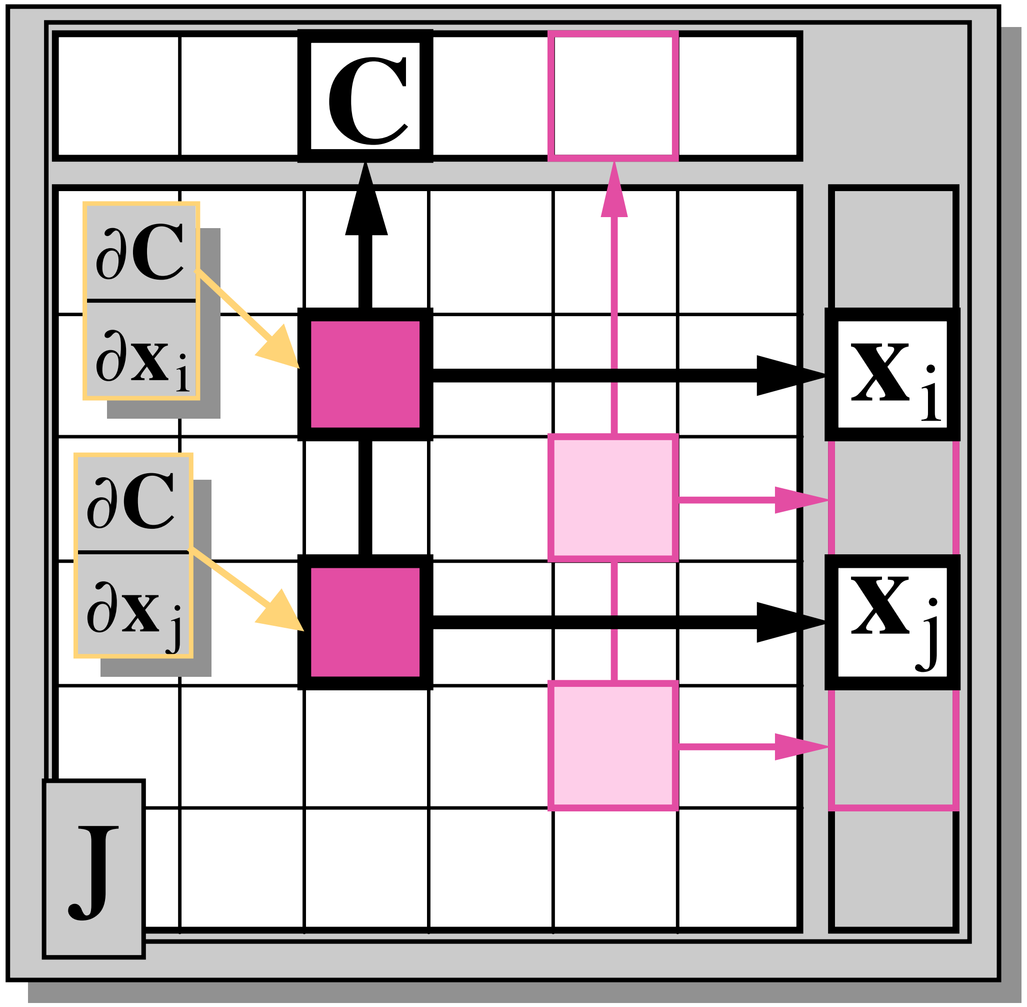
$$\ddot{C} = 0$$

α and β are magic constants.

How do you implement all this?

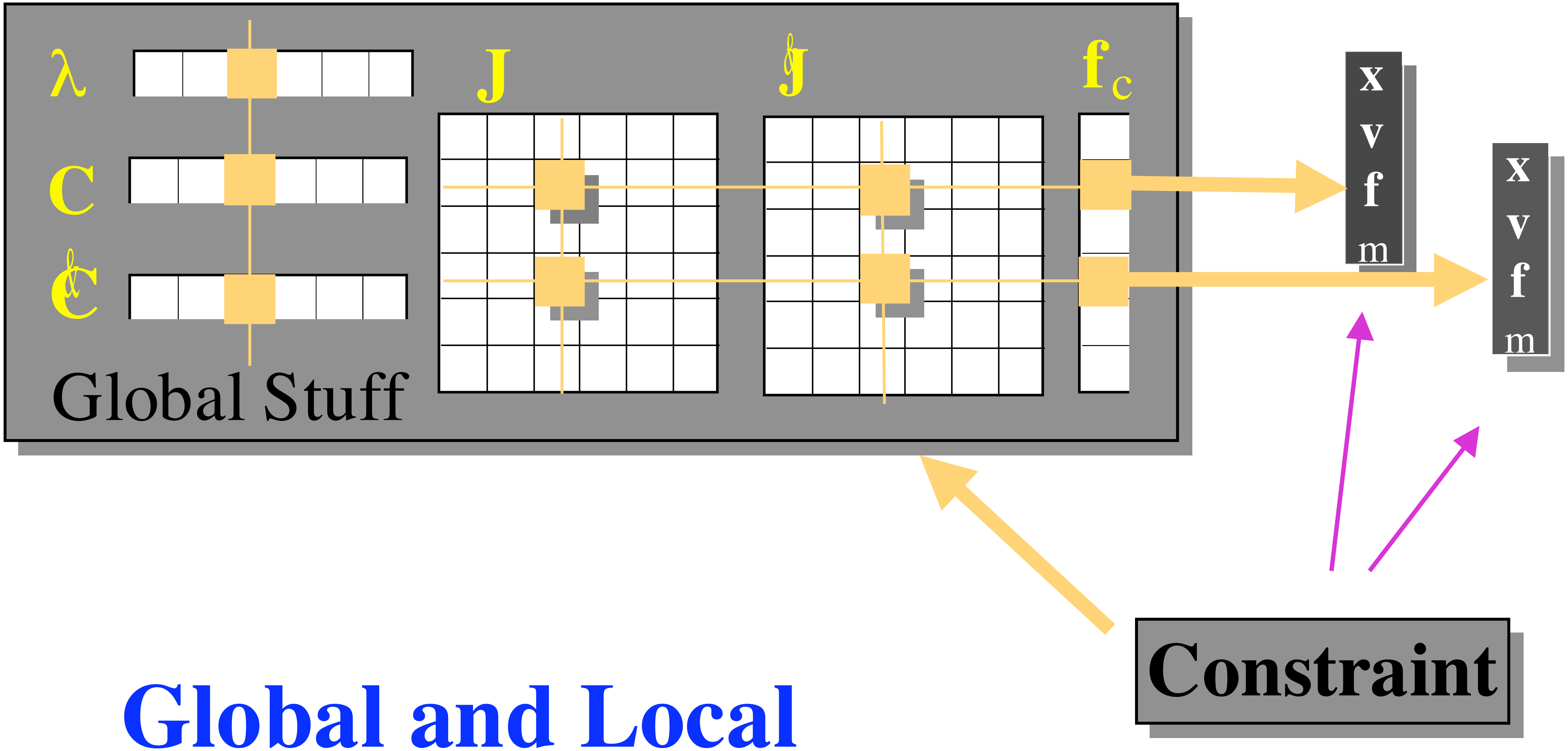
- We have a global matrix equation.
- We want to build models on the fly, just like masses and springs.
- Approach:
  - Each constraint adds its own piece to the equation.

Matrix Block Structure


C
$\frac{\partial C}{\partial \mathbf{x}_i}$
$\frac{\partial C}{\partial \mathbf{x}_j}$
$\mathbf{x}_i$
$\mathbf{x}_j$
J

- 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 and Local

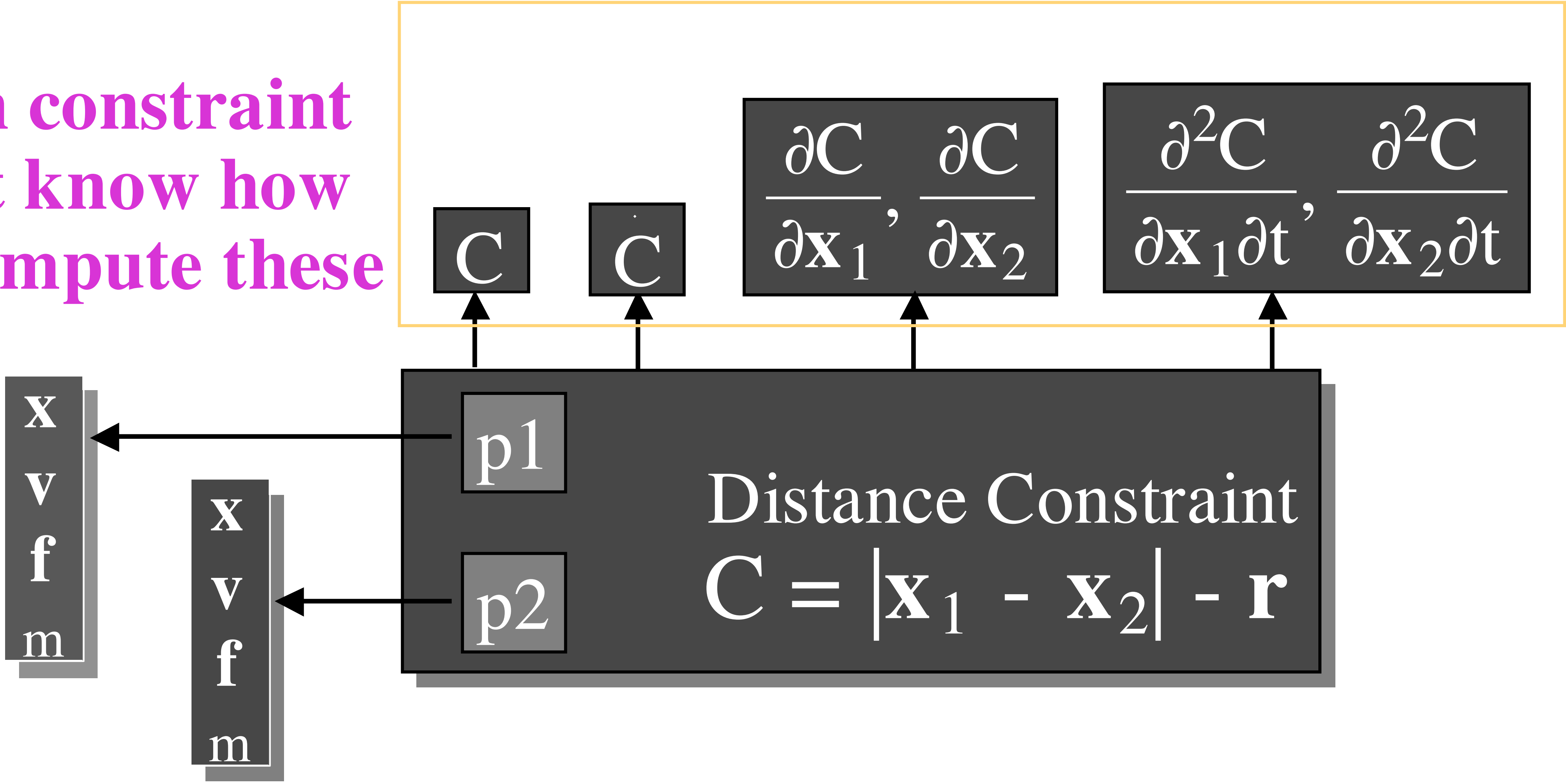


Global Stuff

Constraint

Constraint Structure

Each constraint must know how to compute these



C

$\dot{C}$

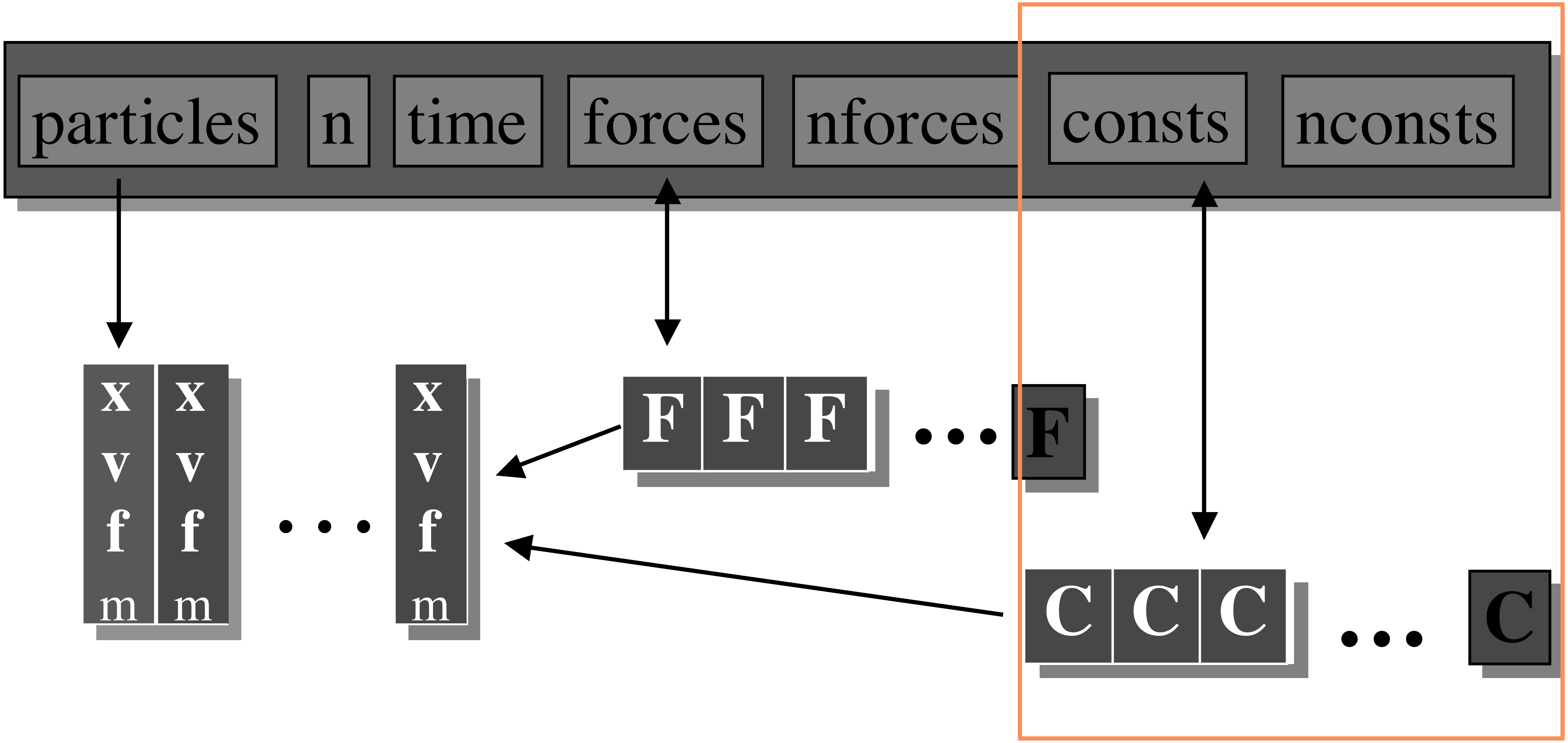
$\frac{\partial C}{\partial \mathbf{x}_1}, \frac{\partial C}{\partial \mathbf{x}_2}$

$\frac{\partial^2 C}{\partial \mathbf{x}_1 \partial t}, \frac{\partial^2 C}{\partial \mathbf{x}_2 \partial t}$

Distance Constraint

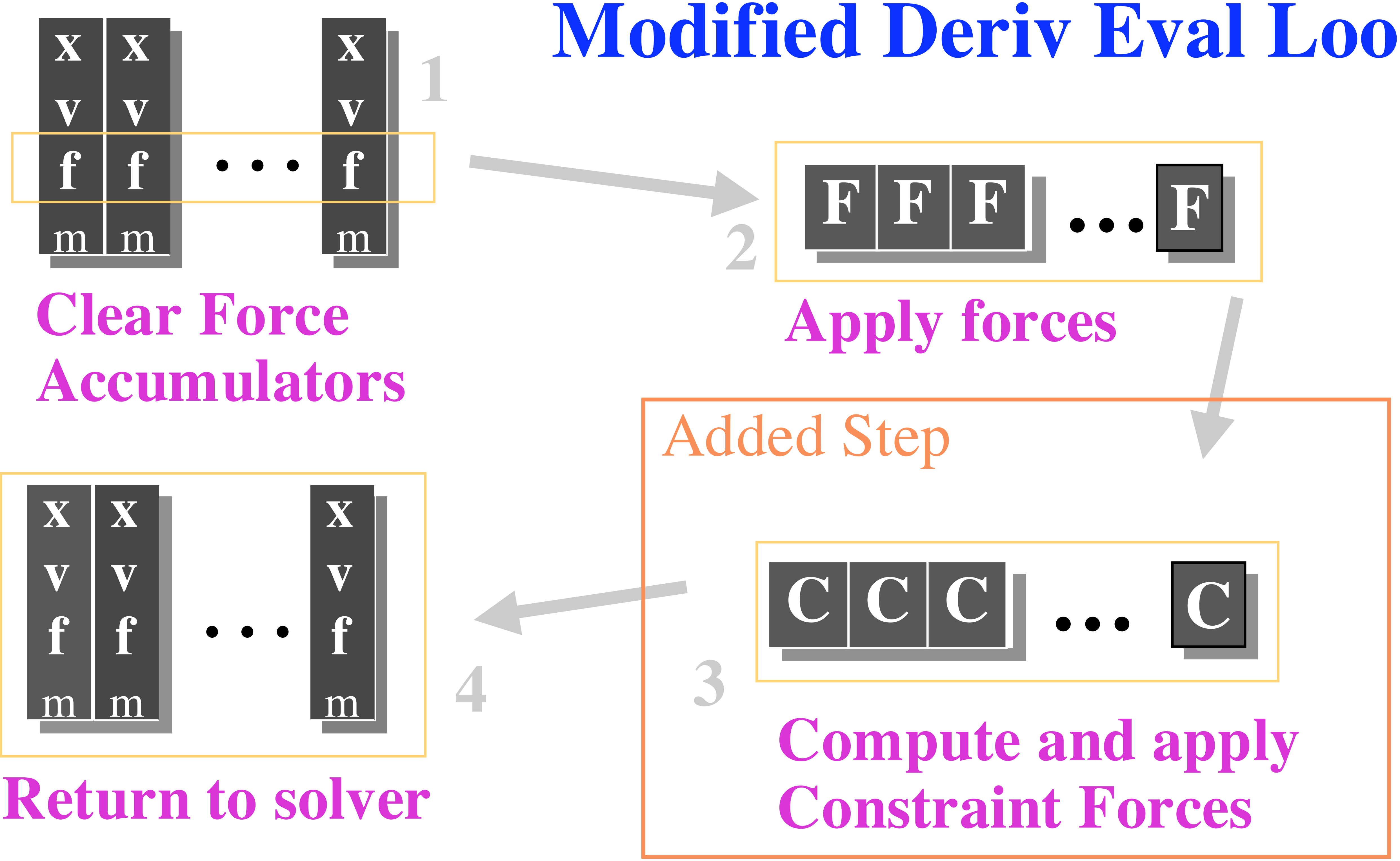
$C = |\mathbf{x}_1 - \mathbf{x}_2| - \mathbf{r}$

Constrained Particle Systems



Added Stuff

Modified Deriv Eval Loop



Clear Force Accumulators

Apply forces

Added Step

Compute and apply Constraint Forces

Return to solver

Constraint Force Eval

- After computing ordinary forces:
  - Loop over constraints, assemble global matrices and vectors.
  - Call matrix solver to get λ, 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 *Principle of Virtual Work.*
- The λ's are called *Lagrange Multipliers.*
- The derivative matrix, J, is called the *Jacobian Matrix.*

Question

- How could you simulate hair?
  - What are the salient properties of hair you’re trying to simulate?