Constrained Dynamics

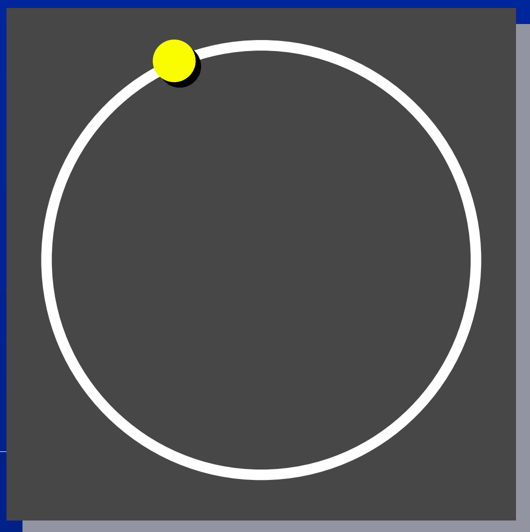
Andrew Witkin

PIXAR
ANIMATION STUDIOS

Beyond Points and Springs

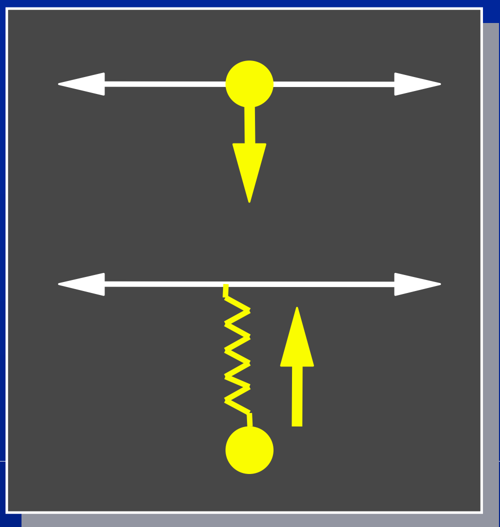
- You can make just about anything out of point masses and springs, *in principle.*
- In practice, you can make anything you want as long as it's jello.
- Constraints will buy us:
  - Rigid links instead of goopy springs.
  - Ways to make interesting contraptions.

A bead on a wire



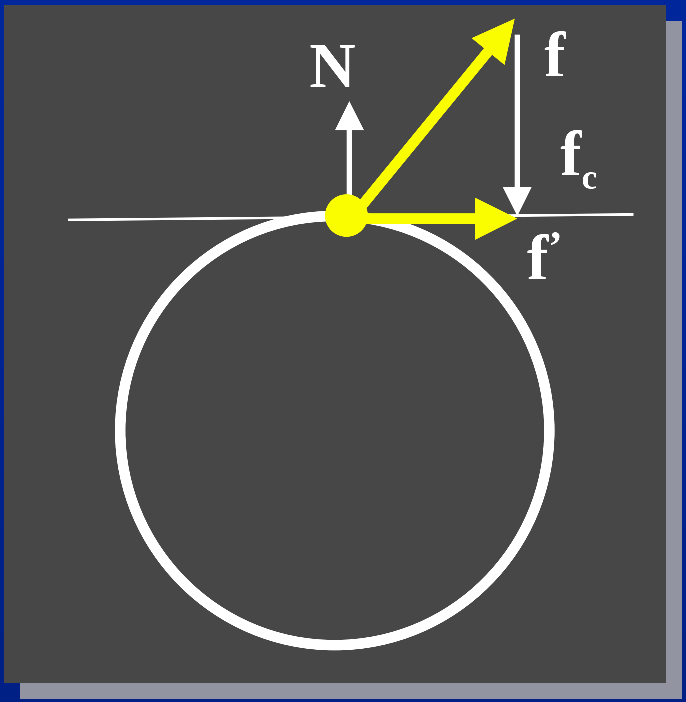
- **Desired Behavior:**
  - **The bead can slide freely *along* the circle.**
  - **It can never come off, however hard we pull.**
- **Question:**
  - **How does the bead move under applied forces?**

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.*

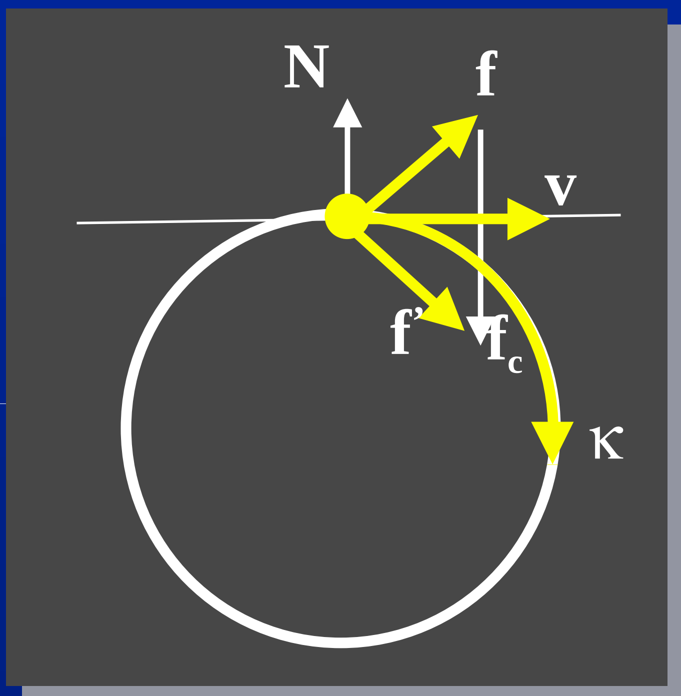
The basic trick (f = mv version)



- 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.

$$\mathbf{f}_C = -\frac{\mathbf{f}\cdot\mathbf{N}}{\mathbf{N}\cdot\mathbf{N}}\mathbf{N} \qquad \mathbf{f}' = \mathbf{f} + \mathbf{f}_C$$

f = ma

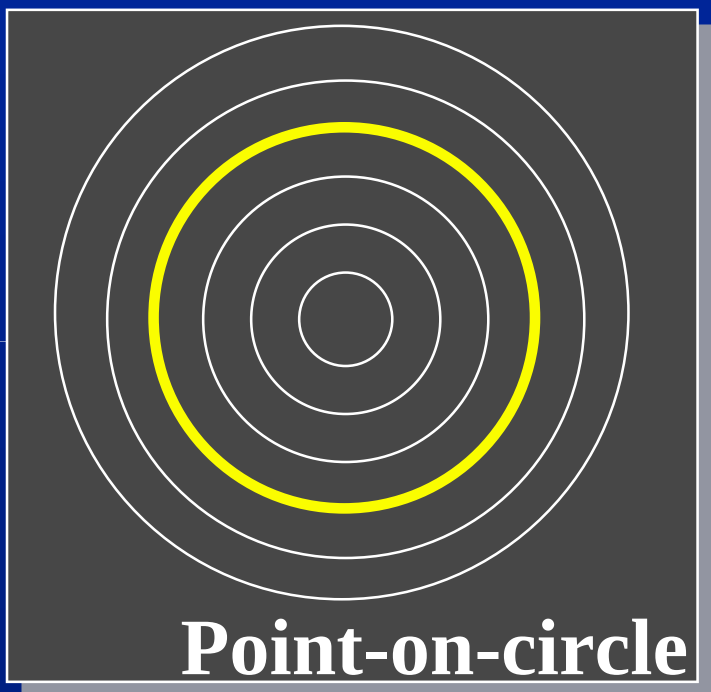


- Same idea, but…
- *Curvature* (κ) has to match.
- κ depends on *both* a and v:
  - the faster you're going, the faster you have to turn.
- Calculate f_c to yield a legal *combination* of a and v.
- Blechh!

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



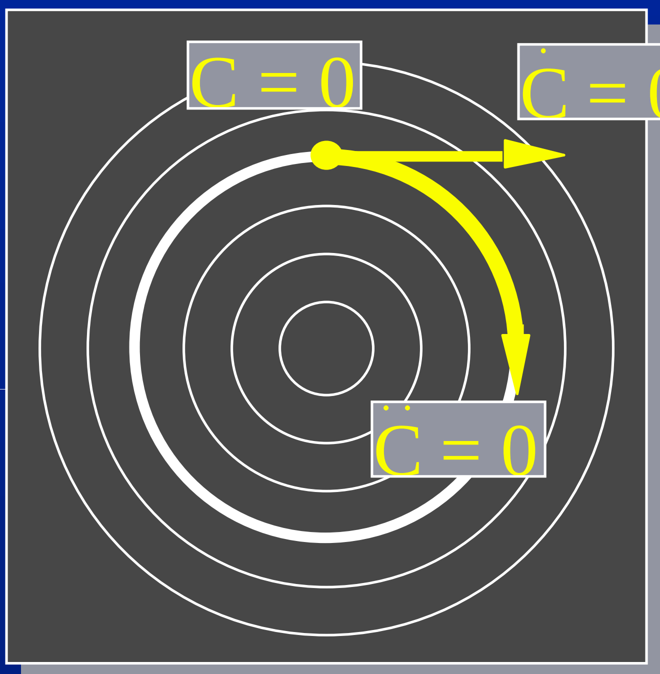
I. Implicit:

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

~~*II. Parametric:*~~

~~$$\mathbf{x} = r[\cos\theta, \sin\theta]$$~~

Maintaining Constraints Differentially



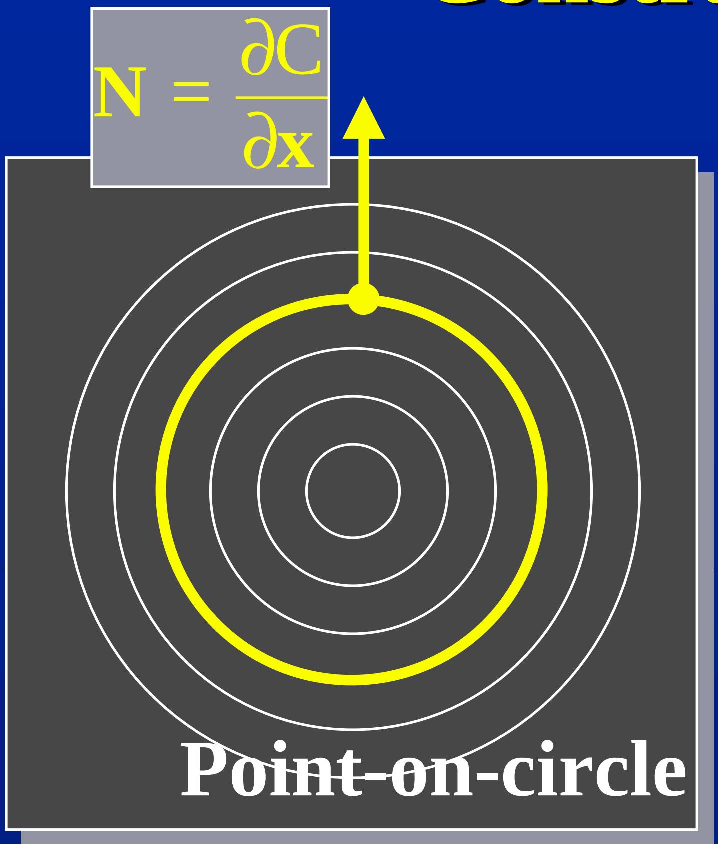
- **Start with legal position and velocity.**
- **Use constraint forces to ensure legal curvature.**

$C = 0$ *legal position*

$\dot{C} = 0$ *legal velocity*

$\ddot{C} = 0$ *legal curvature*

Constraint Gradient



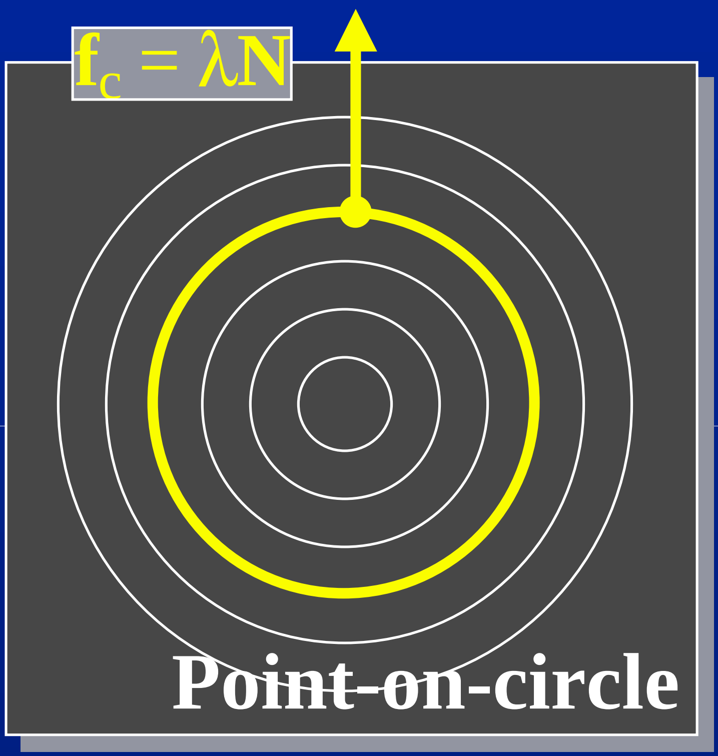
Implicit:

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

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

$$\mathbf{C}(\mathbf{x}(t))$$

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

$$\ddot{\mathbf{C}} = \frac{\partial}{\partial t}[\mathbf{N} \cdot \dot{\mathbf{x}}]$$

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

$$\ddot{\mathbf{x}} = \frac{\dot{\mathbf{f}} + \mathbf{f}_c}{m}$$

$$\mathbf{f}_c = \lambda \mathbf{N}$$

Set $\ddot{\mathbf{C}} = 0$, solve for λ:

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

Constraint force is $\lambda \mathbf{N}$.

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

Example: Point-on-circle

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

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

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

Write down the constraint equation.

Take the derivatives.

Substitute into generic template, simplify.

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

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.

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…**

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

$$\ddot{q} = WQ$$

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

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

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

W: M-inverse (element-wise reciprocal)

$$\mathbf{q} = \left[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\right]$$

$$\mathbf{Q} = \left[\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_n\right]$$

$$\mathbf{M} = \begin{bmatrix} m_1 & & & & & & \\ & m_1 & & & & & \\ & & m_1 & & & & \\ & & & \ddots & & & \\ & & & & m_n & & \\ & & & & & m_n & \\ & & & & & & m_n \end{bmatrix}$$

$$\mathbf{W} = \mathbf{M}^{-1}$$

Particle System Constraint Equations

Matrix equation for λ

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

Constrained Acceleration

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

Derivation: just like bead-on-wire.

More Notation

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

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

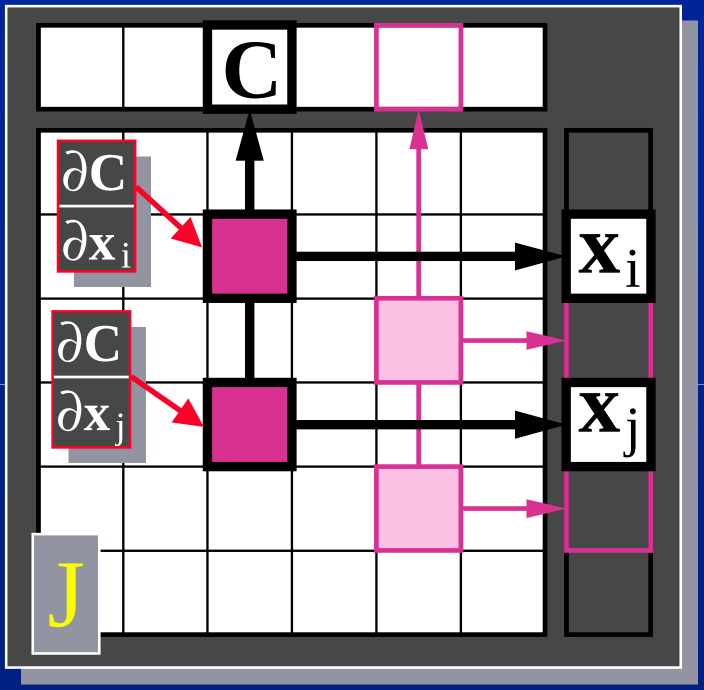
$$\mathbf{J} = \frac{\partial \mathbf{C}}{\partial \mathbf{q}}$$

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

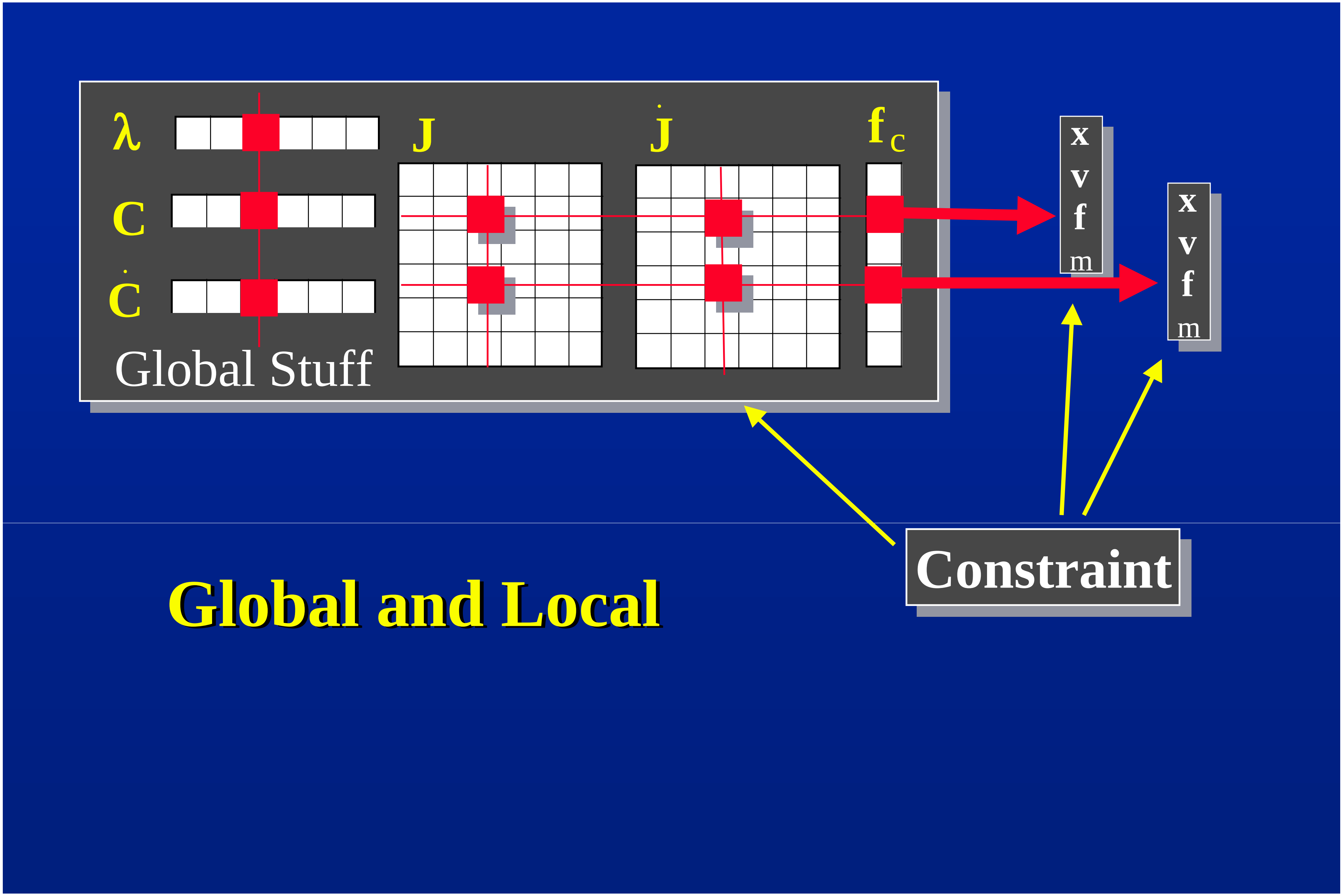
How do you implement all this?

- 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



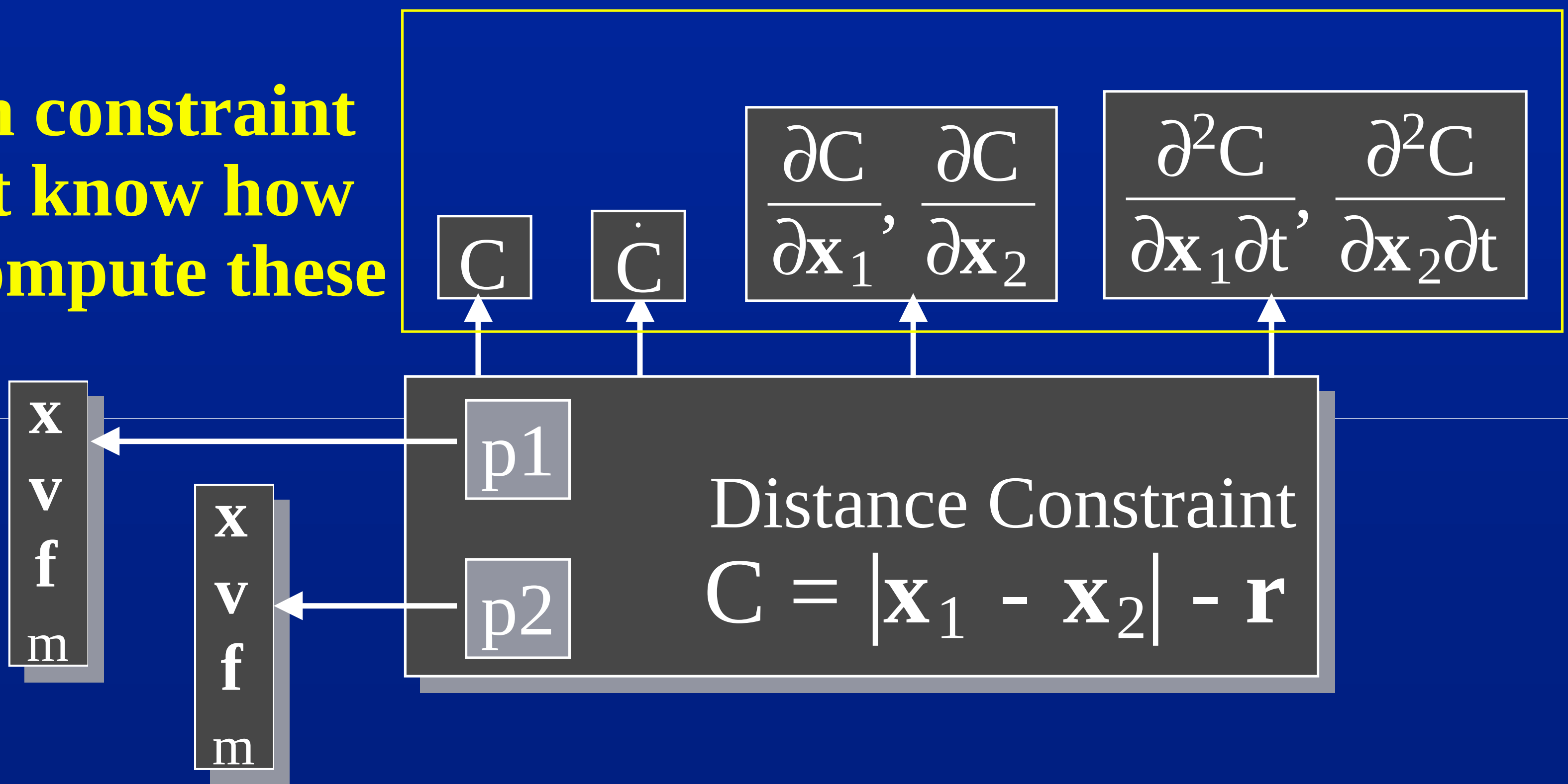
- Each constraint contributes one or more *blocks* to the matrix.
- Sparsity: many empty blocks.
- Modularity: let each constraint compute its own blocks.
- Constraint and particle indices determine block locations.



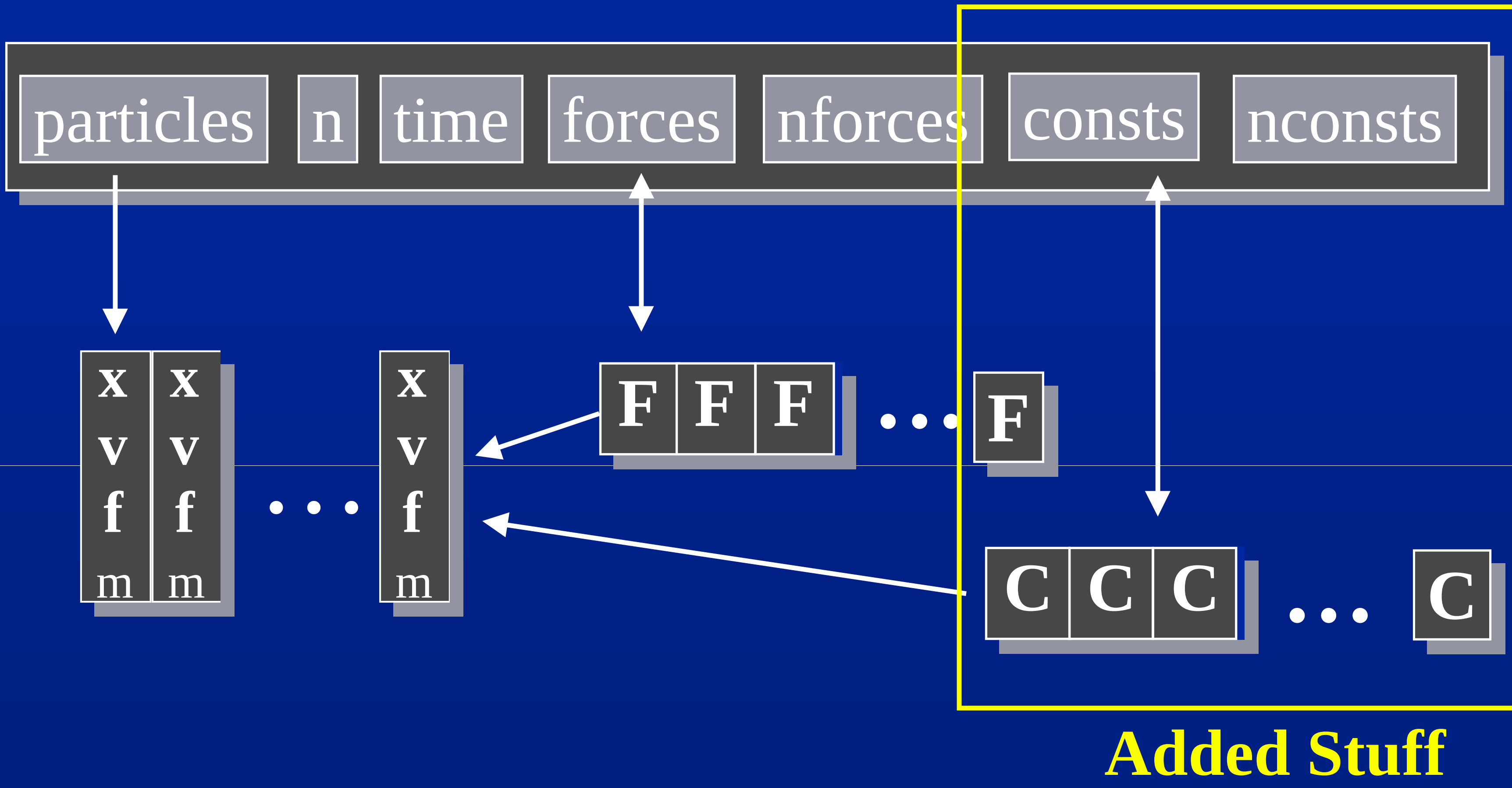
λ
C
Ċ
Global Stuff
J
J̇
f_c
x
v
f
m
x
v
f
m
Constraint
Global and Local

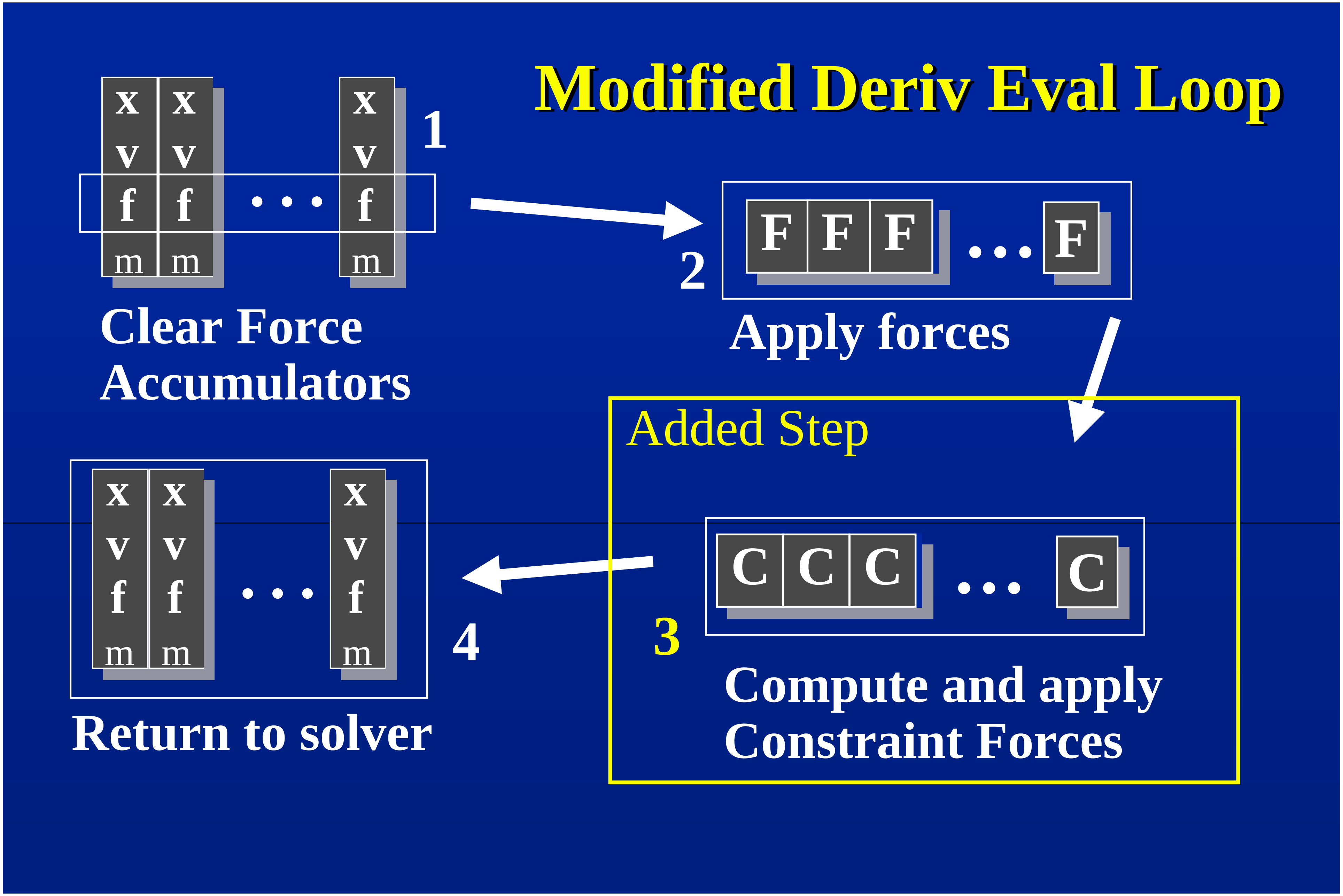
Constraint Structure

Each constraint must know how to compute these



Constrained Particle Systems




Modified Deriv Eval Loop
x
v
f
m
x
v
f
m
...
x
v
f
m
1
Clear Force
Accumulators
2
F
F
F
...
F
Apply forces
Added Step
C
C
C
...
C
3
Compute and apply
Constraint Forces
x
v
f
m
x
v
f
m
...
x
v
f
m
4
Return to solver

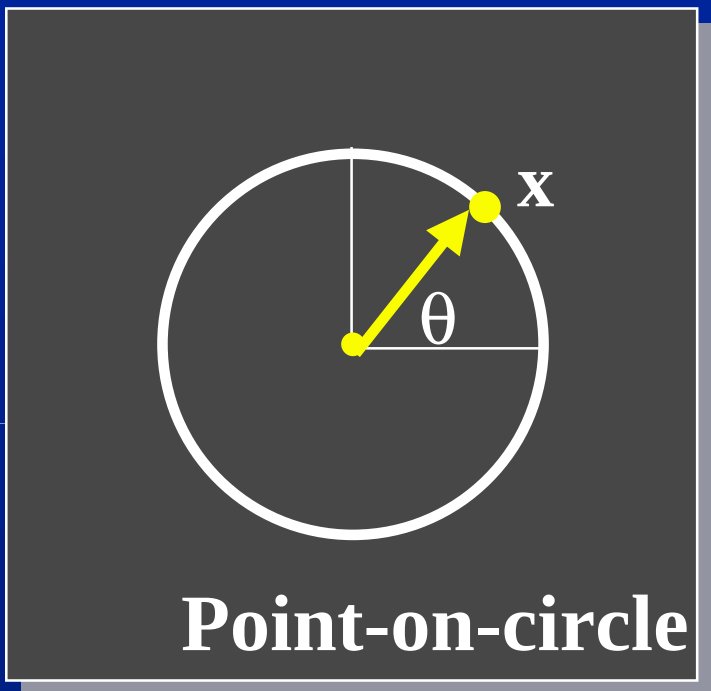
Constraint Force Eval

- 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*.**

A whole other way to do it.



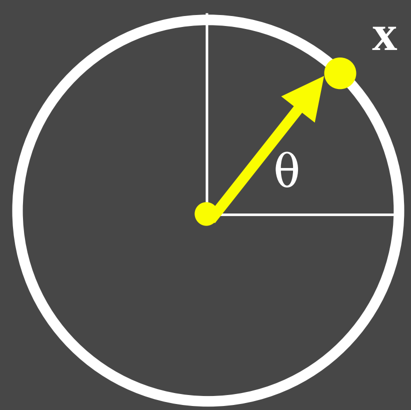
I. Implicit:

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

II. Parametric:

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

Parametric Constraints

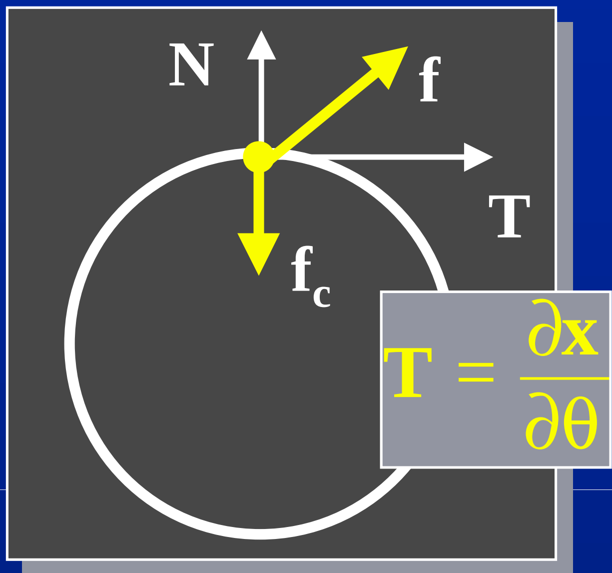


Point-on-circle

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

- **Constraint is always met exactly.**
- **One DOF: θ.**
- **Solve for $\ddot{\theta}$.**

Parametric bead-on-wire (f = mv)



x is not an independent variable.

First step—get rid of it:

$$\dot{\mathbf{x}} = \frac{\mathbf{f} + \mathbf{f}_c}{m}$$ **f = mv *(constrained)***

$$\dot{\mathbf{x}} = \mathbf{T}\dot{\theta}$$ **chain rule**

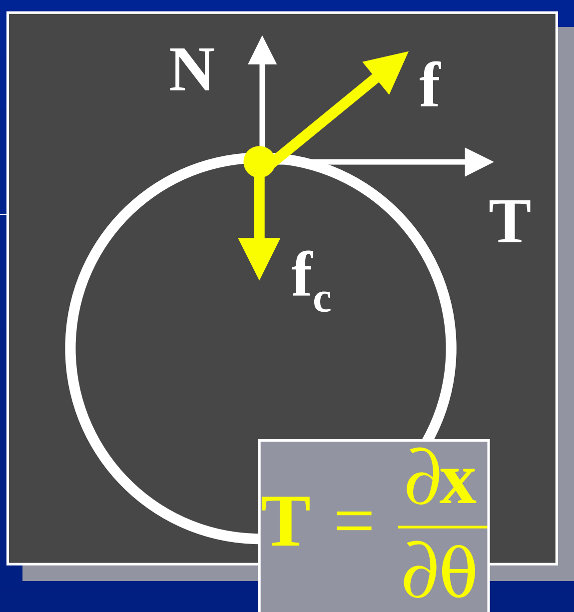
$$\mathbf{T}\dot{\theta} = \frac{\mathbf{f} + \mathbf{f}_c}{m}$$ **combine**

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**
$T \cdot T\dot{\theta} = \frac{T \cdot f + \cancel{T \cdot f_c}}{m}$	**blam!**
$\dot{\theta} = \frac{1}{m}\frac{T \cdot f}{T \cdot T}$	**rearrange.**

Parametric Constraints: Summary

- **Generalizations: f = ma, particle systems**
  - **Like implicit case (see notes.)**
- **Big advantages:**
  - **Fewer DOF's.**
  - **Constraints are always met.**
- **Big *dis*advantages:**
  - **Hard to formulate constraints.**
  - **No easy way to *combine* constraints.**
- **Offical name: *Lagrangian dynamics*.**

Things to try at home:

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