Linear-Time Dynamics Using Lagrange Multipliers

David Baraff
CMU

Presentation by: Elif Tosun
Outline

- Introduction
- Motivation
- Lagrange Multiplier Formulation
- Approach
- Sparse Solution
- Auxiliary Constraints
- Results
Introduction

Problem: “Forward simulation with constraints”

Properties:
- Sparse constraints for articulated figures
  n bodies, O(n) constraints
- Nearly or completely acyclic systems - robot arms, humans...

Methods:
7. *Reduced Coordinate*: Reduce the # of coordinates needed to describe system's state
8. *Constraint Forces*: Introduce additional forces into system to maintain constraints (*)
Very Basic Example

- Given: n bodies

- Primary Constraints
  - n-1
  - Each between 2 bodies
  - Acyclic
  - Can be non-holonomic (can be velocity dependent)
  - Must be equality
Example

- Given: \( n \) bodies
- Auxiliary Constraints
  - Closed loops
Example

- Given: \( n \) bodies

- Auxiliary Constraints
  - Closed loops
  - Between 3+ bodies
Example

- Given: \( n \) bodies

- Auxiliary Constraints
  - Closed loops
  - Between 3+ bodies
  - Inequality Constraints
Overview of method

- Direct (non-iterative) method
- Constraints can be of various dimensions
- Bodies need not be rigid
- Uses a very simple sparse-matrix technique
- No steep learning curve
- Easy to implement
- Auxiliary method can be used with reduced coordinate methods too!
Motivation

- Given: a system with \( m \) d.o.f (called "maximal coordinates") + set of constraints that remove \( c \) d.o.f.

**Reduced Coordinate Methods:**

**Method:** Parameterize remaining \( n=m-c \) d.o.f. using reduced set of \( n \) coordinates (called "generalized coordinates")

**Cons:**
- parameterization very hard
- if one can be found - need \( O(n^3) \) time needed for acceleration computation.

**Pros:**
- loop-free articulated bodies - \( O(n) \) time achievable
- Eliminates drifting problem in multiplier methods (need constraint stabilization)
- “may” run faster due to larger time steps taken by integrator
Motivation

Lagrange Multiplier Methods:

Method:
- use the maximal coordinates \( m \) d.o.f.) + *new* constraint forces.
- Basis for constraint forces known apriori.
- Lagrange multipliers: vector of scalar coordinates for linear combination

Pros:
- Allow an arbitrary set of constraints to be combined.
- Allow/encourage highly modular knowledge/software design (bodies, constraints, geometry)
- Handle non-holonomic constraints(e.g. velocity-dependent)
- No need for parameterization

Cons:
- solving a \( O(n) \times O(n) \) system.

BUT: This method takes linear time!
Lagrange Multiplier Formulation

- **Bodies:** \[ M \ddot{v} = F \]

- **Constraints:** \[ j_{i1} \ddot{v}_1 + \cdots + j_{ik} \ddot{v}_k + \cdots + j_{in} \ddot{v}_n + c_i = 0 \]
  - Linear condition on the acceleration
  - For primary: Only 2 \( j_{ik} \) will be non-zero for constraint \( i \).
  - All constraints: \[ \mathbf{J} \ddot{v} + c = 0 \]
  - “Workless force”

\[
F^c_i = \begin{pmatrix}
\mathbf{j}_{i1}^T \\
\vdots \\
\mathbf{j}_{in}^T
\end{pmatrix} \lambda_i \quad \Rightarrow \quad F^c = \mathbf{J}^T \mathbf{l}
\]

*Want to find \( l \) s.t. constraint force + ext force produce motion that also satisfies constraints*
Lagrange Multiplier Formulation

\[ \mathbf{M} \ddot{\mathbf{v}} = \mathbf{F} \]

\[ \mathbf{M} \ddot{\mathbf{v}} = \mathbf{F}^c = \mathbf{J}^T \mathbf{I} \]

\[ \dot{\mathbf{v}} = \mathbf{M}^{-1} \mathbf{J}^T \mathbf{I} \]

\[ \mathbf{J} \dot{\mathbf{v}} + \mathbf{c} = 0 \]

\[ \mathbf{J} \mathbf{M}^{-1} \mathbf{J}^T \lambda = \mathbf{c} \]
Formulation

- Block-matrix formulation

- Block dimensions: based on body and constraint dimensions:
  - Body dim: # d.o.f. when unconstrained
  - Constraint dim: # of d.o.f. it removes

- Let $p =$ largest dimension among bodies. Block operations take const time.
Approach

\[ M\dot{v} = J^T \lambda + F^{\text{ext}} \]
\[ \dot{v} = M^{-1} J^T \lambda + M^{-1} F^{\text{ext}} \]

\[ J \dot{v} + c = 0 \]

\[ J(M^{-1} J^T \lambda + M^{-1} F^{\text{ext}}) + c = 0 \]

\[ A = JM^{-1} J^T \quad \text{and} \quad b = -(JM^{-1} F^{\text{ext}} + c) \]

Now all we need is to solve is: \[ A l = b \]
\[ A \mathbf{I} = \mathbf{b} \]

- if \( A \) not too large: Cholesky
- if Serial chain - Banded Cholesky – can take \( O(n) \)

\[
\begin{align*}
J & = \\
& \begin{bmatrix}
X & X & X \\
X & X & X \\
X & X & X \\
\end{bmatrix} \\
M & = \\
& \begin{bmatrix}
Y & & & \\
Y & Y & & \\
& Y & Y & \end{bmatrix} \\
JM^{-1}J^T & = \\
& \begin{bmatrix}
A & A & 0 & 0 \\
A & A & A & 0 \\
0 & A & A & A \\
0 & 0 & A & A \\
\end{bmatrix}
\end{align*}
\]
\[ A l = b \]

- If branched – A is completely dense… \( O(n^3) \)

\[
J = \begin{bmatrix}
X & X & X \\
X & X & X \\
X & X & X \\
\end{bmatrix}
\]

\[
M = \begin{bmatrix}
Y & Y & Y \\
Y & Y & Y \\
\end{bmatrix}
\]

\[
JM^{-1}J^T = \begin{bmatrix}
\end{bmatrix}
\]

Need a sparse formulation!
Sparse Formulation

\[
\begin{pmatrix}
M & -J^T \\
-J & 0
\end{pmatrix} \begin{pmatrix}
y \\
\lambda
\end{pmatrix} = \begin{pmatrix}
0 \\
-b
\end{pmatrix}
\]

My - J^Tl = 0 \implies y = M^{-1}J^Tl

-Jy = -b \implies JM^{-1}J^Tl = b \implies Al = b

- Robotics/mech.eng literature
- H is **always** sparse
- Can be solved in O(n)
Non-Linear Solution

- Consider graph of $H$:

- Where the matrix looks like:

- Factor $H = LDL^T (O(n^3))$

- Then solve $(O(n^2))$

$H = LDL^T x = (0; -b)$
Linear Solution

- **Sparse Matrix Theory:**

  "A matrix whose graph is acyclic possesses a **perfect elimination order**"

  ➔ H can be reordered s.t. when factored the matrix L will be just as sparse as H.

\[
H = \begin{pmatrix}
M_1 & 0 & 0 & 0 & 0 & 0 & j_{11}^T & 0 & 0 & j_{41}^T & 0 \\
0 & M_2 & 0 & 0 & 0 & 0 & j_{12}^T & j_{12}^T & j_{22}^T & j_{32}^T & 0 & 0 \\
0 & 0 & M_3 & 0 & 0 & 0 & 0 & j_{23}^T & 0 & 0 & 0 \\
0 & 0 & 0 & M_4 & 0 & 0 & 0 & 0 & j_{34}^T & 0 & 0 \\
0 & 0 & 0 & 0 & M_5 & 0 & 0 & 0 & 0 & j_{45}^T & j_{55}^T \\
0 & 0 & 0 & 0 & 0 & M_6 & 0 & 0 & 0 & 0 & j_{56}^T \\
j_{11} & j_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & j_{22} & j_{23} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & j_{32} & 0 & j_{34} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
j_{41} & 0 & 0 & j_{45} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & j_{55} & j_{56} & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]
Linear Solution

- Sparse Matrix Theory:

“A matrix whose graph is acyclic possesses a "perfect elimination order``

- H can be reordered s.t. when factored the matrix L will be just as sparse as H.

\[
H = \begin{pmatrix}
M_1 & 0 & 0 & 0 & 0 & 0 & j_{11}^T & 0 & 0 & j_{41}^T & 0 \\
0 & M_2 & 0 & 0 & 0 & 0 & j_{12}^T & j_{22}^T & j_{32}^T & 0 & 0 \\
0 & 0 & M_3 & 0 & 0 & 0 & 0 & j_{23}^T & 0 & 0 & 0 \\
0 & 0 & 0 & M_4 & 0 & 0 & 0 & 0 & j_{34}^T & 0 & 0 \\
0 & 0 & 0 & 0 & M_5 & 0 & 0 & 0 & 0 & i_{45}^T & i_{55}^T \\
0 & 0 & 0 & 0 & 0 & M_6 & 0 & 0 & 0 & 0 & 0 \\
0 & j_{11} & j_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & j_{22} & j_{23} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & j_{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & j_{41} & 0 & 0 & j_{45} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & j_{55} & j_{56} & 0 & 0 & 0 & 0
\end{pmatrix}
\]

\[
\begin{pmatrix}
M_3 & j_{23}^T & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
j_{23} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & M_4 & j_{34}^T & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & j_{34} & 0 & j_{32} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & j_{22}^T & 0 & j_{32}^T & M_2 & j_{12}^T & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & j_{12} & 0 & j_{11} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & j_{11}^T & M_1 & j_{41}^T & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & j_{41} & 0 & j_{45} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & j_{45}^T & M_5 & 0 & j_{55}^T \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & j_{55} & j_{56} & 0
\end{pmatrix}
\]
Ordering

- Graph is a rooted tree: parent child relationship between every edge.
- Every node's index greater than its children's indices. (DFS)

$L$ can be computed in $O(n)$ time and $LDL^T x = (b)$ can be solved in $O(n)$ time.

$$\hat{v} = M^{-1} (J^T \lambda + F^{ext})$$
Auxiliary Constraints

Idea:

- While computing multipliers for auxiliary constraints “anticipate” responses of primary constraints due to auxiliary forces.
- Once auxiliary constraints are computed, add primary constraints into system, which won't violate auxiliary constraints due to “anticipation”

Notation:

- For auxiliary constraints
  - $K \sim J$
  - $m \sim l$
Full Algorithm

1. Formulate sparse matrix $H$ for primary constraints, and factor $O(n)$

2. Given $F_{\text{ext}}$,
   - Solve $A I_1 = -(JM^{-1}F_{\text{ext}}+c)$ for $I_1$
   - Compute primary constraint force due to $F_{\text{ext}}$:
     $J^T I_1$
   - Compute system’s acceleration without auxiliary constraints:
     $\mathbf{v}^{\text{aux}} = M^{-1}(J^T I_1 + F_{\text{ext}})$. $O(n)$
$J^a \dot{v} + c^a = a$
$J^a (M^{-1}(F_{\text{resp}} + k_i)m + J^a v_{\text{aux}}) + c^a = a.$

3. For each of $k$ auxiliary constraints
   - Solve $A I_m = -(JM^{-1}k_m)$
   - Compute response force by primary to constraint $k_m$
     $F_{\text{resp}} = J^T I_m$
   - Fill coefficient matrix $O(kn)$

4. Solve for auxiliary multipliers $m$
   $O(k^3)$
5. Given \( K_m \) (the auxiliary constraint force)
Solve \( A I_{\text{final}} = -(JM^{-1}(K_m + F^\text{ext}) + c) \) \( \bigO(n) \)
Compute primary constraints response to \( K_m + F^\text{ext} \)

Total constraint force = \( K_m + J^T I_{\text{final}} \)
Total external forces = \( F^\text{ext} \)
NET FORCE acting = \( K_m + J^T I_{\text{final}} + F^\text{ext} \)

6. Solve for net acceleration of system and move to next step.

TOTAL RUNNING TIME: \( \bigO(n) + \bigO(kn + k^3) \)
Results

- Run on SGI indigo 250Mhz, R4400 processors
  - 2D system: 54 primary constraints 7.75ms
  - 96 primary, 12 auxiliary ~25ms
  - 3D system: 96 primary, 3 auxiliary 18 ms
  - 127 primary: ~45ms

Comparisons:
- Baraff (Lagrange, O(n^3))
  - For smaller systems 2 fold improvement
  - For bigger systems as big as a factor of 40
- Schröder (Reduced Coordinate, O(n))
  - Competitive + no need for SVD which causes ill-conditioning.
- Bramble (Lagrange, Iterative, O(n^2))
  - For smaller matrices competitive, for larger clearly faster