

# Lemke's Algorithm: The Hammer in Your Math Toolbox?

Chris Hecker  
definition six, inc.  
checker@d6.com

# First, a Word About Hammers

“If the only tool you have is a hammer, you tend to see every problem as a nail.”

Abraham Maslow

- requirements for this to be a good idea
  - a way of transforming problems into nails (MLCPs)
  - a hammer (Lemke’s algorithm)
- lots of advanced info + one hour = something has to give
  - majority of lecture is motivating you to care about the hammer by showing you how useful nails can be
  - make you hunger for more info post-lecture
  - very little on how the hammer works in this hour

## Hammers (cont.)

- by definition, not the optimal way to solve problems, BUT
  - computers are very fast these days
  - often don't care about optimality
    - prepro, prototypes, tools, not a profile hotspot, etc.
  - can always move to optimal solution after you verify it's a problem you actually want to solve

# What are “advanced game math problems”?

- problems that are ammenable to mathematical modeling
  - state the problem clearly
  - state the desired solution clearly
  - describe the problem with equations so a proposed solution’s quality is measurable
  - figure out how to solve the equations
- why not hack it?
  - I believe better modeling is the future of game technology development (consistency, not reality)

# Prerequisites

- linear algebra
  - vector, matrix symbol manipulation at least
- calculus concepts
  - what derivatives mean
- comfortable with math notation and concepts

# Overview of Lecture

- random assortment of example problems briefly mentioned
- 5 specific example problems in some depth
  - including one that I ran into recently and how I solved it
- generalize the example models
- transform them all to MLCPs
- solve MLCPs with Lemke's algorithm

# A Look Forward

- linear equations  
 $Ax = b$
- linear inequalities  
 $Ax \geq b$
- linear programming  
 $\min c^T x$   
s.t.  $Ax \geq b$ , etc.
- quadratic programming  
 $\min \frac{1}{2} x^T Q x + c^T x$   
s.t.  $Ax \geq b$   
 $Dx = e$
- linear complementarity problem  
 $a = Af + b$   
 $a \geq 0, f \geq 0$   
 $a_i f_i = 0$

# Applications to Games

graphics, physics, ai, even ui

- computational geometry
- visibility
- contact
- curve fitting
- constraints
- integration
- graph theory
- network flow
- economics
- site allocation
- game theory
- IK
- machine learning
- image processing



# Applications to Games (cont.)

- don't forget...
  - The Elastohydrodynamic Lubrication Problem
  - Solving Optimal Ownership Structures
    - “The two parties establish a relationship in which they exchange feed ingredients,  $q$ , and manure,  $m$ .”

# Specific Examples #1a: Ease Cubic Fitting

- warm up with an ease curve cubic

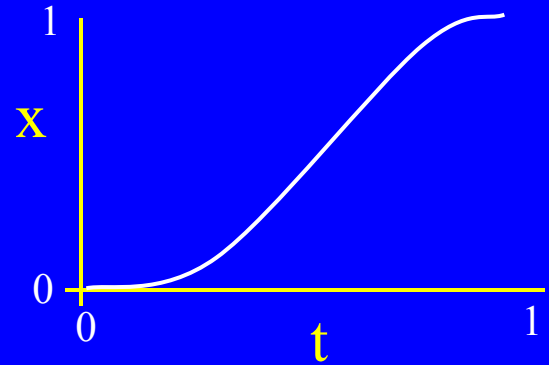
$$x(t) = at^3 + bt^2 + ct + d$$

$$x'(t) = 3at^2 + 2bt + c$$

- 4 unknowns  $a, b, c, d$  (DOFs) we get to set, we choose:

$$x(0) = 0, x(1) = 1$$

$$x'(0) = 0, x'(1) = 0$$



# Specific Examples #1a: Ease Cubic Fitting (cont.)

- $x(t)=at^3+bt^2+ct+d, \quad x'(t)=3at^2+2bt+c$
- $x(0) = a0^3+b0^2+c0+d = d = 0$
- $x(1) = a1^3+b1^2+c1+d = a+b+c+d = 1$
- $x'(0) = 3a0^2+2b0+c = c = 0$
- $x'(1) = 3a1^2+2b1+c = 3a + 2b + c = 0$

## Specific Examples #1a: Ease Cubic Fitting (cont.)

- $d = 0, a+b+c+d = 1, c = 0, 3a + 2b + c = 0$
- $a+b=1, 3a+2b=0$
- $a=1-b \Rightarrow 3(1-b)+2b = 3-3b+2b = 3-b = 0$
- $b=3, a=-2$
- $x(t) = 3t^2 - 2t^3$

# Specific Examples #1a: Ease Cubic Fitting (cont.)

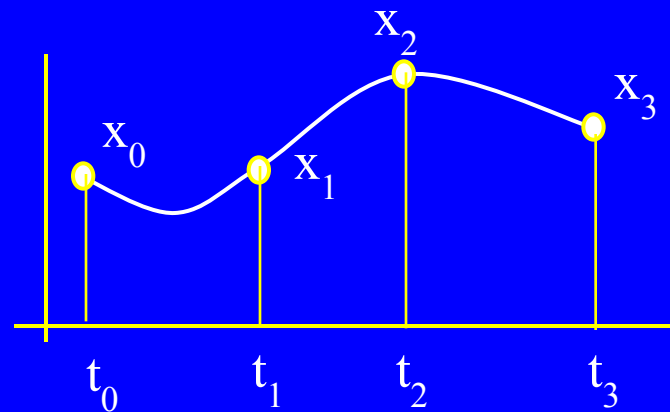
- or,
- $x(0) = d = 0$
- $x(1) = a + b + c + d = 1$
- $x'(0) = c = 0$
- $x'(1) = 3a + 2b + c = 0$

$$\begin{array}{c|c|c|c|c|c} x(0) & 0 & 0 & 0 & 1 & a & 0 \\ x(1) & 1 & 1 & 1 & 1 & b & 1 \\ x'(0) & 0 & 0 & 1 & 0 & c & 0 \\ x'(1) & 3 & 2 & 1 & 0 & d & 0 \end{array} = \begin{array}{c} 0 \\ 1 \\ 0 \\ 0 \end{array} \quad (\text{can solve for any rhs})$$

$Ax = b$ , a system of **linear equations**

# Specific Examples #1b: Cubic Spline Fitting

- same technique to fit higher order polynomials, but they “wiggle”
- piecewise cubic is better “natural cubic spline”
- $x_i(t_i) = x_i$       $x_i(t_{i+1}) = x_{i+1}$   
 $x'_i(t_i) - x'_{i-1}(t_i) = 0$   
 $x''_i(t_i) - x''_{i-1}(t_i) = 0$
- there is coupling between the splines, must solve simultaneously



- 4 DOF per spline
  - 2 endpoint eqns per spline
  - 4 derivative eqns for inside points
  - 2 missing eqns = endpoint slopes

# Specific Examples #1b: Cubic Spline Fitting (cont.)

$$x_i(t_i) = x_i \quad x_i(t_{i+1}) = x_{i+1}$$

$$x'_{i+1}(t_i) - x'_{i-1}(t_i) = 0$$

$$x''_{i+1}(t_i) - x''_{i-1}(t_i) = 0$$

	$=$	$  \begin{bmatrix}  a_0 \\  b_0 \\  c_0 \\  d_0 \\  a_1 \\  b_1 \\  c_1 \\  d_1 \\  \vdots \\  \vdots  \end{bmatrix}  $	$=$	$  \begin{bmatrix}  x_0 \\  x_1 \\  0 \\  0 \\  x_1 \\  x_2 \\  0 \\  0 \\  \vdots \\  \vdots  \end{bmatrix}  $	<p> <math>Ax = b</math>, a            system of  <b>linear equations</b> </p>
--	-----	---	-----	---	---

# Specific Examples #2: Minimum Cost Network Flow

- what is the cheapest flow route(s) from sources to sinks?
- model, want to minimize cost

$c_{ij}$  = cost of  $i$  to  $j$  arc

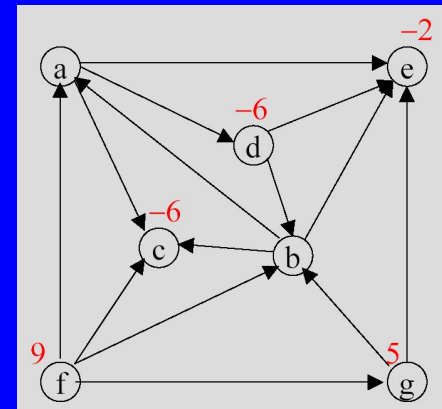
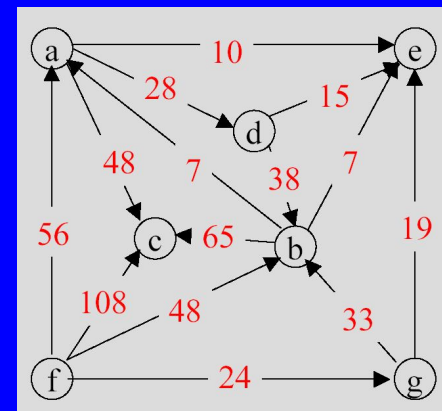
$b_i$  =  $i$ 's supply/demand,  $\sum(b_i)=0$

$x_{ij}$  = quantity shipped on  $i$  to  $j$  arc

$x_{*k} = \sum(x_{ik})$  = flow into  $k$

$x_{k*} = \sum(x_{ki})$  = flow out of  $k$

- flow balance:  $x_{*k} - x_{k*} = -b_k$
- one-way streets:  $x_{ij} \geq 0$



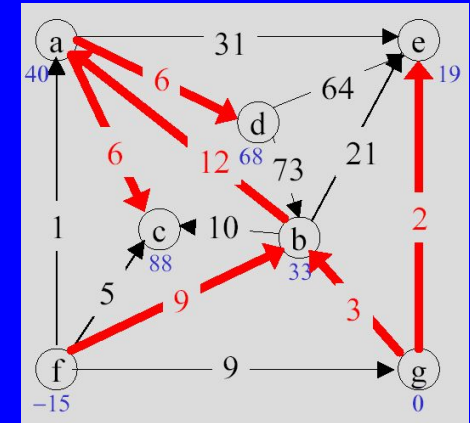


# Specific Examples #2: Minimum Cost Network Flow (cont.)

- **min cost: minimize  $c^T x$** 
  - the sum of the costs times the quantities shipped ( $c^T x = c \cdot x$ )
- **flow balance is coupling: matrix**

$$x_{*k} - x_{k*} = -b_k$$

$$\begin{array}{c}
 \left| \begin{array}{cccccccc}
 -1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & \dots \\
 0 & 0 & 0 & -1 & -1 & -1 & 1 & \dots & & & \\
 \dots & & & & & & & & & & 
 \end{array} \right|
 \begin{array}{c}
 X_{ac} \\
 X_{ad} \\
 X_{ae} \\
 X_{ba} \\
 X_{bc} \\
 X_{be} \\
 X_{db} \\
 \vdots \\
 \vdots
 \end{array}
 = -
 \begin{array}{c}
 \left| \begin{array}{c}
 b_a \\
 b_b \\
 b_c \\
 b_d \\
 \vdots \\
 \vdots
 \end{array} \right|
 \end{array}$$



minimize  $c^T x$   
 subject to  
 $Ax = -b$   
 $x \geq 0$   
 a linear programming problem

# Specific Examples #3: Points in Polys

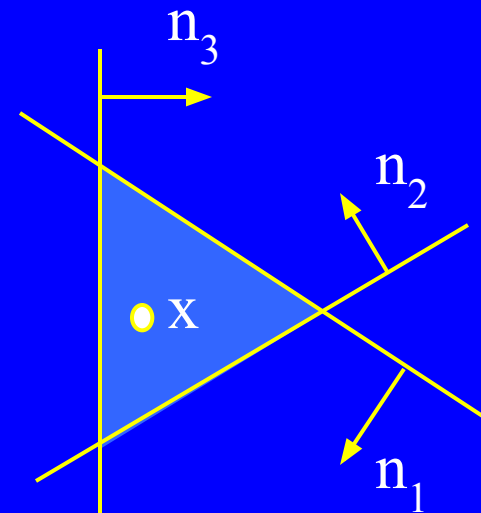
- point in convex poly defined by planes

$$n_1 \cdot x \geq d_1$$

$$n_2 \cdot x \geq d_2$$

$$n_3 \cdot x \geq d_3$$

$Ax \geq b$ ,  
linear inequality



- farthest point in a direction in poly,  $c$ :

$$\min -c^T x$$

$$\text{s.t. } Ax \geq b$$

linear programming

# Specific Examples #3: Points in Polys (cont.)

- closest point in two polys

$$\min (x_2 - x_1)^2$$

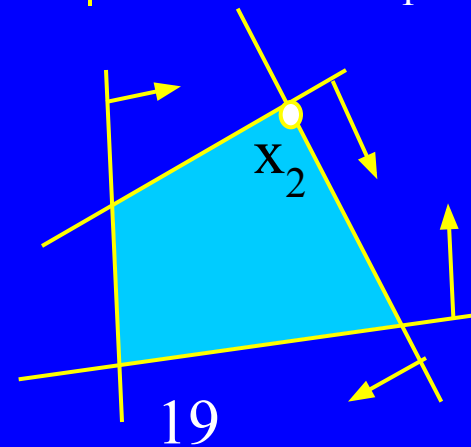
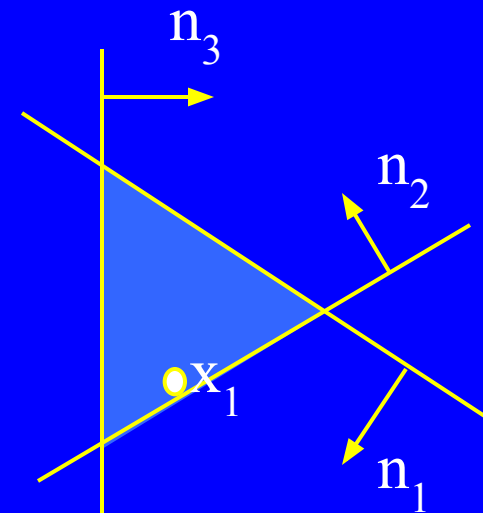
$$\text{s.t. } A_1 x_1 \geq b_1$$

$$A_2 x_2 \geq b_2$$

- stack 'em in blocks,  $Ax \geq b$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad A = \begin{bmatrix} A_1 & A_2 \end{bmatrix}$$

what about  $(x_2 - x_1)^2$ , how do we stack it?



## Specific Examples #3: Points in Polys (cont.)

- how do we stack  $x_1, x_2$  into single  $x$  given  
 $(x_2 - x_1)^2 = x_2^2 - 2x_2 \cdot x_1 + x_1^2$

$$\begin{vmatrix} x_1^T & x_2^T \end{vmatrix} \begin{vmatrix} 1 & -1 \\ -1 & 1 \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \end{vmatrix} = x_2^2 - 2x_2 \cdot x_1 + x_1^2 = x^T Q x$$

$$\begin{aligned} \min x^T Q x \\ \text{s.t. } Ax \geq b \end{aligned}$$

$$\begin{aligned} x^2 &= x^T x = x \cdot x \\ 1 &= \text{identity matrix} \end{aligned}$$

a quadratic programming problem

## Specific Examples #3: Points in Polys (cont.)

- more points, more polys!

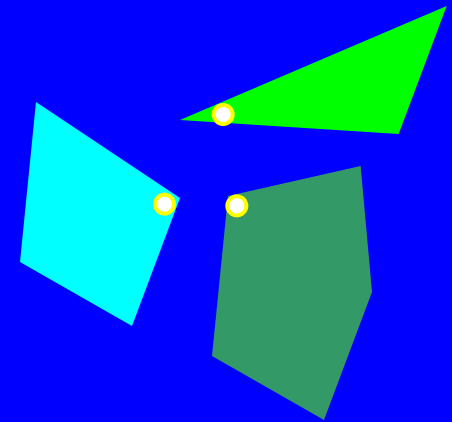
$$\min (x_2 - x_1)^2 + (x_3 - x_2)^2 + (x_3 - x_1)^2$$

$$\begin{array}{c|ccc|c} |x_1^T & 2 & -1 & -1 & |x_1 \\ |x_2^T & -1 & 2 & -1 & |x_2 \\ |x_3^T & -1 & -1 & 2 & |x_3 \end{array} = x^T Q x$$

$$\min x^T Q x$$

$$\text{s.t. } Ax \geq b$$

another quadratic programming problem



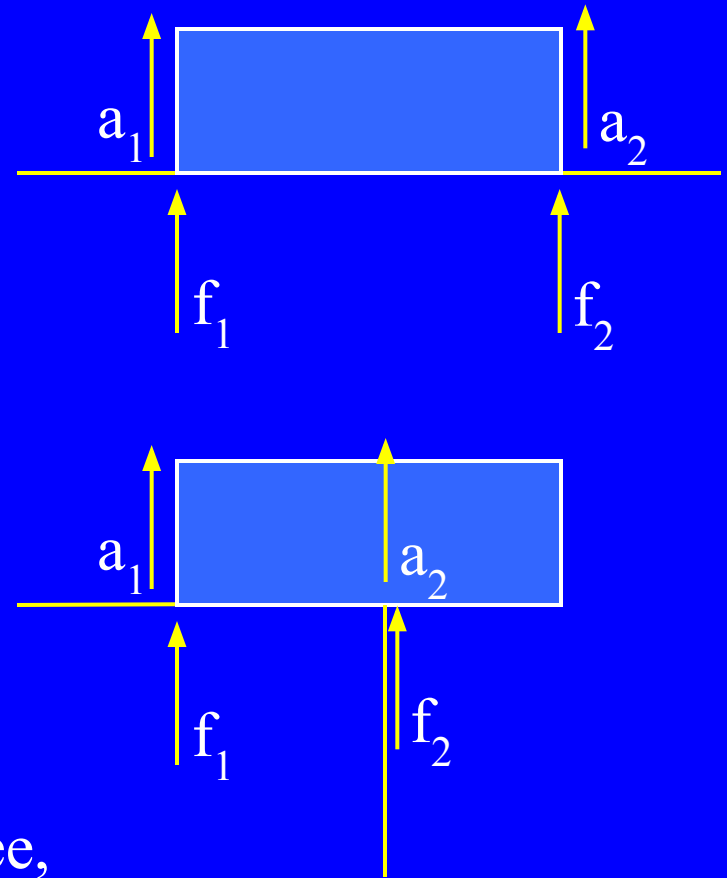
- same form for all these poly problems
- never specified 2d, 3d, 4d, nd!

# Specific Examples #4: Contact

- model like IK constraints  
 $a = Af + b$   
 $a \geq 0$ , no penetrating  
 $f \geq 0$ , no pulling  
 $a_i f_i = 0$ , complementarity  
(can't push if leaving)

linear complementarity problem

it's a **mixed LCP** if some  $a_i = 0$ ,  $f_i$  free,  
like for equality constraints



# Specific Examples #5: Joint Limits in CCD IK

- how to do child-child constraints in CCD?
  - parent-child are easy, but need a way to couple two children to limit them relative to each other

- how to model this & handle all the cases?

- define  $d_n = g_n - a_n$

- $\min (x_1 - d_1)^2 + (x_2 - d_2)^2$

- s.t.  $c_{1\min} \leq a_1 + x_1 - a_2 - x_2 \leq c_{1\max}$

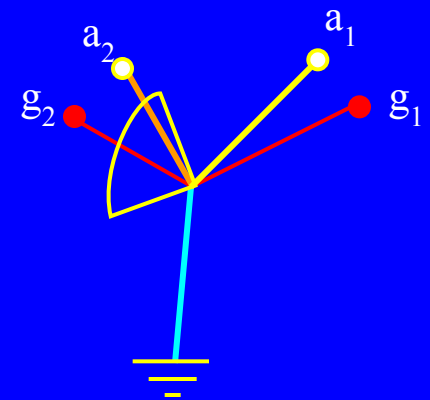
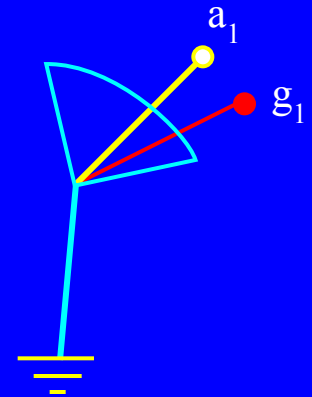
- parent-child are easy in this framework:

$$c_{2\min} \leq a_1 + x_1 \leq c_{2\max}$$

- another quadratic program:

$$\min x^T Q x$$

$$\text{s.t. } A x \geq b$$



# What Unifies These Examples?

- linear equations  
 $Ax = b$
- linear inequalities  
 $Ax \geq b$
- linear programming  
 $\min c^T x$   
s.t.  $Ax \geq b$ , etc.
- quadratic programming  
 $\min \frac{1}{2} x^T Q x + c^T x$   
s.t.  $Ax \geq b$   
 $Dx = e$
- linear complementarity problem  
 $a = Af + b$   
 $a \geq 0, f \geq 0$   
 $a_i f_i = 0$



# QP is a Superset of Most

- quadratic programming  
 $\min \frac{1}{2}x^T Qx + c^T x$   
s.t.  $Ax \geq b$   
 $Dx = e$

but **MLCP** is a superset  
of convex **QP**!

- linear equations
  - $Ax = b$
  - $Q, c, A, b = 0$
- linear inequalities
  - $Ax \geq b$
  - $Q, c, D, e = 0$
- linear programming
  - $\min c^T x$   
s.t.  $Ax \geq b$ , etc.
  - $Q, \text{etc.} = 0$

# Karush-Kuhn-Tucker Optimality

## Conditions get us to MLCP

- for QP  $\min \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x}$   
s.t.  $\mathbf{A} \mathbf{x} - \mathbf{b} \geq 0$
- form “Lagrangian”  $\mathbf{D} \mathbf{x} - \mathbf{e} = 0$

$$L(\mathbf{x}, \mathbf{u}, \mathbf{v}) = \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x} - \mathbf{u}^T (\mathbf{A} \mathbf{x} - \mathbf{b}) - \mathbf{v}^T (\mathbf{D} \mathbf{x} - \mathbf{e})$$

- for optimality (if convex):

$$\partial L / \partial \mathbf{x} = 0$$

$$\mathbf{A} \mathbf{x} - \mathbf{b} \geq 0$$

$$\mathbf{D} \mathbf{x} - \mathbf{e} = 0$$

$$\mathbf{u} \geq 0 \quad \mathbf{u}_i (\mathbf{A} \mathbf{x} - \mathbf{b})_i = 0$$

– this is related to basic calculus  $\min/\max f'(x) = 0$  solve

# Karush-Kuhn-Tucker Optimality Conditions (cont.)

- $L(x,u,v) = \frac{1}{2} x^T Q x + c^T x - u^T (Ax - b) - v^T (Dx - e)$
- $y = \partial L / \partial x = Qx + c - A^T u - D^T v = 0$ ,  $x$  free
- $w = Ax - b \geq 0$ ,  $u \geq 0$ ,  $w_i u_i = 0$
- $s = Dx - e = 0$ ,  $v$  free

$$\begin{array}{c}
 \left| \begin{array}{c} y \\ s \\ w \end{array} \right| = \left| \begin{array}{ccc} Q & -D^T & -A^T \\ D & 0 & 0 \\ A & 0 & 0 \end{array} \right| \left| \begin{array}{c} x \\ v \\ u \end{array} \right| + \left| \begin{array}{c} c \\ -e \\ -b \end{array} \right|
 \end{array}
 \quad
 \begin{array}{l}
 y, s = 0 \\
 x, v \text{ free} \\
 w, u \geq 0 \\
 w_i u_i = 0
 \end{array}$$

# This is an MLCP

$$\begin{array}{c} \left| \begin{array}{c} y \\ s \\ w \end{array} \right| = \left| \begin{array}{ccc} Q & -D^T & -A^T \\ D & 0 & 0 \\ A & 0 & 0 \end{array} \right| \left| \begin{array}{c} x \\ v \\ u \end{array} \right| + \left| \begin{array}{c} c \\ -e \\ -b \end{array} \right| \end{array}$$

$$\begin{array}{l} y, s = 0 \\ x, v \text{ free} \\ w, u \geq 0 \\ w_i u_i = 0 \end{array}$$

$$\begin{array}{c} \downarrow \\ a = A f + b \end{array}$$

$$\begin{array}{l} a_i f_i = 0 \quad \text{some } a \geq 0, \text{ some } = 0 \\ \text{some } f \geq 0, \text{ some free} \\ \text{(but they correspond so complementarity holds)} \end{array}$$

# Modeling Summary

- a lot of interesting problems can be formulated as MLCPs
  - model the problem mathematically
  - transform it to an MLCP
    - on paper or in code with wrappers
  - but what about solving MLCPs?

# Solving MLCPs

(where I hope I made you hungry enough for homework)

- Lemke's Algorithm is only about 2x as complicated as Gaussian Elimination
- Lemke will solve LCPs, which some of these problems transform into
- then, doing an "advanced start" to handle the free variables gives you an MLCP solver, which is just a bit more code over plain Lemke's Algorithm

# Playing Around With MLCPs

- **PATH, a MCP solver (superset of MLCP!)**
  - really stoked professional solver
  - free version for “small” problems
  - matlab or C
- **OMatrix (Matlab clone) free trial (omatrix.com)**
  - only LCPs, but Lemke source is in trial
    - » not a great version, but it’s really small (two pages of code) and quite useful for learning, with debug output
    - » good place to test out “advanced starts”
- **my Lemke’s + advanced start code**
  - not great, but I’m happy to share it
  - it’s in Objective Caml :)

# References for Lemke, etc.

- free pdf book by Katta Murty on LCPs, etc.
- free pdf book by Vanderbei on LPs
- The LCP, Cottle, Pang, Stone
- Practical Optimization, Fletcher
- web has tons of material, papers, complete books, etc.
- email to authors
  - relatively new math means authors are still alive, bonus!





# Specific Examples #5: Constraints for IK

- compute “forces” to keep bones together

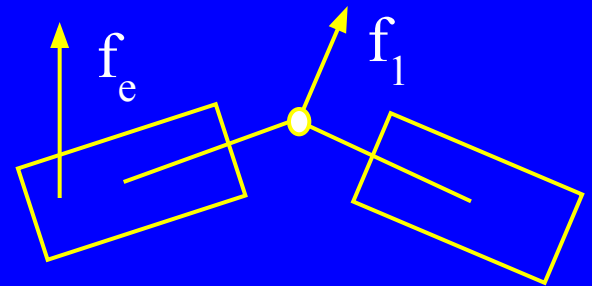
$$a_1 = A_{11} f_1 + b_1$$

$a_1$  : relative acceleration  
at constraint

$f_1$  : force at constraint

$b_1$  : external forces converted to  
accelerations at constraints

$A_{11}$  : force/acceleration relation matrix



# Specific Examples #5: Constraints for IK (cont.)

- multiple bodies gives coupling...

$$\begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$$

$$\mathbf{a} = \mathbf{A}\mathbf{f} + \mathbf{b}$$

$\mathbf{a} = 0$  for rigid constraints

$\mathbf{A}\mathbf{f} = -\mathbf{b}$ , linear equations

