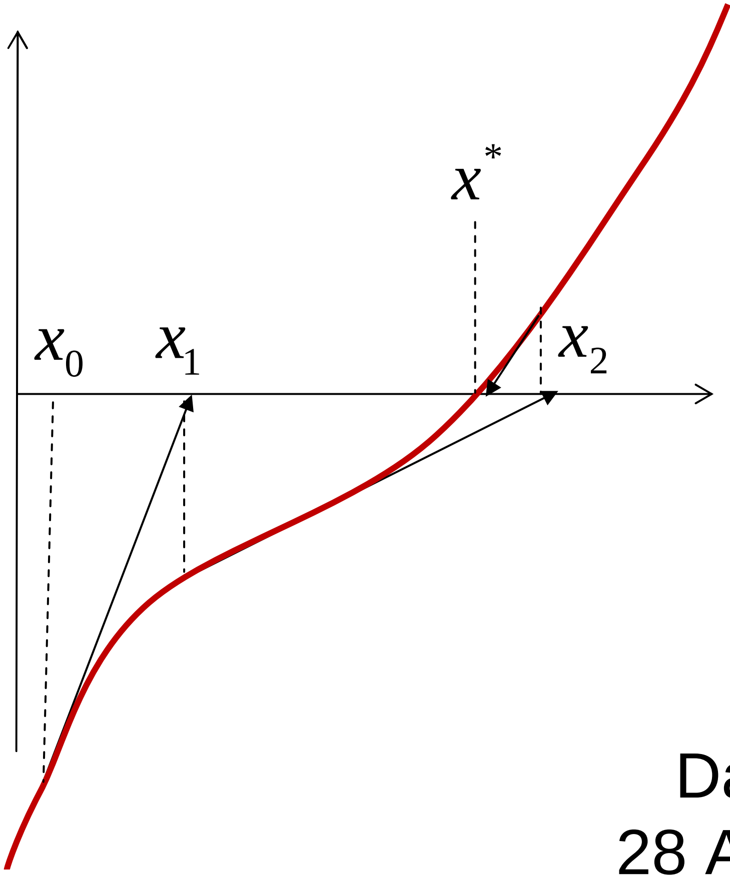
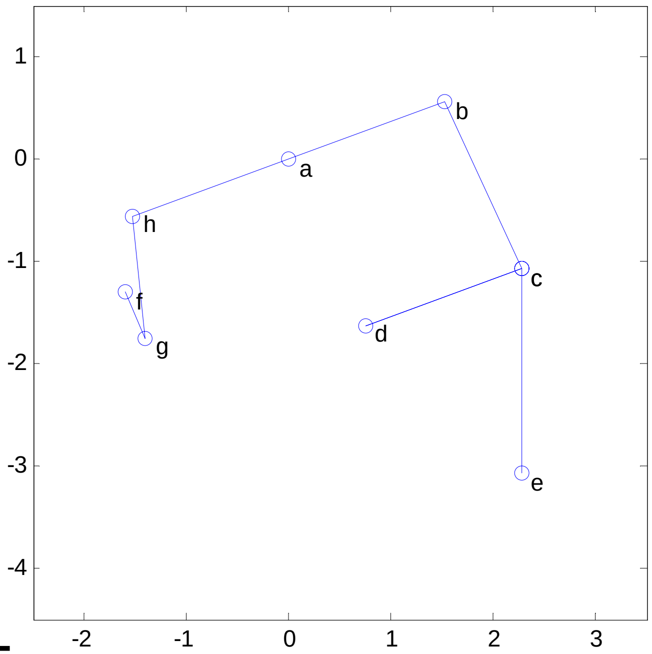
2.007 Design and Manufacturing I
Spring 2009

2.007 –Design and Manufacturing I

Optimization

and Solution of Systems





Dan Frey

28 APR 2009

Today’s Agenda

- Seeding and impounding procedures
- Methods for Solving Systems
  - Newton-Raphson
  - Secant
  - Bisection
- Examples related to mechanism design

Seeding

- Run on the table unopposed
- Timing and set-up as in the actual contest
- Three tries – best of three counts
- Your <u>“seeding card”</u> is essential
  – Get your scores recorded and initialed
  – Don’t lose your card
- “In-lab” competition
  – Basically a way to get round 1 partly finished
  – Same as next Weds but not broadcast

Impounding

- A way to bring the work to an end
- Your machine is checked
  - Safety
  - Wiring
  - Rules issues
- Your <u>“seeding card”</u> is essential
  - Your impound checks are recorded
  - Your card goes in the WOODEN BOX

Linear Systems (Back Solving)

```
A=[1 1 1;
   0 2 3;
   0 0 6];
b=[3; 1; 4];
x(3)=b(3)/A(3,3)
x(2)=(b(2)-x(3)*A(2,3))/A(2,2)
x(1)=(b(1)-x(2)*A(1,2)-x(3)*A(1,3))/A(1,1);
norm(b-A*x')
```

What will happen when I run this code?

Linear Systems (Solving)

```
A=[1 1 1;
   1 2 3;
   1 3 6];
b=[3; 1; 4];
x=A\b
b=[5; 0; -10];
x=A\b
```

What will happen when I run this code?

Linear Systems (Existence of Soln)

```
A=[1 1 1;
    1 2 3;
    1 3 6;
   -1 -1 1];
b=[3; 1; 4; 7];
x=A\b;
norm(b-A*x)
```

What will happen when I run this code?

Linear Systems (Existence of Soln)

```
A=[1 1 1;
    1 2 3;
    1 3 6;
   -1 -1 1];
b=[3; 1; 4; 6];
x=A\b;
norm(b-A*x)
```

What will happen when I run this code?

Linear Systems (Multiple Solutions)

```
A=[1 1 1;
   1 2 3;
   1 3 6;
  -1 -1 1];
b1=[3; 1; 4; 7];
x1=A\b1; norm(b1-A*x1)
b2=[5; 0; -10; -15];
x2=A\b2; norm(b2-A*x2)
```

```
b3=5*b1-2*b2;
x3=A\b3;
norm(b3-A*x3)
norm(x3-(5*x1-2*x2))
```

What will happen when I run this code?

Comparisons

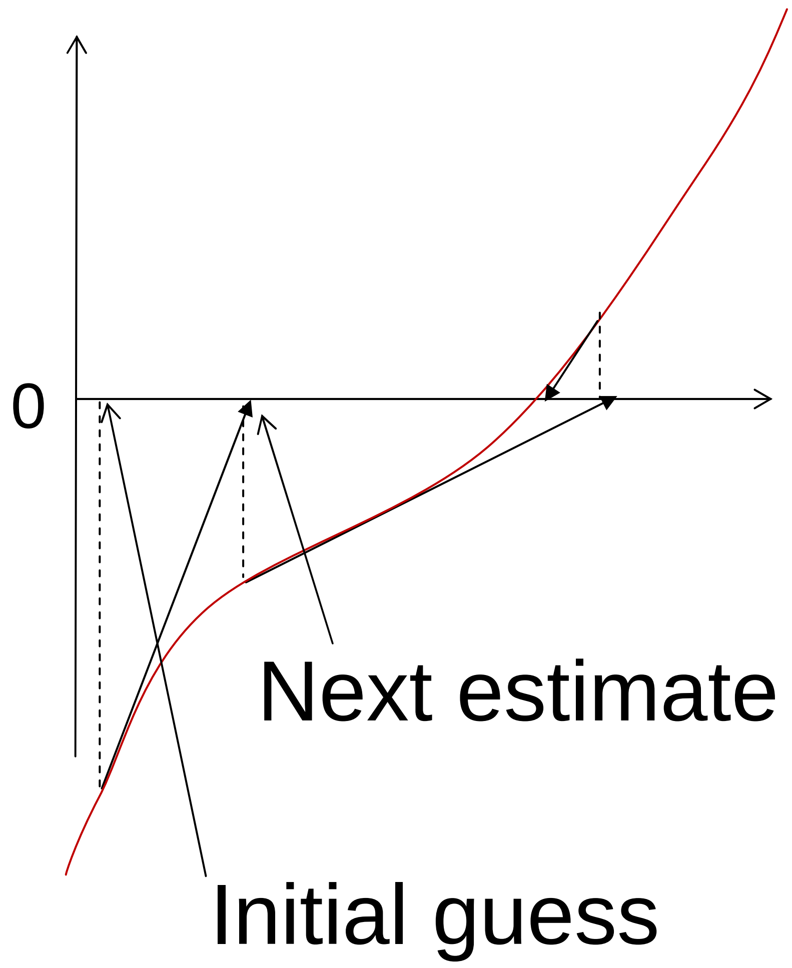
Linear Systems

- Sometimes solved sequentially
- # of equations = # of unknowns
- # of equations > # of unknowns
- When we can find two solutions

Nonlinear systems

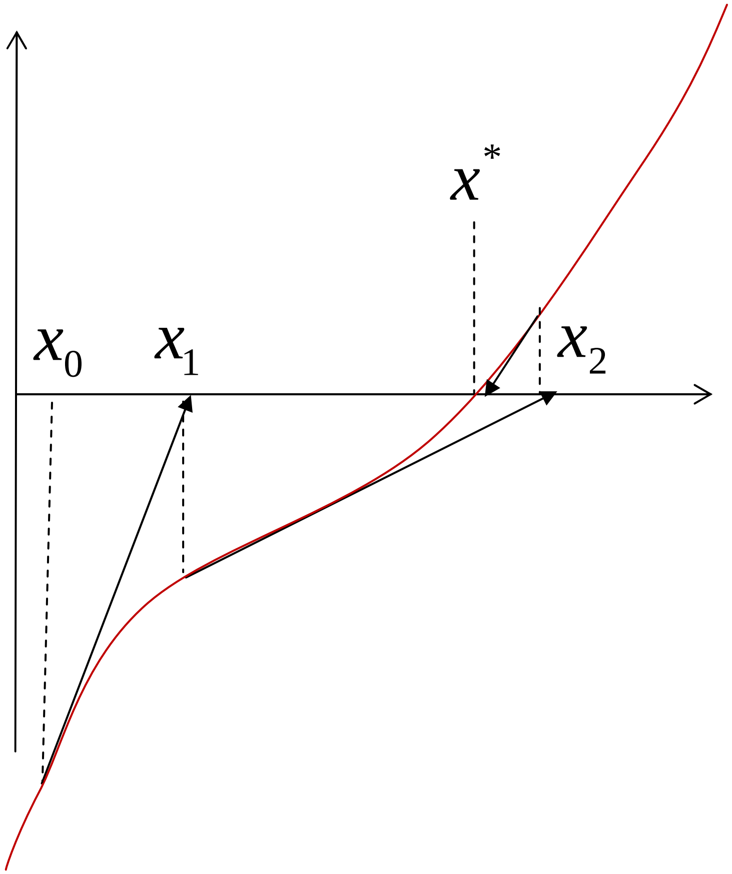
- ?
- ?
- ?
- ?

Newton-Raphson Method



- Make a guess at the solution
- Make a linear approximation of a function by e.g., finite difference
- Solve the linear system
- Use that solution as a new guess
- Repeat until some criterion is met

Newton-Raphson Method



If one equation in one variable

$$x_{k+1} = x_k + \frac{f(x_k)}{f'(x_k)}$$

Generalizing to systems of equations

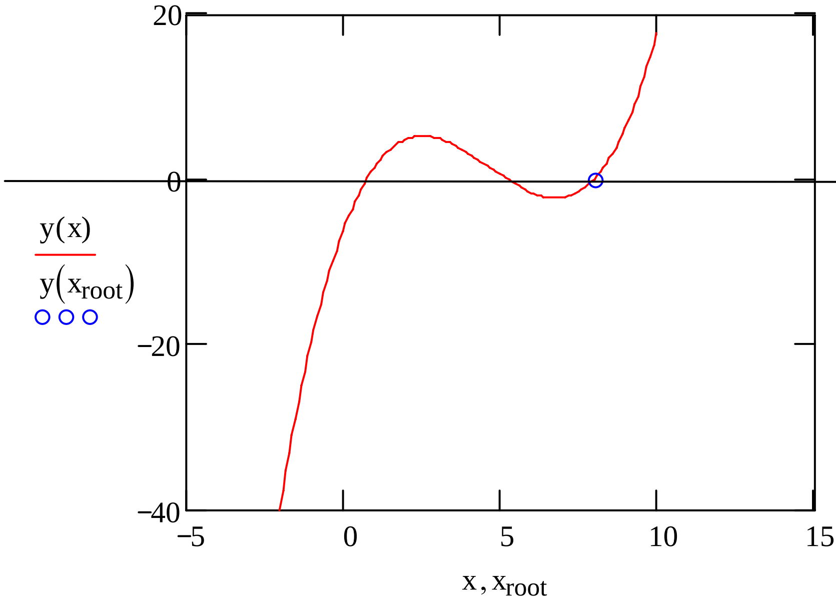
$$\mathbf{J}_F(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\mathbf{F}(\mathbf{x}_k)$$

Solve this system for $\mathbf{x}_{k+1}$

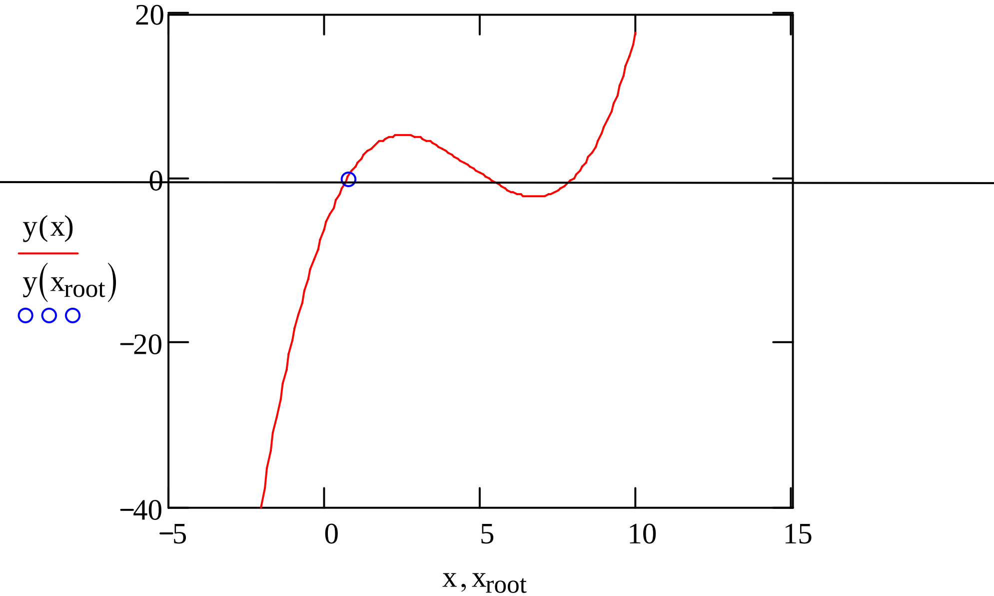
A Fundamental Difficulty

- If there are many solutions, which solution you find will depend on the initial guess

$x_{guess} := 10$ $\quad x_{root} := \mathrm{root}\left(y\left(x_{guess}\right), x_{guess}\right)$



$x_{guess} := -3$ $\quad x_{root} := \mathrm{root}\left(y\left(x_{guess}\right), x_{guess}\right)$

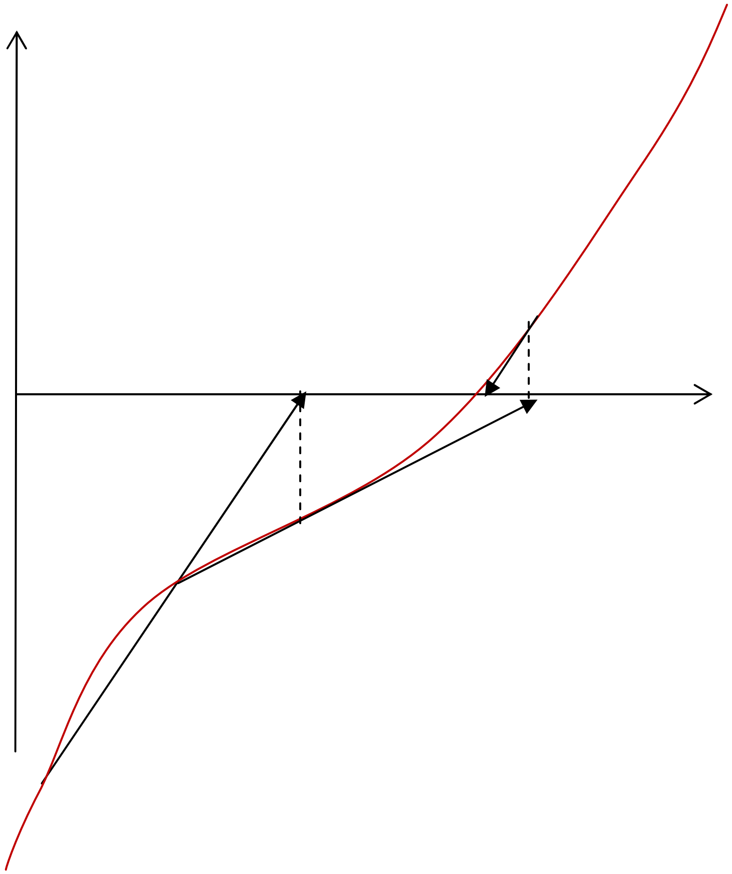


If you seek to find a root of a function $f(x)$, and you use the Newton-Raphson method.

Choose all the numbers corresponding to outcomes that are NOT possible :

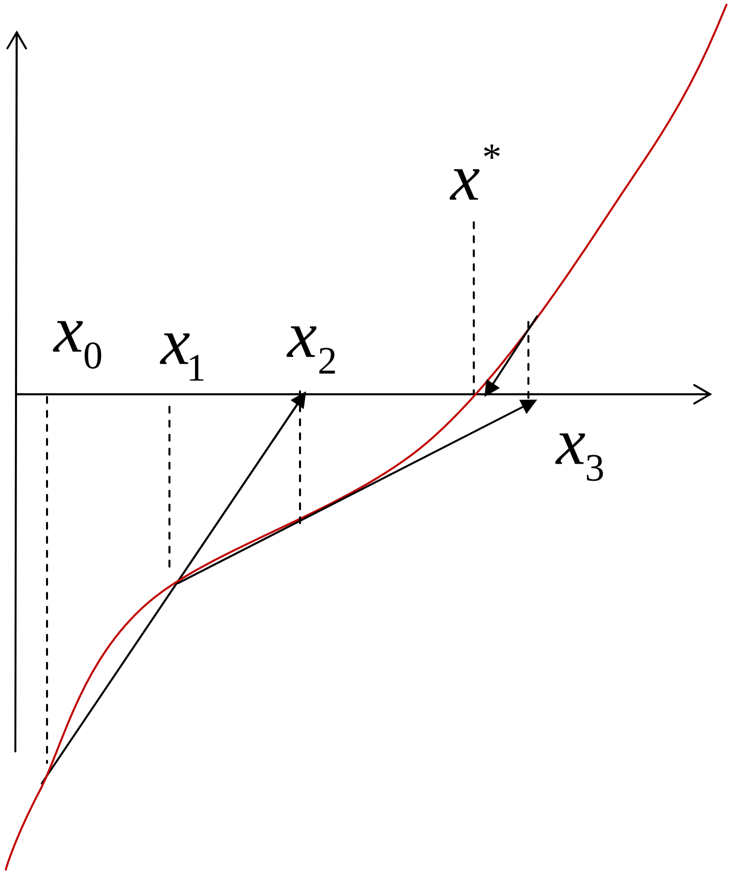
1) You find the same solution no matter what initial guess you use
2) You find many different solutions using many different initial guesses
3) You cannot find a solution because none exists
4) You cannot find a solution even though one exists, even with many, many initial guesses

Secant Method



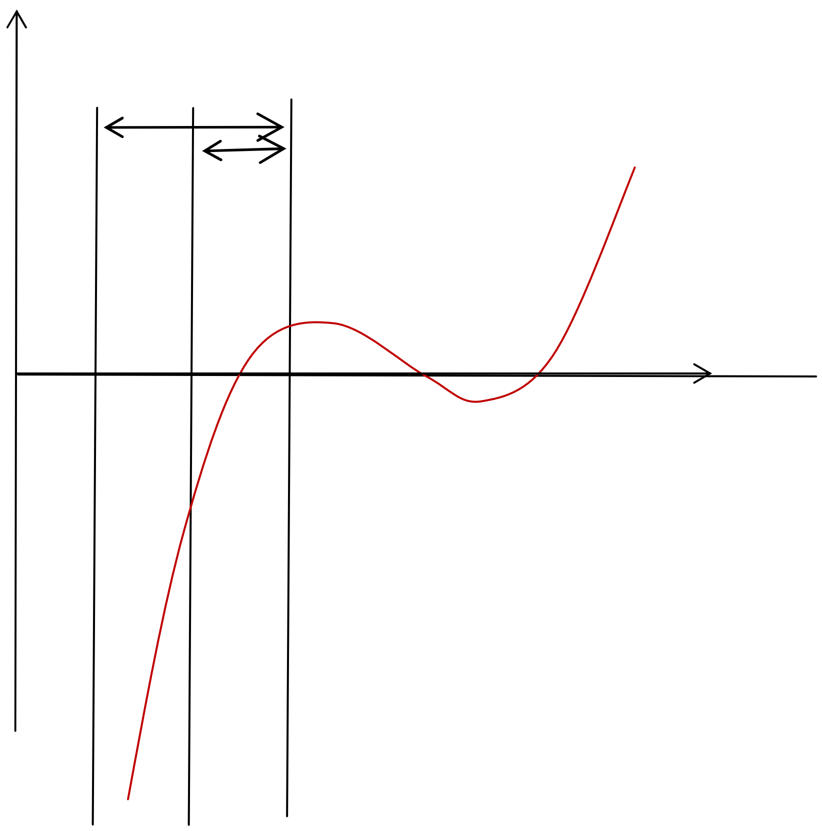
- No derivative needed!
- Uses the current and the last iterate to compute the next one
- Needs two starting values

Secant Method



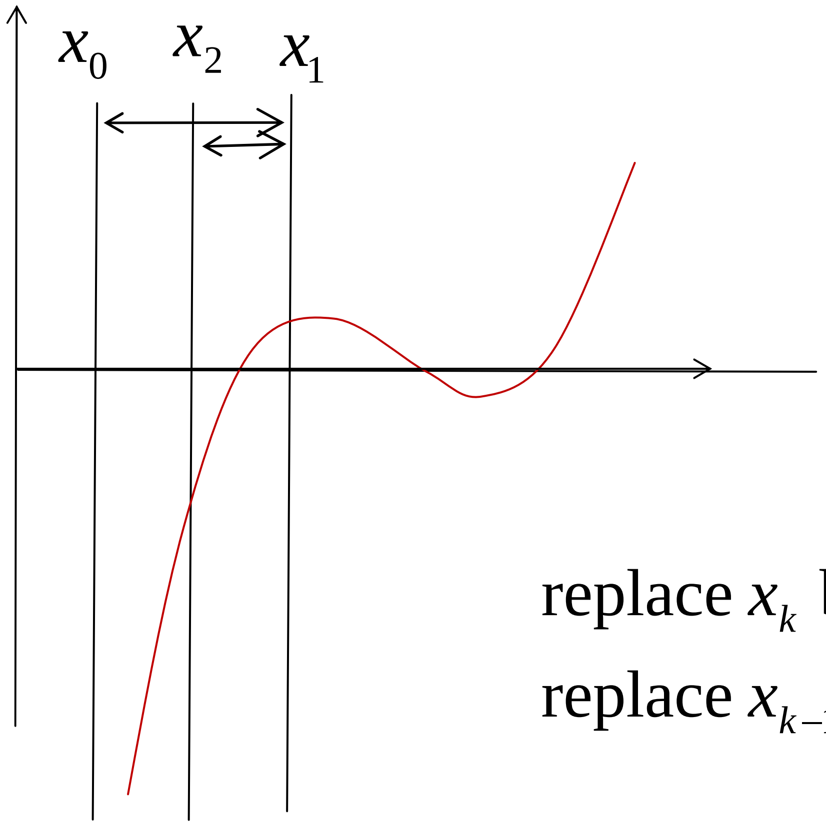
$$x_{k+1} = \frac{x_k f(x_{k-1}) - x_{k-1} f(x_k)}{f(x_{k-1}) - f(x_k)}$$

Bisection Methods



- Given an interval in which a solution is known to lie
- Look in the middle and determine which half has the root
- Iterate until the remaining interval is small enough

Bisection Methods



$$x_{k+1} = \frac{x_k + x_{k-1}}{2}$$

replace x_k by x_{k+1} if $f(x_{k+1})f(x_{k-1}) < 0$

replace x_{k-1} by x_{k+1} if $f(x_{k+1})f(x_{k-1}) > 0$

You seek to find a root of a continuous function $f(x)$, and you use the bisection method. Your initial guesses are such that

$$f(x_0)\,f(x_1) < 0$$

What are the possible outcomes?

Choose all the numbers that apply:

1) You find a solution
2) You cannot find a solution even though one exists
3) You cannot find a solution because no solution exists

Rates of Convergence

- Linear convergence

$$\left|x_k - x^*\right| \le \alpha \left|x_{k-1} - x^*\right|$$

$$\left|x_k - x^*\right| \le \alpha^k \left|x_0 - x^*\right|$$

- Super linear convergence

$$\left|x_k - x^*\right| \le \alpha_{k-1} \left|x_{k-1} - x^*\right|$$

$$\alpha_{k-1} \to 0 \quad \text{as} \quad k \to \infty$$

- Quadratic convergence

$$\left|x_k - x^*\right| \le \alpha \left|x_{k-1} - x^*\right|^2$$

Rates of Convergence

- Linear convergence $$\left|x_k - x^*\right| \le \alpha \left|x_{k-1} - x^*\right|$$
  - Bisection (with α=1/2)
- Super linear convergence $$\left|x_k - x^*\right| \le \alpha_{k-1} \left|x_{k-1} - x^*\right|$$
  - Secant method if x* is simple $$\alpha_{k-1} \to 0 \quad \text{as } k \to \infty$$
- Quadratic convergence $$\left|x_k - x^*\right| \le \alpha \left|x_{k-1} - x^*\right|^2$$
  - Newton-Raphson method if x* is simple

You seek to find a root of a continuous function $f(x)$, and you use the bisection method. Your initial guesses are such that

$$x_0 - x_1 = 10$$

You want to know that your estimated solution satisfies $\left| x_k - x^* \right| < 10^{-5}$

About how many iterations (i.e. k=?)

1) ~2
2) ~20
3) ~200
4) ~10^5

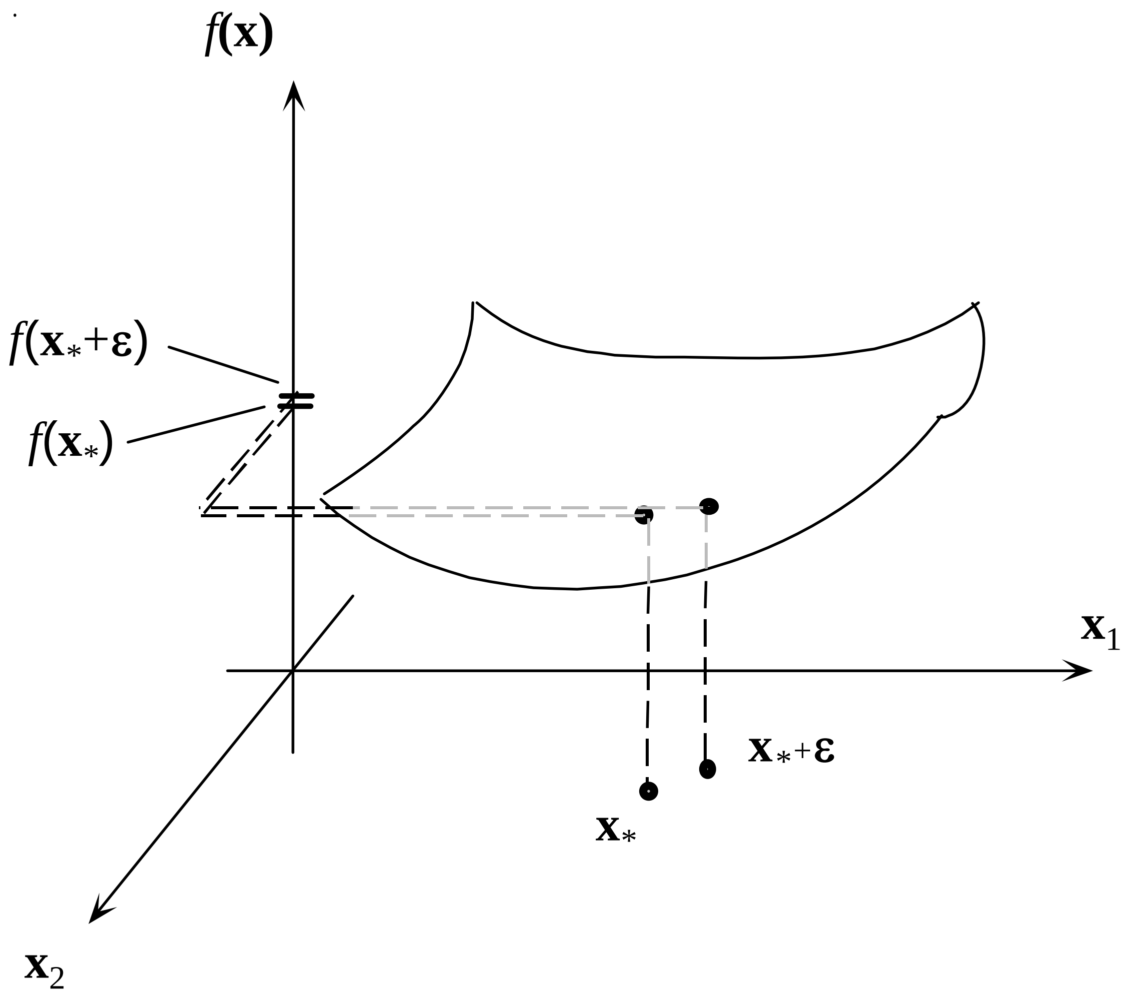
Optimization

- You seek

$$\min \ f(\mathbf{x})$$

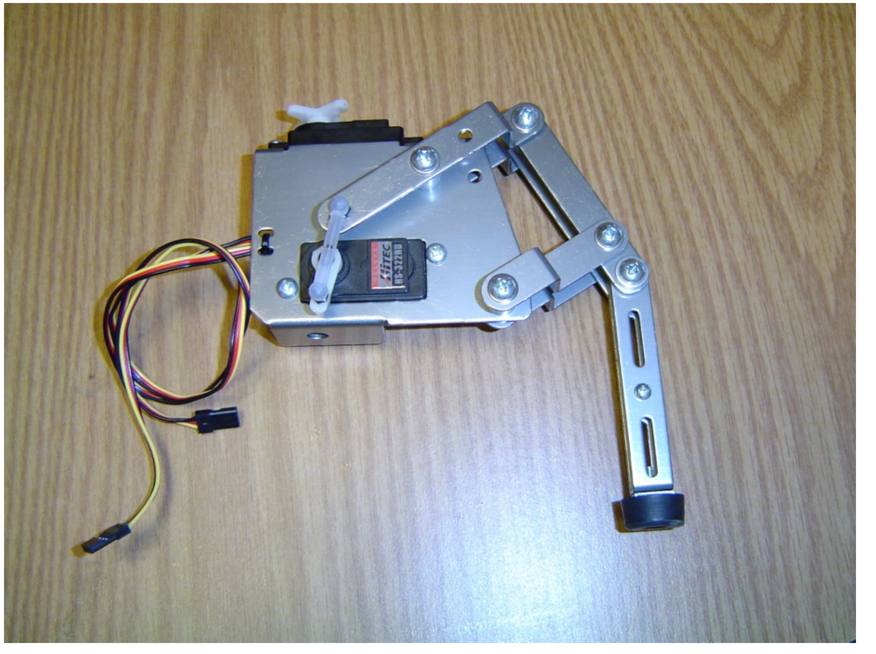
- The first order optimilality condition is

$$\nabla f(\mathbf{x}_*) = \mathbf{0}^T$$



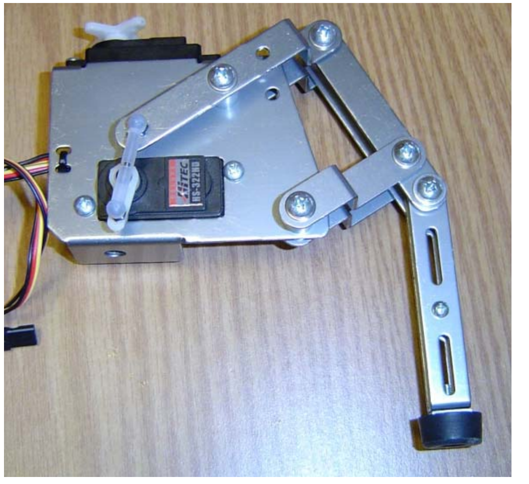
Example Problem

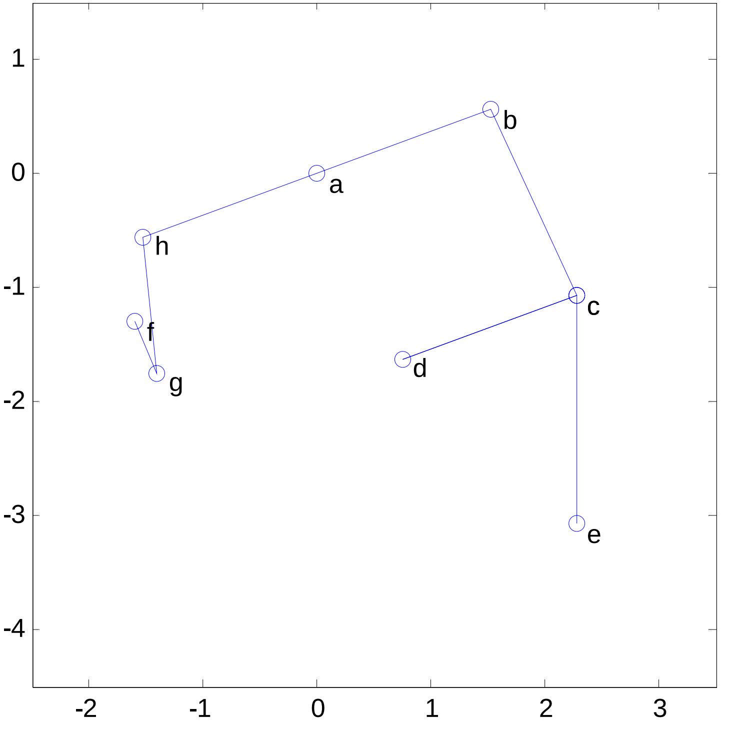
- Here is a leg from a simple robot
- If the servo motor starts from the position shown and rotates 45 deg CCW
- How far will the “foot” descend?



Representing the Geometry

```
a=[0 0 0 1]';
b=[1.527 0.556 0 1]';
c=[2.277 -1.069 0 1]';
d=[0.75 -1.625 0 1]';
e=[2.277 -3.069 0 1]';
f=[-1.6 -1.3 0 1]';
g=[-1.4 -1.75 0 1]';
h=[-1.527 -0.556 0 1]';
leg=[f g h a b c d c e];
names=char('f','g','h','a','b
','c','d','c','e');
plot(leg(1,:),leg(2,:),'o-b')
axis equal
axis([-2.5 3.5 -4.5 1.5]);
for i=1:length(leg)
    text(leg(1,i)+0.1,
leg(2,i)-0.1, names(i))
end
```

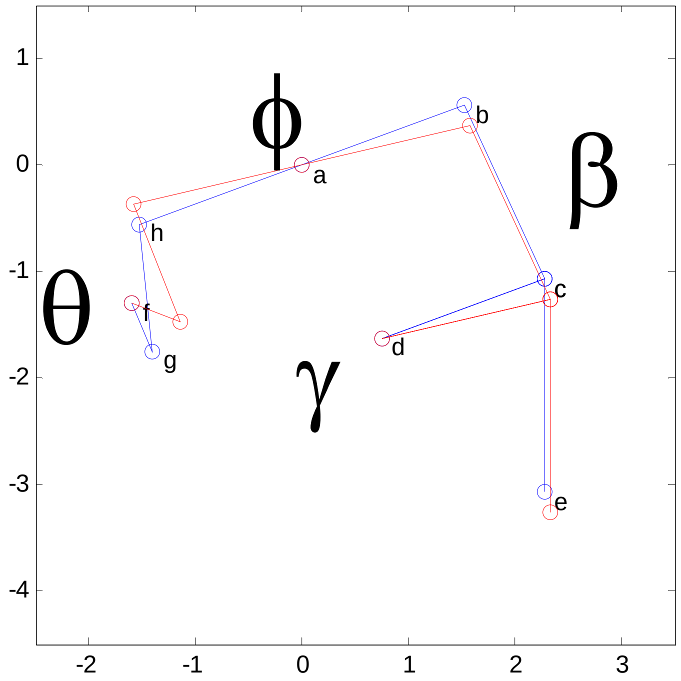




Define a Few Functions

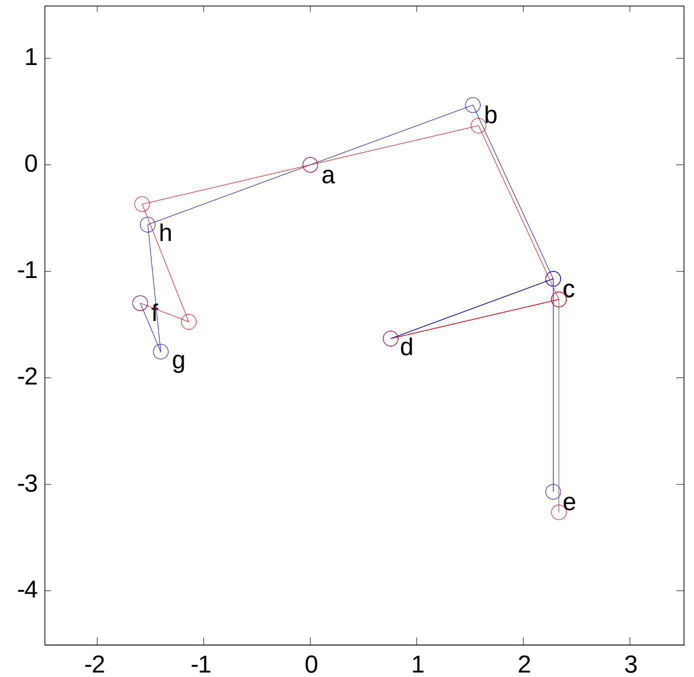
```
R=@(theta) [cos(theta) -sin(theta) 0 0;
            sin(theta) cos(theta)  0 0;
            0              0       1 0;
            0              0       0 1];

T=@(p) [1 0 0 p(1);
        0 1 0 p(2);
        0 0 1 p(3);
        0 0 0  1];

Rp=@(theta,p) T(p)*R(theta)*T(-p);
```

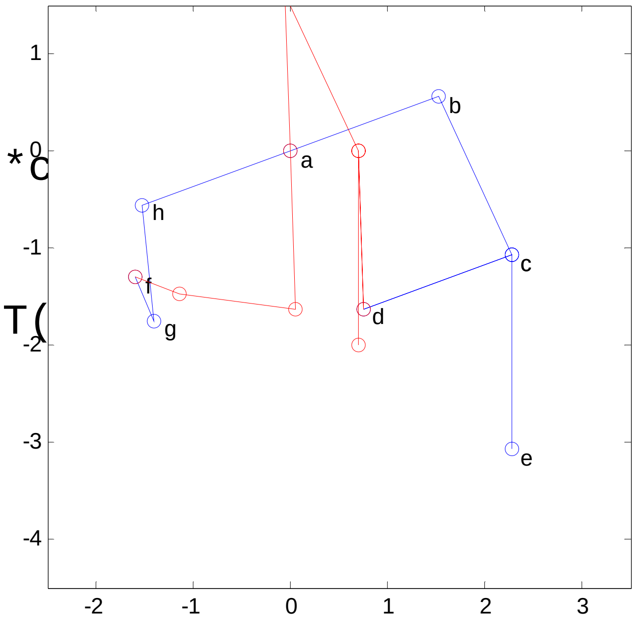
Compute a Solution




```
theta=45*pi/180;
g2=Rp(theta,f)*g;
link1=@(phi) norm(g-h)-norm(g2-Rp(phi,a)*h);
phi=fzero(link1,0);
h2=Rp(phi,a)*h;
b2=Rp(phi,a)*b;
link2=@(gamma) norm(b-c)-norm(b2-Rp(gamma,d)*c);
gamma=fzero(link2,0);
c2=Rp(gamma,d)*c;
link3=@(beta) norm(b-c)-norm(b2-Rp(beta,b2)*T(b2-b)*c);
beta=fzero(link3,0);
e2=Rp(beta,b2)*T(b2-b)*e;
leg2=[f g2 h2 a b2 c2 d c2 e2];
hold on
plot(leg2(1,:),leg2(2,:),'o-r')
```

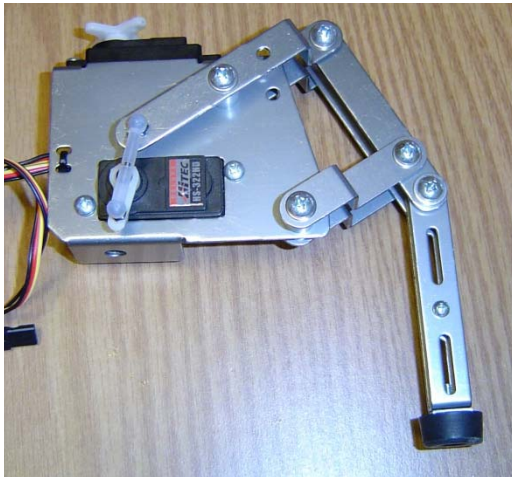
Compute Another Solution

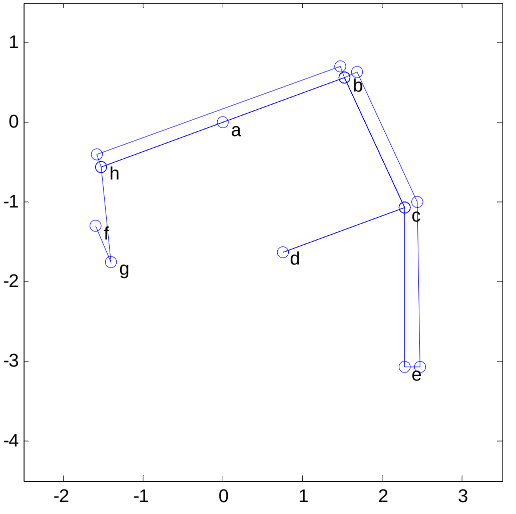





```
theta=45*pi/180;
g2=Rp(theta,f)*g;
link1=@(phi) norm(g-h)-norm(g2-Rp(phi,a)*h);
phi=fzero(link1,pi);
h2=Rp(phi,a)*h;
b2=Rp(phi,a)*b;
link2=@(gamma) norm(b-c)-norm(b2-Rp(gamma,d)*c
gamma=fzero(link2,0);
c2=Rp(gamma,d)*c;
link3=@(beta) norm(b-c)-norm(b2-Rp(beta,b2)*T(
beta=fzero(link3,0);
e2=Rp(beta,b2)*T(b2-b)*e;
leg2=[f g2 h2 a b2 c2 d c2 e2];
hold on
plot(leg2(1,:),leg2(2,:),'o-r')
```

Representing the Geometry

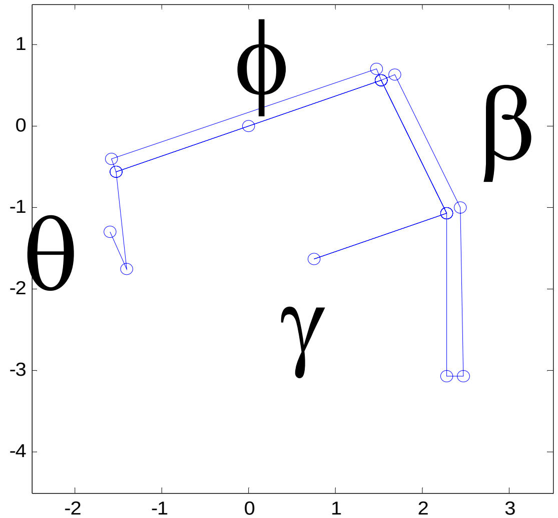





```
a=[0 0 0 1]';
b=[1.527 0.556 0 1]';
c=[2.277 -1.069 0 1]';
d=[0.75 -1.625 0 1]';
e=[2.277 -3.069 0 1]';
f=[-1.6 -1.3 0 1]';
g=[-1.4 -1.75 0 1]';
h=[-1.527 -0.556 0 1]';
leg=[f g h a b c b b+0.05*Rp(-pi/2,b)*(h-b)
h+0.05*Rp(pi/2,h)*(b-h) h b c d c e e+0.1*Rp(-pi/2,e)*(c-e)
c+0.1*Rp(-pi/2,c)*(b-c) b+0.1*Rp(pi/2,b)*(c-b) b];
names=char('f','g','h','a','b','c','d','e');
plot(leg(1,:),leg(2,:),'o-b')
axis equal
axis([-2.5 3.5 -4.5 1.5]);
loc=[1 2 3 4 5 6 13 15];
for i=1:8
    text(leg(1,loc(i))+0.1, leg(2,loc(i))-0.1, names(i))
end
```

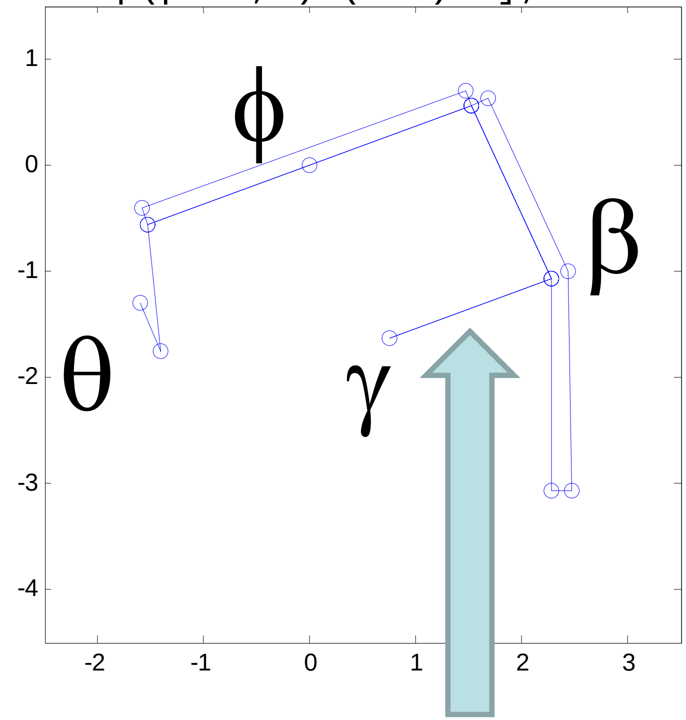
Animate the Leg Mechanism

```
instant = 0.0001;  % pause between frames
leg=[f g h a b c b b+0.05*Rp(-pi/2,b)*(h-b) h+0.05*Rp(pi/2,h)*(b-h) h b c d c
e e+0.1*Rp(-pi/2,e)*(c-e) c+0.1*Rp(-pi/2,c)*(b-c) b+0.1*Rp(pi/2,b)*(c-b) b];
p = plot(leg(1,:),leg(2,:),'o-b',...
    'EraseMode', 'normal');
axis equal
axis([-2.5 3.5 -4.5 1.5]);
options = optimset('Display','off');
for theta=0:0.5*pi/180:210*pi/180
  g2=Rp(theta,f)*g;
  link1=@(phi) norm(g-h)-norm(g2-Rp(phi,a)*h);
  phi=fzero(link1,0);
  h2=Rp(phi,a)*h;
  b2=Rp(phi,a)*b;
  link2=@(gamma) norm(b-c)-norm(b2-Rp(gamma,d)*c);
  gamma=fzero(link2,0);
  c2=Rp(gamma,d)*c;
  link3=@(beta) norm(c2-Rp(beta,b2)*T(b2-b)*c);
  beta=fsolve(link3,0,options);
  e2=Rp(beta,b2)*T(b2-b)*e;
 leg=[f g2 h2 a b2 c2 b2 b2+0.05*Rp(-pi/2,b2)*(h2-b2) h2+0.05*Rp(pi/2,h2)*(b2-
h2) h2 b2 c2 d c2 e2 e2+0.1*Rp(-pi/2,e2)*(c2-e2) c2+0.1*Rp(-pi/2,c2)*(b2-c2)
b2+0.1*Rp(pi/2,b2)*(c2-b2) b2];
  set(p,'XData',leg(1,:), 'YData',leg(2,:))
  pause(instant)
end
```



Back-Drive the Leg with Link cd

```
instant = 0.0001;  % pause between frames
leg=[f g h a b c b b+0.05*Rp(-pi/2,b)*(h-b) h+0.05*Rp(pi/2,h)*(b-h) h b c d c
e e+0.1*Rp(-pi/2,e)*(c-e) c+0.1*Rp(-pi/2,c)*(b-c) b+0.1*Rp(pi/2,b)*(c-b) b];
p = plot(leg(1,:),leg(2,:),'o-b',...
    'EraseMode', 'normal');
axis equal
axis([-2.5 3.5 -4.5 1.5]);
for gamma=0:-0.5*pi/180:-50*pi/180
  c2=Rp(gamma,d)*c;
  link1=@(phi) norm(b-c)-norm(Rp(phi,a)*b-c2);
  phi=fzero(link1,0);
  b2=Rp(phi,a)*b;
  h2=Rp(phi,a)*h;
  link2=@(theta) norm(g-h)-norm(Rp(theta,f)*g-h2);
  theta=fzero(link2,0);
  g2=Rp(theta,f)*g; leg=[f g2 h2 a b2 c2 d c2 e2];
  link3=@(beta) norm(c2-Rp(beta,b2)*T(b2-b)*c);
  beta=fsolve(link3,0,options);
  e2=Rp(beta,b2)*T(b2-b)*e;
 leg=[f g2 h2 a b2 c2 b2 b2+0.05*Rp(-pi/2,b2)*(h2-b2) h2+0.05*Rp(pi/2,h2)*(b2-
h2) h2 b2 c2 d c2 e2 e2+0.1*Rp(-pi/2,e2)*(c2-e2) c2+0.1*Rp(-pi/2,c2)*(b2-c2)
b2+0.1*Rp(pi/2,b2)*(c2-b2) b2];
  set(p,'XData',leg(1,:), 'YData',leg(2,:))
  pause(instant)
end
```



Matlab's fsolve

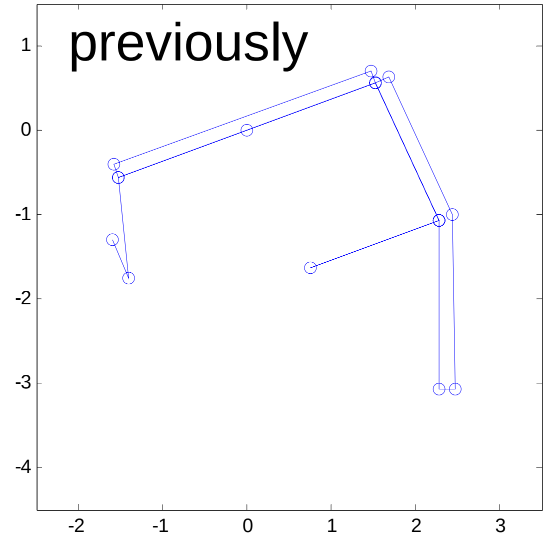
```
myfun=inline('[2*x(1) - x(2) - exp(-x(1));
-x(1) + 2*x(2) - exp(-x(2))]');

x0 = [-5; -5]; % Make a starting guess at
the solution
options=optimset('Display','iter');
[x,fval] = fsolve(myfun,x0,options)
```

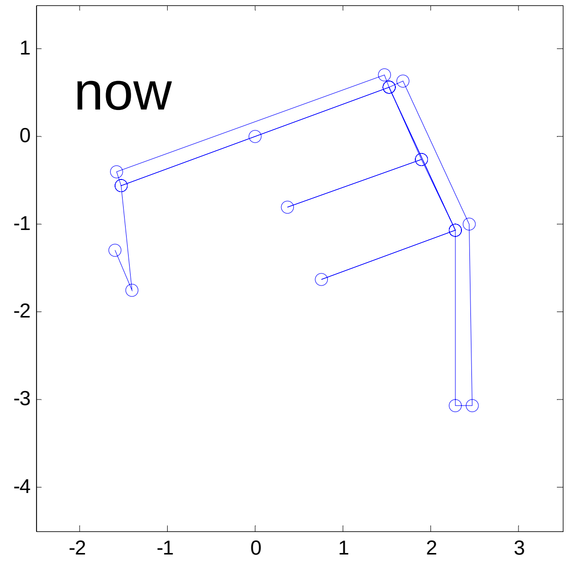
```
                                         Norm of      First-order   Trust-region
 Iteration  Func-count     f(x)          step         optimality    radius
     0          3         47071.2                     2.29e+004               1
     1          6         12003.4              1      5.75e+003               1
     2          9         3147.02              1      1.47e+003               1
     3         12         854.452              1            388               1
     4         15         239.527              1            107               1
     5         18         67.0412              1           30.8               1
     6         21         16.7042              1           9.05               1
     7         24         2.42788              1           2.26               1
     8         27        0.032658       0.759511          0.206             2.5
     9         30    7.03149e-006       0.111927        0.00294             2.5
    10         33    3.29525e-013     0.00169132      6.36e-007             2.5
Optimization terminated: first-order optimality is less than options.TolFun.
```

```
x =

    0.5671
    0.5671


fval =

  1.0e-006 *

   -0.4059
   -0.4059
```

Add a Link

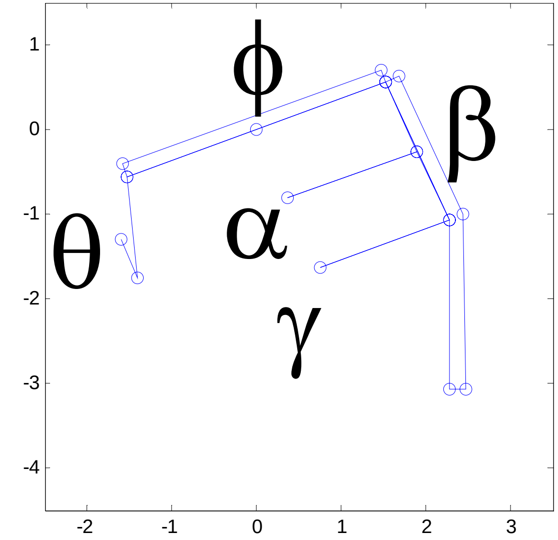
```
a=[0 0 0 1]';
b=[1.527 0.556 0 1]';
c=[2.277 -1.069 0 1]';
d=[0.75 -1.625 0 1]';
e=[2.277 -3.069 0 1]';
f=[-1.6 -1.3 0 1]';
g=[-1.4 -1.75 0 1]';
h=[-1.527 -0.556 0 1]';
i=a+(c-b)/2;
j=b+(c-b)/2;
leg=[f g h a b j i j c b b+0.05*Rp(-
pi/2,b)*(h-b) h+0.05*Rp(pi/2,h)*(b-h) h b c d
c e e+0.1*Rp(-pi/2,e)*(c-e) c+0.1*Rp(-
pi/2,c)*(b-c) b+0.1*Rp(pi/2,b)*(c-b) b];
plot(leg(1,:),leg(2,:),'o-b')
axis equal
axis([-2.5 3.5 -4.5 1.5]);
```





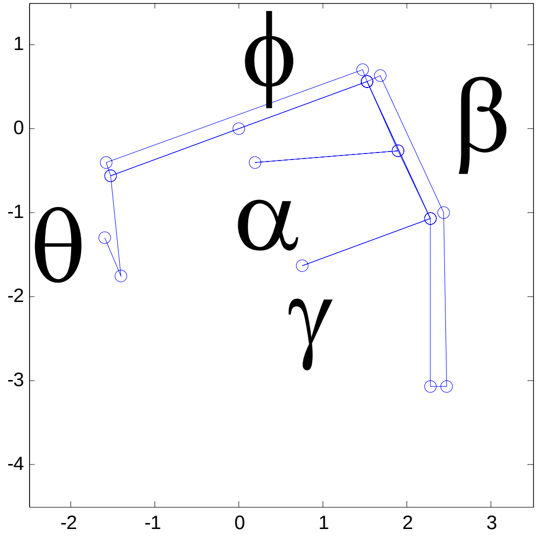
Animate the New Mechanism

```
instant = 0.0001;  % pause between frames
leg=[f g h a b j i j c b b+0.05*Rp(-pi/2,b)*(h-b) h+0.05*Rp(pi/2,h)*(b-h) h b c d c e
e+0.1*Rp(-pi/2,e)*(c-e) c+0.1*Rp(-pi/2,c)*(b-c) b+0.1*Rp(pi/2,b)*(c-b) b];
p = plot(leg(1,:),leg(2,:),'o-b',...
    'EraseMode', 'normal');
axis equal
axis([-2.5 3.5 -4.5 1.5]);
options = optimset('Display','on','TolX',10^-6, 'TolFun',10^-6);
for theta=0:0.5*pi/180:210*pi/180
  g2=Rp(theta,f)*g;
  link1=@(phi) norm(g-h)-norm(g2-Rp(phi,a)*h);
  phi=fzero(link1,0);
  h2=Rp(phi,a)*h;
  b2=Rp(phi,a)*b;
  link2=@(gamma) norm(b-c)-norm(b2-Rp(gamma,d)*c);
  gamma=fzero(link2,0);
  c2=Rp(gamma,d)*c;
  link3=@(beta) norm(c2-Rp(beta,b2)*T(b2-b)*c);
  beta= fsolve(link3,0,options);
  e2=Rp(beta,b2)*T(b2-b)*e;
  joint3=@(alpha) norm(Rp(beta,b2)*T(b2-b)*j -Rp(alpha,i)*j);
  alpha=fsolve(joint3,0, options);
  j2= Rp(alpha,i)*j;
  leg=[f g2 h2 a b2 j2 i j2 c2 b2 b2+0.05*Rp(-pi/2,b2)*(h2-b2) h2+0.05*Rp(pi/2,h2)*(b2-
h2) h2 b2 c2 d c2 e2 e2+0.1*Rp(-pi/2,e2)*(c2-e2) c2+0.1*Rp(-pi/2,c2)*(b2-c2)
b2+0.1*Rp(pi/2,b2)*(c2-b2) b2];
set(p,'XData',leg(1,:), 'YData',leg(2,:))
  pause(instant)
end
```



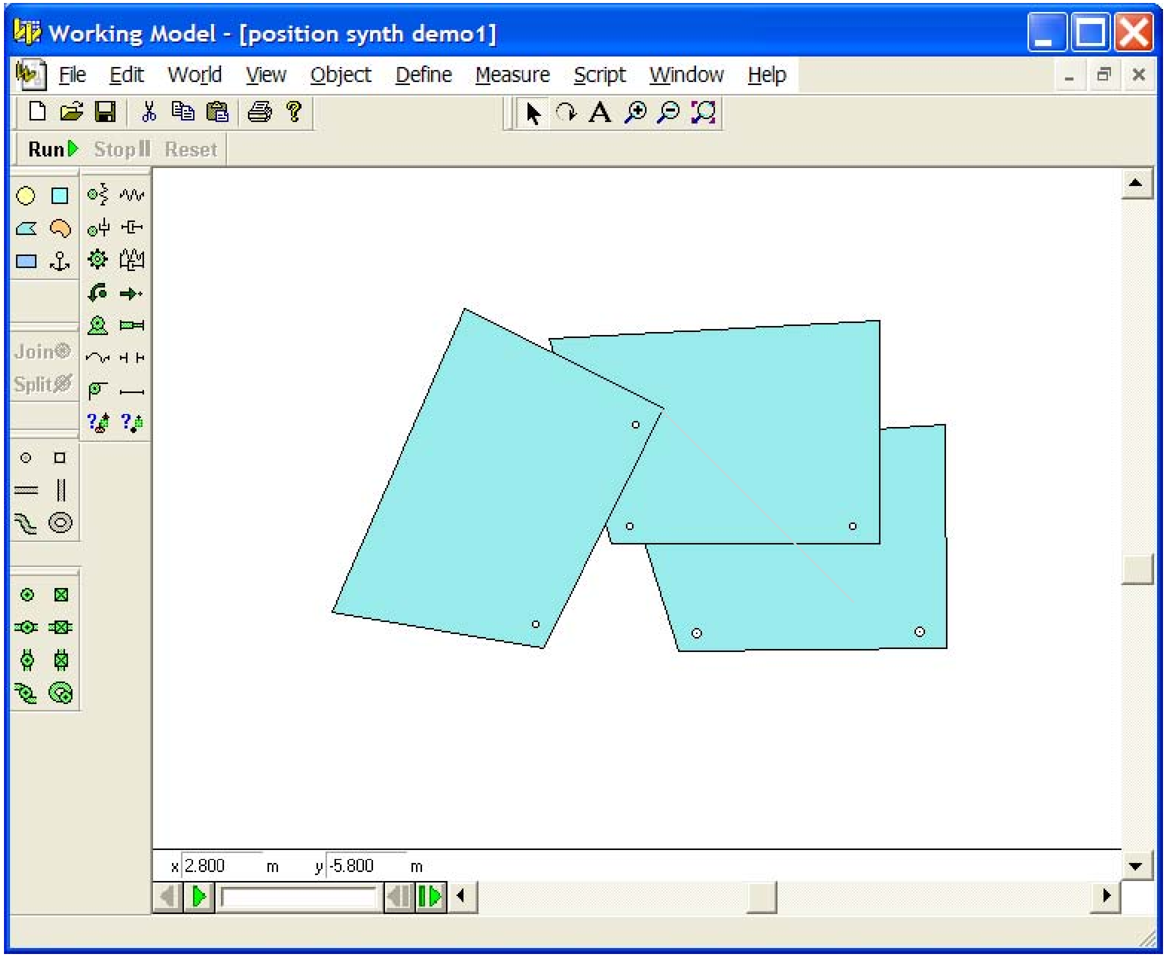
Try Another Geometry

```
i=a+(c-b)/4; j=b+(c-b)/2;
instant = 0.0001;  % pause between frames
leg=[f g h a b j i j c b b+0.05*Rp(-pi/2,b)*(h-b) h+0.05*Rp(pi/2,h)*(b-h) h b c d c e
e+0.1*Rp(-pi/2,e)*(c-e) c+0.1*Rp(-pi/2,c)*(b-c) b+0.1*Rp(pi/2,b)*(c-b) b];
p = plot(leg(1,:),leg(2,:),'o-b',...
    'EraseMode', 'normal');
axis equal; axis([-2.5 3.5 -4.5 1.5]);
for theta=0:0.5*pi/180:210*pi/180
  g2=Rp(theta,f)*g;
  link1=@(phi) norm(g-h)-norm(g2-Rp(phi,a)*h);
  phi=fzero(link1,0);
  h2=Rp(phi,a)*h;
  b2=Rp(phi,a)*b;
  link2=@(gamma) norm(b-c)-norm(b2-Rp(gamma,d)*c);
  gamma=fzero(link2,0);
  c2=Rp(gamma,d)*c;
  beta=acos((b-c)'*(b2-c2)/norm(b-c)^2);
  e2=Rp(beta,b2)*T(b2-b)*e;
  joint3=@(alpha) norm(Rp(beta,b2)*T(b2-b)*j -Rp(alpha,i)*j);
  options = optimset('Display','on','TolX',10^-6, 'TolFun',10^-6);
  alpha=fsolve(joint3,0, options);
  j2= Rp(alpha,i)*j;
 leg=[f g2 h2 a b2 j2 i j2 c2 b2 b2+0.05*Rp(-pi/2,b2)*(h2-b2) h2+0.05*Rp(pi/2,h2)*(b2-
h2) h2 b2 c2 d c2 e2 e2+0.1*Rp(-pi/2,e2)*(c2-e2) c2+0.1*Rp(-pi/2,c2)*(b2-c2)
b2+0.1*Rp(pi/2,b2)*(c2-b2) b2];
  set(p,'XData',leg(1,:), 'YData',leg(2,:))
  pause(instant)
end
```



3 Position Synthesis

- Say we want a mechanism to guide a body in a prescribed way
- Pick 3 positions
- Pick two attachment points
- The 4 bar mechanism can be constructed graphically

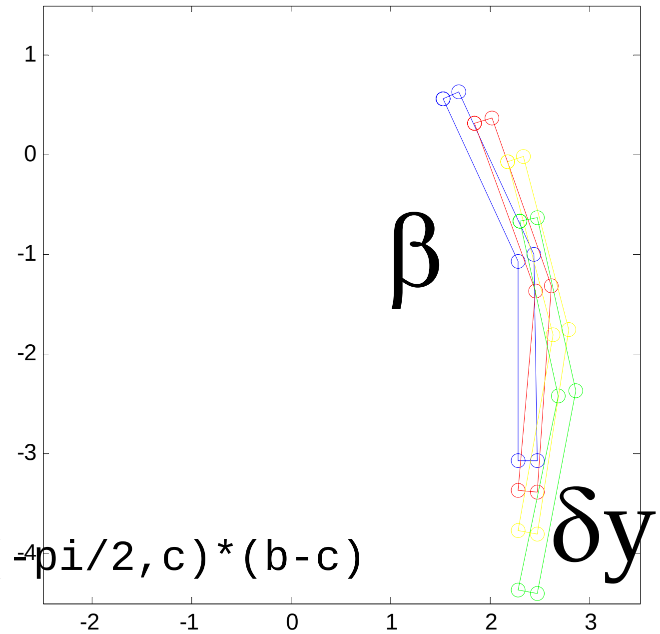


Discussion Question

- If you do not specify the attachment point, how many positions can you specify and still generally retain the capability to synthesize a mechanism?

1) 3
2) 4
3) 5
4) >5

Representing the Desired Motions



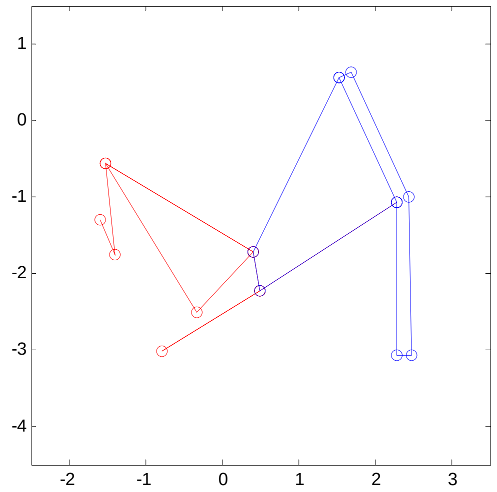

```
b=[1.527 0.556 0 1]';
c=[2.277 -1.069 0 1]';
e=[2.277 -3.069 0 1]';
leg=[b c e e+0.1*Rp(-pi/2,e)*(c-e) c+0.1*Rp(-pi/2,c)*(b-c)
b+0.1*Rp(pi/2,b)*(c-b) b];

Beta12=-5*pi/180; Beta13=-10*pi/180; Beta14=-12*pi/180;
dy12=-0.3; dy13=-0.7; dy14=-1.3;

leg2= T([0,dy12,0])*Rp(Beta12,e)*leg;
leg3= T([0,dy13,0])*Rp(Beta13,e)*leg;
leg4= T([0,dy14,0])*Rp(Beta14,e)*leg;

plot(leg(1,:),leg(2,:),'o-b'); hold on;
plot(leg2(1,:),leg2(2,:),'o-r')
plot(leg3(1,:),leg3(2,:),'o-y')
plot(leg4(1,:),leg4(2,:),'o-g')
axis equal; axis([-2.5 3.5 -4.5 1.5]);
```

Synthesize the Leg Mechanism

```
ax=0; ay=0;
bx=1.527; by=0.556;
cx=2.277; cy=-1.069;
dx=0.75; dy=-1.625;
links=@(x)...
    [norm([x(1); x(2); 0; 1]-[x(3); x(4); 0; 1])-norm([x(1); x(2);
0; 1]-T([0,dy12,0])*Rp(Beta12,e)*[x(3); x(4); 0; 1]);...
    norm([x(1); x(2); 0; 1]-[x(3); x(4); 0; 1])-norm([x(1); x(2);
0; 1]-T([0,dy13,0])*Rp(Beta13,e)*[x(3); x(4); 0; 1]);...
    norm([x(1); x(2); 0; 1]-[x(3); x(4); 0; 1])-norm([x(1); x(2);
0; 1]-T([0,dy14,0])*Rp(Beta14,e)*[x(3); x(4); 0; 1]);...
    norm([x(7); x(8); 0; 1]-[x(5); x(6); 0; 1])-norm([x(7); x(8);
0; 1]-T([0,dy12,0])*Rp(Beta12,e)*[x(5); x(6); 0; 1]);...
    norm([x(7); x(8); 0; 1]-[x(5); x(6); 0; 1])-norm([x(7); x(8);
0; 1]-T([0,dy13,0])*Rp(Beta13,e)*[x(5); x(6); 0; 1]);...
    norm([x(7); x(8); 0; 1]-[x(5); x(6); 0; 1])-norm([x(7); x(8);
0; 1]-T([0,dy14,0])*Rp(Beta14,e)*[x(5); x(6); 0; 1])];

xg=[ax;ay;bx;by;cx;cy;dx;dy];
x=fsolve(links,xg);
a=[x(1); x(2); 0; 1]; bs=[x(3); x(4); 0; 1];
cs=[x(5); x(6); 0; 1]; d=[x(7); x(8); 0; 1];
```

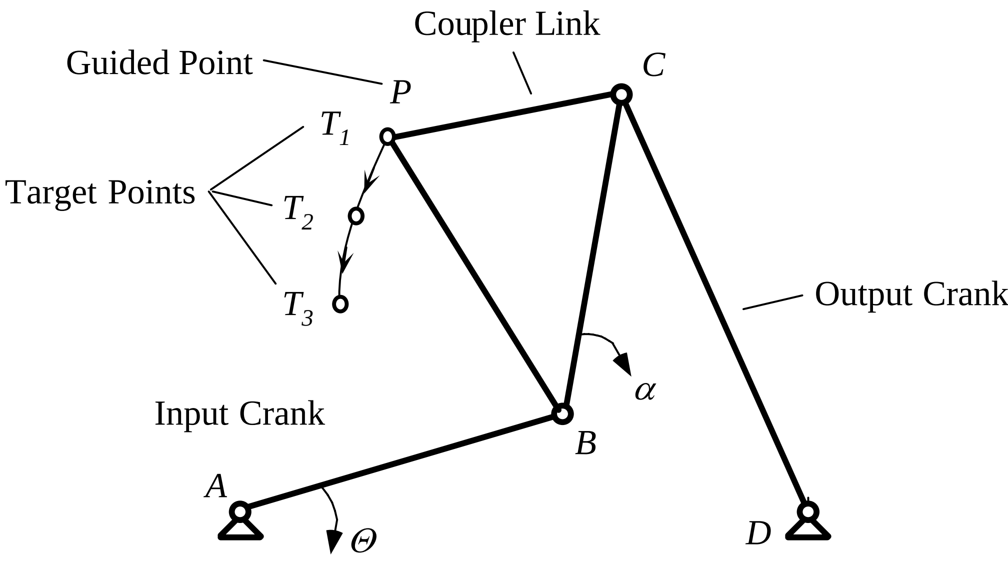
Animate the Synthesized Mechanism

```
instant = 0.0001;  % pause between frames
leg=[b c e e+0.1*Rp(-pi/2,e)*(c-e) c+0.1*Rp(-pi/2,c)*(b-c) b+0.1*Rp(pi/2,b)*(c-b) b bs cs
c];
mech=[f g h a bs h bs cs d cs c];
p2 = plot(mech(1,:),mech(2,:),'o-r','EraseMode', 'normal'); hold on;
p1 = plot(leg(1,:),leg(2,:),'o-b','EraseMode', 'normal');
axis equal
axis([-2.5 3.5 -4.5 1.5]);
for theta=0:0.5*pi/180:70*pi/180
  g2=Rp(theta,f)*g;
  link1=@(phi) norm(g-h)-norm(g2-Rp(phi,a)*h);
  phi=fzero(link1,0);
  h2=Rp(phi,a)*h;
  bs2=Rp(phi,a)*bs;
  link2=@(gamma) norm(bs-cs)-norm(bs2-Rp(gamma,d)*cs);
  gamma=fzero(link2,0);
  cs2=Rp(gamma,d)*cs;
  link3=@(beta) norm(cs2-Rp(beta,bs2)*T(bs2-bs)*cs);
  beta=fsolve(link3,0,options);
  b2=Rp(beta,bs2)*T(bs2-bs)*b;
  c2=Rp(beta,bs2)*T(bs2-bs)*c;
  e2=Rp(beta,bs2)*T(bs2-bs)*e;
  leg=[b2 c2 e2 e2+0.1*Rp(-pi/2,e2)*(c2-e2) c2+0.1*Rp(-pi/2,c2)*(b2-c2)
b2+0.1*Rp(pi/2,b2)*(c2-b2) b2 bs2 cs2 c2];
  set(p1,'XData',leg(1,:), 'YData',leg(2,:))
  mech=[f g2 h2 a bs2 h2 bs2 cs2 d cs2 c2];
  set(p2,'XData',mech(1,:), 'YData',mech(2,:))
  set(p1,'XData', leg(1,:), 'YData',leg(2,:))
  pause(instant)
end
```



Path Generation

- Define a set of points through which a location on a moving body should travel
- Allow this point to be freely selected on the moving body
- Allow the body to rotate as needed
- Solve the system of equations



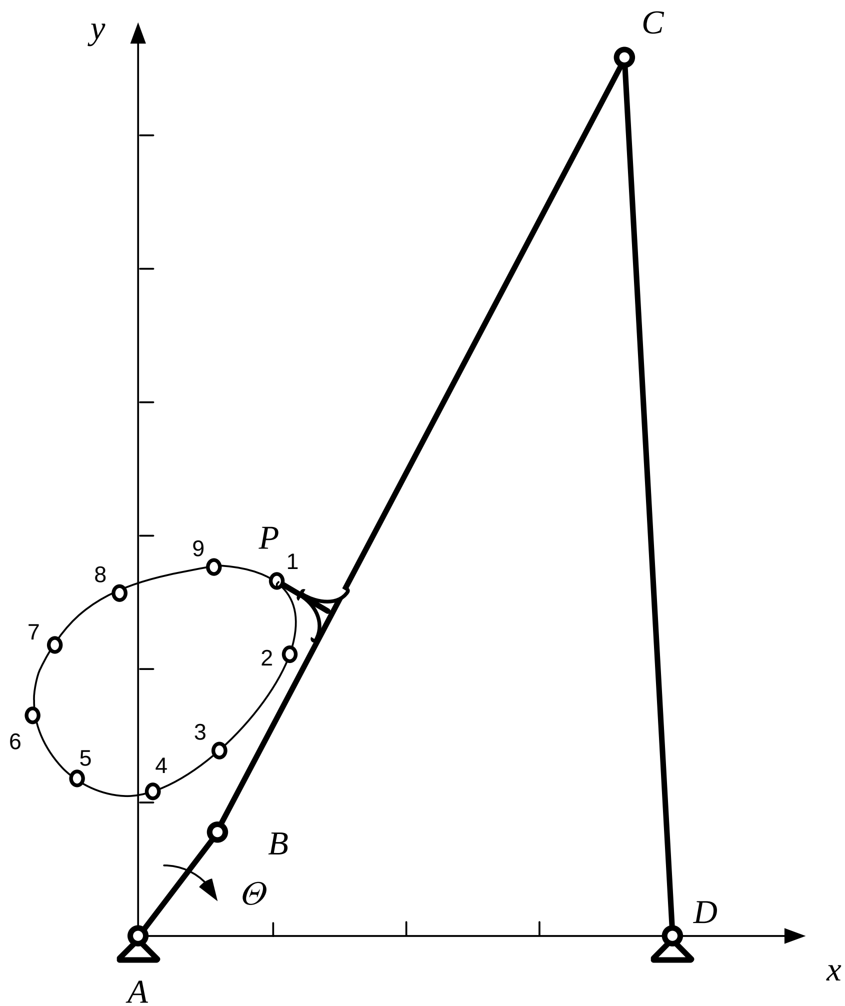
Discussion Question

- How many points can you specify and still generally retain the capability to synthesize a mechanism?

1)4
2)5-7
3)7-9
4)>9

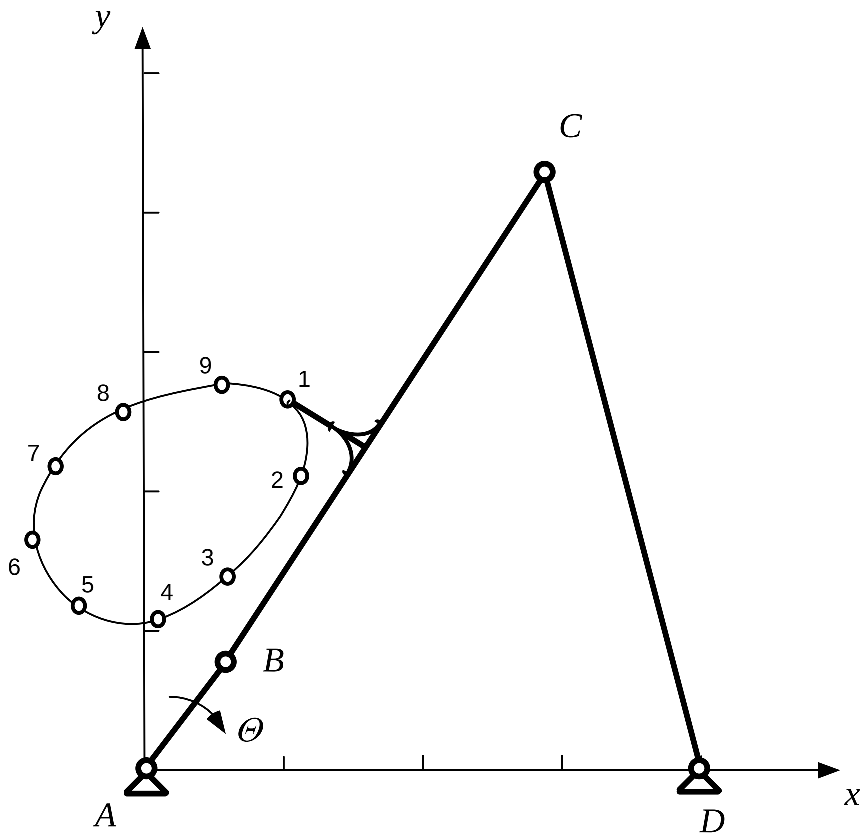
Optimization

- An “optimal” mechanism if the goal is to minimize the sum squared deviations



Optimization Under Constraints

- An “optimal” mechanism if the goal is to minimize the sum squared deviations
- AND limit the link lengths to less than a specified amount



Next Steps

- Thursday 30 April
  - Exam discussion
  - Professional ethics
- Tuesday 5 May
  - Contest procedures
- Weds 6 May (First night)
- Thursday 7 May
  - No lecture
  - Second night of contest