ME352A

A handout on simple movies in Matlab, and a first introduction to four bar linkages

Anindya Chatterjee
August 2, 2017

In what follows, Matlab commands will be in red. Matlab m-files will be in blue. My comments will be in black, like this line.

The basic movie command

Try the following commands in Matlab:

```
>> x=linspace(0,1,300); for k=1:20; plot(x,sin(6*x.^(k/5))); M(k)=getframe; end
>> movie(M,6,20)
>> movie(M,6,200)
>> movie(M,6,5)
>> movie(M,-6,50)
```

The function sin(6*x.^(k/5)) above is arbitrary.

In the "movie" command, "M" contains the twenty-frame movie, the first number is the number of times to play it (with a negative number, Matlab plays it both backwards and forwards), and the second number is the frame rate. The numbers are optional.

The Newton-Raphson method

Save the following in your directory as "newton.m" but also learn how to write this out on your own.

```
function x=newton(fun,x)
n=length(x);
epsil=(1e-6*max(1,norm(x)));
pert=eye(n)*epsil;
iter=0;
nmax=60;

ee=feval(fun,x);
while (norm(ee)*max(1,norm(x))>1e-10)*(iter<nmax)
iter=iter+1;
for k=1:n
D(:,k)=(feval(fun,x+pert(:,k))-ee)/epsil;
end
x=x-(D\ee);
if nargin == 3
disp(x)
end
ee=feval(fun,x);
end

if (iter == nmax)+(abs(x)==inf)
x=inf;
disp('did not converge')
end
```

The above program is a general purpose numerical solver for small systems of simultaneous equations of the form "fun(x) = 0" where "fun" represents some m-file you have created, or some existing function. The first input argument is the name of the function, and the second one is an initial guess. For example, "sin" is a built-in function and sin(pi)=0, so:

```
>> newton('sin',3)
ans =
   3.141592653589793
```

Now suppose we want to solve the following two equations:

$$x^2 + y^2 = 1, \qquad x + y = 0.7$$

Then we could create the following "junk" m-file:

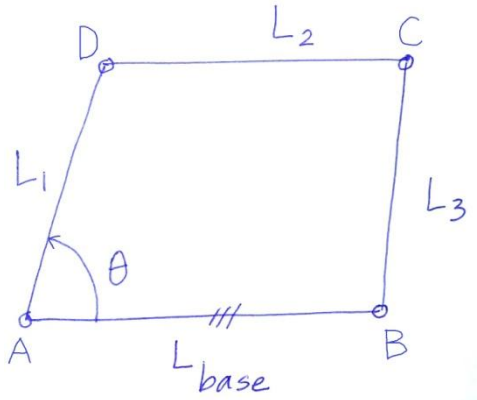
```
function z=junk(x)

y=x(2); x=x(1); z=[x^2+y^2-1; x+y-0.7];
```

Starting off the Newton-Raphson method with initial conditions of (1,-0.3), we find

```
>> newton('junk',[1;-0.3])
ans =
   0.964410286372709
  -0.264410286372710
>> junk(ans)
ans =
  1.0e-011 *
   0.118971499318832
  -0.000011102230246
```

Note that the input is a column matrix, [1;-0.3]. The answer satisfies the equations.

Now consider a four bar linkage as shown below.



The base link is fixed. Given the link lengths and the crank angle Θ, can we draw the linkage? The base link and link L_1 are now located. Points A, B and D are known. The far point C must be located, that is all.

A new file, FBpoint, is created. The coordinates of the sought point are called X (containing both x and y), and we use two equations. The crank angle is available as a global variable.

```
function z=FBpoint(X,p)

global theta

% parameters
Lbase=0.4; L1=0.8; L2=0.75; L3=0.65;

A=[0;0];
B=[Lbase;0];
D=L1*[cos(theta);sin(theta)];

z=[norm(X-D)-L2; norm(X-B)-L3];
```

Using Newton-Raphson, the point C can be found easily for given Θ.

```
>> global theta
>> theta=1.4
theta =
   1.400000000000000
>> a=newton('FBpoint',[1;1])
a =
   0.824894064717124
   0.491899414279410
>> FBpoint(a)
ans =
  1.0e-012 *
   0.283550960489265
   0.204503081135954
```

We have a starting point. Now how about a movie of the four bar linkage? We can keep incrementing the crank angle by small amounts, and keep using the last location of the point C as an initial guess for the new location.

The first thing to do it to add plotting to the same file FBpoint using an optional argument. Like this:

```
function z=FBpoint(X,p)

global theta

% parameters
Lbase=0.4; L1=0.8; L2=0.75; L3=0.65;

A=[0;0];
B=[Lbase;0];
D=L1*[cos(theta);sin(theta)];

z=[norm(X-D)-L2; norm(X-B)-L3];

% so far, the same as before; new stuff follows
if nargin==2
   C=X;
   figure(1)
   clf
   y=[A,B]; plot(y(1,:),y(2,:),'m','linewidth',3)
   axis('image')  % important
   axis([-1,1.3,-1,1.4])  % can be adjusted
   hold on
   y=[B,C]; plot(y(1,:),y(2,:),'k','linewidth',3)
   y=[C,D]; plot(y(1,:),y(2,:),'r','linewidth',3)
   y=[D,A]; plot(y(1,:),y(2,:),'b','linewidth',3)
   hold off
end
```

Note that the second argument is optional. If no second argument is included in the function call, then the last if-end part is not executed. The old command for finding the position still works (with Newton-Raphson).

FBpoint(a) will return a tiny number (zero). FBpoint(a,1) will return a tiny number *and* generate a plot.

In the plotting commands, try out other combinations of “axes” and “hold on” and “hold off” to see what happens.

Now write another m-file that will prepare the movie.

```
function M = FBmovie
global theta
```

```
% use initial points already found
Xguess=[0.824894064717124; 0.491899414279410];

t=linspace(1.4,1.4+2*pi,100); t=t(1:99);
for k=1:99
   theta=t(k);
   Xguess=newton('FBpoint',Xguess);
   FBpoint(Xguess,1);
   figure(1)
   M(k)=getframe;
end
```

Try this:

```
>> M=FBmovie;
>> movie(M,10,200)
```

Note in the movie that all three moving links make a complete revolution. You will find that for many other combinations of link lengths, the crank is unable to complete a full revolution. Sometimes the crank will complete a full revolution but the other links will not. And so on.

You can extend one or more of the links to get an idea of how this mechanism might be used to perform various tasks.