ECE 111 - Homework #3

Math 105: Trigonometry.
Due Monday, February 3rd. Please submit via email or on BlackBoard

Polar to Rectangular Conversions

1) Determine the final position of A: (x,y)

$$A = (6\angle -93^0) + (11\angle 70^0) + (8\angle 87^0)$$

In Matlab

```
>> x1 = 6*cos(-93*pi/180)
x1 =   -0.3140

>> y1 = 6*sin(-93*pi/180)
y1 =   -5.9918

>> x2 = 11*cos(70*pi/180)
x2 =    3.7622

>> y2 = 11*sin(70*pi/180)
y2 =   10.3366

>> x3 = 8*cos(87*pi/180)
x3 =    0.4187

>> y3 = 8*sin(87*pi/180)
y3 =    7.9890

>> Ax = x1 + x2 + x3
Ax =    3.8669

>> Ay = y1 + y2 + y3
Ay =   12.3339
```

On an HP Prime

- Setting - Entry - RPN

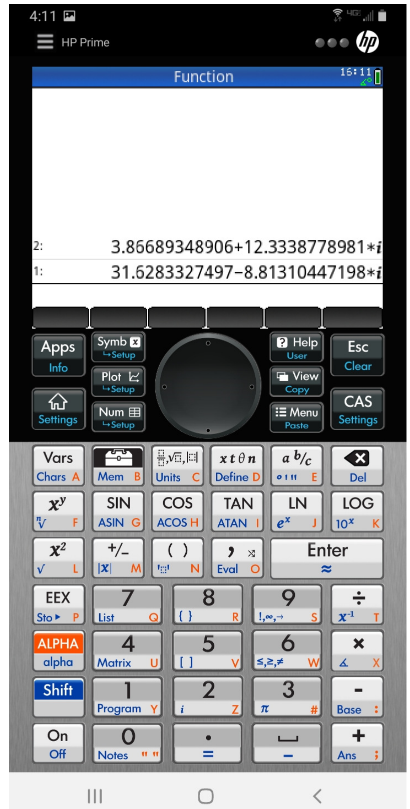
```
6 angle -93
enter
11 angle 70
+
8 angle 87
+
angle
```

2) Determine final position of B: (x,y)

$$B = (5\angle -22^0) + (22\angle 31^0) + (20\angle -66^0)$$

In Matlab

```
>> x1 = 5 * cos(-22*pi/180)
x1 =     4.6359

>> y1 = 5*sin(-22*pi/180)
y1 =   -1.8730

>> x = 22*cos(31*pi/180)
x =   18.8577

>> x2 = 22*cos(31*pi/180)
x2 =   18.8577

>> y2 = 22*sin(31*pi/180)
y2 =   11.3308

>> x3 = 20*cos(-66*pi/180)
x3 =     8.1347

>> y3 = 20*sin(-66*pi/180)
y3 =  -18.2709

>> Bx = x1 + x2 + x3
Bx =   31.6283

>> By = y1 + y2 + y3
By =   -8.8131

>>
```



On an HP-Prime calculator

```
5 angle -22
enter
22 angle 31
+
20 angle -66
+
angle
```

3) Where is B relative to A (i.e. what is C = B - A?)

- In (x,y) coordinates
- In polar coordinates

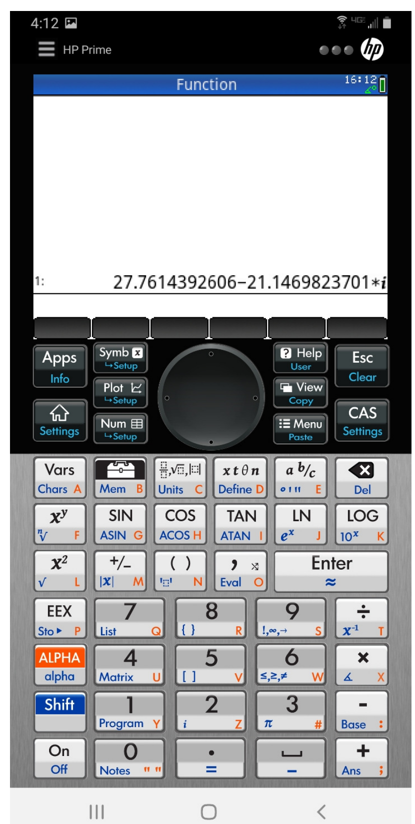
In Matlab

```
>> Cx = Bx - Ax

Cx =    27.7614

>> Cy = By - Ay

Cy =   -21.1468

>> Cr = sqrt(Cx^2 + Cy^2)

Cr =    34.8982

>> Cq = atan2(Cx, Cy) * 180/pi

Cq =  127.2978
```

On an HP-Prime

- A and B are already on the stack. Just subtract

```
-
+/-
```



Plotting Polar Functions

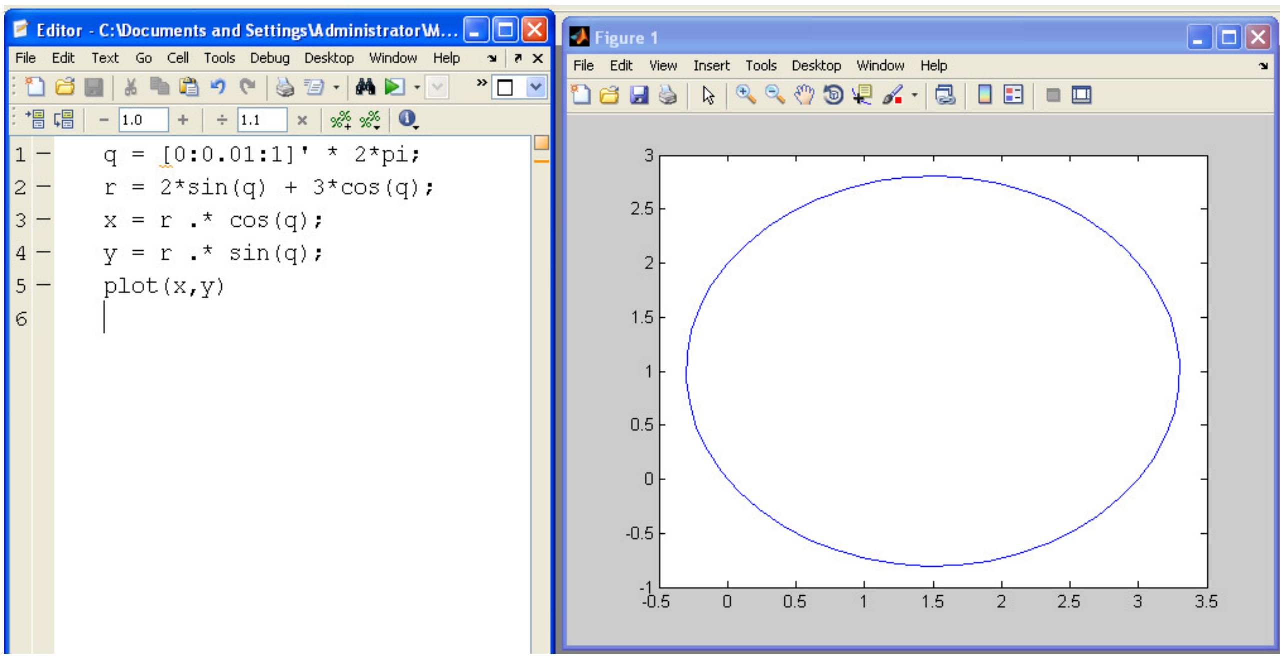
4) Plot the following functions in Matlab for $-2\pi < \theta < 2\pi$

- Note: plot() plots in cartesian coordinates. Each function needs to be converted from polar to rectangular.

a) $r = 2\sin(\theta) + 3\cos(\theta)$

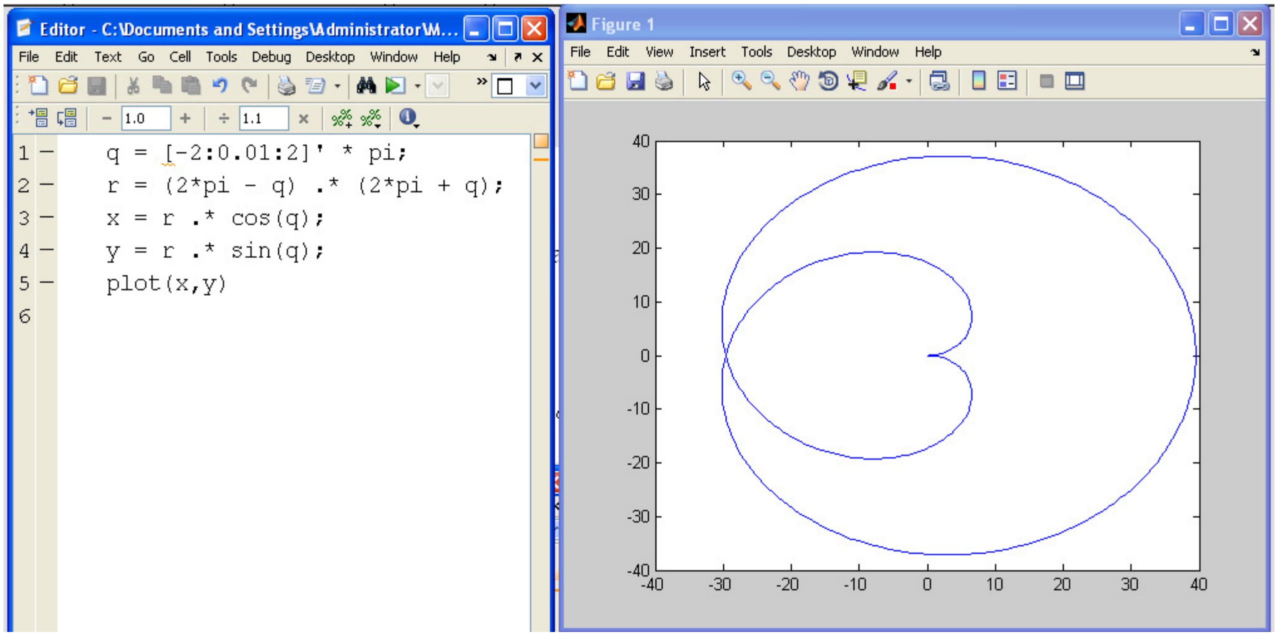
Matlab Script & Figure:

not surprisingly, this plots as a circle. Trig functions are all about circles.



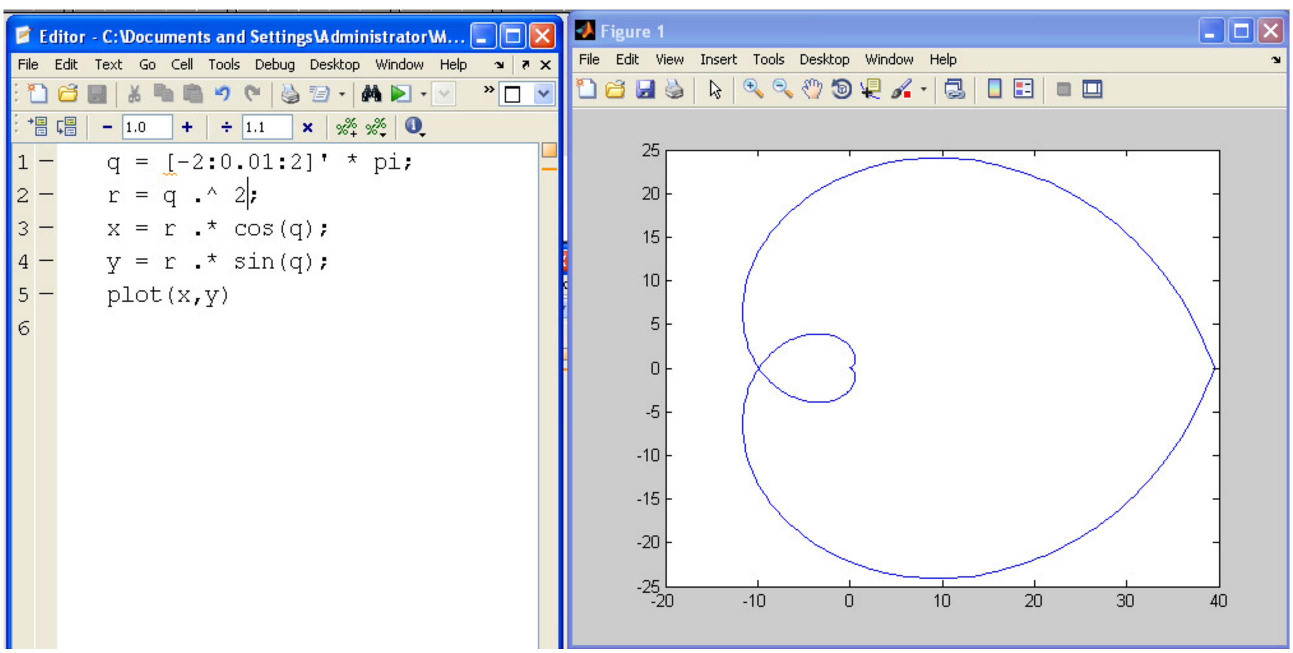
b) $r = (2\pi - \theta)(2\pi + \theta)$

sort of a heart - happy Valentine's day!



c) $r = \theta^2$

another variation of a heart



Robot Tip Position (Forward Kinematics)

A 2D robot has three arms with lengths of {3.0, 2.0, 1.0} meters. The final tip positionis

$$x_1 = 3\cos(\theta_1) \qquad y_1 = 3\sin(\theta_1)$$

$$x_2 = x_1 + 2\cos(\theta_1 + \theta_2) \qquad y_2 = y_1 + 2\sin(\theta_1 + \theta_2)$$

$$x_3 = x_2 + \cos(\theta_1 + \theta_2 + \theta_3) \qquad y_3 = y_2 + \sin(\theta_1 + \theta_2 + \theta_3)$$

5) Plot the tip position (x3, y3) for

$$\theta_1 = 41^0 \quad \theta_2 = -94^0 \quad \theta_3 = -45^0$$

Matlab program:

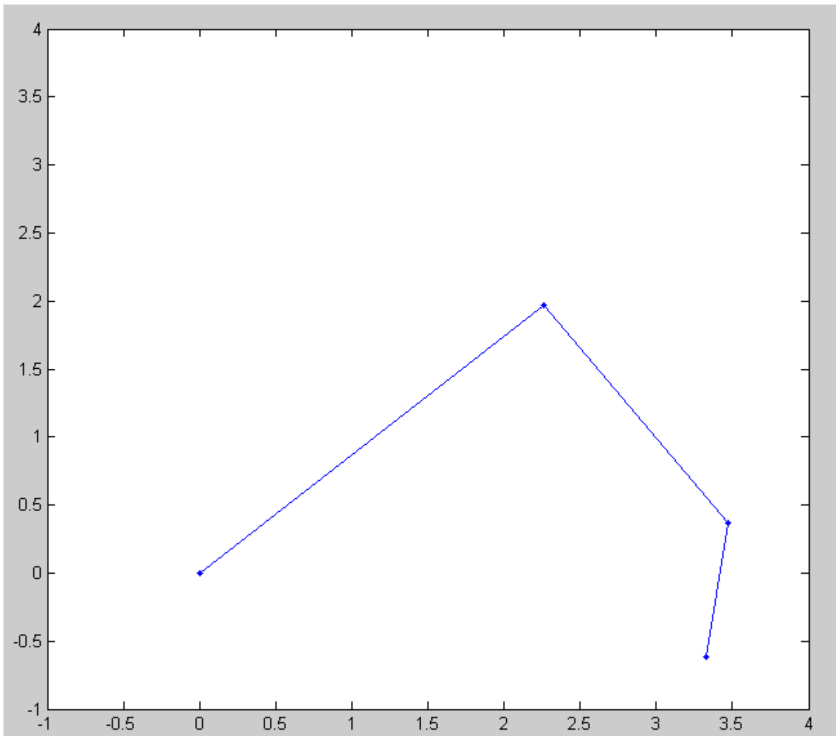
- Pass the angles in degrees
- Return the tip position (x3, y3)
- Just for fun, also plot the position of the robot (not required but fun to see)

File RRR.m

```
function [x3, y3] = RRR(q1, q2, q3)

   % convert to radians
   q1 = q1 * pi / 180;
   q2 = q2 * pi / 180;
   q3 = q3 * pi / 180;

   % compute the joint positions
   x0 = 0;
   y0 = 0;

   x1 = x0 + 3*cos(q1);
   y1 = y0 + 3*sin(q1);

   x2 = x1 + 2*cos(q1+q2);
   y2 = y1 + 2*sin(q1+q2);

   x3 = x2 + 1*cos(q1+q2+q3);
   y3 = y2 + 1*sin(q1+q2+q3);

   % just for fun, plot the resulting robot position
   plot([x0,x1,x2,x3],[y0,y1,y2,y3],'b.-');
   ylim([-1,4]);
   xlim([-1,4]);
   pause(0.01);

   end
```

From the command window, call the subroutine

```
>>> [x3, y3] = RRR(41, -94, -45)

x3 =     3.3286

y3 =    -0.6194
```




```
Tip Position
    3.3286   -0.6194
```

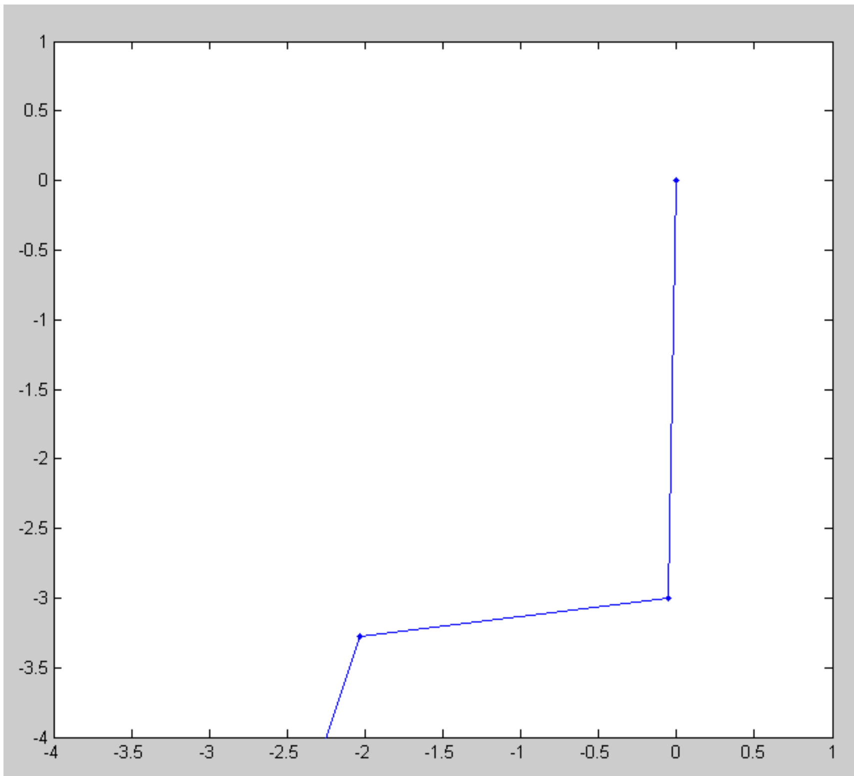
6) Plot the tip position (x3, y3) for

$$\theta_1 = -91^0 \qquad \theta_2 = -81^0 \qquad \theta_3 = 65^0$$

```
>> [x3, y3] = RRR(-91, -81, 65)
```

x3 =   -2.3253

y3 =   -4.2342



Robot Tip Position (Inverse Kinematics & fminsearch())

7) Write a Matlab function which

- Is passed the angles $(\theta_1, \theta_2, \theta_3)$,
- Computes the tip position, and
- Returns the distance from the tip position and point (x = 2.0, y = 0.0)

Comment: Part of the power of Matlab is once you write a function, that function becomes part of the Matlab library of functions you can use. That lets you build up a rather powerful set of instructions.

For example, use the previous function from problem #5 (RRR.m) to create a new function; RRR_Cost

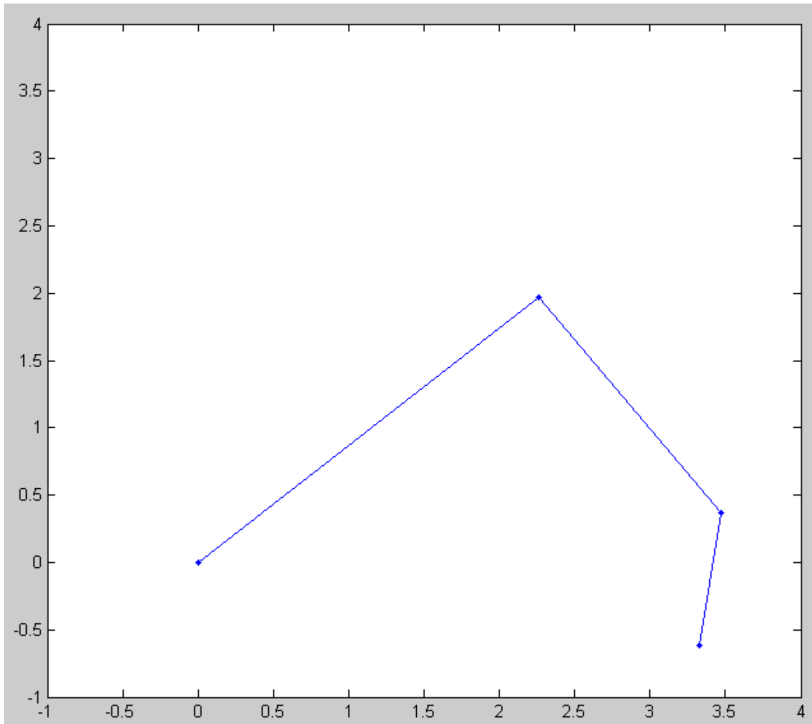
Matlab Function: RRR_Cost.m

- uses RRR.m (problem #5)

```
function [J] = RRR_Cost(z)

   q1 = z(1);
   q2 = z(2);
   q3 = z(3);

   [x3, y3] = RRR(q1, q2, q3);

   J = sqrt( (x3-2)^2 + (y3-0)^2 );
   end
```

Checking: The tip is 1.4659 meters from the point (2,0)

```
>> J = RRR_Cost([41, -94, -45])

J =    1.4659
```


4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4

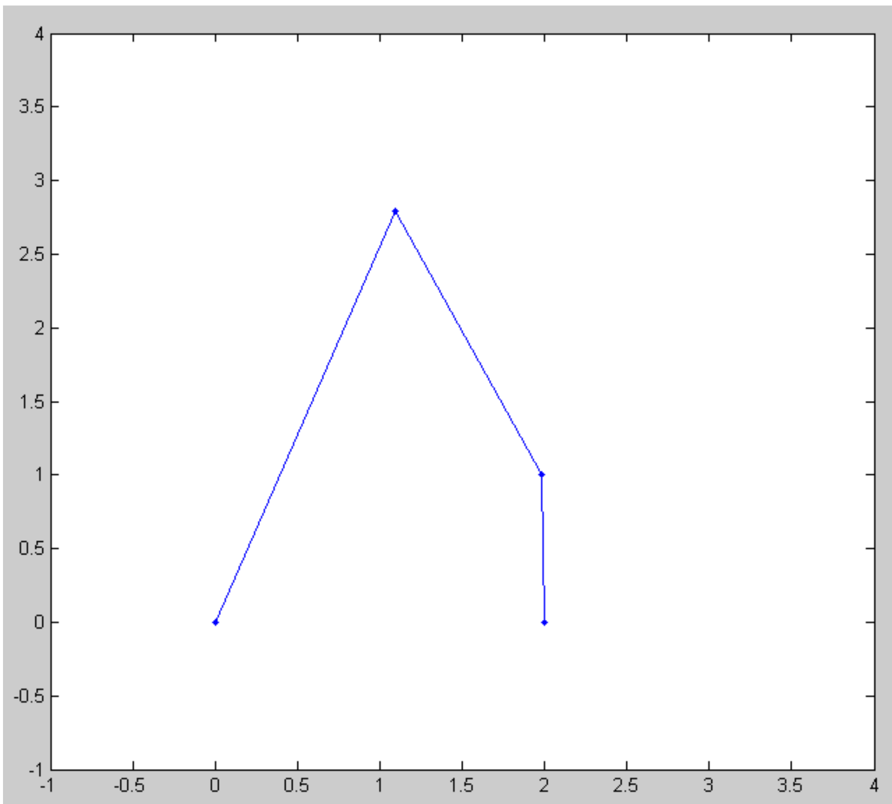
8) Use the fminsearch() to determine the joint angles which place the robot at (x3 = 2.0, y3 = 1.0)

From the command window:

```
>> [Q,e] = fminsearch('RRR_Cost', [41, -95, -45])

Q =    68.5320 -132.1667  -25.5707

e =   8.5387e-008
```

One solution is

- q1 = 68.5320 degrees
- q2 = -132.1667 degrees
- q3 = -25.5707 degrees



This solution isn't unique. With 3 degrees of freedom and two constraints, there are an infinite number of solutions. Change the initial guess and you converge to another valid solution

```
>> [Q,e] = fminsearch('RRR_Cost', [21, -195, -85])

Q =    7.4057 -153.5350 -167.1550

e = 3.7470e-009
```