Derek Frank

[dmfrank@ucsc.edu](mailto:dmfrank@ucsc.edu)

AMS 147

Due: 2/3/10

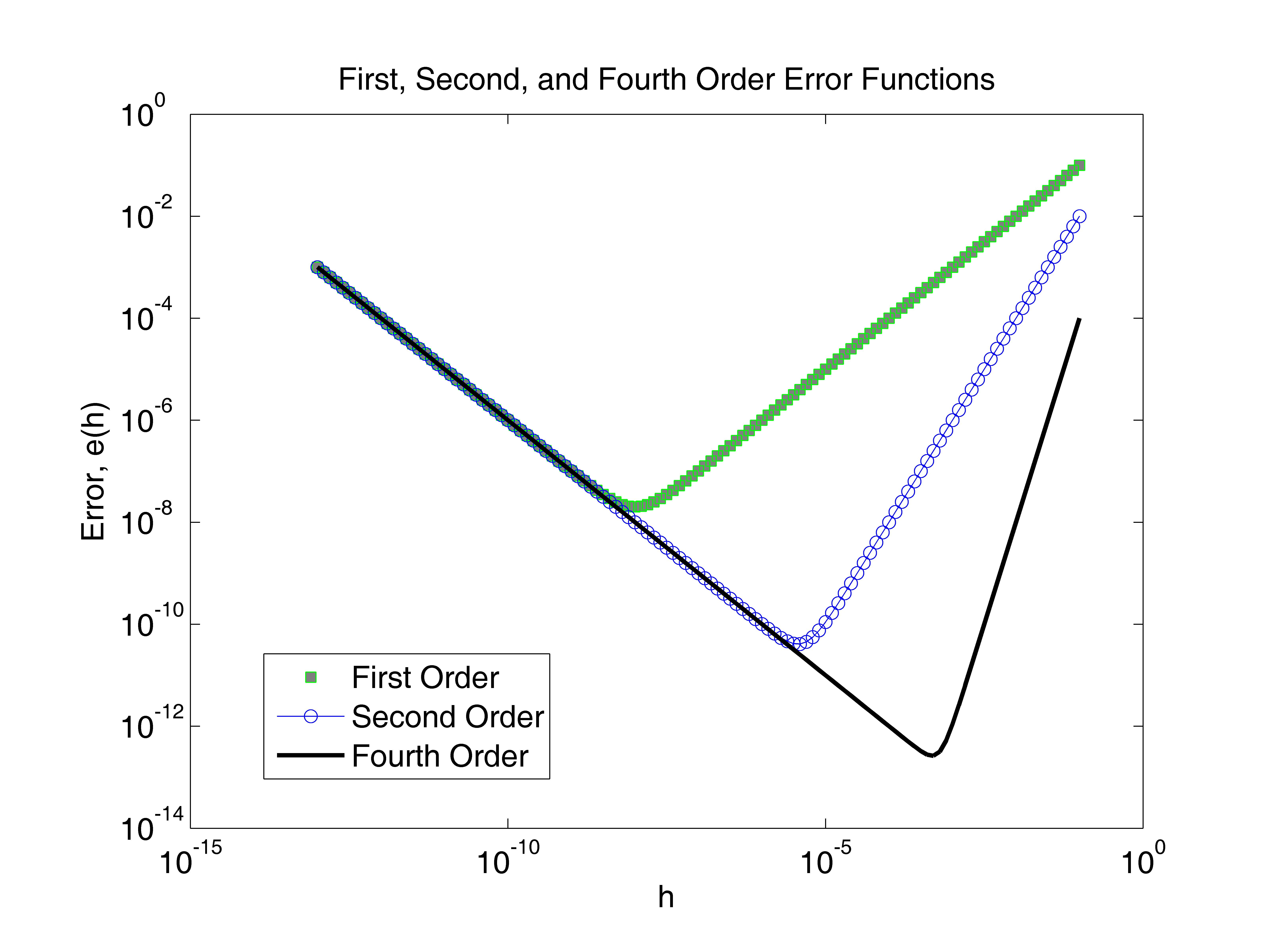
Homework #3

Part B:

(1):

i) In this problem, I am to graph three functions: first, second, and fourth order error functions as functions of *h*.

ii) Using Matlab, I set the vector/array *h* to values between [10-13, 10-1]. I then create the three functions of *h* and plot them.

iii)

iv) The results show that the higher the order of total error, the smaller the error is. As well, there is a value *h*, which is not too big nor too small, that produces the smallest total error for each order function. The minimum for the first order total error is approximately *ET* 2\*10-16 when *h* 10­­­-8­. The minimum for the second order total error is approximately *ET* 10-10.4 when *h* 10­­­-5.4. The minimum for the fourth order total error is approximately *ET* 10-12.6 when *h* 10­­­-3.3­.

(2):

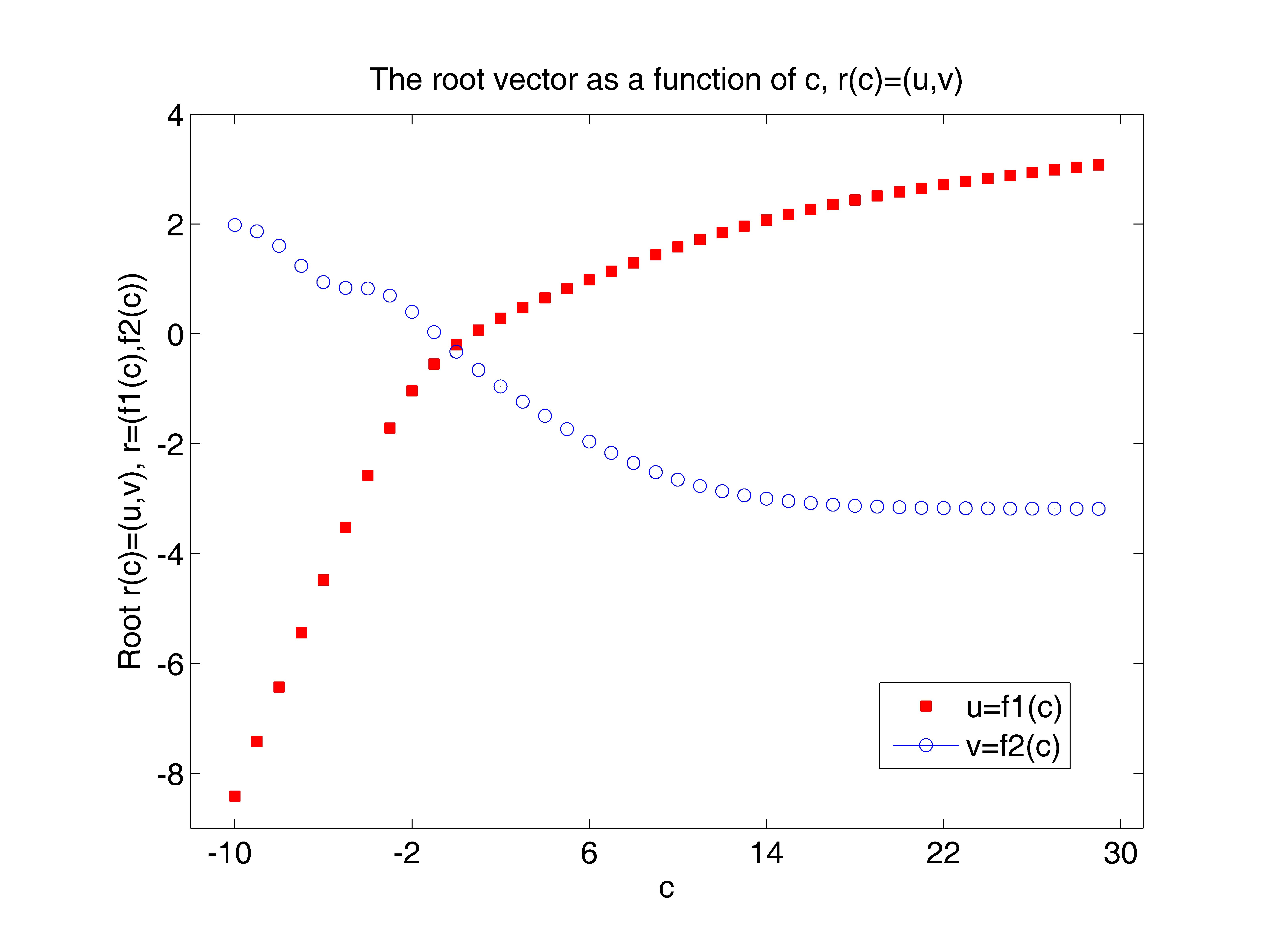
1. The problem I am to solve is to take the non-linear system:

*eu - cos(v) + u - v - c = 0*

*ev + sin(u) + u + v = 0*

*u* and *v* are functions of *c* and I am to plot the vectors corresponding to *c* values.

*c* is between [-10,30].

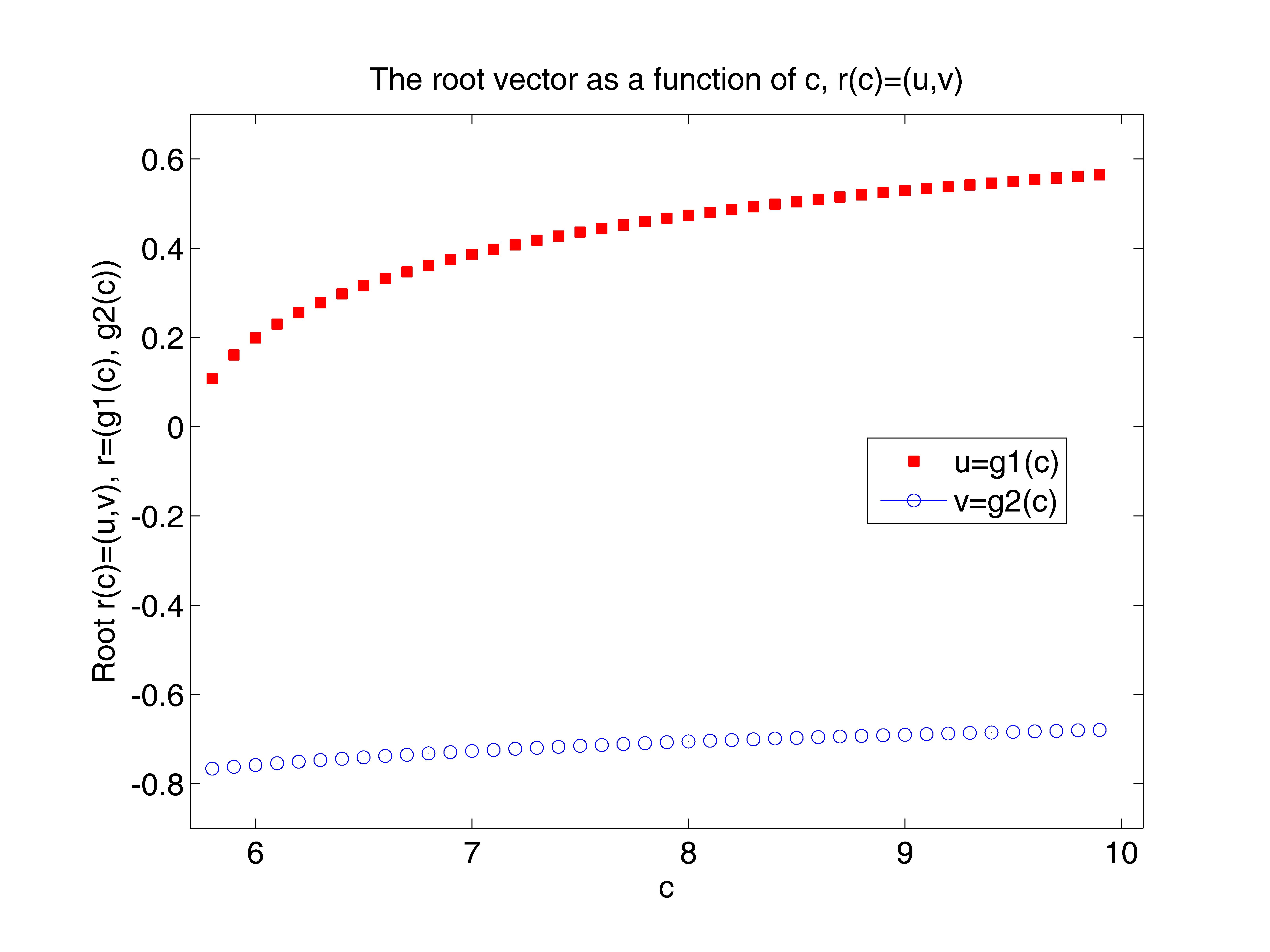
1. I use Newton’s method to solve the system. I attain a 2N matrix (N is the number of *c* values inputted so it is 40 in this case). Values for *u* and *v* are stored in each row of the root vector solved by Newton’s method and I simply plot the points for each *c*.
2. The results I obtain plot a root vector *r(c)=(u, v)* for each c. Since are 40 values of *c*, there exists 40 plotted roots. *c* is between [-10, 30]. The root vector is broken into two curves, one for *u=f1(c)* and the other for *v=f2(c)*. The length of the root vector approaches a minimum, *u≈-0.198562977* and *v≈-0.325989234*, when *c≈1*. As c is small, the length of the vector is very large, but as *c* gets larger (but not too big) the length becomes smaller until it reaches the minimum around *c≈1*, then the length the of vector increases again, although not as rapid as before the minimum value.

(3):

i) In this problem I am to plot the root vector of a given “g.m” file for about 40 values of *c* between [5.8, 10].

ii) Using Matlab and Newton’s method, I will solve the system for *g(r, c)=0*

(*g1(r, c=0*, *g2(r, c)=0*). Incrementing *c* by 0.1, I attain 42 points that graph *r(c)=(u, v)*.

iii)

iv) The results show the root vector as a function of *c* (*u(c)* resembles the graph of *y=√x*, while *v(c)* resembles *y=x/a*, but slightly curved and not a straight line). As *c* increases, the length of the root vector increases, but not rapidly. A minimum on this plot occurs when *c=5.8*, resulting in *u≈.107426025* and *v≈-0.766365027*.

APPENDIX

(Part B1):

“plot\_error.m”

% The first, second, and fourth order errors are calculated

% and plotted as a function of h.

clear

figure(1)

clf

axes('position',[0.15,0.13,0.75,0.75])

% h is between [10.^-13, 10.^-1]

h=10.^([-13:.1:-1]);

% This function calculates the first order error error as a

% function of h.

e1=h+(1\*(10.^-16))./h;

% This function calculates the second order error error as a

% function of h.

e2=(h.^2)+(1\*(10.^-16))./h;

% This function calculates the fourth order error error as a

% function of h.

e4=(h.^4)+(1\*(10.^-16))./h;

loglog(h, e1, 'gs', 'markerfacecolor', [.5,.5,.5])

hold on

loglog(h, e2, 'b-o')

hold on

loglog(h, e4, 'k-', 'linewidth',2.0)

set(gca,'fontsize',14)

xlabel('h')

ylabel('Error, e(h)')

title('First, Second, and Fourth Order Error Functions')

legend('First Order', 'Second Order', 'Fourth Order')

(Part B2):

“f.m”

function [y]=f(x,c)

% This function calculates f(x) for a given x and a parameter c.

u=x(1);

v=x(2);

y=zeros(2,1);

y(1)=exp(u)-cos(v)+u-v-c;

y(2)=exp(v)+sin(u)+v+u;

“fp\_2.m”

function [D]=fp\_2(x,c)

% This function calculates df(x)/dx for a given x and a parameter

% c, using a second finite difference method.

D=zeros(2,2);

h=1.0e-5;

D(:,1)=(f(x+[h,0]',c)-f(x-[h,0]',c))/(2\*h);

D(:,2)=(f(x+[0,h]',c)-f(x-[0,h]',c))/(2\*h);

“newton\_sys.m”

function [ru, rv, n]=newton\_sys(funct\_name, deriv\_name, c,u,v,tol)

% This function finds a root of f(x) = 0 using Newton's method.

%

% Input:

% funct\_name: the name of the .m file for calculating the

% function f(x)

% deriv\_name: the name of the .m file for calculating df(x)/dx

% c: a parameter in functions "f" and "fp"

% u, v: the starting point for Newton's method

% tol: the error tolerance

% Output:

% r: the root found

% n: the number of iterations

err=1.0;

n=0;

x0=[u v]';

%

while(err > tol),

n=n+1;

f\_x0=feval(funct\_name,x0,c);

fp\_x0=feval(deriv\_name,x0,c);

Dx=-fp\_x0\f\_x0;

x0=x0+Dx;

err=norm(Dx);

end

%

ru=x0(1);

rv=x0(2);

“run\_newton.m”

% Consider the non-linear system

% exp(u) - cos(v) + u - v - c = 0

% exp(v) + sin(u) + v + u = 0

% This code calculates the root vector (u, v) for c = [-10, 30]

clear

figure(2)

clf

axes('position',[0.15,0.13,0.75,0.75])

%

c=[-10:1:30];

N=40;

r\_u=zeros(N);

r\_v=zeros(N);

n=zeros(N);

tol=1.0e-10;

%

for i=1:N,

r\_u(i)=1;

r\_v(i)=1;

[r\_u(i), r\_v(i), n(i)]=newton\_sys('f', 'fp\_2', c(i), r\_u(i), r\_v(i), tol);

end

%

for j=1:N,

r=[r\_u(j) r\_v(j)]';

disp(' ')

disp([' The root found is (u, v) = (',num2str(r(1),'%16.8e'),', ',num2str(r(2),'%16.8e'),').'])

disp([' It takes n = ',num2str(n(j)),' iterations to reach err <= ',...

num2str(tol),'.'])

disp(' ')

plot(c(j), r\_u(j), 'sr', 'MarkerFaceColor', 'r')

hold on

plot(c(j), r\_v(j), 'b-o')

end

%

axis([-12,31,-9,4])

set(gca,'xtick',[-10:8:30])

set(gca,'ytick',[-8:2:4])

set(gca, 'fontsize', 14)

xlabel('c')

ylabel('Root r(c)=(u,v), r=(f1(c),f2(c))')

title('The root vector as a function of c, r(c)=(u,v)')

legend('u=f1(c)','v=f2(c)')

(Part B3):

“g.m”

function [z]=g(r,c)

% r=(u,v)'

% z=g(r,c)=(g1(u,v,c), g2(u,v,c))'

%

% DO NOT MODIFY THIS FILE!!!!!

N=128;

da=pi/N;

a=[0:N]\*da;

[x,y]=meshgrid(a,a);

%

alpha=c;

mu=0.6;

alpha0=1;

E=1;

r1=r(1);

r3=r(2);

%

h1=sin(y).\*cos(x);

h3=cos(y);

g0=sin(y).\*exp(alpha\*r1\*h1+(alpha\*r3+mu\*E)\*h3+0.5\*alpha0\*E^2\*h3.^2);

%

g=g0;

g(:,1)=0.5\*(g(:,1)+g(:,N+1));

u1=sum(g(:,1:N),2)\*da;

%

g=h1.\*g0;

g(:,1)=0.5\*(g(:,1)+g(:,N+1));

u2=sum(g(:,1:N),2)\*da;

%

g=h3.\*g0;

g(:,1)=0.5\*(g(:,1)+g(:,N+1));

u3=sum(g(:,1:N),2)\*da;

%

u=[u1, u2, u3];

u(1,:)=0.5\*(u(1,:)+u(N+1,:));

v1=sum(u(1:N,:),1)\*da;

v2=sum(u(1:2:N,:),1)\*2\*da;

v3=sum(u(1:4:N,:),1)\*4\*da;

v4=sum(u(1:8:N,:),1)\*8\*da;

v5=sum(u(1:16:N,:),1)\*16\*da;

v6=sum(u(1:32:N,:),1)\*32\*da;

v1=(4\*v1-v2)/3;

v2=(4\*v2-v3)/3;

v3=(4\*v3-v4)/3;

v4=(4\*v4-v5)/3;

v5=(4\*v5-v6)/3;

v1=(16\*v1-v2)/15;

v2=(16\*v2-v3)/15;

v3=(16\*v3-v4)/15;

v4=(16\*v4-v5)/15;

v1=(64\*v1-v2)/63;

v2=(64\*v2-v3)/63;

v3=(64\*v3-v4)/63;

v1=(256\*v1-v2)/255;

v2=(256\*v2-v3)/255;

v1=(512\*v1-v2)/511;

err=(v1-v2)/norm(v1);

%

z=[v1(2)/v1(1)-r1; v1(3)/v1(1)-r3];

“gp\_2.m”

function [D]=gp\_2(r,c)

% This function calculates df(x)/dx for a given x and a parameter

% c, using a second finite difference method.

D=zeros(2,2);

h=1.0e-5;

D(:,1)=(g(r+[h,0]',c)-g(r-[h,0]',c))/(2\*h);

D(:,2)=(g(r+[0,h]',c)-g(r-[0,h]',c))/(2\*h);

“run\_newton\_g.m”

% Consider the non-linear system

% g1(u,v,c)=0

% g2(u,v,c)=0

% This code calculates the root vector (u, v) for c = [5.8, 10]

clear

figure(3)

clf

axes('position',[0.15,0.13,0.75,0.75])

%

c=[5.8:.1:10];

N=42;

r\_u=zeros(N);

r\_v=zeros(N);

n=zeros(N);

tol=1.0e-10;

%

r\_u(1)=0.1;

r\_v(1)=-0.8;

[r\_u(1), r\_v(1), n(1)]=newton\_sys('g', 'gp\_2', c(1), r\_u(1), r\_v(1), tol);

for i=2:N,

r\_u(i)=r\_u(i-1);

r\_v(i)=r\_v(i-1);

[r\_u(i), r\_v(i), n(i)]=newton\_sys('g', 'gp\_2', c(i), r\_u(i), r\_v(i), tol);

end

for j=1:N,

r=[r\_u(j) r\_v(j)]';

disp(' ')

disp([' The root found is (u, v) = (',num2str(r(1),'%16.8e'),', ',num2str(r(2),'%16.8e'),').'])

disp([' It takes n = ',num2str(n(j)),' iterations to reach err <= ',...

num2str(tol),'.'])

disp(' ')

plot(c(j), r\_u(j), 'sr', 'MarkerFaceColor', 'r')

hold on

plot(c(j), r\_v(j), 'b-o')

end

%

axis([5.7,10.1,-.9,.7])

set(gca,'xtick',[6:1:10])

set(gca,'ytick',[-.8:.2:.6])

set(gca, 'fontsize', 14)

xlabel('c')

ylabel('Root r(c)=(u,v), r=(g1(c), g2(c))')

title('The root vector as a function of c, r(c)=(u,v)')

legend('u=g1(c)','v=g2(c)')