Derek Frank

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

AMS 147

Due: 1/27/10

Homework #2

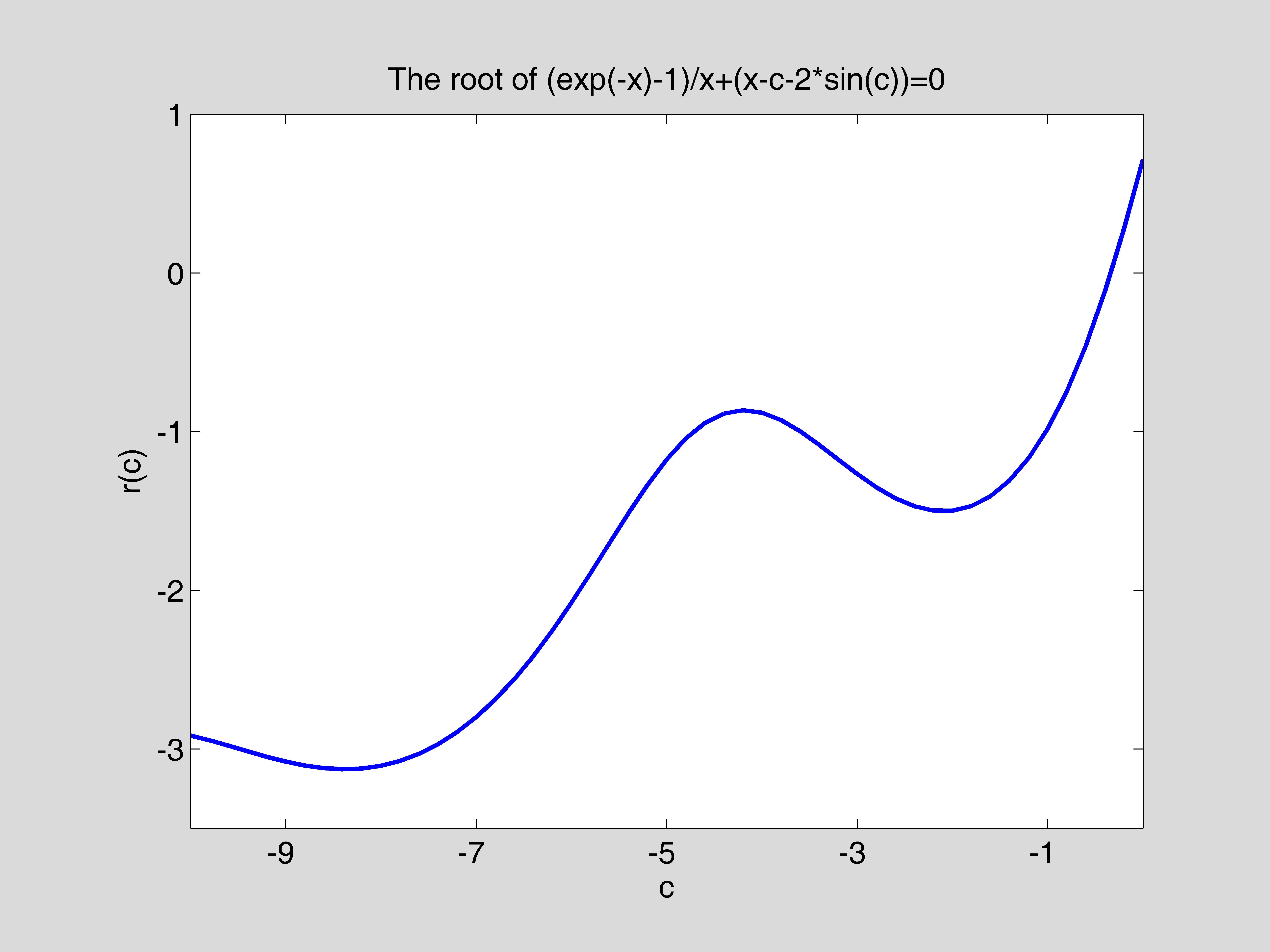
Part B:

(1):

i) The problem I am to solve is the non-linear equation

*(exp(-x)-1)/x+(x-c-2\*sin(c))=0*. I am also to plot the function's root as a function of *c* contained in [-10,0].

ii) Using Matlab, I am to use Newton's method to solve the given equation. Newton's method will produce a root as a function of *c*.

iii)

iv) The root of the given function is shown above contained on an interval where *c* is

between [-10,0]. The root, on this interval, does not exceed a value above 1 or below -4. Although the root is wave-like and slightly decreases after a steady increase, it overall becomes increasingly larger as *c* gets bigger.

(2):

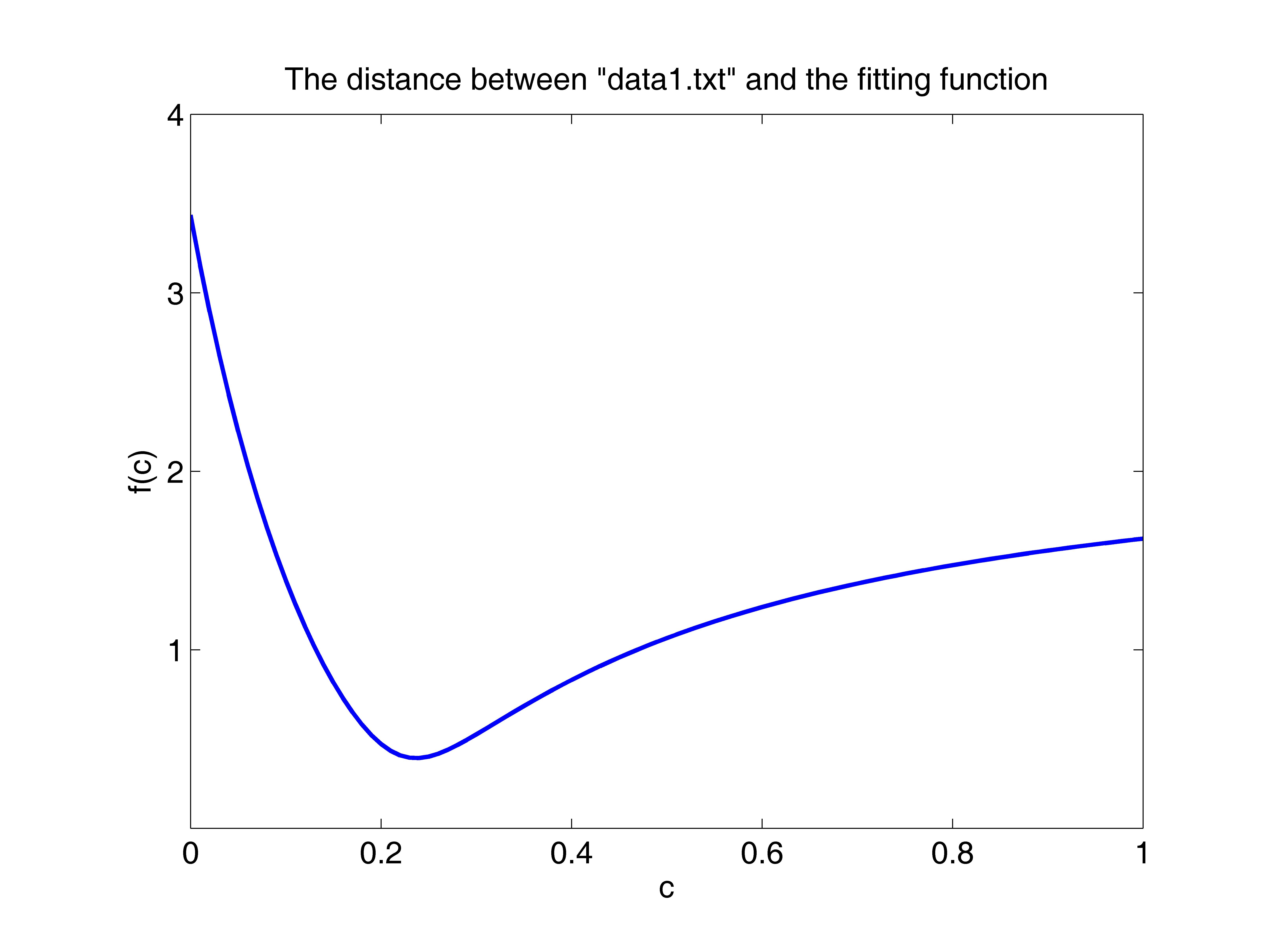
1. The problem I am to solve is to find the distance between data points found in "data1.txt" and the fitting function

*f(x)=exp(-c\*x)\*cos(2\*x)*. I am to plot this distance function,

*dd(c)=sqrt(sum of (f(x(i))-y(i))^2)* (*i=0,1,…,50*), as a function of *c* in [0,1].

1. Using Matlab, I am to graph the distance between the data and fitting function

with a varying *c* value. Using a for loops to increment the *c* vector and its corresponding *d* (distance) vector. Each time around the loop I send to “dd.m” a *c* and all *x* and y values. The funtion calculates the fitting function *y* value and sends back a distance which I store in the *d* vector according to its corresponding *c* value. I am able to save information in the form of vectors and use these vectors to plot the curve.

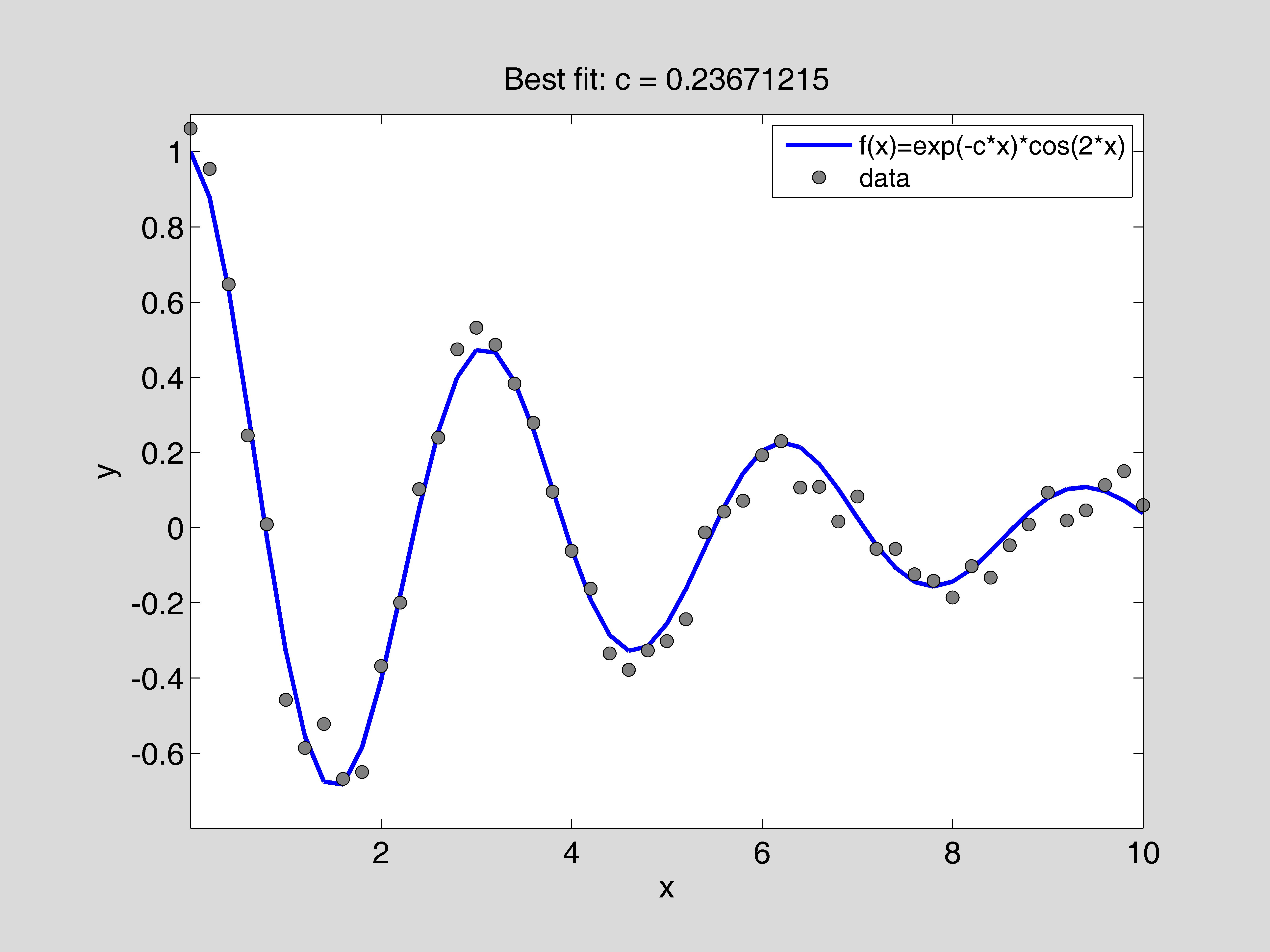
iii)

1. These results show the distance (*f(c)*) between the fitting funcion and the given data points. This graph only shows the values of *f(c)* when *c* is between [0,1]. The greatest distance, on this graph, between the data points and the fitting function is greatest when c is 0 and quadratically decreases to a minimum (optimal value) at a little after *c*=.2. After the distance reaches a minimum and *c* keeps increasing, it does not grow like a quadratic funtion, but much more slowly and even appears to possibly be asymptotic at around *y*=1.8.

(3):

i) The problem I am to solve is find the optimal fitting function from the previous problem. That is find the best value of *c* that minimizes the distance between the given data in "data1.txt" and the fitting function *f(x)=exp(-c\*x)\*cos(2\*x)*.

ii) Using the golden search method to find the best value for c, I enter code in Matlab to calculate and graph my problems.

iii)

iv) The results show that the best value for *c* to optimize the distance is

*c* = .23671215 (this value is very close to the average for *c* in homework assignment #1).

APPENDIX

(Part B1):

"f.m"

function [y]=f(x,c)

%

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

%

y=(exp(-x)-1)./x+(x-c-2.\*sin(c));

%

"fp.m"

function [y]=fp(x,c)

%

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

%

y=-1.\*(exp(-x)./(x.^2))-1.\*(exp(-x)./x)+1./(x.^2)+2;

%

"fp\_2.m"

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

%

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

%

h=1.0e-5;

y=(f(x+h,c)-f(x-h,c))/(2\*h);

%

"newton.m"

function [r, n]=newton(funct\_name, deriv\_name, c, x0, 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"

% x0: 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;

%

while(err > tol),

n=n+1;

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

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

x1=x0-f\_x0/fp\_x0;

err=abs(x1-x0);

x0=x1;

end

%

r=x0;

%

"calc\_data.m"

%

% Consider the non-linear equation (exp(-x)-1)/x+(x-c-2\*sin(c))=0.

% Here "c" is a parameter in the equation. The root of the

% equation varies with "c" and thus the root is a function of "c".

% This code calculates the root for "c" in [-10,0] and stores the

% data in data1.mat. Later on the data is used in "plot\_curve.m

% to plot the root as a function of "c".

%

clear

%

c\_v=[-10:0.2:0];

nc=size(c\_v,2);

r\_v=zeros(1,nc);

tol=1.0e-10;

r=1;

%

for i=1:nc,

c=c\_v(i);

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

r\_v(i)=r;

end

%

save data1 c\_v r\_v

%

"plot\_curve.m"

%

% Consider the non-linear equation (exp(-x)-1)/x+(x-c-2\*sin(c))=0.

% Here "c" is a parameter in the equation. The root of the

% equation varies with "c" and thus the root is a function of "c".

% The code "calc\_data.m" calculates the root for "c" in [-10,0]

% and stores the data in data1.mat.

% This code loads in data1.mat and plots the root as a function

% of "c".

%

clear

clf

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

%

load data1.mat

plot(c\_v, r\_v,'linewidth',2.0)

axis([-10,0,-3.5,1.0])

set(gca,'xtick',[-9:2:0])

set(gca,'ytick',[-3:1:1])

set(gca,'fontsize',14)

xlabel('c')

ylabel('r(c)')

title('The root of (exp(-x)-1)/x+(x-c-2\*sin(c))=0')

%

(Part B2):

"data1.txt"

% x y

%

0.0000000e+00 1.0616627e+00

2.0000000e-01 9.5438313e-01

4.0000000e-01 6.4726669e-01

6.0000000e-01 2.4543142e-01

8.0000000e-01 8.6474234e-03

1.0000000e+00 -4.5847332e-01

1.2000000e+00 -5.8629116e-01

1.4000000e+00 -5.2270060e-01

1.6000000e+00 -6.6897708e-01

1.8000000e+00 -6.5025323e-01

2.0000000e+00 -3.6829629e-01

2.2000000e+00 -1.9946104e-01

2.4000000e+00 1.0214847e-01

2.6000000e+00 2.3939828e-01

2.8000000e+00 4.7441685e-01

3.0000000e+00 5.3204202e-01

3.2000000e+00 4.8645531e-01

3.4000000e+00 3.8277756e-01

3.6000000e+00 2.7859469e-01

3.8000000e+00 9.5304196e-02

4.0000000e+00 -6.1937557e-02

4.2000000e+00 -1.6264995e-01

4.4000000e+00 -3.3465224e-01

4.6000000e+00 -3.7848057e-01

4.8000000e+00 -3.2684557e-01

5.0000000e+00 -3.0202142e-01

5.2000000e+00 -2.4418751e-01

5.4000000e+00 -1.2606921e-02

5.6000000e+00 4.2709432e-02

5.8000000e+00 7.1364914e-02

6.0000000e+00 1.9256373e-01

6.2000000e+00 2.2975779e-01

6.4000000e+00 1.0651973e-01

6.6000000e+00 1.0873230e-01

6.8000000e+00 1.6313790e-02

7.0000000e+00 8.2753168e-02

7.2000000e+00 -5.6705147e-02

7.4000000e+00 -5.6552321e-02

7.6000000e+00 -1.2417349e-01

7.8000000e+00 -1.4140695e-01

8.0000000e+00 -1.8606224e-01

8.2000000e+00 -1.0253530e-01

8.4000000e+00 -1.3337982e-01

8.6000000e+00 -4.7083796e-02

8.8000000e+00 8.5379099e-03

9.0000000e+00 9.3244738e-02

9.2000000e+00 1.8795176e-02

9.4000000e+00 4.5569780e-02

9.6000000e+00 1.1312777e-01

9.8000000e+00 1.5024618e-01

1.0000000e+01 5.9238496e-02

%

"dd.m"

function [dist]=dd(c,x,y)

%

% This function calculates the distance between the data

% and the function f(x)=exp(-c\*x)\*cos(2x)

%

fx=exp(-c\*x).\*cos(2\*x);

dist=norm(fx-y);

"calc\_data.m”

%

% Consider the non-linear equation sqrt(sum of (f(x(i))-y(i))^2), % where, depending on c, x(i) is an array of values produced by

% the fitting funtion found in "dd.m" and y(i) is an array of

% values found in "data1.txt". Here "c" is a parameter in the

% equation. The distance of the equation varies with "c" and thus

% the distance is a function of "c".

% This code calculates the distance for "c" in [0,1] and stores

% the data in data1.mat. Later on the data is used "plot\_curve.m"

% to plot he distance as a function of "c".

%

load -ascii data1.txt

% vector of x values found in "data1.txt"

x=data1(:,1);

% array/vector of y values found in "data1.txt"

y1=data1(:,2);

% array/vector of 100 c values between [0,1]

c\_v=[0:.01:1];

% scalar number of c values in array/vector c\_v

nc=size(c\_v,2);

% initialization for array/vector of distance values

d\_v=zeros(1,nc);

for i=1:nc,

c=c\_v(i);

d(i)=dd(c,x,y1);

end

%

save data1 c\_v y\_v

“plot\_curve.m”

%

% Consider the non-linear equation fitting function

% f(x)=exp(-c\*x)\*cos(2\*x). With a given x, "c" is a parameter in

% the equation. The code "calc\_data.m" calculates

% sqrt(sum of (f(x(i))-y(i))^2), the distance between the data

% given in "datal.txt" and the fitting function for "c" in [0,1]

% and stores the data in data1.mat. This code loads in data1.mat

% and plots the distance as a function of "c".

%

clear

figure(3)

clf

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

%

load data1.mat

plot(c\_v, y\_v,'linewidth',2.0)

axis([0,1,0,4])

set(gca,'xtick',[0:.2:1])

set(gca,'ytick',[1:1:4])

set(gca,'fontsize',14)

xlabel('c')

ylabel('f(c)')

title('The distance between "data1.txt" and the fitting function')

%

(Part B3):

"dd.m"

function [dist]=dd(c,x,y)

%

% This function calculates the distance between the data

% and the function f(x)=exp(-c\*x)\*cos(2x)

%

fx=exp(-c\*x).\*cos(2\*x);

dist=norm(fx-y);

"golden.m"

%

% This code first reads in data (x, y) from "data1.txt."

% Then it uses the golden search method to find the value of c

% such that the distance between the data and the function

% f(x)=exp(-c\*x)\*cos(2\*x) is minimized.

% Finally, it plots the data along with the best fit.

%

clear

clf reset

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

%

load -ascii data1.txt

x=data1(:,1);

y=data1(:,2);

%

a=0;

b=2;

tol=1.0e-10;

n=0;

%

g=(sqrt(5)-1)/2;

r1=a+(b-a)\*(1-g);

f1=dd(r1,x,y);

r2=a+(b-a)\*g;

f2=dd(r2,x,y);

%

while (b-a) > tol,

n=n+1;

if f1 < f2,

b=r2;

r2=r1;

f2=f1;

r1=a+(b-a)\*(1-g);

f1=dd(r1,x,y);

else

a=r1;

r1=r2;

f1=f2;

r2=a+(b-a)\*g;

f2=dd(r2,x,y);

end

end

c0=(a+b)/2;

%

fx=exp(-c0\*x).\*cos(2\*x);

plot(x,fx,'b-','linewidth',2.0)

hold on

plot(x,y,'ko','markerfacecolor',[0.5,0.5,0.5])

axis([0,10,-.8,1.1])

set(gca,'xtick',[2:2:10])

set(gca,'ytick',[-.6:0.2:1.1])

set(gca,'fontsize',14)

xlabel('x')

ylabel('y')

title(['Best fit: c = ',num2str(c0,8)])

h1=legend('f(x)=exp(-c\*x)\*cos(2\*x)','data');

set(h1,'fontsize',12)

%

disp(' ')

disp([' The function attains a minimum at c0 = ',num2str(c0,'%24.16e'),'.'])

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

disp(' ')