Homework 2 - Solutions
MA/CS 375, Fall 2005

1. Use the bisection method, Newton's method, and the MATLAB® function `fzero` to compute a positive real number x satisfying:

$$\sinh x = \cos x.$$

For each of the three methods use a tolerance of 10^{-8}. List your initial approximation (or interval in the case of bisection) and the number of iterations needed. Also print at least nine digits of the approximate roots.

Solution:

Here general-purpose bisection script similar to the one in the text, although it is simpler and does not rely on using the `feval` function.

```
function [xz,res,iter]=bisect(f,a,b,tol,maxit)

iter=0;
fa=f(a); fb=f(b);
if fa*fb>0
    error(['The sign of the the function at the extrema of the', ...
    'interval must be different']);
elseif fa==0
    xz=a; res=0;
    return
elseif fb==0
    xz=b; res=0;
    return
end
x=(a+b)/2;  fx=f(x);    dx=(b-a)/2;
while (iter<maxit & dx>tol)
    iter=iter+1;
    if fa*fx<0
        b=x;    fb=fx;
    elseif fx*fb<0
        a=x;    fa=fx;
    else  % done
    end
    dx=(b-a)/2;
    x=a+dx; fx=f(x);
end
```

```
if iter>maxit
    fprintf(['bisection stopped without converging to the desired ',...
        'tolerance because the maximum number of iterations was reached']);
end
xz=x;
res=f(x);
```

To use Newton's method, we define $f(x) = \sinh(x) - \cos(x)$ so that $f'(x) = \cosh(x) + \sin(x)$. The Newton iteration is simply

$$x_{k+1} = x_k - \frac{\sinh(x_k) - \cos(x_k)}{\cosh(x_k) + \sin(x_k)}$$

By inspecting the functions $\sinh(x)$ and $\cos(x)$ one can immediately see that the zero must lie somewhere between 0 and $\pi/2$, so we can use these to define our initial bisection interval and take the midpoint $\pi/4$ as the initial point for the Newton's method. Computing to 9-digit precision, one obtains for the root:

$$x_{\text{bisect}} = 0.703290663199065, \quad x_{\text{Newton}} = 0.703290658873561, \quad x_{\texttt{fzero}} = 0.703290658863965$$

Table 1: Finding the root of $\sinh(x) = \cos(x)$, $|f(x_n)|$

iterations	bisection	Newton's method
1	0.52100915059422	1.19331410800446
2	0.207760588557854	1.31032700775238
3	0.0304020718929473	0.189609665360664
4	0.0637188965127375	0.00667324312269613
5	0.0161973962740738	9.32165948031027e-06
6	0.00721702718877359	1.82739379184227e-11
7	0.00446144162724882	1.11022302462516e-16
8	0.00138496968849744	$O(\epsilon_M)$
9	0.00153644063981195	
10	7.52867810949187e-05	
$\vdots$	$\vdots$	$\vdots$
25	2.8881665947722e-09	$O(\epsilon_M)$

2. Use the bisection method, Newton's method, and the MATLAB® function `fzero` to compute all three real numbers x satisfying:

$$5x^2 - e^x = 0.$$

For each of the three roots and each of the three methods use a tolerance of 10^{-8}. List your initial approximation (or interval in the case of bisection) and the number of iterations needed. Also print at least nine digits of the approximate roots.

Solution: This function has three roots, denoted as $x^{(1)}$, $x^{(2)}$,$x^{(3)}$. Plotting on a graph, one sees that

$$-1 < x^{(1)} < 0, \quad 0 < x^{(2)} < 1, \quad 4 < x^{(3)} < 6$$

As with problem 1, we use these bounds to set the initial bisection intervals and the midpoints, $-1/2$, $1/2$ and 5 as the initial values for Newton's method. For the leftmost zero, $x^{(1)}$, the bisection method converged to within tolerance in 22 iterations and Newton's method converged in 6 iterations. The roots obtained were

$$x^{(1)}_{\text{bisect}} = -0.371417641639709, \quad x^{(1)}_{\text{Newton}} = -0.371417756797242, \quad x^{(1)}_{\texttt{fzero}} = -0.371417752459174$$

For the middle root, bisection converged in 23 iterations and Newton's method converged in 5 iterations, although the original guess for Newton's method 1/2 returned $x^{(1)}$. To obtain $x^{(2)}$, an initial guess of $x0 = 2$ was used. The roots obtained were

$$x^{(2)}_{\text{bisect}} = 0.605267107486725, \quad x^{(2)}_{\text{Newton}} = 0.605267121314618, \quad x^{(2)}_{\texttt{fzero}} = 0.605267121314618$$

In computing $x^{(3)}$, bisection converged in 27 iterations and Newton's method converged in 5 iterations.

$$x^{(3)}_{\text{bisect}} = 4.70793791860342, \quad x^{(3)}_{\text{Newton}} = 4.70793791814078, \quad x^{(3)}_{\texttt{fzero}} = 4.70793791812886$$

Newton's method outperformed the method of bisection in all three cases, although in case $x^{(2)}$ a different initial guess was needed to find the correct root.

3. Consider the use of Newton's method to solve:

$$e^{-x^2} - 1 = 0,$$

for the root $x = 0$. Reformulate the method as a fixed point iteration and find the rate at which it will converge. (Hint: use l'Hopital's rule to evaluate the derivative of the iteration function as $x \to 0$.)

Solution: This can be written as a fixed point method as

$$x = \phi(x) = x - \frac{f(x)}{f'(x)} \Rightarrow x_{n+1} = \phi(x_n) = x_n - \frac{f(x_n)}{f'(x_n)}$$

The fixed point function is

$$\phi(x) = \frac{1 + 2x^2 - e^{x^2}}{2x}$$

It has the limit at the fixed point

$$\lim_{x \to 0} \phi(x) = 0$$

since the numerator goes to zero quadratically whereas the denominator goes to zero linearly. The derivative of the fixed point function is

$$\phi'(x) = \frac{(2x^2 - 1)(1 - e^{x^2})}{2x^2}$$

and it has the limit at the fixed point

$$\lim_{x \to 0} \phi'(x) = \frac{1}{2}$$

The order of convergence is determined by the equation

$$\lim_{n \to \infty} \frac{e_{n+1}}{e_n} = \lim_{n \to \infty} \frac{x_{n+1} - x^*}{(x_n - x^*)^p} = \frac{1}{p!}\phi^{(p)}(x^*) \neq 0$$

Where p is the lowest order of derivative for which the fixed point function is nonzero. Since the we have shown that the first derivative has a nonzero limit, then $p = 1$ so the convergence is linear instead of quadratic. The asymptotic error constant is

$$c = \frac{1}{p!}\phi^{(p)}(x^*) = \frac{1}{2}$$

In other words, the theory predicts that when Newton's method converges, that the error of the $n + 1$ iteration will be half of the error of the n iteration. Assuming we take a point x closed to the fixed point $x^* = 0$ we can approximate the ration $f(x)/f'(x)$ using l'Hopitals rule to get $f'(x)/f''(x)$

$$x_{n+1} = x_n - \frac{x_n}{2x_n^2 - 1}$$

Eventually the convergence breaks down due to the errors of evalating the e^{-x^2} functions. If the l'Hopital approximation above is used, then the root can be evaluated tp machine precision.

4. Consider a 4-bar planar linkage ABCD where the four rods have lengths $AB = a_1$, $BC = a_2$, $CD = a_3$ and $DA = a_4$. If we introduce the angles $\alpha = \angle ABC$ and $\beta = \pi - \angle DAB$, we have the system described in the text, **problem 2.3**, p. 38 (see also Fig. 2.1 in the text). This is called a planar linkage system, and it can exist in various shapes. As the angle α is varied that will result in turn in changes in the angle β; the two angles are related by equation (text, eq. 2.2):

$$\frac{a_1}{a_2}\cos(\beta) - \frac{a_1}{a_4}\cos(\alpha) - \cos(\beta - \alpha) = -\frac{a_1^2 + a_2^2 - a_3^2 + a_4^2}{2a_2a_4}$$

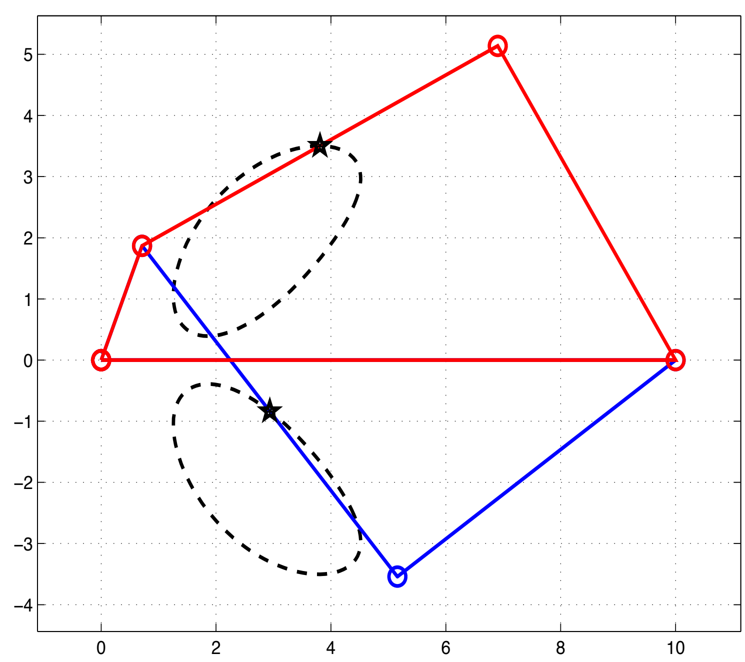
Apply Newton's method to solve this problem for $\alpha \in [0, 2\pi]$ with a tolerance of 10^{-6}. Assume that the lengths of the rods are $a_1 = 10$, $a_2 = 2$, $a_3 = 7$, $a_4 = 6$. For this arrangement, and to each value of α in the given range there correspond two possible values of β. Use an initial value of $\beta = -\pi/2$, which ensures that you will get a solution for β in the same contiguous range. Plot the position of the mid-point of the rod CD as α takes values in $[0, 2\pi]$.

Solution: The Newton solver with linkage plotting is

```
function [mx1,my1,mx2,my2]=linkage(a,k)
% Evaluate the midpoint of CD for the vector of angles in a (alpha)
%
% Draw the linkage system in the configuration dictated by the kth
% value of a (alpha)

n=length(a);

% Two possible configurations
b1=-pi/2*ones(n,1);
b2=pi/2*ones(n,1);

a1=10; a2=2; a3=7;  a4=6;

c1=a1/a2; c2=a1/a4;
c3=(a1^2+a2^2-a3^2+a4^2)/(2*a2*a4);

% Newton iteration for each configuration for every a (alpha)
for iter=1:7
    f1=c1*cos(b1)-c2*cos(a)-cos(b1-a)+c3;    f1p=sin(b1-a)-c1*sin(b1);
    f2=c1*cos(b2)-c2*cos(a)-cos(b2-a)+c3;    f2p=sin(b2-a)-c1*sin(b2);
    b1=b1-f1./f1p;    b2=b2-f2./f2p;
end

if(1) % Do the plotting

    Cx=a2*cos(a);         Cy=a2*sin(a);
    Dx1=a1+a4*cos(b1);   Dx2=a1+a4*cos(b2);
    Dy1=a4*sin(b1);      Dy2=a4*sin(b2);

    mx1=(Cx+Dx1)/2;    mx2=(Cx+Dx2)/2;
    my1=(Cy+Dy1)/2;    my2=(Cy+Dy2)/2;

    % The loops followed by the midpoints
```

```
    plot(mx1,my1,'k--',mx2,my2,'k--','LineWidth',2);
    grid on;
    hold on

    % The linkage for a specific angle
    x1=[0 a1 Dx1(k) Cx(k) 0];    y1=[0 0 Dy1(k) Cy(k) 0];
    x2=[0 a1 Dx2(k) Cx(k) 0];    y2=[0 0 Dy2(k) Cy(k) 0];
    plot(x1,y1,'bo-',mx1(k),my1(k),'kp-','LineWidth',2,'MarkerSize',10)
    plot(x2,y2,'ro-',mx2(k),my2(k),'kp-','LineWidth',2,'MarkerSize',10)
    hold off;
    axis([-4 12 -6 6])
end
```

Note the Newton method works with arrays of α's simultaneously, so one can simply choose
`>>a=linspace(0,2*pi,100)';`
and solve the system for all 100 values of α at one time. The second input parameters determines which particular value of α to use to draw the linkage system.



Explaination of grading scheme:

Problem 1 (15 pts):
Bisection (7)
Newton (7)
`fzero` (1)

Problem 2 (30 pts):
10 pts per root

Problem 3 (15 pts):
Effort at determining the order of convergence (10)
Determining it correctly (5)

Problem 4 (40 pts):
Code present
Any kind of graph
The correct graph

Table 2: Finding the root of $\exp(-x^2) - 1$

iterations	e_n	e_n/e_{n-1}
1	0.140859085770477	
2	0.0697261925839738	0.495006709738192
3	0.0347782110958669	0.498782592409333
4	0.0173785850367956	0.499697497059039
5	0.00868798023700563	0.4999244885939
6	0.00434382617002007	0.499981129275359
7	0.00217189259415103	0.49999528276266
8	0.00108594373579125	0.499998820713202
9	0.000542971547745431	0.499999705187126
10	0.000271485733865846	0.499999926318663
11	0.000135742861985003	0.499999981774658
12	6.78714303895193e-05	0.49999999555791
13	3.39357151682078e-05	0.499999999608792
14	1.6967857193955e-05	0.499999988503294
15	8.48392965485562e-06	0.500000062346006
16	4.24196314361372e-06	0.49999980152899
17	2.12097904619955e-06	0.499999404613566
18	1.06049882090937e-06	0.500004383734771
19	5.30250270994825e-07	0.500000811448466
20	2.65074082013324e-07	0.499903718136735
21	1.32512805322296e-07	0.499908570146948
22	6.63247726019351e-08	0.500515949689699
23	3.28463909634952e-08	0.495235636322964
24	1.59461624747848e-08	0.48547685170365
25	8.98384143747749e-09	0.563385795904397
26	2.8048419952473e-09	0.31220965048943