1. The requested MATLAB® functions are as follows.

```
  function [x k] = forward(a,b,c,x0,tol)
% function [x k] = forward(a,b,c,x0,tol)
% Uses "forward" fixed-point iteration x(k+1) = -(1/a)(b + c/x(k)),
% to find a root x of the quadratic equation ax^2 + bx + c = 0. x0
% is the initial iterate, and |x(k+1)-x(k)| <= tol/2 controls the
% termination. Also returned is the total number k of iterations.
x   = -(b + c/x0)/a; k = 1; % single iteration so far.
err = abs(x-x0);
while err > 0.5*tol
   x0  = x;
   x   = -(b + c/x0)/a; k = k+1;
   err = abs(x-x0);
end
```

```
  function [x k] = backward(a,b,c,x0,tol)
% function [x k] = backward(a,b,c,x0,tol)
% Uses "backward" fixed-point iteration x(k+1) = -c/(b + ax(k)),
% to find a root x of the quadratic equation ax^2 + bx + c = 0. x0
% is the initial iterate, and |x(k+1)-x(k)| <= tol/2 controls the
% termination. Also returned is the total number k of iterations.
x   = -c/(b + a*x0); k = 1; % single iteration so far.
err = abs(x-x0);
while err > 0.5*tol
   x0  = x;
   x   = -c/(b + a*x0); k = k+1;
   err = abs(x-x0);
end
```

The quadratic equation $x^2 - bx - 1 = 0$ in question has roots $x_\pm = \frac{1}{2}(1 \pm \sqrt{5})$, or

$$x_+ \simeq 1.618033988749895\text{e+}00, \qquad x_- \simeq -6.180339887498949\text{e-}01.$$

At the command line we verify that each of our fixed-point schemes finds one of these roots to the desired tolerance.

```
>> format compact; a=1; b=-1; c=-1; x0=1; tol=1e-10; % These will not change between uses of forward/backward.
>> format long e; [x k] = forward(a,b,c,x0,tol)
x =
     1.618033988738303e+00
k =
    26
>> format short e; err = abs(0.5*(1+sqrt(5))-x)
err =
   1.1592e-11
>> format long e; [x k] = backward(a,b,c,x0,tol)
x =
    -6.180339887383031e-01
k =
    28
>> format short e; err = abs(0.5*(1-sqrt(5))-x)
err =
   1.1592e-11
```

Clearly, `forward` finds x_+ and `backward` finds x_-. In both cases the iterations `k` required to achieve the `1e-10` tolerance is about 30. Note that `1.1592e-11` $<$ $\frac{1}{2}$ `tol = 5.0e-11`. So, despite the termination condition being between *successive iterates*, the tolerance is achieved relative to the *actual root*. To test whether these results depend on `x0`, we try the following (now `x0 = -10`).

```
>> format compact; a=1; b=-1; c=-1; x0=-10; tol=1e-10; % These will not change between uses of forward/backward.
>> format long e; [x k] = forward(a,b,c,x0,tol)
x =
     1.618033988755378e+00
k =
    28
>> format short e; err = abs(0.5*(1+sqrt(5))-x)
err =
   5.4829e-12
>> format long e; [x k] = backward(a,b,c,x0,tol)
x =
    -6.180339887405341e-01
k =
    27
>> format short e; err = abs(0.5*(1-sqrt(5))-x)
err =
   9.3608e-12
```

So the results are essentially the same. Indeed, experimentation with several choices for `x0` in the range $[-100, 100]$ always yields the same roots in about 30 or so iterations.

We now perform the following experiment. We use the returned root from `forward` as the `x0` for `backward`, in both cases with `tol = 1e-10`. Here is the result.

```
>> format compact; a=1; b=-1; c=-1; tol=1e-10; % These will not change between uses of forward/backward.
>> format long e; x0 = 1; [x k] = forward(a,b,c,x0,tol)
x =
     1.618033988738303e+00
k =
    26
>> x0 = x; % Overwrite x0 with approximate root found by forward.
>> [x k] = backward(a,b,c,x0,tol)
x =
     1.618033988780243e+00
k =
     1
>> format short e; abs(x-x0)
ans =
   4.1940e-11
>> err = abs(0.5*(1+sqrt(5))-x)
err =
   3.0348e-11
```

So after 1 iteration, the `backward` iteration returns `x`, so let us denote this `x` by x_1. Then, as is evident, $|x_0 - x_1| \simeq$ `4.1940e-11` $\leq$ `5.0-11`, and so the termination from `backward` is indeed what was specified. Moreover, we have also demonstrated above that $|x_+ - x_1| \simeq$ `3.0348e-11`. Note this is *larger* than the error $|x_+ - x_0| \simeq$ `1.1592e-11` associated with `x0` (the approximate root returned from `forward` obeyed this estimate, as seen on the first page). Now, performance of the same experiment with `tol = 1e-13` for the `backward` iteration yields (we print out `x0` just to confirm that it remains the root returned by the `forward` iteration).

```
>> tol = 1e-13; x0
x0 =
     1.618033988738303e+00
>> [x k] = backward(a,b,c,x0,tol)
x =
    -6.180339887499086e-01
k =
    61
```

We have converged to the x_- root!

The explanation for this phenomena is as follows. In fact, *both* x_- and x_+ are fixed-points for the `backward` iteration. Indeed, for x_+ one can check that

$$-\frac{c}{b+ax_+} = \frac{1}{-1+x_+} = \frac{1}{-1+\frac{1}{2}(1+\sqrt{5})} = \frac{1}{\frac{1}{2}(\sqrt{5}-1)} = \frac{2}{\sqrt{5}-1}\left(\frac{\sqrt{5}+1}{\sqrt{5}+1}\right) = \tfrac{1}{2}(1+\sqrt{5}) = x_+.$$

Here $g_{\text{backward}}(x) = 1/(-1+x)$, so the last equation is $g_{\text{backward}}(x_+) = x_+$. However,

$$g'_{\text{backward}}(x_+) = -\frac{1}{(-1+x_+)^2} = -\frac{1}{\left(\frac{1}{2}(\sqrt{5}-1)\right)^2} = -\frac{4}{(\sqrt{5}-1)^2} \simeq -2.6180.$$

Likewise, for the other root x_-, we find

$$x_- = g_{\text{backward}}(x_-), \qquad g'_{\text{backward}}(x_-) \simeq \text{-3.8197e-01}.$$

So far, we have just confirmed that the `backward` iteration is locally convergent for x_- and locally divergent for x_+, as expected. Moreover, from this analysis, we see that the errors associated with `backward` behave as

$$e_{k+1} \simeq 2.6180 e_k$$

near x_+. This is clearly not convergent behavior. Nevertheless, in the first case above, we called `backward` with an `x0` for which $e_0 = |x_+ - x_0| =$ `1.1592e-11`. Therefore, after one iteration, we expect that

$$e_1 = |x_+ - x_1| \simeq (2.6180)(1.1592\text{e-}11) \simeq 3.0348\text{e-}11.$$

This is precisely what we found above. The error has gotten worse with one iteration, but the new iterate x_1 still yields an exit from `backward` since $|x_1 - x_0| \leq$ `5.0e-11`. However, when `backward` is called with the tighter `tol = 1e-13`, then the iteration can continue until the region of attraction for the fixed point x_- is reached (this fixed point is convergent for `backward`).

2. First we have written the following function.

```
% Returns y = sinh(x) - cos(x), first function (fun1) for hw03
% function y = hw03_fun1(x)
  function y = hw03_fun1(x)
  y = sinh(x) - cos(x);
```

With this MATLAB® function for $f(x) = \sinh x - \cos x$, the required tasks set down in the problem are performed by the following script.

```
% Script: hw03Problem2.m
% Solves Problem 2 for hw03. Uses hw03_fun1, external function sinh(x)-cos(x).

% First used bisection starting with bracketing interval [a,b] = [0,1].
format long e; format compact;
a = 0; b = 1; tol = 1e-8; kmax = 1000;
[r_bisection k] = bisection(@hw03_fun1,a,b,tol,kmax);
disp(['with ',num2str(k),' chops bisection found    ',num2str(r_bisection,'%1.15e')])

% Next use fzero to find same root. First alter the default tolerance.
MyOptions = optimset('TolX',tol);
r_fzero   = fzero(@hw03_fun1,1,MyOptions);
disp(['fzero with TolX = ',num2str(tol),' found    ',num2str(r_fzero,'%1.15e')])
```

```
%
r_fzero_long   = fzero(@hw03_fun1,1);
disp(['fzero with default TolX found    ',num2str(r_fzero_long,'%1.15e')])

%
disp([' '])
disp(['error |r_bisection - r_fzero(1e-8tol)|    ',num2str(abs(r_bisection - r_fzero),'%1.4e')])
disp(['error |r_bisection - r_fzero(Default)|    ',num2str(abs(r_bisection - r_fzero_long),'%1.4e')])
```

Here is the output from the script.

```
>> hw03Problem2
with 26 chops bisection found    7.032906562089920e-01
fzero with TolX = 1e-08 found    7.032906590051011e-01
fzero with default TolX found    7.032906588639654e-01

error |r_bisection - r_fzero(1e-8tol)|    2.7961e-09
error |r_bisection - r_fzero(Default)|    2.6550e-09
```

3. To get some idea where the roots lie, we have first made the plot shown in Fig. 1. Using the

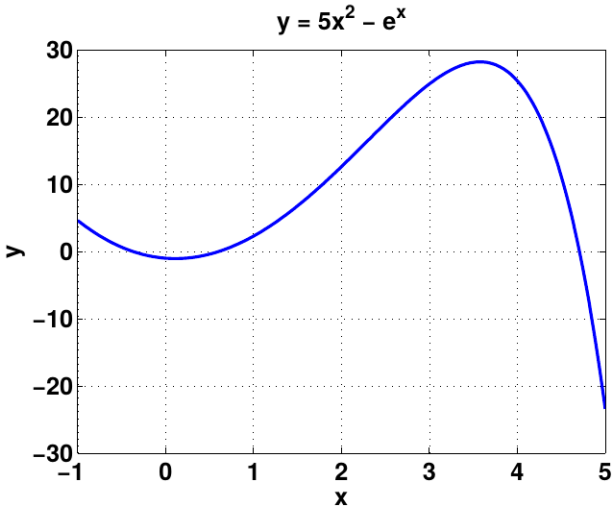


Figure 1: Plot for Problem 3.

figure, we have written the following script.

```
% Script: hw03Problem3.m
% Solves Problem 3 for hw03. Uses hw03_fun2, external function 5x^2 - e^x.

% This will not change, so fix upfront.
tol = 1e-8; kmax = 1000;

% Compute the roots using bisection.
% From plot, [-1,0] brackets leftmost root.
a1 = -1; b1 = 0; [r1_bisect k1] = bisection(@hw03_fun2,a1,b1,tol,kmax);
% From plot, [0,1] brackets middle root.
a2 = 0; b2 = 1; [r2_bisect k2] = bisection(@hw03_fun2,a2,b2,tol,kmax);
% From plot [4,5] brackets rightmost root.
a3 = 4; b3 = 5; [r3_bisect k3] = bisection(@hw03_fun2,a3,b3,tol,kmax);
```

```
clc
fprintf('Roots found with bisection algorithm\n')
fprintf('-------------------------------------------------------------------------------------------\n')
fprintf('with %2i chops and bracket interval [a,b]=[%2i,%2i] bisection found %+1.15e\n',k1,a1,b1,r1_bisect)
fprintf('with %2i chops and bracket interval [a,b]=[%2i,%2i] bisection found %+1.15e\n',k2,a2,b2,r2_bisect)
fprintf('with %2i chops and bracket interval [a,b]=[%2i,%2i] bisection found %+1.15e\n',k3,a3,b3,r3_bisect)

% Next use fzero to find same roots. First alter the default tolerance.
MyOptions = optimset('TolX',tol);
start1 = -1; r1_fzero   = fzero(@hw03_fun2,start1,MyOptions);
start2 =  1; r2_fzero   = fzero(@hw03_fun2,start2,MyOptions);
start3 =  5; r3_fzero   = fzero(@hw03_fun2,start3,MyOptions);
fprintf('\n')
fprintf('Roots found with fzero algorithm\n')
fprintf('-------------------------------------------------------------------------------------------\n')
fprintf('fzero with TolX = %1.1e and start %+2i found                    %+1.15e\n',tol,start1,r1_fzero)
fprintf('fzero with TolX = %1.1e and start %+2i found                    %+1.15e\n',tol,start2,r2_fzero)
fprintf('fzero with TolX = %1.1e and start %+2i found                    %+1.15e\n',tol,start3,r3_fzero)

% Display the errors between roots found by both methods.
err1 = abs(r1_bisect - r1_fzero);
err2 = abs(r2_bisect - r2_fzero);
err3 = abs(r3_bisect - r3_fzero);
fprintf('\n')
fprintf('Errors between two methods\n')
fprintf('-----------------------------------------------\n')
fprintf('|r1_bisect - r1_fzero(1e-8tol)|: %1.4e\n',err1)
fprintf('|r2_bisect - r2_fzero(1e-8tol)|: %1.4e\n',err2)
fprintf('|r3_bisect - r3_fzero(1e-8tol)|: %1.4e\n',err3)
```

Here is the output from the script.

```
Roots found with bisection algorithm
-------------------------------------------------------------------------------------------
with 26 chops and bracket interval [a,b]=[-1, 0] bisection found -3.714177533984184e-01
with 26 chops and bracket interval [a,b]=[ 0, 1] bisection found +6.052671223878860e-01
with 26 chops and bracket interval [a,b]=[ 4, 5] bisection found +4.707937918603420e+00

Roots found with fzero algorithm
-------------------------------------------------------------------------------------------
fzero with TolX = 1.0e-08 and start -1 found                    -3.714177524344662e-01
fzero with TolX = 1.0e-08 and start +1 found                    +6.052671213146193e-01
fzero with TolX = 1.0e-08 and start +5 found                    +4.707937925444780e+00

Errors between two methods
-----------------------------------------------
|r1_bisect - r1_fzero(1e-8tol)|: 9.6395e-10
|r2_bisect - r2_fzero(1e-8tol)|: 1.0733e-09
|r3_bisect - r3_fzero(1e-8tol)|: 6.8414e-09
```

4. The answer is $g'(1) = 1$ which corresponds to the arrangement shown in Fig. 2. Let us verify this result theoretically, but first argue graphically that it is so. We are told that $x = 1$ is a fixed point, and so the graphs $y = x$ and $y = g(x)$ *must* touch for $x = 1$, and there are two ways this can happen: (i) a tangent touch or (ii) a crossing touch. Fig. 2 depicts the type (i) touch for which $y = x$ is tangent to the graph $y = g(x)$ at $x = 1$, so indeed $g'(1) = 1$. Moreover, the figure clearly also shows that the other conditions [that is $g'(-3) = 2.4$ and $|g'(2)| < 1$] are achievable when $g'(1) = 1$. However, as indicated in the plots shown in Fig. 3, for (ii) the crossing touch at $x = 1$ always leads to $g'(2) \geq 1$, as seen in the two depicted possibilities. In Fig. 3b we have $y = x$ tangent to $y = g(x)$ at $x = 2$. Note that in all three plots $g'(-3) = 2.4$ (or at least $g'(-3) > 1$ obviously).

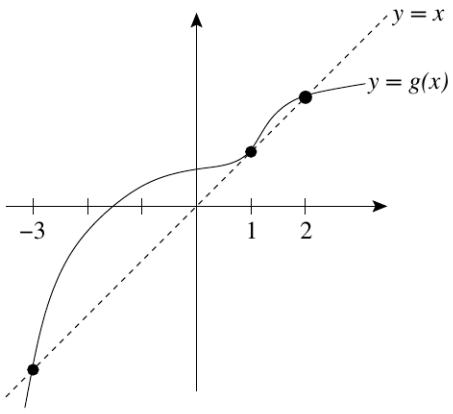
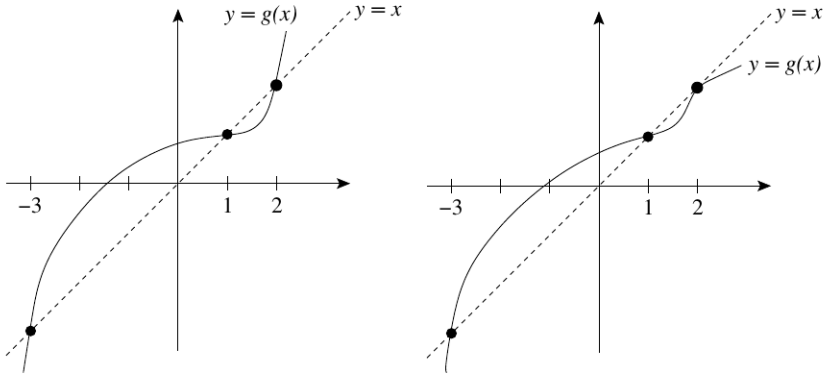


Figure 2: Sketch for Problem 4. Here $|g'(2)| < 1$ is possible.



(a) Here $g'(2) > 1$.

(b) Here $g'(2) = 1$.

Figure 3: Sketch for Problem 4.

To obtain the result theoretically, we write

$$g(x) = w(x)(x+3)(x-1)(x-2) + x,$$

which manifestly yields fixed points at $x = -3, 1, 2$. Since $g(x)$ is continuously differentiable, $w(x)$ is at least continuous. Now

$$g'(-3) = 2.4 \implies w(-3) = 7/100 > 0.$$

Moreover, since $|g'(2)| < 1$ (we are told the fix-point iteration is locally convergent at $x = 2$), we have

$$-1 < g'(2) < 1 \implies -2 < 5w(2) < 0 \implies w(2) < 0.$$

Since $w(x)$ is continuous, and $[-3, 2]$ is a bracketing interval, by the *Intermediate Value Theorem* there must be at least one root r on $(-3, 2)$. However, if $w(r) = 0$, then r is a fixed point of $x = g(x)$; whence, since we are told that there are exactly three fixed points (namely, $-3, 1, 2$), it must be the case that $r = 1$. That is $w(1) = 0$. Clearly then, $g'(1) = -4w(1) + 1 = 1$, as claimed.