

NUMERICAL METHODS

(1)

SOLUTIONS TO ASSIGNMENT #2

QUESTION 1(A) $x - \exp(-x) = 0$

a	c	b	
0.5 ⁻	0.55 ⁻	0.6 ⁺	} superscripts indicate the sign of the function value at the given value of x
0.55 ⁻	0.575 ⁺	0.6 ⁺	
0.55 ⁻	0.5625 ⁻	0.575 ⁺	
0.5625 ⁻	0.56875 ⁻	0.575 ⁺	

The approximate solution after 4 iterations is 0.56875

QUESTION 1B

see the M-file bisect2.m for complete function

In matlab the number of steps is

$$nSteps = \text{floor}((\log(b-a) - \log(2.0 * \text{tolerance})) / (\log(2.0))) + 1;$$

and the while loop is replaced by the for loop

```

[ nSteps = .....
  for n = 1 : nSteps
    :
  end

```

QUESTION 1(c)

See txt file output 1. txt [next page]

NOTE: If you didn't do question 1(B) then use bisect (instead of bisectr) and use a suitable maximum number of iterations

=====
SOLUTIONS TO QUESTION 1(c), ASSIGNMENT 2
=====

```
>> format long
>> f = @(x) x^3 + 3*x - 1;
>> r = bisection(f,0,1,0.5E-6, true)
```

n	a	c	b
1	0.0000000000e+000	5.0000000000e-001	1.0000000000e+000
2	0.0000000000e+000	2.5000000000e-001	5.0000000000e-001
3	2.5000000000e-001	3.7500000000e-001	5.0000000000e-001
4	2.5000000000e-001	3.1250000000e-001	3.7500000000e-001
5	3.1250000000e-001	3.4375000000e-001	3.7500000000e-001
6	3.1250000000e-001	3.2812500000e-001	3.4375000000e-001
7	3.1250000000e-001	3.2031250000e-001	3.2812500000e-001
8	3.2031250000e-001	3.2421875000e-001	3.2812500000e-001
9	3.2031250000e-001	3.2226562500e-001	3.2421875000e-001
10	3.2031250000e-001	3.2128906250e-001	3.2226562500e-001
11	3.2128906250e-001	3.2177734375e-001	3.2226562500e-001
12	3.2177734375e-001	3.2202148438e-001	3.2226562500e-001
13	3.2202148438e-001	3.2214355469e-001	3.2226562500e-001
14	3.2214355469e-001	3.2220458984e-001	3.2226562500e-001
15	3.2214355469e-001	3.2217407227e-001	3.2220458984e-001
16	3.2217407227e-001	3.2218933105e-001	3.2220458984e-001
17	3.2217407227e-001	3.2218170166e-001	3.2218933105e-001
18	3.2218170166e-001	3.2218551636e-001	3.2218933105e-001
19	3.2218170166e-001	3.2218360901e-001	3.2218551636e-001
20	3.2218360901e-001	3.2218456268e-001	3.2218551636e-001

r =
0.32218456268311

```
>> g = @(x) x^3 - 2*sin(x);
>> r = bisection(g,0.5,2,0.5E-5, true)
```

n	a	c	b
1	5.0000000000e-001	1.2500000000e+000	2.0000000000e+000
2	5.0000000000e-001	8.7500000000e-001	1.2500000000e+000
3	8.7500000000e-001	1.0625000000e+000	1.2500000000e+000
4	1.0625000000e+000	1.1562500000e+000	1.2500000000e+000
5	1.1562500000e+000	1.2031250000e+000	1.2500000000e+000
6	1.2031250000e+000	1.2265625000e+000	1.2500000000e+000
7	1.2265625000e+000	1.2382812500e+000	1.2500000000e+000
8	1.2265625000e+000	1.2324218750e+000	1.2382812500e+000
9	1.2324218750e+000	1.2353515625e+000	1.2382812500e+000
10	1.2353515625e+000	1.2368164063e+000	1.2382812500e+000
11	1.2353515625e+000	1.2360839844e+000	1.2368164063e+000
12	1.2360839844e+000	1.2364501953e+000	1.2368164063e+000
13	1.2360839844e+000	1.2362670898e+000	1.2364501953e+000
14	1.2360839844e+000	1.2361755371e+000	1.2362670898e+000
15	1.2361755371e+000	1.2362213135e+000	1.2362670898e+000
16	1.2361755371e+000	1.2361984253e+000	1.2362213135e+000
17	1.2361755371e+000	1.2361869812e+000	1.2361984253e+000
18	1.2361755371e+000	1.2361812592e+000	1.2361869812e+000

r =

1.23618125915527

```
>> h = @(x) x + 10 - x*cosh(50/x);
>> r = bisect2(h,120,130,0.5E-4, true)
```

n	a	c	b
1	1.2000000000e+002	1.2500000000e+002	1.3000000000e+002
2	1.2500000000e+002	1.2750000000e+002	1.3000000000e+002
3	1.2500000000e+002	1.2625000000e+002	1.2750000000e+002
4	1.2625000000e+002	1.2687500000e+002	1.2750000000e+002
5	1.2625000000e+002	1.2656250000e+002	1.2687500000e+002
6	1.2656250000e+002	1.2671875000e+002	1.2687500000e+002
7	1.2656250000e+002	1.2664062500e+002	1.2671875000e+002
8	1.2656250000e+002	1.2660156250e+002	1.2664062500e+002
9	1.2660156250e+002	1.2662109375e+002	1.2664062500e+002
10	1.2662109375e+002	1.2663085938e+002	1.2664062500e+002
11	1.2663085938e+002	1.2663574219e+002	1.2664062500e+002
12	1.2663085938e+002	1.2663330078e+002	1.2663574219e+002
13	1.2663085938e+002	1.2663208008e+002	1.2663330078e+002
14	1.2663208008e+002	1.2663269043e+002	1.2663330078e+002
15	1.2663208008e+002	1.2663238525e+002	1.2663269043e+002
16	1.2663238525e+002	1.2663253784e+002	1.2663269043e+002
17	1.2663238525e+002	1.2663246155e+002	1.2663253784e+002

```
r =
1.266324615478516e+002
```

```
>>
```

QUESTION 2(A)

$$x_0 = 0.5, f(x) = e^x - 2 \cos x = 0$$

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

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

The hand calculations are

$$x_0 = 0.5$$

$$x_1 = x_0 - \frac{e^{x_0} - 2 \cos x_0}{e^{x_0} + 2 \sin x_0} = 0.540821055$$

$$x_2 = x_1 - \frac{e^{x_1} - 2 \cos x_1}{e^{x_1} + 2 \sin x_1} = 0.539785831$$

$$x_3 = x_2 - \frac{e^{x_2} - 2 \cos x_2}{e^{x_2} + 2 \sin x_2} = 0.539785161$$

The result accurate to 5 significant figures is 0.53978

QUESTION 2(B)

output2.txt

6



=====
SOLUTIONS TO QUESTION 2(b), ASSIGNMENT 2
=====

```
>> format long
>> f = @(x) exp(x) - 2*cos(x);
>> fp = @(x) exp(x) + 2 * sin(x);
>> newton( f, fp, 0.5, 0.5E-5, 10, true)
   n          x          err
   1  5.408210545590396e-001  4.082105455903962e-002
   2  5.397858310677990e-001  1.035223491240657e-003
   3  5.397851608095621e-001  6.702582369493129e-007
ans =
   0.53978516080956
>>
```

This is question 2(a) using our Matlab newton function. See output2.txt

The value correct to 5 sig figs is 0.53978

Note: In the newton function we used $0.5E-5$ to get 5 significant figures

QUESTION 2(c)

7

3.2.1 (page 101): $f(x) = x^2 - R$, $f'(x) = 2x$

Therefore

$$\begin{aligned}x_{n+1} &= x_n - \frac{x_n^2 - R}{2x_n} = \frac{2x_n^2 - x_n^2 + R}{2x_n} \\ &= \frac{x_n^2 + R}{2x_n} = \frac{1}{2} \left(x_n + \frac{R}{x_n} \right)\end{aligned}$$

3.2.2 (page 101): From problem 3.2.1

$$\begin{aligned}x_{n+1}^2 &= \left(x_n - \frac{x_n^2 - R}{2x_n} \right)^2 = x_n^2 - 2x_n \left(\frac{x_n^2 - R}{2x_n} \right) \\ &\quad + \left(\frac{x_n^2 - R}{2x_n} \right)^2\end{aligned}$$

$$x_{n+1}^2 = R + \left(\frac{x_n^2 - R}{2x_n} \right)^2 \quad (*)$$

Define a measure of the error by $E_n = x_n^2 - R$.

$$\text{Then } E_{n+1} = x_{n+1}^2 - R = \left(\frac{x_n^2 - R}{2x_n} \right)^2 \quad \text{From } (*)$$

$$\text{Therefore } E_{n+1} = \left(\frac{E_n}{2x_n} \right)^2 = \frac{1}{4x_n^2} E_n^2 \approx \frac{1}{4R} E_n^2$$

Thus the error E_{n+1} is approximately a constant multiple of the square of the previous error so we obtain quadratic convergence.

QUESTION 2(D)

Problem 3.2.16: $f(x) = e^{-x} - \cos x$,

$f'(x) = -e^{-x} + \sin x$. The iteration formula is

$$x_{n+1} = x_n + \frac{e^{-x_n} - \cos x_n}{e^{-x_n} - \sin x_n} = x_n + \frac{1 - e^{x_n} \cos x_n}{1 - e^{x_n} \sin x_n}$$

When using a calculator we have

$$x \leftarrow x + ((1 - e^x \cos x) / (1 - e^x \sin x))$$

The results to get 3 significant figures are

$$x_0 = \pi/2$$

$$x_1 = 1.308362\dots$$

$$x_2 = 1.292790\dots$$

$$x_3 = 1.292695\dots$$

Therefore the root to 4 significant figures is

$$1.29$$

QUESTION 2(E)

Computer problem 3.2.5 page 106 using
Newton's method :

$$f = @(x) x^3 - 2*x - 5;$$

$$fp = @(x) 3*x*x - 2;$$

newton(f, fp, 2, 0.5E-5, 10, true)

see output3.txt.

=====
SOLUTIONS TO QUESTION 2(e), ASSIGNMENT 2
=====

```
>> format long
>> f = @(x) x^3 - 2*x - 5;
>> fp = @(x) 3*x*x - 2;
>> newton(f,fp,2, 0.5E-5, 10, true)
    n          x          err
    1  2.1000000000000000e+000  1.0000000000000000e-001
    2  2.094568121104185e+000  5.431878895814855e-003
    3  2.094551481698199e+000  1.663940598573203e-005
    4  2.094551481542327e+000  1.558725019392802e-010
ans =
    2.09455148154233
>>
```

output3.txt

QUESTION 3(A)

Computer Problem 3.1.11 page 88

see next 3 pages

QUESTION 3(A) script

11

1/8/10 9:53 AM C:\web\badams\mc3416\assignments\solutions2\question3.m

1 of 1

```
% question3.m
% Find first three positive roots of
% the function  $f(x) = \tanh(x) + \tan(x)$ 

% First define data for the plots

x = linspace(0,12,100); % 100 points between 0 and 12
y1 = tanh(x);
y2 = -tan(x);

% plot intersections in window with
%  $0 \leq x \leq 12$ , and  $-2 \leq y \leq 2$ 

plot(x,y1,'-', x,y2,'--');

axis([0,12,-2,2]); % note the square brackets

% Estimate intervals on which the following
% function  $f$  changes sign and find first 3 roots using fzero

f = @(x) tanh(x) + tan(x);
format long

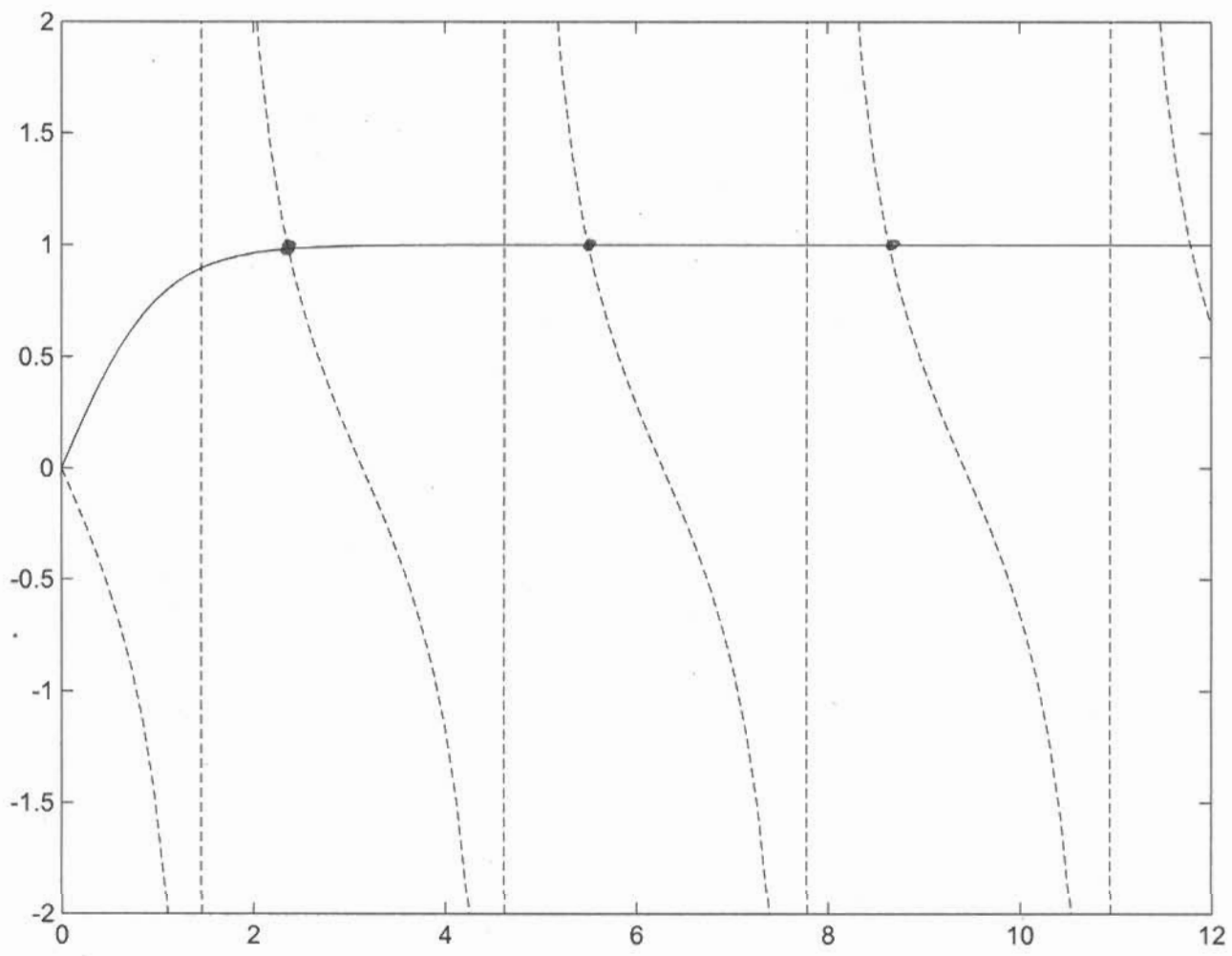
r1 = fzero(f, [2,3])
r2 = fzero(f, [5,6])
r3 = fzero(f, [8,10])

%r1 = fzero(f, 3)
%r2 = fzero(f, 6)
%r3 = fzero(f, 9)

sr1 = secant(f, 2, 3, 0.5E-5, 10, true)
sr2 = secant(f, 5, 6, 0.5E-5, 10, true)
sr3 = secant(f, 8, 10, 0.5E-5, 10, true)
```

$$y_1 = \tanh(x)$$

$$y_2 = -\tan(x)$$



Script output

=====
OUTPUT FOR QUESTION 3(a)
=====

>> question3

r1 =
2.36502037243135
r2 =
5.49780391900084
r3 =
8.63937982869974

n	x	err
1	2.588859520239958e+000	4.111404797600421e-001
2	2.270699983722995e+000	3.181595365169632e-001
3	2.385015383006802e+000	1.143153992838069e-001
4	2.366919606536348e+000	1.809577647045365e-002
5	2.364983329702061e+000	1.936276834287391e-003
6	2.365020441510995e+000	3.711180893381017e-005
7	2.365020372433866e+000	6.907712860004781e-008

sr1 =
2.36502037243387

n	x	err
1	5.770525495375958e+000	2.294745046240416e-001
2	5.401548476396417e+000	3.689770189795405e-001
3	5.522720117410098e+000	1.211716410136806e-001
4	5.500261188793355e+000	2.245892861674298e-002
5	5.497743255111537e+000	2.517933681818266e-003
6	5.497804067944307e+000	6.081283276982779e-005
7	5.497803919009871e+000	1.489344357217518e-007

sr2 =
5.49780391900987

n	x	err
1	9.557372508784262e+000	4.426274912157387e-001
2	8.583237122831584e+000	9.741353859526777e-001
3	8.675867061224519e+000	9.262993839293540e-002
4	8.641443155036917e+000	3.442390618760099e-002
5	8.639305513014921e+000	2.137642021995291e-003
6	8.639379981935585e+000	7.446892066387010e-005
7	8.639379828711128e+000	1.532244563504611e-007

sr3 =
8.63937982871113

>>

QUESTION 3 (B)

Problem 3.3.10 on page 120 of text book.

The two formulas for the secant method are

$$x_{n+1} = x_n - \left(\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \right) f(x_n)$$

$$x_{n+1} = \frac{x_{n-1} f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}$$

The second formula is obtained by taking a common denominator in the first formula and simplifying

An important rule for iterative numerical methods is that the quantities you are calculating at each iteration should be expressed as an approximate value +/- a small correction.

However, the second formula is not of this form and there could be a loss of significance in the subtraction