HW-3

1. Use the Matlab program in sakai that implements the fixed point iteration method $x_{(k+1)}=g(x_k)$ k=1,2,… with a stopping criterion, the $|x_k-x_{(k-1)}|<tol$, and a max number of iterations kmax. Then solve the equation $f(x)=x^2-3x+2$ by using the following iteration functions.
   a. Newton's method for g(x)=x-f(x)/f'(x)
   b. $g(x)=(x^2+2)/3$
   c. $g(x)=\sqrt{3x-2}$
   d. g(x)=3-2/x
   e. $g(x)=(x^2-2)/(2x-3)$

   Select the initial points by using ezplot to first plot f(x) and identify the regions for the root.

   Do these methods converge, if YES or NOT provide an explanation. If they converge identify the interval of convergence, i.e. any initial starting point in that region will converge to a root.

   What are the rates of convergence for each method? If they converge?

   SOLUTION: solve('x^2-3*x+2')

   ans =

   1

   2

   So we have two roots 1 and 2.

   a. Newton's method: [root result] = iteration(3,10^-8,100,1)
      result =

0	3.000000000000000	-0.666666666666667
1.000000000000000	2.333333333333334	-0.266666666666667
2.000000000000000	2.066666666666666	-0.062745098039215
3.000000000000000	2.003921568627451	-0.003906309605554
4.000000000000000	2.000015259021897	-0.000015258789066
5.000000000000000	2.000000000232831	-0.000000000232831
6.000000000000000	2.000000000000000	-0.000000000232831

Prove convergence: syms x

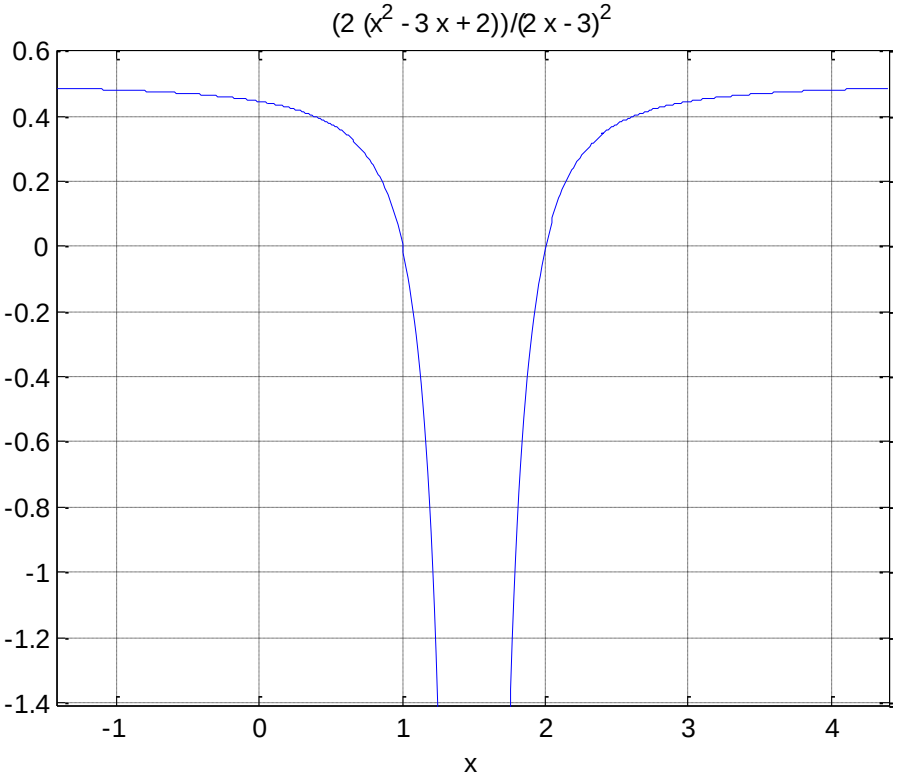
```
>> diff(x-(x^2-3*x+2)/(2*x-3))

ans =

(2*(x^2 - 3*x + 2))/(2*x - 3)^2

>> ezplot(ans)
>> grid
```

Even though we can see that Newton's method iteration the derivative in absolute value becomes greater than 1 in some region between [1,2] the method converges for all x0 except for x0=1.5. This is because x1 for any other value jumps into the convergence region see the example below:



result =

0	1.440000000000000	-2.053333333333335
1.000000000000000	-0.613333333333335	0.997518401682441
2.000000000000000	0.384185068349105	0.445881719929524

3.000000000000000	0.830066788278630	0.148380842066684
4.000000000000000	0.978447630345314	0.021107059974625
5.000000000000000	0.999554690319939	0.000445111555803
6.000000000000000	0.999999801875742	0.000000198124218
7.000000000000000	0.999999999999961	0.000000000000040
8.000000000000000	1.000000000000000	0.000000000000040

b. $g'(x) = \frac{2x}{3} ==> |g'(x| < 1 ==> \quad -1 < \frac{2x}{3} < 1 ==> -\frac{3}{2} < x < \frac{3}{2}$

The only root in that interval is 1 so it will converge to that root if x is selected in that interval. It actually converges for -2<x<2 and diverges for everything else. To prove that you need to show that every point is monotonically decreases if positive or if negative it jumps to the convergence interval(run few examples to see it).

result =

0	1.400000000000000	-0.080000000000000
1.000000000000000	1.320000000000000	-0.072533333333333
2.000000000000000	1.247466666666667	-0.062075638518519
3.000000000000000	1.185391028148148	-0.050340398276774
4.000000000000000	1.135050629871375	-0.038937319080907
5.000000000000000	1.096113310790468	-0.028958514093121
6.000000000000000	1.067154796697347	-0.020881676659295
7.000000000000000	1.046273120038052	-0.014710639466665
8.000000000000000	1.031562480571387	-0.010188763463856
9.000000000000000	1.021373717107531	-0.006972293774846
10.000000000000000	1.014401423332685	-0.004731340779559
11.000000000000000	1.009670082553126	-0.003192190685514
12.000000000000000	1.006477891867612	-0.002145309594854
13.000000000000000	1.004332582272757	-0.001437937001202
14.000000000000000	1.002894645271555	-0.000962088766769
15.000000000000000	1.001932556504786	-0.000642940576714
16.000000000000000	1.001289615928072	-0.000429317606277
17.000000000000000	1.000860298321795	-0.000286519402864

18.000000000000000	1.000573778918931	-0.000191149898894
19.000000000000000	1.000382629020037	-0.000127494205023
20.000000000000000	1.000255134815014	-0.000085023240413
21.000000000000000	1.000170111574600	-0.000056694212218
22.000000000000000	1.000113417362383	-0.000037801499628
23.000000000000000	1.000075615862754	-0.000025203381665
24.000000000000000	1.000050412481089	-0.000016803313224
25.000000000000000	1.000033609167865	-0.000011202679430
26.000000000000000	1.000022406488436	-0.000007468662128
27.000000000000000	1.000014937826307	-0.000004979201056
28.000000000000000	1.000009958625251	-0.000003319508692
29.000000000000000	1.000006639116559	-0.000002213024160
30.000000000000000	1.000004426092398	-0.000001475357603
31.000000000000000	1.000002950734796	-0.000000983575363
32.000000000000000	1.000001967159433	-0.000000655718521
33.000000000000000	1.000001311440912	-0.000000437146397
34.000000000000000	1.000000874294515	-0.000000291431250
35.000000000000000	1.000000582863265	-0.000000194287642
36.000000000000000	1.000000388575623	-0.000000129525157
37.000000000000000	1.000000259050466	-0.000000086350133
38.000000000000000	1.000000172700333	-0.000000057566768
39.000000000000000	1.000000115133565	-0.000000038377851
40.000000000000000	1.000000076755715	-0.000000025585236
41.000000000000000	1.000000051170478	-0.000000017056825
42.000000000000000	1.000000034113653	-0.000000011371217
43.000000000000000	1.000000022742436	-0.000000007580812
44.000000000000000	1.000000015161624	-0.000000007580812

c. diff(sqrt(3*x-2))

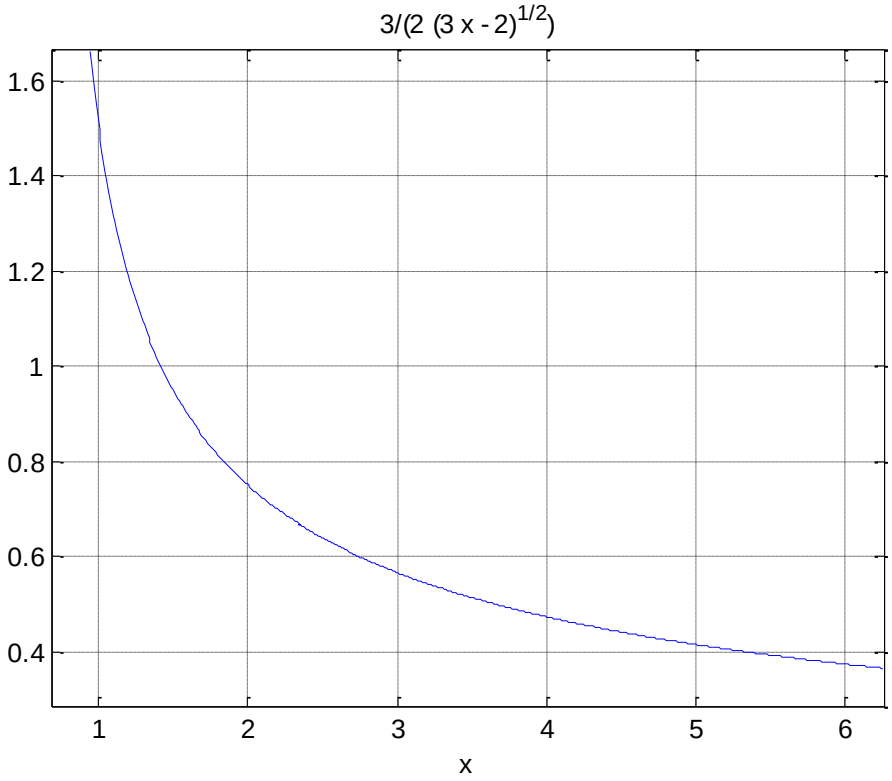
ans =

3/(2*(3*x - 2)^(1/2))

>> ezplot(ans)

So this method does not converge for the root 1 but converges for 2 for any x>1

>> grid



d. >> diff(3-2/x)
ans =2/x^2

It does not converge for x=1 but always converges to x=2. We know the for x>sqrt(2) the above derivative is less than one. Also for x<-sqrt(2). So it converges in that interval. In between you need to show that the iteration jumps to the interval of convergence. See the results below:

r7esult =

0	0.900000000000000	-0.122222222222222
1.000000000000000	0.777777777777778	-0.349206349206349
2.000000000000000	0.428571428571428	-2.095238095238097
3.000000000000000	-1.666666666666669	5.866666666666667
4.000000000000000	4.199999999999998	-1.676190476190475
5.000000000000000	2.523809523809524	-0.316262353998203
6.000000000000000	2.207547169811321	-0.113530075794227
7.000000000000000	2.094017094017094	-0.049119134833420
8.000000000000000	2.044897959183674	-0.022941871359322
9.000000000000000	2.021956087824351	-0.011097252681212

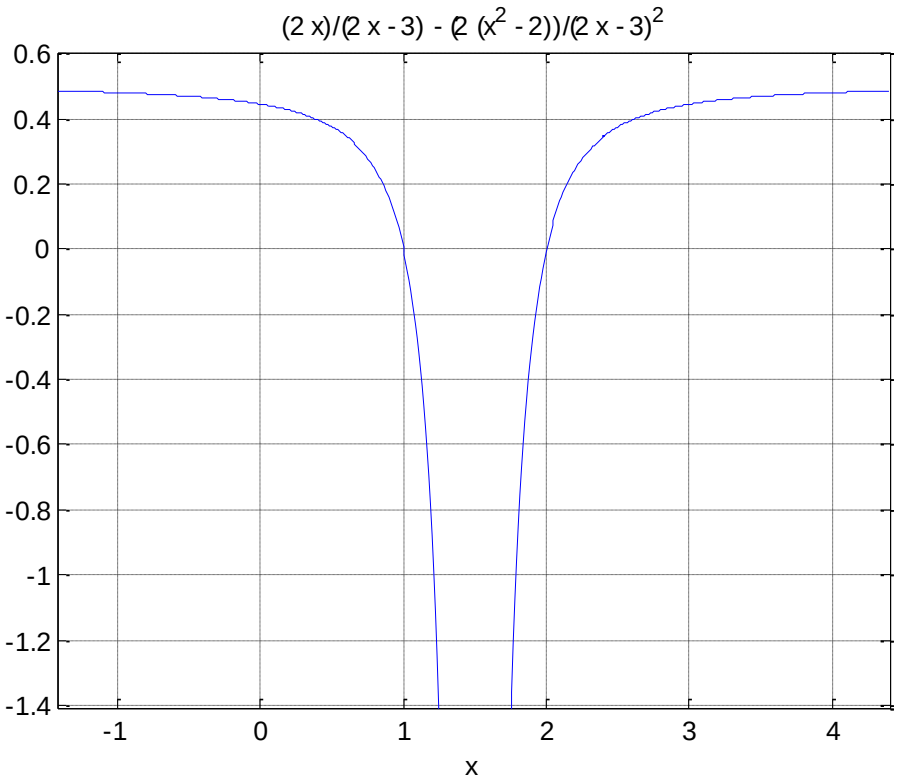
10.000000000000000	2.010858835143139	-0.005458736959536
11.000000000000000	2.005400098183603	-0.002707319725831
12.000000000000000	2.002692778457773	-0.001348199555438
13.000000000000000	2.001344578902335	-0.000672741120621
14.000000000000000	2.000671837781714	-0.000336031694465
15.000000000000000	2.000335806087249	-0.000167931230324
16.000000000000000	2.000167874856925	-0.000083944473363
17.000000000000000	2.000083930383562	-0.000041966952784
18.000000000000000	2.000041963430777	-0.000020982155612
19.000000000000000	2.000020981275165	-0.000010490747635
20.000000000000000	2.000010490527530	-0.000005245291278
21.000000000000000	2.000005245236252	-0.000002622625004
22.000000000000000	2.000002622611248	-0.000001311307344
23.000000000000000	2.000001311303905	-0.000000655652382
24.000000000000000	2.000000655651522	-0.000000327825869
25.000000000000000	2.000000327825654	-0.000000163912854
26.000000000000000	2.000000163912800	-0.000000081956407
27.000000000000000	2.000000081956393	-0.000000040978198
28.000000000000000	2.000000040978195	-0.000000020489098
29.000000000000000	2.000000020489097	-0.000000010244549
30.000000000000000	2.000000010244548	-0.000000005122274
31.000000000000000	2.000000005122274	-0.000000005122274

e. >> diff((x^2-2)/(2*x-3))

ans =

(2*x)/(2*x - 3) - (2*(x^2 - 2))/(2*x - 3)^2

>> ezplot(ans)
>> grid



This is a fast convergent method

r7esult =

0	0.900000000000000	0.091666666666667
1.000000000000000	0.991666666666667	0.008265027322404
2.000000000000000	0.999931693989071	0.000068301345855
3.000000000000000	0.999999995334926	0.000000004665074
4.000000000000000	1.000000000000000	0.000000004665074

Notice that the derivative is zero for x=1 and x=2. So the method is at least as fast as Newton's method and converges at least quadratic. You can show that it is exactly quadratic by showing that its second derivative is non zero.

```
function [root result] =
iteration(x0,error_bd,max_iterate,index_f)

result=[]
```

```
format long
error = 1;
it_count = 0;
while abs(error) > error_bd && it_count <= max_iterate
    gx = g(x0,index_f);


    x1 = gx;
    error = x1 - x0;
%   Internal print of newton method. Tap the carriage
%   return key to continue the computation.
    iteration = [it_count x0 error];
    result=[result
        iteration];
    x0 = x1;
    it_count = it_count + 1;
end

if it_count > max_iterate
    disp('The number of iterates calculated exceeded')
    disp('max_iterate.  An accurate root was not')
    disp('calculated.')
else
root=x0;
result=[result;it_count x0 error];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function value = g(x,index)

% function to define equation for rootfinding problem.

switch index
case 1
    value = x-(x^2-3*x+2)/(2*x-3);
case 2
    value =(x^2+2)/3 ;
case 3
    value=sqrt(3*x-2);
case 4
   value=3-2/x;
case 5
    value=(x^2-2)/(2*x-3);
end
```

2. We would like to solve $e^{-x} = x$, determine the rate of convergence for Newton's method for points that are farther to the root and points near the root. Will newton's method converge for any starting point? Can you propose a method that is faster than Newton's method?

It converges quadratic near the root for sure but also for all positive values since the first iteration is close to the root as well. See http://en.wikipedia.org/wiki/Halley%27s_method for a cubic convergent method.

r7esult =

1.0e+02 *

0	1.000000000000000	-1.000000000000000
0.010000000000000	0	0.005000000000000
0.020000000000000	0.005000000000000	0.000663110031972
0.030000000000000	0.005663110031972	0.000008321618376
0.040000000000000	0.005671431650349	0.000000001253749
0.050000000000000	0.005671432904098	0.000000000000000
0.060000000000000	0.005671432904098	0.000000000000000

3. Show that $x_{n+1} = \frac{x_n(x_n^2+3a)}{3x_n^2+a}$ is a third order method for computing $\sqrt{a}$. Implement the above method by modifying Newton's program in sakai. Compare newton's method bisection method and the above third order method for the computation of $\sqrt{5}$ within $eps = 10^{-12}$. Write conclusions of your comparison.

```
>> syms x
>> syms a
>> diff(x*(x^2+3*a)/(3*x^2+a))

ans =

(2*x^2)/(3*x^2 + a) + (x^2 + 3*a)/(3*x^2 + a) - (6*x^2*(x^2 + 3*a))/(3*x^2 + a)^2
```

```
>> x=sqrt(a);
>> (2*x^2)/(3*x^2 + a) + (x^2 + 3*a)/(3*x^2 + a) - (6*x^2*(x^2 + 3*a))/(3*x^2 + a)^2

ans =

0

>>
```