Homework 6

Daniel Gollahon
Ryan Dejesus
Ron Quan

Contents

Review 5.2

True. We can view Newton's method as a systematic way of transforming a nonlinear equation f(x) = 0 into a fixed-point problem x = g(x) where g(x) = x - f(x)/f'(x). (p. 230)

Review 5.3

False. This would still be a linear convergence rate because the solution would be gaining a **constant** number of digits of accuracy per iteration. (p. 223)

Review 5.4

False. If the number of digits of accurate information we obtain per iteration in a linear scheme is relatively large and the number of digits of accuracy we're eventually trying to obtain is relatively small, we may compute the necessary digits faster through a linear method as compared to a superlinear method. I.e. if we want 4 digits of accuracy and we can gain 4 digits per iteration with a linear solution method, it is probably faster just to use one iteration of that function.

Review 5.5

For an ill-conditioned problem, `f(xk)` can be small without xk being close to the true solution, x*. We want the distance between `xk - x*` to be small so we would be better off choosing `xk - xk-1`.

Review 5.7

We must use an absolute rather than a relative condition number in assessing sensitivity, because the function value at the solution is zero by definition.

Review 5.8(a)

The convergence rate r of an iterative method tells us that we will have r times as many correct digits after each iteration as it had the previous iteration. (p. 223)

Review 5.9

```
(a) Quadratic
(b) Linear
(p. 223)
```

Review 5.13

A quadratically convergent method doubles the number of correct digits with each iteration. (p. 223)

Review 5.14

In this case, r would be sqrt(2). (We would multiply r times itself over the course of two iterations and since sqrt(2)*sqrt(2) = 2, we have our answer).

Review 5.15

```
(a) A solution is a multiple root if it solves f(x) = 0 as well as one or more of it's derivatives f'(x) = f''(x) = ... = 0. (p. 220)
(b) The convergence rate remains unchanged. This is because interval bisection does not take into account the derivative and has a constant rate of converge
(c) It causes the convergence rate to move from quadratic to linear. (p. 231)
```

Review 5.16

```
All (a), (b), and (c) may occur. (p. 231)
```

Review 5.17

(a) The root has a multiplicity of two. Since this is a multiple root, Newton's method will have a linear convergence rate of C = 1 - (1/2) = 1/2.

(b) In this case, there is not a multiple root so Newton's method will have quadratic convergence.

Review 5.20

```
(a) If x* = g(x*) and |g'(x*)| < 1, the iterative scheme is locally convergent. (p. 227)
(b) The asymptotic vonvergence rate is linear, with C = |g'(x*)| (p. 228[end])
(c) The rate is at least quadratic when g'(x*) = 0. (p. 229)
(d) Yes. g(x) = x - f(x)/f'(x) (p. 230)
```

Review 5.23

```
Advantage: The secant method is superlinearly convergent instead of only linearly convergent. (p. 233)
Disadvantage: It is not guaranteed that the secant method will converge if the initial guess is not good. (p. 233)  The bisection method will converge so lo
```

Review 5.24

```
Advantage: The secant method requires only one new function evaluation per iteration (can be computed faster). (p. 233)
Disadvantage: It requires two starting geusses and converges somewhat more slowly. (Though this is often offset by the cheaper computation cost) (p. 233)
```

Review 5.25

```
(a)
   1. Iterpolation with more than two points can give misleading results.
      A line may increase when it gives the notion that it may be decreading because of the lower-degree.
   2. The secant method relies on the two guesses provided; therefore it is based off these two guess. The method has a steady convergence rate because of r
   3. With more guesses, it creates more possibility there will be convergence to a new point.
(b)  Start with two initial starting guesses, but instead use k points for the degree k of the polynomial.
```

Review 5.27

```
1. Bisection (slowest)
2. Newton's method (faster) [or is it the fastest? consider p. 233 offset]
3. Secant (fastest) [verify, as above]
```

Review 5.28

```
(a) Bisection: 1 bit per iteration. (p. 226)
(b) Newton's method: The number of correct digits is doubled at each iteration. (p. 231)
```

Refer to images for the answers to the exercises

Computer Problem 5.3

Newton's method

```
function [root, convergence, convergence_rate] = Newton( f, df, x_guess, tol, maxIterations )

if nargin == 3
  tol = 1e-5;
  maxIterations = 1e1;
elseif nargin == 4
  maxIterations = 1e1;
end

%f = inline(f);
%df = inline(df);
root = x_guess;

root = [root;x_guess - (f(x_guess)/df(x_guess))];
convergence = abs(root(2,1) - x_guess );
x_guess = root(2,1);
rel_change = abs( root(2,1) - root(1,1) ) / root(1,1);
k = 3;
while ( rel_change >= tol ) && (k <= maxIterations)
  root = [root;x_guess - (f(x_guess)/df(x_guess))];
  convergence = [convergence; abs(root(k,1)-x_guess)];
  rel_change = abs( root(k,1) - root(k-1,1) );
  x_guess = root(k,1);
  k = k+1;
end

[m,n] = size(convergence);
sum_of_errors = sum(convergence);
convergence_rate = sum_of_errors/convergence(1,1);
root = root(k-1,1);

end %function%
```

Bisection method

```
function [root, convergence] = Bisection(a,b,f)
 tol = 10e-10;
 while((b - a) > tol)
   m = a + (b - a)/2;
   if sign(f(a)) == sign(f(m))
     a = m;
   else
     b = m;
   end
 end
 root = m;
 convergence = 2;
end
```

Secant method

```
function [root,hx] = Secant( f, x0, x1, tol, maxIterations )

  root = x0;
  root = [root; x1];
  root = [root; x1 - ( ( f(x1)*(x1 - x0) )/ ( f(x1)-f(x0) ) )];

  hx = [0; root(3,1) - root(2,1)];

  k = 3;

  while( abs(f(root(k,1))) >= tol )
      x1 = root(k,1);
      x0 = root(k-1,1);
      root = [root;x1 - ( ( f(x1)*( x1 - x0 ) ) / ( f(x1)-f(x0) ) )];
      hx= [hx; root(k+1) - x1];
      k = k+1;
  end%while

end %function
```

```
a = @(x)x^3-2*x-5;
da = @(x)3*x^2-2;
b = @(x)exp(-x)-x;
db = @(x)-1*exp(-x);
c = @(x)x*sin(x)-1;
dc = @(x)x*cos(x)+sin(x);
d = @(x)x^3-3*x^2+3*x-1;
dd = @(x)3*x^2-6*x+3;
tol = 1e-8;
maxIterations = 500;
```

Computer Problem 5.3(a)

(a)

```
disp('Bisection (a)');
[root,convergence] = Bisection(2,4,a)
disp('Newton (a)');
[root,convergence,convergence_rate] = Newton(a,da,2,tol,maxIterations)
disp('Secant (a)');
[root,hx] = Secant(a,2,4,tol,maxIterations)
disp('Library function (a)');
[x,fval,exitflag] = fsolve(a,2)
```

```
Bisection (a)

root =

    2.0946


convergence =

   -11

Newton (a)

root =

    2.0946


convergence =

    0.1000
    0.0054
    0.0000
    0.0000
```

```
convergence_rate =

    1.0545

Secant (a)

root =

    2.0000
    4.0000
    2.0385
    2.0615
    2.0956
    2.0945
    2.0946
    2.0946


hx =

         0
   -1.9615
    0.0231
    0.0341
   -0.0011
    0.0000
    0.0000

Library function (a)

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.




x =

    2.0946


fval =

   1.7441e-09


exitflag =

     1
```

Computer Problem 5.3(b)

(b)

```
disp('Bisection (b)');
[root,convergence] = Bisection(0.2,1,b)
disp('Newton (b)');
[root,convergence,convergence_rate] = Newton(b,db,0.53,tol,maxIterations)
disp('Secant (b)');
[root,hx] = Secant(b,0.87,1,tol,maxIterations)
disp('Library function (b)');
[x,fval,exitflag] = fsolve(b,2)
```

```
Bisection (b)

root =

    0.5671


convergence =

   -11

Newton (b)

root =

    0.7787


convergence =
```

```
0.0996
0.1816
0.2988
0.5759
0.7973
1.5493
1.3250
0.5653
0.7864
1.5332
1.3219
0.6016
0.8229
1.5849
1.3314
0.4846
0.6997
1.3898
1.2880
0.9240
1.0815
1.6772
1.3453
0.2784
0.4402
0.8711
1.0465
1.7056
1.3488
0.2162
0.3509
0.6839
0.8997
1.6709
1.3444
0.2923
0.4595
0.9115
1.0735
1.6853
1.3463
0.2607
0.4152
0.8187
1.0091
1.7178
1.3502
0.1898
0.3114
0.6019
0.8232
1.5853
1.3315
0.4837
0.6987
1.3880
1.2875
0.9279
1.0840
1.6745
1.3449
0.2845
0.4486
0.8888
1.0585
1.6979
1.3479
0.2332
0.3757
0.7358
0.9442
1.7027
1.3484
0.2227
0.3605
0.7039
0.9172
1.6852
1.3463
0.2609
0.4156
0.8194
1.0097
1.7178
1.3502
0.1900
0.3116
0.6023
0.8236
1.5858
```

```
1.3316
0.4825
0.6974
1.3857
1.2869
0.9331
1.0873
1.6707
1.3444
0.2928
0.4601
0.9128
1.0743
1.6845
1.3462
0.2624
0.4177
0.8238
1.0129
1.7174
1.3501
0.1909
0.3129
0.6050
0.8263
1.5894
1.3322
0.4745
0.6883
1.3692
1.2822
0.9692
1.1093
1.6408
1.3403
0.3590
0.5484
1.0961
1.1777
1.4863
1.3121
0.7081
0.9208
1.6879
1.3466
0.2551
0.4073
0.8020
0.9966
1.7182
1.3502
0.1890
0.3101
0.5992
0.8205
1.5818
1.3309
0.4917
0.7077
1.4039
1.2919
0.8929
1.0613
1.6958
1.3476
0.2376
0.3822
0.7493
0.9553
1.7081
1.3491
0.2109
0.3430
0.6675
0.8850
1.6572
1.3426
0.3226
0.5005
0.9973
1.1256
1.6130
1.3361
0.4212
0.6259
1.2514
1.2436
1.2147
1.2296
1.2853
1.2556
```

```
1.1466
1.2011
1.4063
1.2925
0.8875
1.0577
1.6985
1.3479
0.2318
0.3737
0.7316
0.9407
1.7007
1.3482
0.2270
0.3666
0.7168
0.9282
1.6931
1.3473
0.2437
0.3910
0.7678
0.9701
1.7136
1.3497
0.1989
0.3251
0.6302
0.8505
1.6197
1.3371
0.4062
0.6076
1.2155
1.2299
1.2837
1.2551
1.1498
1.2025
1.4011
1.2911
0.8992
1.0654
1.6925
1.3472
0.2449
0.3927
0.7714
0.9730
1.7145
1.3498
0.1971
0.3223
0.6246
0.8451
1.6133
1.3361
0.4207
0.6252
1.2502
1.2431
1.2171
1.2305
1.2808
1.2541
1.1557
1.2051
1.3910
1.2884
0.9215
1.0799
1.6789
1.3455
0.2747
0.4350
0.8602
1.0390
1.7095
1.3492
0.2079
0.3384
0.6580
0.8763
1.6485
1.3414
0.3419
0.5261
1.0504
1.1546
1.5502
```

```
1.3252
0.5632
0.7843
1.5301
1.3213
0.6088
0.8299
1.5941
1.3330
0.4637
0.6760
1.3466
1.2755
1.0180
1.1373
1.5900
1.3323
0.4731
0.6867
1.3664
1.2814
0.9753
1.1129
1.6351
1.3394
0.3718
0.5647
1.1294
1.1933
1.4347
1.2999
0.8243
1.0133
1.7173
1.3501
0.1910
0.3131
0.6054
0.8266
1.5898
1.3323
0.4734
0.6871
1.3670
1.2815
0.9740
1.1121
1.6364
1.3396
0.3689
0.5610
1.1220
1.1899
1.4465
1.3028
0.7978
0.9934
1.7180
1.3502
0.1894
0.3106
0.6004
0.8217
1.5833
1.3311
0.4882
0.7038
1.3971
1.2900
0.9080
1.0712
1.6874
1.3466
0.2560
0.4086
0.8048
0.9987
1.7183
1.3502
0.1889
0.3099
0.5989
0.8202
1.5813
1.3308
0.4927
0.7088
1.4060
1.2924
0.8882
1.0581
```

```
1.6982
1.3479
0.2325
0.3748
0.7339
0.9426
1.7018
1.3483
0.2247
0.3633
0.7098
0.9222
1.6889
1.3468
0.2528
0.4040
0.7952
0.9914
1.7179
1.3502
0.1897
0.3112
0.6015
0.8227
1.5847
1.3314
0.4850
0.7001
1.3906
1.2882
0.9224
1.0805
1.6783
1.3454
0.2760
0.4369
0.8641
1.0417
1.7082
1.3491
0.2107
0.3427
0.6668
0.8844
1.6566
1.3425
0.3240
0.5024
1.0011
1.1278
1.6089
1.3354
0.4305
0.6370
1.2729
1.2513
1.1718
1.2120
1.3633
1.2805
0.9820
1.1168
1.6287
1.3385
0.3862
0.5828
1.1661
1.2096
1.3731
1.2833
0.9606
1.1042
1.6486
1.3414
0.3418
0.5260
1.0500
1.1544
1.5506
1.3253
0.5622
0.7833
1.5285
1.3210
0.6124
0.8334
1.5987
1.3337
0.4534
0.6640
1.3244
```

```
    1.2686
    1.0653
    1.1623
    1.5303
    1.3213
    0.6083
    0.8295
    1.5936
    1.3329
    0.4650
    0.6774
    1.3493
    1.2763
    1.0122
    1.1340
    1.5967
    1.3334
    0.4580
    0.6693
    1.3343
    1.2717
    1.0444
    1.1515
    1.5579
    1.3266
    0.5456
    0.7660
    1.5018
    1.3155
    0.6728
    0.8898
    1.6619
    1.3432
    0.3123
    0.4867
    0.9686
    1.1090
    1.6414
    1.3404
    0.3577
    0.5466
    1.0926
    1.1759
    1.4915
    1.3132


convergence_rate =

   5.0047e+03

Secant (b)

root =

    0.8700
    1.0000
    0.5462
    0.5687
    0.5671
    0.5671


hx =

         0
   -0.4538
    0.0225
   -0.0015
   -0.0000

Library function (b)

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.




x =

    0.5671


fval =

   6.9273e-09
```

```
exitflag =

     1
```

Computer Problem 5.3(c)

(c)

```
disp('Bisection (c)');
[root,convergence] = Bisection(2,3,c)
disp('Newton (c)');
[root,convergence,convergence_rate] = Newton(c,dc,3.1,tol,maxIterations)
disp('Secant (c)');
[root,hx] = Secant(c,1.8,3,tol,maxIterations)
disp('Library function (c)');
[x,fval,exitflag] = fsolve(c,3)
```

```
Bisection (c)

root =

    2.7726


convergence =

   -11

Newton (c)

root =

    2.7726


convergence =

    0.2851
    0.0412
    0.0011
    0.0000
    0.0000


convergence_rate =

    1.1485

Secant (c)

root =

    1.8000
    3.0000
    2.4796
    2.7274
    2.7841
    2.7723
    2.7726
    2.7726


hx =

         0
   -0.5204
    0.2478
    0.0567
   -0.0119
    0.0003
    0.0000

Library function (c)

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.



x =

    2.7726
```

```
fval =

  -1.6924e-07


exitflag =

     1
```

Computer Problem 5.3(d)

```
disp('Bisection (d)');
[root,convergence] = Bisection(-6,7,d)
disp('Newton (d)');
[root,convergence,convergence_rate] = Newton(d,dd,1.03,tol,maxIterations)
disp('Secant (d)');
[root,hx] = Secant(d,2.3,1.03,tol,maxIterations)
disp('Library function (d)');
[x,fval,exitflag] = fsolve(d,223)
```

```
Bisection (d)

root =

    1.0000


convergence =

   -11

Newton (d)

root =

    1.0000


convergence =

    0.0100
    0.0067
    0.0044
    0.0030
    0.0020
    0.0013
    0.0009
    0.0006
    0.0004
    0.0003
    0.0002
    0.0001
    0.0001
    0.0001
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
         0


convergence_rate =

    3.0038

Secant (d)

root =

    2.3000
    1.0300
    1.0300
    1.0200
    1.0158
    1.0117
    1.0089
```

```
    1.0067
    1.0051
    1.0038
    1.0029
    1.0022
    1.0016


hx =

         0
   -0.0000
   -0.0100
   -0.0042
   -0.0041
   -0.0028
   -0.0022
   -0.0016
   -0.0012
   -0.0009
   -0.0007
   -0.0005

Library function (d)

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.




x =

    1.0358


fval =

   4.6012e-05


exitflag =

     1
```

Published with MATLAB® R2013b

5.10.) $x_1 = x_0 + s_0$

s_0 determined by $J_f(x_0)s_0 = -f(x_0)$

$$J_f(x_0)s_0 = \begin{bmatrix} 2x_1 & -2x_2 \\ 2x_2 & 2x_1 \end{bmatrix}\begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = -\begin{bmatrix} x_1^2 \\ 2x_1x_2 - 1 \end{bmatrix} - f(x_0)$$

$$x_0 = [0 \; 1]^T$$

$$\begin{bmatrix} 0 & -2 \\ 2 & 0 \end{bmatrix}\begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = -\begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$s_0 = [0.5 \; -0.5]^T$$

$$x_1 = x_0 + s_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

5.13.) $J(x) = \begin{bmatrix} 1 & 0 \\ x_2 & x_1 \end{bmatrix}$

newton's method will fail for starting points of the form $x = [0 \;\; \alpha]^T$ α is arbitrary since $J(x)$ is singular at those points.

5.4.)

$f(x) = x^{-1} - y$

$f'(x) = -x^{-2}$

$f(x) = 0$

$$x_{k+1} = x_k - \frac{(x_k^{-1} - y)}{(-x_k^{-2})} = x_k + x_k^2(x_k^{-1} - y) = 2x_k - x_k^2 y$$

no divisions

5.9.)

a.) $J_f(x_k)s_k = \begin{bmatrix} 2x_1 & 2x_2 \\ 2x_1 & -1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = -\begin{bmatrix} x_1^2 + x_2^2 - 1 \\ x_1^2 - x_2 \end{bmatrix} = -f(x_k)$

b.) $J_f(x_k)s_k = \begin{bmatrix} 2x_1 + x_2^3 & 3x_1x_2^2 \\ 6x_1x_2 & 3x_1^2 - 3x_2^2 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = -\begin{bmatrix} x_1^2 + x_1x_2^3 - 9 \\ 3x_1^2x_2 - x_2^3 - 4 \end{bmatrix} = -f(x_k)$

c.) $J_f(x_k)s_k = \begin{bmatrix} 1 - 2x_2 & 1 - 2x_1 \\ 2x_1 - 2 & 2x_2 + 2 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = -\begin{bmatrix} x_1 + x_2 - 2x_1x_2 \\ x_1^2 + x_2^2 - 2x_1 + 2x_2 + 1 \end{bmatrix} - f(x_k)$

d.) $J_f(x_k)s_k = \begin{bmatrix} 3x_1^2 & -2x_2 \\ 1 + 2x_1x_2 & x_1^2 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = -\begin{bmatrix} x_1^3 - x_2^2 \\ x_1 + x_1^2x_2 - 2 \end{bmatrix} = -f(x_k)$

e.) $\cancel{2} J_f(x_k)s_k = \begin{bmatrix} 2\cos(x_1) - 5 & -\sin(x_2) \\ -4\sin(x_1) & 2\cos(x_2) - 5 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = -\begin{bmatrix} 2\sin(x_1) + \cos(x_2) - 5x_1 \\ 4\cos(x_1) + 2\sin(x_2) - 5x_2 \end{bmatrix}$

~~Sto~~.

Exercises 5.1, 5.2, 5.3, 5.4, 5.9, 5.10, 5.13

5.1.)

A.) $f(x) = x^2 - 2 \quad x_0 = 1$

$f'(x) = 2x$

$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 1 + \frac{1}{2} = 3/2$

b.) $x_0 = 1,\ x_1 = 2$

Secant method -> $x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)}$

$x_2 = 2 - \frac{(2(2-1))}{(2-(-1))} = 4/3$

5.2.)

a.) $x_{k+1} = x_k - \frac{(x_k^3 - 2x_k - 5)}{3x_k^2 - 2}$

b.) $x_{k+1} = x_k + \frac{(e^{-x_k} - x_k)}{(e^{-x_k} + 1)}$

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

5.3.)

a.) $f(x) = x^2 - y$ $\quad f(x) = 0$ is $x_{k+1} = x_k - \frac{(x_k^2 - y)}{2x_k}$

$f'(x) = 2x$

b.) must satisfy $4 \cdot 2k = m$

4 bits -> $k = \lceil \log_2(24/4) \rceil = 3$ iterations

53 bits -> $k = \lceil \log_2(53/4) \rceil = 4$ iterations