The University of British Columbia

CPSC 406

Computational Optimization

Homework #3

Student name:

ALI SIAHKOOHI

Student ID:

84264167

Instructor:

Professor Ascher

Date Submitted:

October 18th, 2016

Question #1

Part (a)

Since matrix W is SPD, all of its eigenvalues are positive. Therefore if we construct the spectral eigenvalues of W we would get:

$$W = X\Lambda X^T \tag{1 – 1}$$

Since the eigenvalues are positive we can define the matrix below:

$$B = X\Lambda^{\frac{1}{2}}X^T \tag{1 – 2}$$

Note that matrix B is symmetric positive definite also. Now if we define the matrix $\hat{A}$ and vector $\hat{b}$ as below, we can write the weighted least squares problem as a regular least squares problem:

$$\hat{A} = BA, \quad \hat{b}=Bb, \quad W = B^TB = \left(X\Lambda^{\frac{1}{2}}X^T \right)\left(X\Lambda^{\frac{1}{2}}X^T \right) = X\Lambda X^T$$

$$f(x) = \frac{1}{2}(Ax-b)^T W(Ax-b) = \frac{1}{2}(Ax-b)^T B^TB(Ax-b),$$

$$\Rightarrow f(x) = \frac{1}{2}\left(B(Ax-b)\right)^T \left(B(Ax-b)\right)$$

$$\Rightarrow f(x) = \frac{1}{2}\left(\hat{A}x-\hat{b}\right)^T \left(\hat{A}x-\hat{b}\right). \tag{1 – 3}$$

In addition to spectral eigenvalues one can use Cholesky factorization to obtain $W = R^TR$ which R is an upper triangular matrix. The same equations for reformulating the weighted least squares go as well to Cholesky factorization.

Part (b)

Part (i)

For finding the normal equations we set the gradient of the objective function equal to zero:

$$f(x) = \frac{1}{2}(Ax-b)^T W(Ax-b) = \frac{1}{2}\left(x^TA^TWAx - x^TA^TWb - b^TWAx + b^TWb\right)$$

$$\Rightarrow \nabla f(x) = \frac{1}{2}\left(x^T\left(A^TWA + A^TWA\right) - b^TWA - b^TWA\right)$$

$$\Rightarrow \nabla f(x) = x^TA^TWA - b^TWA = 0$$

$$\Rightarrow A^TWAx - A^TWb = 0$$

$$\Rightarrow A^TWAx = A^TWb \Rightarrow x^* = \left(A^TWA\right)^{-1}A^TWb. \tag{1 – 4}$$

With the formulation above I solved the problem by inversing the left hand side matrix of the normal equations.

Part (ii)

In this section I used the reformulated problem using cholesky decomposition and solved the new least squares problem with $\hat{\mathrm{A}}$ and $\hat{\mathrm{b}}$ using the backslash operator in MATLAB.

There is not such difference between two results. And the reason is we have constructed our model (matrix A) using rand command which usually means that our matrix is not ill-conditioned. Since the matrix is well-conditioned, I would prefer to use the normal equations because it's computationally cheaper.

Advantage of method in (ii) comparing to (i) is its more robust, meaning that when the operator is ill-conditioned, in other words, the condition number is very large; using the normal equations could cost us some accuracy.

Elapsed time is 0.026653 seconds. Part (i)
Elapsed time is 0.035543 seconds. Part (ii)
2-norm of the solution using normal equations = 1.64804
2-norm of the solution using reformulated normal equation and MATLABs back slash = 1.64804

MATLAB code used in this question:

```
clear; clc

mx = 20;
m = mx^2;
n = 300;
A = randn(m, n);
b = randn(m, 1);
W = delsq(numgrid('S', mx + 2));
spy(W)

tic
xls = (A'*W*A)^-1*(A'*W*b);
toc

tic
R = chol(W);
A_new = R*A;
b_new = R*b;
xqr = A_new\b_new;
toc

fprintf('2-norm of the solution using normal equations = %g \n', norm(xls))
```

```
fprintf('2-norm of the solution using reformulated normal equation and
MATLABs back slash = %g \n', norm(xqr))
```

Part (c)

As mentioned earlier, if the problem (Matrix A) is well-conditioned and the dimension of matrix A is very big, normal equations are better that QR-based method. And keeping the matrix well-conditioned, as the dimension grows, normal equations gets better and better comparing to QR-based method. And the reason is computational cost.

I used a larger matrix, 2500x1000 we the same structure and got the following results. The norm of solutions is still the same.

Elapsed time is 1.272688 seconds.
Elapsed time is 2.333753 seconds.
2-norm of the solution using normal equations = 0.883124
2-norm of the solution using reformulated normal equation and MATLABs back slash = 0.883124

Question #3

Part (a)

The function f(**z**) is non-differentiable at any point where one of the elements of **r** is zero. Ignoring this fact for a moment, derivatives of f(**z**) at other point are are:

$$\frac{\partial f(z)}{\partial z_k} = \sum_{i=1}^{n} \frac{\partial |r_i|}{\partial z_k} = \sum_{i=1}^{n} J_{i,k} \operatorname{sign}(r_i) = J^T \operatorname{sign}(r) \qquad (3-1)$$

Part (b)

We can re write the equation above as a gradient of a weighted least squares problem.

$$\frac{\partial f(z)}{\partial z_k} = \sum_{i=1}^{n} J_{i,k} \operatorname{sign}(r_i) = J^T \operatorname{sign}(r) = \sum_{i=1}^{n} J_{i,k} \frac{r_i}{|r_i|} = \sum_{i=1}^{n} J_{i,k} \frac{1}{|r_i|} r_i$$

$$\Rightarrow \frac{\partial f(z)}{\partial z_k} = J^T W (Jz - d)$$

$$W_{i,i} = \frac{1}{|r_i| + \varepsilon} \qquad (3-2)$$

For solving the above problem I will write the diagonal weight matrix W as $W = B^T B$ which $B = W^{\frac{1}{2}}$.

$$\Rightarrow \frac{\partial f(z)}{\partial z_k} = J^T W(Jz - d) \Rightarrow J^T \left(B^T B\right)(Jz - d) = 0$$

$$\begin{cases} \hat{J} = BJ \\ \hat{d} = Bd \end{cases} \Rightarrow \hat{J}^T(\hat{J}z - \hat{d}) = 0 \Rightarrow \hat{J}^T \hat{J} z = \hat{J}^T d. \qquad (3-3)$$

The IRLS MATLAB code function is as below:

```
function [zn, tmpz] = IRLS(J, d, atol, kmax)

z0 = J\d;
eta = 1e-12;

zn = z0;
tmpz(1, :) = norm(J*zn-d, 1);
z = 0;
k = 0;
while (norm(zn - z) >  atol*(1+norm(z))) && k<kmax

    z = zn;
    r = J*z-d;
    diagW = 1./(abs(r) + eta);
    W = diag(diagW);
    zn = (sqrt(W) * J)\(sqrt(W) * d);
    k = k +1;
    tmpz(k+1, :) = norm(J*zn-d, 1);
end
```

Part (c)

The total 8 runs for this part are as below

```
Run (1)

Size of J is =

ans =

   100    50

Tolerance is = 0.0001
1-norm of Jz-d in every 10th iteration =

ans =

  62.558585855501008
  54.000001094904164
  53.739385495663996
  53.596177852233112
  53.517900167460084
  53.473568831724798
  53.444318108057701
  53.425117747758591
  53.413007149140668
  53.404932956480465
  53.399798398093672
  53.397507069144538
```

```
Run (2)

Size of J is =

ans =

   200    20

Tolerance is = 0.0001
1-norm of Jz-d in every 10th iteration =

ans =

   1.0e+02 *

   1.550375089954235
   1.494942281178250
   1.493391424908961
   1.492899333575093
   1.492860843081828
   1.492858722412926
```

```
Run (3)

Size of J is =

ans =

   200    50

Tolerance is = 0.0001
1-norm of Jz-d in every 10th iteration =

ans =

   1.0e+02 *

   1.316271757680765
   1.203888707063562
   1.200819891886280
   1.200041707276868
   1.199438943012619
   1.199074142483433
   1.198840296767982
   1.198720538831042
   1.198684653716744
```

```
Run (4)

Size of J is =

ans =

   200   100

Tolerance is = 0.0001
1-norm of Jz-d in every 10th iteration =

ans =

   1.0e+02 *

   1.054218964041704
   0.893440208131291
   0.890120818912766
   0.889229001393183
   0.888778743845066
   0.888574889181346
```

```
   0.888506131029741
   0.888476160320584
   0.888454901661449
   0.888438425830705
   0.888426550667022
   0.888418590535499
   0.888412803047244
   0.888407946833218
   0.888403588424727
   0.888399590138129
   0.888395888197709
   0.888392440687825
   0.888389216994079
   0.888386195085511
   0.888383359633872
   0.888380700027264
   0.888378208526169
   0.888375878836524
   0.888373705186540
```

```
Run (5)

Size of J is =

ans =

   100    50

Tolerance is = 1e-06
1-norm of Jz-d in every 10th iteration =

ans =

  41.791765620882792
  35.772978459877095
  35.575683292914029
  35.541399809978053
  35.532449911673496
  35.531180928318605
  35.530466475433805
  35.529829717167914
  35.529269566941267
  35.528806236354782
  35.528471761252575
  35.528283455371195
  35.528206745254721
  35.528182522046194
  35.528175718407546
```

Run (6)

Size of J is =

ans =

```
   200    20
```

Tolerance is = 1e-06
1-norm of Jz-d in every 10th iteration =

ans =

```
   1.0e+02 *

   1.563301030886292
   1.526319196012898
   1.522892952149651
   1.522076832389271
   1.521720264340486
   1.521498927365026
   1.521461350692464
   1.521460486517506
   1.521449593269062
   1.521426063322131
   1.521400507413630
   1.521380425500167
   1.521371801270077
   1.521370491477122
```

Run (7)

ans =

```
   200    50
```

Tolerance is = 1e-06
1-norm of Jz-d in every 10th iteration =

ans =

```
   1.0e+02 *

   1.525220596558480
```

```
   1.407187708030022
   1.403809246406960
   1.403311492173082
   1.403040845539044
   1.402926306152101
   1.402900424067968
   1.402884203037388
   1.402871489381820
   1.402862095957987
   1.402855109571612
   1.402849489865019
   1.402844625082016
   1.402840308913486
   1.402836716544836
   1.402833987284774
   1.402831744427157
   1.402829749606588
   1.402827815538348
   1.402825897510603
   1.402824016591337
   1.402822195257910
   1.402820451866967
   1.402818802670388
   1.402817263577641
   1.402815851227069
   1.402814583321715
   1.402813477868318
   1.402812550667613
   1.402811810653427
   1.402811254217120
   1.402810862097415
   1.402810602502593
   1.402810439481133
   1.402810341078125
   1.402810283256646
   1.402810249854518
```

```
Run (8)

   200   100

Tolerance is = 1e-06
1-norm of Jz-d in every 10th iteration =

ans =

   1.0e+02 *
```

1.220106289961938
1.026070773149288
1.021431986241094
1.019401949525617
1.018433107442186
1.017983839092648
1.017768473310329
1.017621417983627
1.017510076881245
1.017431641947025
1.017385636022152
1.017365070690206
1.017356680088845
1.017352200596456
1.017349203561085
1.017346993686707
1.017345277112322
1.017343897933922
1.017342765019189
1.017341821522108
1.017341029888910
1.017340363504618
1.017339801781352
1.017339327444802
1.017338925287168
1.017338581789268
1.017338285124809
1.017338025217957
1.017337793700965
1.017337583751676
1.017337389857303
1.017337207559826
1.017337033222539
1.017336863836541
1.017336696871608
1.017336530167760
1.017336361860785
1.017336190334045
1.017336014189399
1.017335832231217
1.017335643457946
1.017335447057181
1.017335242401122
1.017335029040313
1.017334806694923
1.017334575242914
1.017334334706204
1.017334085234804

1.017333827090373
1.017333560629658
1.017333286288678
1.017333004568030
1.017332716019627
1.017332421235432
1.017332120837556
1.017331815470045
1.017331505792438
1.017331192474374
1.017330876191672
1.017330557623039
1.017330237447830
1.017329916344230
1.017329594988129
1.017329274051957
1.017328954203934
1.017328636107202
1.017328320419065
1.017328007789988
1.017327698862779
1.017327394271386
1.017327094640171
1.017326800583252
1.017326512704811
1.017326231600091
1.017325957858436
1.017325692068286
1.017325434825527
1.017325186745384
1.017324948478894
1.017324720734596
1.017324504304053
1.017324300091369
1.017324109140820
1.017323932653520
1.017323771975218
1.017323628524433
1.017323503622927
1.017323398201872
1.017323312419456
1.017323245337102
1.017323194871124
1.017323158128910
1.017323131981061
1.017323113578366
1.017323100627476
1.017323091432556

The MALAB code used in this question is as below:

```
clear; clc; close all
format long
kmax = 1000;

atol = 1e-4;

J = randn(100, 50);
d  = randn(100, 1);
[zn, tmpz] = IRLS(J, d, atol, kmax);
fprintf('Size of J is = \n')
size(J)
fprintf('Tolerance is = %g \n', atol)
fprintf('1-norm of Jz-d in every 10th iteration = \n')
tmpz(1:10:end)


J = randn(200, 20);
d  = randn(200, 1);
[zn, tmpz] = IRLS(J, d, atol, kmax);
fprintf('Size of J is = \n')
size(J)
fprintf('Tolerance is = %g \n', atol)
fprintf('1-norm of Jz-d in every 10th iteration = \n')
tmpz(1:10:end)


J = randn(200, 50);
d  = randn(200, 1);
[zn, tmpz] = IRLS(J, d, atol, kmax);
fprintf('Size of J is = \n')
size(J)
fprintf('Tolerance is = %g \n', atol)
fprintf('1-norm of Jz-d in every 10th iteration = \n')
tmpz(1:10:end)


J = randn(200, 100);
d  = randn(200, 1);
[zn, tmpz] = IRLS(J, d, atol, kmax);
fprintf('Size of J is = \n')
size(J)
fprintf('Tolerance is = %g \n', atol)
fprintf('1-norm of Jz-d in every 10th iteration = \n')
tmpz(1:10:end)


atol = 1e-6

J = randn(100, 50);
d  = randn(100, 1);
```

```
[zn, tmpz] = IRLS(J, d, atol, kmax);
fprintf('Size of J is = \n')
size(J)
fprintf('Tolerance is = %g \n', atol)
fprintf('1-norm of Jz-d in every 10th iteration = \n')
tmpz(1:10:end)


J = randn(200, 20);
d  = randn(200, 1);
[zn, tmpz] = IRLS(J, d, atol, kmax);
fprintf('Size of J is = \n')
size(J)
fprintf('Tolerance is = %g \n', atol)
fprintf('1-norm of Jz-d in every 10th iteration = \n')
tmpz(1:10:end)


J = randn(200, 50);
d  = randn(200, 1);
[zn, tmpz] = IRLS(J, d, atol, kmax);
fprintf('Size of J is = \n')
size(J)
fprintf('Tolerance is = %g \n', atol)
fprintf('1-norm of Jz-d in every 10th iteration = \n')
tmpz(1:10:end)


J = randn(200, 100);
d  = randn(200, 1);
[zn, tmpz] = IRLS(J, d, atol, kmax);
fprintf('Size of J is = \n')
size(J)
fprintf('Tolerance is = %g \n', atol)
fprintf('1-norm of Jz-d in every 10th iteration = \n')
tmpz(1:10:end)
```