PHYS 5794 Homework 4

Thomas Edwards

February 23, 2016

1 Problem 1

1.1 Problem Statement

Solve the following linear algebraic equations by using the singular value decomposition (SVD) method. As you noticed, the equations are the same as those in HW3. This is a chance to solve the same problem with a different method. For diagonalization of a matrix, use the Jacobi method discussed in the class. At least, the following items must be discussed in the report: (i) the column- orthogonal matrix U , the square diagonal matrix W , and the orthogonal matrix V . (ii) Show that A = UWV T . (iii) Write down the solution from your code. (iv) Confirm that your solution satisfies the original set of the equations. Is the solution of this problem the same as that in HW3? (v) The number of iterations used for the Jacobi transformation, where one iteration means after you go over all the off-diagonal elements once. (vi) The tolerance used in the Jacobi transformation. (vii) Confirm that the Jacobi transformation diagonalizes the given matrix which you would like to diagonalize. For both the SVD method and the Jacobi method, you need to mention how you tested your code.

1.2 Method

This problem requires two distinct parts to provide a solution. The first is the diagonalization through Jacobi's method, and the second is the single value decomposition.

The diagonalization through Jacobi's method is done in a similar way to the lecture notes. A matrix $\mathbf{A}$ is diagonalized iteratively by creating a sequence of similarity transforms $\mathbf{P}_{i,i-1,\ldots,0}$, such that

$$\mathbf{P}_i^{-1}\mathbf{P}_{i-1}^{-1}\mathbf{P}_{i-2}^{-1}\ldots\mathbf{P}_0^{-1}\mathbf{A}\mathbf{P}_0\ldots\mathbf{P}_{i-2}\mathbf{P}_{i-1}\mathbf{P}_i = \mathbf{A}.$$

These matrices $\mathbf{P}$ are chosen such that after the transformations, the diagonal elements of $\mathbf{A}$ are the eigenvalues of $\mathbf{A}$. A convenient and important consequence of choosing these similarity transforms in this was is that when combined, the matrices also transform the identity matrix $\mathbf{I}$ into a matrix of orthogonal eigenvectors $\mathbf{V}$:

$$\mathbf{V} = \mathbf{P}_0\ldots\mathbf{P}_{i-2}\mathbf{P}_{i-1}\mathbf{P}_i$$

The solution for this particular algorithm begins by iterating through all upper-triangular elements of $\mathbf{A}$ and finding the largest value and its index $\mathbf{A}_{p,q}$. The matrix is then transformed for all $r \neq p, q$ by

$$\begin{aligned}
a'_{pq} &= a_{qp} = 0 \\
a'_{pp} &= a_{pp} - ta_{pq} \\
a'_{qq} &= a_{qq} + ta_{pq} \\
a'_{rp} &= a_{rp} - s(a_{rq} + \tau a_{rp}) \\
a'_{rq} &= a_{rq} + s(a_{rp} - \tau a_{rq})
\end{aligned}$$

where

$$\begin{aligned}
\theta &= \frac{a_{pp} - a_{qq}}{2a_{pq}} \\
t &= \frac{\text{sign}(\theta)}{|\theta| + \sqrt{\theta^2 + 1}} \\
c &= \frac{1}{\sqrt{1 + t^2}} \\
s &= tc \\
\tau &= \frac{s}{1 + c}
\end{aligned}$$

This is done iteratively until the absolute value of all diagonal values are below a given tolerance, which in this case is chosen to be 1×10^{-13}.

From here, the Single Value Decomposition can be done. We separate $\mathbf{A}$ by required that

$$\mathbf{A} = \mathbf{U}\mathbf{W}\mathbf{V}^T$$

From the Jacobi method, we have already recovered $\mathbf{V}$. $\mathbf{W}$ in this case is simply a matrix who's diagonal is the eigenvalues found by Jacobi's method. $\mathbf{U}$ is found by the relation

$$u_i = w_i^{-1} A v_i$$

where w_i is one of the eigenvalues, and v_i is the associated eigenvector.

We can now solve for x by the relation

$$\mathbf{x} = \mathbf{V}\mathbf{W}^{-1}\mathbf{U}^T b$$

where $\mathbf{W}^{-1}$ is 1 divided by the diagonal elements of $\mathbf{W}$. This then provides our solution.

The specific implementation of this code also includes on number of verification routines, and some safeguards to make sure numerical issues do not stop the procedure.

1.3 Verification of Program

To verify this program, a simple test case from the lecture notes was used. The results of this verification are below.

```
----------------------------------------
 A simple test case, from lecture notes
----------------------------------------
****************************************
           Starting A and b
****************************************
 A =
[[ 1.  1.  4.]
 [ 3.  2.  1.]
 [ 2.  3.  1.]]
 b =
[ 12.  11.  13.]
****************************************
     Number of Sweeps (Iterations)
****************************************
2
****************************************
             Diagonalized A
****************************************
[[  1.   0.   0.]
 [  0.  36.   0.]
 [  9.   0.   9.]]
****************************************
           V and Eigenvalues
****************************************
 V =
[[ 0.70710678  0.57735027 -0.40824829]
 [-0.70710678  0.57735027 -0.40824829]
 [ 0.          0.57735027  0.81649658]]
 Eigenvalues =
[  1.  36.   9.]
```

```
****************************************
                  W
****************************************
 W =
[[ 1.  0.  0.]
 [ 0.  6.  0.]
 [ 0.  0.  3.]]
 1/sqrt(W) =
[[ 1.          0.          0.        ]
 [ 0.          0.16666667  0.        ]
 [ 0.          0.          0.33333333]]
****************************************
                  U
****************************************
 U =
[[ 0.          0.57735027  0.81649658]
 [ 0.70710678  0.57735027 -0.40824829]
 [-0.70710678  0.57735027 -0.40824829]]
****************************************
         Testing is A = U W V_T
****************************************
 A =
[[ 1.  1.  4.]
 [ 3.  2.  1.]
 [ 2.  3.  1.]]
 U W V_T =
[[ 1.  1.  4.]
 [ 3.  2.  1.]
 [ 2.  3.  1.]]
****************************************
                  x
****************************************
[ 1.  3.  2.]
****************************************
          Check if Ax = b
****************************************
 Ax =
[ 12.  11.  13.]
 b =
[ 12.  11.  13.]
****************************************
             Known Solution:
****************************************
[ 1.  3.  2.]
```

From this we see that the solution is as expected. In addition, the values for the intermediate matrix values (such as **U**, **V**, and **W**) are found to be the same as the lecture notes. Further, the eigenvalues were checked against numpy's own SVD method, and were found to be correct.

1.4 Data

The solution for the problem statement at hand is below.

```
----------------------------------------
       Problem Statement Solution
----------------------------------------
****************************************
           Starting A and b
****************************************
 A =
[[  2.   3.  10.  -1.]
 [ 10.  15.   3.   7.]
 [ -4.   1.   2.   9.]
 [ 15.  -3.   1.   3.]]
 b =
[ 1.  2.  3.  4.]
****************************************
     Number of Sweeps (Iterations)
****************************************
5
****************************************
             Diagonalized A
****************************************
[[ 483.37630327    0.            0.            0.        ]
 [   0.          198.07063946    0.            0.        ]
 [   0.            0.           68.84597834    0.        ]
 [   0.            0.            0.           92.70707893]]
****************************************
           V and Eigenvalues
****************************************
 V =
[[ 0.71936614 -0.69064832 -0.0742309   0.00264959]
 [ 0.54696394  0.62858534 -0.5498717  -0.05789672]
 [ 0.25066026  0.19830167  0.55675296 -0.76672813]
 [ 0.34714872  0.29759337  0.61818851  0.63935039]]
 Eigenvalues =
[ 483.37630327  198.07063946   68.84597834   92.70707893]
****************************************
                   W
****************************************
 W =
[[ 21.9858205    0.           0.           0.        ]
 [  0.          14.07375712   0.           0.        ]
 [  0.           0.           8.29734767   0.        ]
 [  0.           0.           0.           9.62845153]]
 1/sqrt(W) =
[[ 0.04548386  0.          0.          0.        ]
 [ 0.          0.07105423  0.          0.        ]
 [ 0.          0.          0.12052044  0.        ]
 [ 0.          0.          0.          0.10385886]]
****************************************
                   U
****************************************
 U =
```

```
[[ 0.23829349  0.15560044  0.37979175 -0.88020619]
 [ 0.84509661  0.36950726 -0.36069431  0.13847641]
 [ 0.0589088   0.45944535  0.77425397  0.43124299]
 [ 0.47492914 -0.79256726  0.3552304   0.14174211]]
****************************************
         Testing is A = U W V_T
****************************************
 A =
[[  2.   3.  10.  -1.]
 [ 10.  15.   3.   7.]
 [ -4.   1.   2.   9.]
 [ 15.  -3.   1.   3.]]
 U W V_T =
[[  2.   3.  10.  -1.]
 [ 10.  15.   3.   7.]
 [ -4.   1.   2.   9.]
 [ 15.  -3.   1.   3.]]
****************************************
                   x
****************************************
[ 0.14498382 -0.17346278  0.16116505  0.38122977]
****************************************
          Check if Ax = b
****************************************
 Ax =
[ 1.  2.  3.  4.]
 b =
[ 1.  2.  3.  4.]
****************************************
             Known Solution:
****************************************
[ 0.14498382 -0.17346278  0.16116505  0.38122977]
```

1.5 Analysis

The Jacobi method appears to converge to a solution very quickly, based partially on the recommendations made within the lecture notes to improve the scheme. During debugging it was found that many values got below the tolerance, but were not being eliminated, so a number of safeguard routines had to be implemented in order to establish a stable, converging solution.

1.6 Interpretation

It appears that even with the safeguard modifications, the routine is very accurate. In particular, the results seen in the debugging were not showing 0's, and instead values on the scale of 10^{-30}, which is well below machine tolerance for my setup.

We also see that the matrices do in fact recover **A** when multiplied, and that the Jacobi method diagonalized the matrix as expected. There is a small exception, however, where the sample problem was not completely diagonalized. This may be due to an indexing issue, but ultimately did not have an impact on the solution. On further inspection, it was found that this matrix had some non-real eigenvalues, which explained the case of the non-diagonal result. Regardless, the SVD method still provided results that were reasonable.

1.7 Critique

A significant part of the problem was actually getting the Jacobi method to work, and then understanding the indexing of **U**, **V**, and **W** from the lecture notes. In the future, I would suggest splitting this into two problems: one about Jacobi's transformation and another about SVD with both square and non-square matrices.

1.8 Log

In total, both problems took about 14 hours.

2 Problem 2

2.1 Problem Statement

Solve the following linear algebraic equations by using the singular value decomposition (SVD) method. As you noticed, the equations are the same as those in HW3. This is a chance to solve the same problem with a different method. For diagonalization of a matrix, use the Jacobi method discussed in the class. At least, the following items must be discussed in the report: (i) the column- orthogonal matrix U , the square diagonal matrix W , and the orthogonal matrix V . (ii) Show that A = UWV T . (iii) Write down the solution from your code. (iv) Confirm that your solution satisfies the original set of the equations. Is the solution of this problem the same as that in HW3? (v) The number of iterations used for the Jacobi transformation, where one iteration means after you go over all the off-diagonal elements once. (vi) The tolerance used in the Jacobi transformation. (vii) Confirm that the Jacobi transformation diagonalizes the given matrix which you would like to diagonalize. For both the SVD method and the Jacobi method, you need to mention how you tested your code.

2.2 Method

This problem followed the same method of Problem 1, except that the input matrix **A** and vector **b** needed to modified with 0's for the missing equation from previously.

2.3 Verification of Program

To verify this program, a simple test case from the lecture notes was used. The results of this verification are below.

```
----------------------------------------
 A simple test case, from lecture notes
----------------------------------------
****************************************
           Starting A and b
****************************************
 A =
[[ 3.  1.  1.]
 [-1.  3.  1.]
 [ 0.  0.  0.]]
 b =
[ 1.  1.  0.]
****************************************
     Number of Sweeps (Iterations)
****************************************
5
****************************************
             Diagonalized A
```

```
****************************************
[[  1.00000000e+01   0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   1.20000000e+01   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00  -1.68361192e-16]]
****************************************
          V and Eigenvalues
****************************************
 V =
[[ 0.89442719  0.40824829 -0.18257419]
 [-0.4472136   0.81649658 -0.36514837]
 [ 0.          0.40824829  0.91287093]]
 Eigenvalues =
[ 10.  12.   0.]
****************************************
                  W
****************************************
 W =
[[ 3.16227766  0.          0.        ]
 [ 0.          3.46410162  0.        ]
 [ 0.          0.          0.        ]]
 1/sqrt(W) =
[[ 0.31622777  0.          0.        ]
 [ 0.          0.28867513  0.        ]
 [ 0.          0.          0.        ]]
****************************************
                  U
****************************************
 U =
[[ 0.70710678  0.70710678  0.        ]
 [-0.70710678  0.70710678  0.        ]
 [ 0.          0.          0.        ]]
****************************************
         Testing is A = U W V_T
****************************************
 A =
[[ 3.  1.  1.]
 [-1.  3.  1.]
 [ 0.  0.  0.]]
 U W V_T =
[[ 3.  1.  1.]
 [-1.  3.  1.]
 [ 0.  0.  0.]]
****************************************
                  x
****************************************
[ 0.16666667  0.33333333  0.16666667]
****************************************
          Check if Ax = b
****************************************
 Ax =
[ 1.  1.  0.]
 b =
[ 1.  1.  0.]
****************************************
```

```
             Known Solution:
****************************************
[ 1.  3.  2.]
```

The results are as expected, and when compared to the lecture notes appear to return the correct results.

2.4 Data

The solution for the problem statement at hand is below.

```
----------------------------------------
       Problem Statement Solution
----------------------------------------
****************************************
           Starting A and b
****************************************
 A =
[[  2.   3.  10.  -1.]
 [ 10.  15.   3.   7.]
 [ -4.   1.   2.   9.]
 [  0.   0.   0.   0.]]
 b =
[ 1.  2.  3.  0.]
****************************************
     Number of Sweeps (Iterations)
****************************************
5
****************************************
             Diagonalized A
****************************************
[[ -8.13166745e-15   0.00000000e+00   0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   4.15377511e+02   0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   8.39399635e+01   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   9.96825255e+01]]
****************************************
           V and Eigenvalues
****************************************
 V =
[[ 0.68014868  0.46632981 -0.37563506 -0.42288601]
 [-0.6361643   0.74714394 -0.19012377 -0.03039509]
 [ 0.09011434  0.29087873  0.89407506 -0.32847946]
 [ 0.35294782  0.3737673   0.15290859  0.84400522]]
 Eigenvalues =
[   0.          415.37751099   83.93996347   99.68252554]
****************************************
                  W
****************************************
 W =
[[  0.           0.           0.           0.        ]
 [  0.          20.38081232   0.           0.        ]
 [  0.           0.           9.16187554   0.        ]
 [  0.           0.           0.           9.98411366]]
 1/sqrt(W) =
[[ 0.          0.          0.          0.        ]
 [ 0.          0.04906576  0.          0.        ]
```

```
 [ 0.          0.          0.10914796  0.        ]
 [ 0.          0.          0.          0.10015912]]
****************************************
                  U
****************************************
 U =
[[ 0.          0.28012188  0.81492054 -0.50738176]
 [ 0.          0.9498868  -0.31168529  0.02381901]
 [ 0.          0.13873283  0.48862746  0.86139213]
 [ 0.          0.          0.          0.        ]]
****************************************
        Testing is A = U W V_T
****************************************
 A =
[[  2.   3.  10.  -1.]
 [ 10.  15.   3.   7.]
 [ -4.   1.   2.   9.]
 [  0.   0.   0.   0.]]
 U W V_T =
[[  2.   3.  10.  -1.]
 [ 10.  15.   3.   7.]
 [ -4.   1.   2.   9.]
 [  0.   0.   0.   0.]]
****************************************
                  x
****************************************
[-0.09853585  0.05430877  0.12890061  0.25486073]
****************************************
         Check if Ax = b
****************************************
 Ax =
[ 1.  2.  3.  0.]
 b =
[ 1.  2.  3.  0.]
****************************************
            Known Solution:
****************************************
[ 0.14498382 -0.17346278  0.16116505  0.38122977]
```

2.5 Analysis

The method completed in the same manner as Problem 1, but required more handling of the small values in the matrices. Since many of the values were very small but still manageable by python, they had to be set to 0 by relating them to a tolerance value.

2.6 Interpretation

The diagonalization of the matrix appears to be correct, and during the verification **U**, **V**, and **W** all appeared to be correct. The results are not going to be exact, since the missing information requires that we find the minimum of $\mathbf{Ax} = \mathbf{b}$, instead of the exact solution. With that in mind, it is confirmed that $\mathbf{Ax} = \mathbf{b}$ manually to double check to make sure the solution is still valid.

2.7 Log

In total, both problems took about 14 hours.