

LAB 5: Positive-Definite Matrices, Cholesky Factorization, Singular Value Decomposition, and Digital Image Processing

In this lab you will use MATLAB to study the following topics:

- Tests for positive-definiteness of a real symmetric matrix
- The Cholesky decomposition of a positive-definite matrix
- The singular value decomposition (SVD) of a matrix
- Digital image processing using the SVD

MATLAB Preliminaries

Tcodes: For this lab you will need the Teaching Codes

```
house.m, plot2d.m, slu.m
```

Before opening MATLAB to work on the Lab questions you should copy these codes to your directory.

Random Seed: When you start your MATLAB session, initialize the random number generator by typing

```
rand('seed', abcd)
```

where *abcd* are the last four digits of your Student ID number. This will ensure that you generate your own particular random vectors and matrices.

BE SURE TO INCLUDE THIS LINE IN YOUR LAB WRITE-UP

Question 1. Positive-Definite Matrices

(a) **Tests for Positivity:** (see Strang, page 318, 6B) Generate a random 3×3 symmetric matrix A with integer entries by

```
B = fix(5*rand(3)); A = B + B' - ones(3,3)
```

Determinant Test: Calculate the three *principal minors* of A by

```
D1 = A(1,1), D2 = det(A(1:2,1:2)), D3 = det(A)
```

One test for A to be *positive definite* is

$$D_1 > 0, \quad D_2 > 0, \quad D_3 > 0$$

Does the matrix A pass this test? If it fails this test, use the up-arrow key to generate another symmetric matrix A until you get one that passes this test. Since A has entries that are small integers that are very likely to be positive, you can see at a glance without MATLAB whether $D_1 > 0$ and $D_2 > 0$. Use a matrix A that passes this test in the next two steps.

Eigenvalue Test: Another test for A to be positive definite is that all the *eigenvalues* of A are positive. Calculate these eigenvalues by the MATLAB command `Lambda = eig(A)`. Are the results of this test consistent with the Determinant Test?

Pivot Test: A third test for A to be positive definite is that all the *pivots* in the LU decomposition of A are positive. Check this by calculating

```
[L, U] = slu(A)
```

Note that you are using the special T-code `slu.m`, which calculates the LU factorization so that the pivots of A are the diagonal entries of U (this is not the case with the built-in MATLAB `lu` function). Are the pivots all positive? Use MATLAB to calculate the pivots in terms of the principal minors D_1, D_2, D_3

(b) Rayleigh Quotients: Let A be the matrix from (a) that passed all the tests for positivity. Generate a vector $\mathbf{x} = 2*\text{rand}(3,1) - \text{ones}(3,1)$ with random entries between -1 and 1 . The smallest eigenvalue of A is `Lambda(1)` and the largest eigenvalue is `Lambda(3)`.

(i) What inequalities hold between `Lambda(1)`, `Lambda(3)`, and the *Rayleigh quotient* $(\mathbf{x}'*A*\mathbf{x})/(\mathbf{x}'*\mathbf{x})$?

Use MATLAB to verify these inequalities for your random vector \mathbf{x} .

Generate two more random vectors \mathbf{x} and calculate the Rayleigh quotients for each vector. What information does this give you about the eigenvalues of A (see Strang, page 342)? Is this consistent with your MATLAB calculations in part (a)?

(c) Law of Inertia: Generate a random 3×3 real symmetric matrix A by the method of part (a) that is *not* positive definite. Use the T-code `slu.m` and command `eig` to verify that A has the same number of negative pivots as negative eigenvalues (see Strang, page 325, 6G).

Question 2. Cholesky Factorization of Positive-Definite Matrices

(a) If B has independent columns, then the matrix $A = B^T B$ is always positive definite. Check this by generating a random 4×3 matrix

```
B = rmat(4,3) - 5*ones(4,3)
```

Calculate `rank(B)` to determine if B has independent columns. If this is the case, set $A = B'*B$ (otherwise, generate another matrix B and repeat).

(b) Use MATLAB to calculate an orthonormal basis of eigenvectors for A by the command

```
[Q, D] = eig(A)
```

How do you know from this calculation that A is positive definite? Use MATLAB to verify that Q is an orthogonal matrix and that $Q'*A*Q = D$.

(c) The positive-definite matrix A has a *Cholesky factorization* $A = R^T R$, where R is upper triangular with positive diagonal entries. One way to obtain this factorization is to start with the LU factorization. Compute

```
[L, U] = slu(A), H = diag((diag(U)))
```

(i) Show by hand calculation why $A = LHL^T$, and then verify this by MATLAB.

(ii) The matrices A and H are congruent. Do they have the same eigenvalues? Explain.

(iii) Let $R = \text{sqrt}(H)*L'$ (take the square root of the diagonal entries of H). Show by hand calculation why $A = R^T * R$, and then verify this by MATLAB.

Finally, calculate the Cholesky decomposition of A by the MATLAB m-file `chol(A)` and verify that you get the same matrix R ; this built-in MATLAB function takes advantage of the positive-definiteness of A and is considerably faster than the LU factorization method for large matrices.

Question 3. Singular Value Decomposition

(a) Graphic Demo of SVD: Generate a random 2×2 matrix $A = \text{rand}(2,2)$. Then type `eigshow(A)` at the MATLAB prompt. A graphics window should open. Click on the `svd` button on the right side of the window. Your matrix A should appear (in MATLAB notation) in the menu bar above the graph. Underneath the graph the statement

```
Make A*x perpendicular to A*y
```

should appear (if it does not, then click on the `svd` button again). The graph shows a pair of orthogonal unit vectors \mathbf{x} and \mathbf{y} , together with the transformed vectors $A\mathbf{x}$ and $A\mathbf{y}$.

Move the pointer onto the vector \mathbf{x} , and then make the pair of vectors \mathbf{x} , \mathbf{y} go around a circle. The transformed vectors $A\mathbf{x}$ and $A\mathbf{y}$ then move around an ellipse, but generally $A\mathbf{x}$ is not perpendicular to $A\mathbf{y}$. Search for the position of \mathbf{x} and \mathbf{y} so that $A\mathbf{x}$ is perpendicular to $A\mathbf{y}$. When this happens, then the *singular values* σ_1 and σ_2 of A are the lengths of the vectors $A\mathbf{x}$ and $A\mathbf{y}$. Give a rough estimate of these lengths from the graph (remember that \mathbf{x} and \mathbf{y} have length one).

(b) Calculation of SVD: Let A be the random matrix you generated in part (a). Use MATLAB to calculate the singular value decomposition of A by

```
[U, S, V] = svd(A)
```

Verify that $A = U * S * V'$ (up to numerical roundoff error).

The diagonal entries of S are the *singular values* σ_1, σ_2 of A . Compare the calculated singular values with your graphical estimates for σ_1, σ_2 from part (a).

The singular values are the square roots of the eigenvalues of $A^T A$. Check this by calculating

```
sqrt(eig(A'*A))
```

(c) Geometric Meaning of SVD: Close the SVD demo window, and at the MATLAB prompt type

```
H = house;
figure(1), plot2d(H)
```

A graphics window should open and display a crude drawing of a house (the matrix H contains the coordinates of the line segments of the figure).

Generate an orthogonal matrix V by

```
t = pi/6; V = [cos(t), -sin(t); sin(t), cos(t)]
```

Generate a new plot by `figure(1), plot2d(V'*H)` (be sure to use the transpose matrix V'). How has the house been changed?

Next generate a diagonal matrix S by

```
S = [5/4, 0; 0, 3/4 ]
```

Generate a new plot by `figure(1), plot2d(S*V'*H)`. How does this change the previous picture?

Now generate another orthogonal matrix U by

```
t = pi/4; U = [cos(t), -sin(t); sin(t), cos(t)]
```

(use \uparrow to edit a previous command). Generate a new plot by `figure(1), plot2d(U*S*V'*H)`. How does this change the previous picture? Print this final version of Figure 1 and include it in your lab write up.

Set $A = U*S*V'$. Use MATLAB to calculate the singular value decomposition of A by

```
[U1, S1, V1] = svd(A)
```

Verify that $S = S1$ and that $U1*S1*V1' = A$ (up to numerical roundoff error). Note that the U, V matrices in the SVD are *not* unique (since the eigenvectors of $A^T A$ and AA^T can be multiplied by ± 1).

Question 4. Digital Image Processing by SVD

(a) Digital Images: Monochromatic digital images can be encoded as integer matrices with elements between 1 and 64 indicating the gray-scale. The image of the 4×4 magic square in Albrecht Dürer's etching *Melancholia II* is stored as a 359×371 matrix in the MATLAB file `detail.mat`. Load and display it by the commands

```
load detail
figure(2), subplot(2,2,1)
image(X), colormap(gray(64)), axis image, axis off
```

(this image file is included with the normal MATLAB installation). Calculate the rank of the image matrix X and label the displayed image by the commands

```
size(X), r = rank(X)
title(['rank = ' int2str(r)])
```

Notice that X is of full rank.

(b) SVD of Image Matrix: Calculate the singular value decomposition of the image and plot the singular values (on a logarithmic scale) by the commands (**be sure to use the semicolons ; as indicated so that the matrices are not displayed**)

```
[U, S, V] = svd(X,0);
sigma = diag(S);
figure(3), semilogy(sigma, '.')
```

Notice that the singular values decrease very rapidly (the labeling on the horizontal axis indicates the relative size of the singular values, with 1 being the largest). Look at the vector `sigma(1:10)` to answer the following questions:

- (i) How many singular values are greater than 10^4 ?
- (ii) How many singular values are greater than 10^3 ?

Now use the magnifying feature on the figure tool bar to zoom in on the 100 largest singular values (you may have to experiment with this to get a satisfactory graph—click with the cursor on the 50th singular value). Label and print figure 3 showing the graph of the 100 largest singular values (but don't try to include the largest singular value in your graph).

(c) Principal Component Approximations to Image: The singular value matrix S has the singular values σ_k arranged on its diagonal in decreasing order. The *rank-one* principal component approximation to X is given by the matrix $X_1 = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T$, where \mathbf{u}_1 is the first column of U and \mathbf{v}_1 is the first column of V . Generate and plot this approximation in the first figure window by the commands (**be careful with ; and with ')**

```
S_1 = S(1,1); U_1 = U(:,1); V_1 = V(:,1);
X_1 = U_1*S_1*V_1';
figure(2), subplot(2,2,2), image(X_1)
axis image, axis off
```

Use the same method as in part (a) to label this subplot `rank = 1`. Notice that the rank-one approximation shows the horizontal and vertical lines in the image, but no details.

Now generate the rank-20 approximation and graph it by (**be careful with ; and with ')**

```
S_20 = S(1:20,1:20); U_20 = U(:,1:20); V_20 = V(:,1:20);
X_20 = U_20*S_20*V_20';
figure(2), subplot(2,2,3), image(X_20)
axis image, axis off
```

Verify the rank of X_{20} by MATLAB. Use the same method as in part (a) to label this subplot `rank = 20`. Notice how much of the detail of the original image is already contained in the rank-20 approximation.

- (i) What is the percentage error between the matrix X and the approximation X_{20} ? (Measure the error using `norm`.)
- (ii) If X_{20} is stored as the pair of matrices U_{20} , V_{20} together with the diagonal elements (the singular values) of S_{20} , what *compression ratio* is achieved, compared with the original matrix X ? (Use the `size` command to find the total number of entries in U_{20} and V_{20} . Add to this the number of diagonal entries in S_{20} . Now take the ratio of the number of entries in X to this total.)

Modify these calculations to generate the rank-80 approximation `X_80`; use the graphing commands

```
figure(1), subplot(2,2,4), image(X_80)
axis image, axis off
```

to display the plot. Use the same method as in part (a) to label this subplot `rank = 80`. Notice that the rank-80 approximation looks almost identical to the original image. Answer questions (i) and (ii) for the rank-80 approximation. Print figure 2 with the original image and the rank 1, 20, and 80 approximations.

Remark. Although the low-rank approximations to images obtained by the SVD do achieve excellent compression ratios, there are more effective techniques that do not require as much computation (for example, wavelet transforms). The most important use of the SVD in image processing is for feature recognition.

Final Editing of Lab Write-up:

After you have worked through all the parts of the lab assignment, edit your diary file. Include the MATLAB calculations, but remove errors that you made in entering commands and remove other material that is not directly related to the questions.