Homework 7

Math 3607, Autumn 2021

Marco LoPiccolo

Table of Contents


```
clc
```

Problem 1.

1. (Using `eig`; **FNC 7.2.3**) Use `eig` to find the EVD of each matrix. Then for each eigenvalue λ, use the `rank` command to verify that $\lambda I - A$ is singular.

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & -1 & -1 \\ -2 & 2 & -1 \\ -1 & -2 & 2 \end{bmatrix}, \quad C = \begin{bmatrix} 4 & -3 & -2 & -1 \\ -2 & 4 & -2 & -1 \\ -1 & -2 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix}.$$

```
A = [2 -1 0; -1 2 -1; 0 -1 2];
B = [2 -1 -1; -2 2 -1; -1 -2 2];
C = [4 -3 -2 -1; -2 4 -2 -1; -1 -2 4 -1; -1 -2 -1 4];
[V, D] = eig(A)
```

```
V = 3×3
                 0.5      -0.707106781186547                    -0.5
   0.707106781186547    4.88509860010542e-17       0.707106781186547
                 0.5       0.707106781186547                    -0.5
D = 3×3
   0.585786437626905                       0                       0
                   0                       2                       0
                   0                       0        3.41421356237309
```

```
AcheckEVD = norm(A - V*D/V)
```

```
AcheckEVD =
    4.00296604248672e-16
```

```
for i = 1:3
    rank(D(i,i)*eye(3) - A)
```

```
end
```

```
ans =
     2
ans =
     2
ans =
     2
```

```
%rank is less than number of columns therefore the matrix is singular
[S, T] = eig(B)
```

```
S = 3×3 complex
          0.473810655208375 +                      0i ···
          0.604431820749537 +                      0i
          0.640441751509385 +                      0i
T = 3×3 complex
         -0.627365084711833 +                      0i ···
                          0 +                      0i
                          0 +                      0i
```

```
BcheckEVD = norm(B - S*T/S)
```

```
BcheckEVD =
      1.29417586711847e-15
```

```
for i = 1:3
    rank(T(i,i)*eye(3) - B)
end
```

```
ans =
     2
ans =
     2
ans =
     2
```

```
%rank is less than number of columns therefore the matrix is singular
[L, M] = eig(C)
```

```
L = 4×4
         0.596679678521752         0.617859345276244         -0.56548073559032 ···
         0.520060031310845         0.270644956514782         0.740568557833172
         0.432152128926922        -0.522015869954125        -0.256699969638804
         0.432152128926922        -0.522015869954125        -0.256699969638804
M = 4×4
        -0.787554501253493                         0                         0 ···
                         0          5.22052493999402                         0
                         0                         0          6.56702956125948
                         0                         0                         0
```

```
CcheckEVD = norm(C - L*M/L)
```

```
CcheckEVD =
      3.64072695440716e-15
```

```
for i = 1:4
    rank(M(i,i)*eye(4) - C)
end
```

```
ans =
```

```
        3
ans =
        3
ans =
        3
ans =
        3
```

```
%rank is less than number of columns therefore the matrix is singular
```

Problem 2.

Part a.)

2. (Polynomial evaluation of matrices; **FNC** 7.2.5 and Su20 final exam) Let $p(z) = c_1 + c_2 z + \cdots + c_n z^{n-1}$. The value of p for a square matrix input is defined as

$$p(X) = c_1 I + c_2 X + \cdots + c_n X^{n-1}.$$

(a) Show that if $X \in \mathbb{R}^{k\times k}$ has an EVD, then $p(X)$ can be found using only evaluations of p at the eigenvalues and two matrix multiplications.

2a.) $p(z) = c_1 + c_2 z + \ldots\ldots + c_n z^{n-1}$

$p(X) = c_1 I + c_2 X + \ldots\ldots + c_n X^{n-1}$

If $X \in \mathbb{R}^{k \times k}$ has an EVD $A = VDV^{-1}$,

then $A^2 = (VDV^{-1})(VDV^{-1}) = VD^2V^{-1}$

$A^3 = \ldots\ldots\ldots = VD^3V^{-1}$

$\vdots$

$A^n = \ldots\ldots = VD^nV^{-1}$

This means that $p(X)$ can be found using the Matrix of the eigenvectors V and its inverse V^{-1} with the evaluations of the eigenvalues which is D the diagonal matrix of all the eigenvalues.

V and V^{-1} are unchanged and D just has to be put to the power of n

$$D = \begin{bmatrix} \lambda_1^n & & 0 \\ & \lambda_2^n & \\ 0 & & \ddots \; \lambda_k^n \end{bmatrix}$$

$X = VDV^{-1}$

So $p(X) = c_1 VD^0V^{-1} + c_2 VD^1V^{-1} + c_3 VD^2V^{-1} + \ldots\ldots + c_n VD^{n-1}V^{-1}$

$$= V \Big[\underbrace{c_1 I + c_2 D + c_3 D^2 + \ldots\ldots\ldots + c_n D^{n-1}}_{P(D)} \Big] V^{-1}$$

$$P(D) = c_1 I + c_2 D + c_3 D^2 + \cdots\cdots + c_n D^{n-1}$$

$$= c_1 \begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ & & 1 & \\ 0 & & & \ddots\ 1 \end{bmatrix} + c_2 \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_k \end{bmatrix}$$

$$+ c_3 \begin{bmatrix} \lambda_1^2 & & & 0 \\ & \lambda_2^2 & & \\ & & \ddots & \\ 0 & & & \lambda_k^2 \end{bmatrix} + \cdots\cdots + c_n \begin{bmatrix} \lambda_1^{n-1} & & & 0 \\ & \lambda_2^{n-1} & & \\ & & \ddots & \\ 0 & & & \lambda_k^{n-1} \end{bmatrix}$$

$$= \begin{bmatrix} c_1 + c_2\lambda_1 + c_3\lambda_1^2 + \cdots + c_n\lambda_1^{n-1} & & & 0 \\ & c_1 + c_2\lambda_2 + c_3\lambda + \cdots + c_n\lambda_2^{n-1} & & \\ & & \ddots & \\ 0 & & & c_1 + c_2\lambda_k + c_3\lambda_k^2 + \cdots\cdots + c_n\lambda_k^{n-1} \end{bmatrix}$$

$$= \begin{bmatrix} P(\lambda_1) & & & 0 \\ & P(\lambda_2) & & \\ & & \ddots & \\ 0 & & & P(\lambda_k) \end{bmatrix}$$

So $$\boxed{P(X) = V \begin{bmatrix} P(\lambda_1) & & & 0 \\ & P(\lambda_2) & & \\ & & \ddots & \\ 0 & & & P(\lambda_k) \end{bmatrix} V^{-1}}$$

Part b.)

(b) Complete the following program which, given coefficients $\mathbf{c} = (c_1, c_2, \ldots, c_n)^{\mathrm{T}}$, evaluates the corresponding polynomial at $\mathbf{x}$, which can be a number, a vector, or a square matrix. If $\mathbf{x}$ is a scalar or a vector, use *Horner's method*[1]; if $\mathbf{x}$ is a square matrix, use the result from the previous part.

```
function y = mypolyval(c, x)
%MYPOLYVAL evaluates a polynomial at points  x  given its coeffs.
% Input:
%   c   coefficient vector (c_1, c_2, ..., c_n)^T
%   x   points of evaluation
%        - if x is a scalar or a vector, use Horner's method
%        - if x is a square matrix, use the result from (a)
%        - otherwise, produce an error message.
```

```
K = [1 2; 3 4]                  %test matrix
```

```
K = 2×2
     1     2
     3     4
```

```
c = [2 3 4 5].^2;               %coefficient vector
x = [1 2 3 4 5 6 7];            %test vector
l = 2;                          %test scalar
y = mypolyval(c, K);
```

```
ans = 1×2
     2     1
```

```
z = flip(polyvalm(c, K));
p = mypolyval(c, x);
k = flip(polyval(c,x));
n = mypolyval(c, l);
m = polyval(c,l);
```

Problem 3.

3. (Singular values by hand) Calculate the singular values of

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix}$$

by solving a 2×2 eigenvalue problem.

3.) calculate singular values of

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix}$$

by solving a 2×2 eigenvalue problem.

We know that all matrices have an SVD so $A = U\Sigma V^*$ and $B = A^*A$

- $B \in \mathbb{C}^{n \times n}$ is a hermitian matrix, $B^* = B$
- B has an EVD

$$B = A^*A$$
$$= (U\Sigma V^*)^*(U\Sigma V^*)$$
$$= V\Sigma^* \underbrace{U^*U}_{\hookrightarrow I} \Sigma V^*$$
$$= V\Sigma^T\Sigma V^{-1} = VDV$$

-The squares of singular values of A are eigenvalues of B.

$$A^* = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

$$B = A^* \cdot A = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \qquad B - \lambda \cdot I = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = 0$$

$$\det\left(\begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix}\right) = 0$$

$$(2-\lambda)^2 - (1 \cdot 1) = 0$$

$$4 - 4\lambda + \lambda^2 - 1 = 0$$

$$\lambda^2 - 4\lambda + 3 = 0$$

$$\lambda = \frac{4 \pm \sqrt{16 - 4(1)(3)}}{2(1)} = \frac{4 \pm 2}{2} = 3, 1$$

We know that the eigenvalues are the squared values of the singular values of matrix A so we need to

$$\sqrt{3} = \sqrt{3}$$

$$\sqrt{1} = 1$$

$$\boxed{\text{Singular values of } A = \sqrt{3} \text{ and } 1}$$

Problem 4.

4. (SVD and the 2-norm) ✏ Let $A \in \mathbb{R}^{n \times n}$. Show that

(a) A and A^{T} have the same singular values.

(b) $\|A\|_2 = \left\|A^{\mathrm{T}}\right\|_2$.

4a.) Let $A \in \mathbb{R}^{n \times n}$ show that A and A^T have the same singular values.

Let $A = U \Sigma V^T$ be the SVD of A.

then

$$A^T = (U \Sigma V^T)^T = V \Sigma^T U^T$$

Σ are the diagonal entries of the singular values of A

$$\begin{bmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ 0 & & & \sigma_n \end{bmatrix}$$

Since we have a savore matrix, we know that a sauare diagonal matrix is symmetric which means that the transpose of a square diagonal matrix is itself

$B^T = B$

therefore Σ^T is the same as Σ

$$\begin{bmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ 0 & & & \sigma_n \end{bmatrix}$$

Which means the A and A^T have the same singular values.

b.) Show that
$\|A\|_2 = \|A^T\|_2$
We know from the properties of SVD and theorem 1 that
$\|A\|_2 = \sigma_1$

by the fact established in the previous part we know that σ_1 of A is the same as σ_1 of A^T since $\Sigma = \Sigma^T$ meaning they have the same singular values which means that
$\|A^T\|_2 = \sigma_1$

$$\boxed{\text{which means that } \|A\|_2 = \|A^T\|_2}$$

Problem 5.

5. (Vandermonde matrix, SVD, and rank) Let $\mathbf{x}$ be a vector of 1000 equally spaced points between 0 and 1, and let A_n be the $1000 \times n$ Vandermonde-type matrix whose (i, j) entry is x_i^{j-1} for $j = 1, \ldots, n$.

   (a) Print out the singular values of A_1, A_2, and A_3.

   (b) Make a semi-log plot of the singular values of A_{25}.

   (c) Use `rank` to find the rank of A_{25}. How does this relate to the graph from part (b)? You may want to use the help document for the `rank` command to understand what it does.

Part a.)

```
format long g
x = linspace(0, 1, 1000)';
n = 1;
A = x .^ (0:n-1);
```

```
s = svd(A)
```

```
s =
        31.6227766016838
```

```
n = 2;
A = x .^ (0:n-1);
s = svd(A)
```

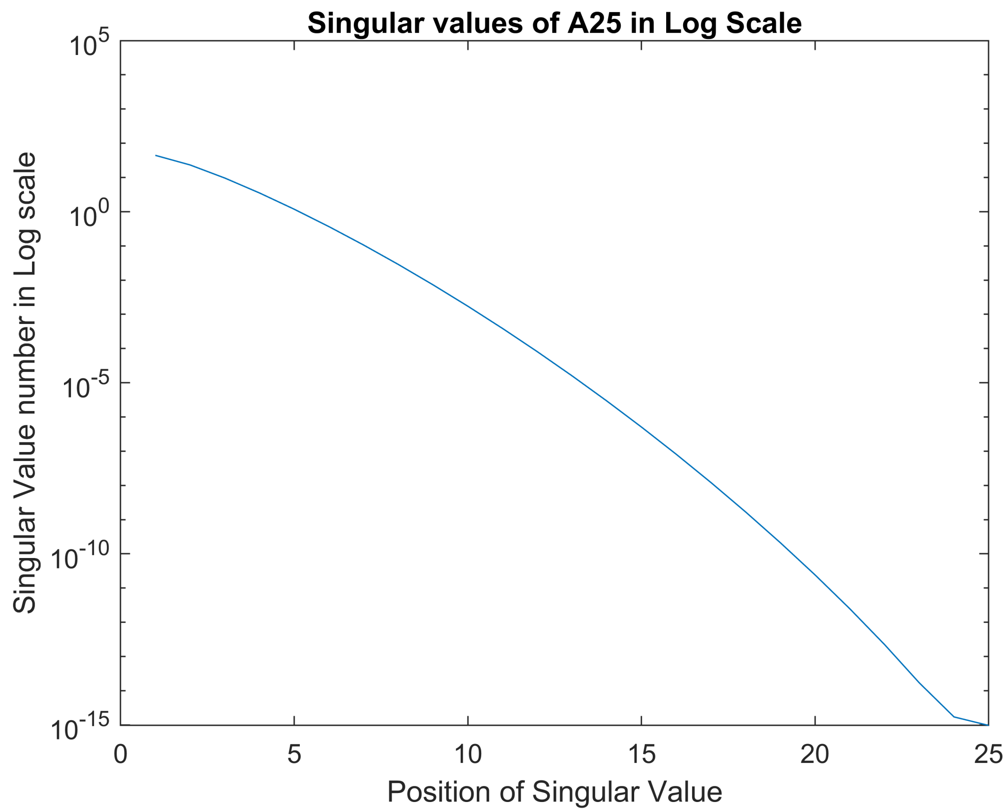
```
s = 2×1
        35.6037782926897
        8.11610362911264
```

```
n = 3;
A = x .^ (0:n-1);
s = svd(A)
```

```
s = 3×1
        37.5306448362094
        11.0703713596197
        1.64258230810186
```

Part b.)

```
n = 25;
A = x .^ (0:n-1);
s = svd(A);
semilogy(1:25, s)
xlabel('Position of Singular Value')
ylabel('Singular Value number in Log scale')
title('Singular values of A25 in Log Scale')
```



Part c.)

```
rank(A)
```

```
ans =
    20
```

```
givenTol = max(size(A)) * eps(norm(A))
```

```
givenTol =
       7.105427357601e-12
```

As we see used in the documentation of the rank command it is checking to see the number of singular values of A larger than a tolerance which if you don't provide a tolerance, it provides one for you which is the calculation you see above under the variable givenTol. This helps us to understand why rank of A is 20 since if we look at the graph there are values that are below 7.12x10^-12 which means that they are not considered to be within the tolerance that is given in the rank command, which means that some of the values are just considered 0 and since the rank is meant to produce all nonzero singular values since those last few are considered to be 0 then it just thinks that there actually 20 singular values when in reality there are more as you get to much smaller numbers in the vector of all of the singular values. So we see that there can be some error in the rank command if you try to get singular values that are smaller than the given tolerance put into the rank command.

Function mypolyval

```
function y = mypolyval(c, x)
%MYPOLYVAL evaluates a polynomial at points x given its coeffs.
% Input:
% c coefficient vector (c_1, c_2, ..., c_n)^T
% x points of evaluation
% - if x is a scalar or a vector, use Horner’s method
% - if x is a square matrix, use the result from (a)
% - otherwise, produce an error message

sizeOfX = size(x);

if sizeOfX(:,1) == 1 || sizeOfX(:,2) == 1
    n = max(size(c));
    for i = 1:max(size(x))
        y(i) = c(n);          % This algorithm is called Horner's rule.
        for j = n-1:-1:1
            y(i) = y(i)*x(i) + c(j);
        end
    end
elseif sizeOfX(:,1) > 1 && sizeOfX(:,2) > 1
    [V, ~] = eig(x);
    lambda = eig(x);
    size(lambda)
    changedLambda = mypolyval(c,lambda);
    y = V * diag(changedLambda) / V;
else
    fprintf('Invalid Syntax for function, please produce a scalar, vector, or matrix')
```

```
end
end
```