smsa2

February 11, 2020

1 Lecture 2: Introduction to Linear Algebra and MATLAB

Sound and Music Signal Analysis, SMC8, Aalborg University, 2020

By Jesper Kjær Nielsen (jkn@create.aau.dk), Audio Analysis Lab, Aalborg University.

Last edited: 2020-02-11

Table of Contents

1.1 Notation

An N-dimensional **column** vector $\boldsymbol{v}$ is written as

$$\boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix}$$

where the v_n's are all scalers.

An $N \times M$ matrix $\boldsymbol{A}$ is written as

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix}$$

where the a_{nm}'s are all scalers.

Note that

- an N-dimensional column vector is an $N \times 1$ matrix
- an $N \times M$ matrix contains M N-dimensional column vectors $\{\boldsymbol{a}_m\}_{m=1}^{M}$, i.e.,

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_M \end{bmatrix} .$$

- instead of writing $N \times M$ matrix $\boldsymbol{A}$, people often write $\boldsymbol{A} \in \mathbb{R}^{N \times M}$ instead where $\mathbb{R}$ describes that all entries are **real-valued** scalars ($\mathbb{C}$ is used for complex-valued scalars).
- if
  - $N < M$, the matrix is said to be **fat**
  - $N = M$, the matrix is said to be **square**
  - $N > M$, the matrix is said to be **skinny**
- please avoid using **row** vectors if you are already using column vectors. It will only confuse the reader (and yourself)!

[1]:
```
% a column vector in MATLAB (separate entries with semicolons)
v1 = [1; 2; 3]
```

```
v1 =

     1
     2
     3
```

[2]:
```
% a row vector in MATLAB (separate entries with commas or spaces)
v2 = [1, 2, 3]
```

```
v2 =

     1     2     3
```

[3]:
```
% a NxM matrix in MATLAB
A = [
    11, 12, 13;
```

```
    21, 22, 23;
    31, 32, 33;
    41, 42, 43;
]
```

```
A =

    11    12    13
    21    22    23
    31    32    33
    41    42    43
```

[4]:
```
% what size is my matrix?
size(v1)
size(v2)
size(A)
```

```
ans =

     3     1


ans =

     1     3


ans =

     4     3
```

Example: writing a finite data set as a vector Assume that we observe $x(n)$ for $n = 0, 1, \ldots, N-1$.

How can we write this as a vector?

We simply stack the observations as

$$\boldsymbol{x} = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}.$$

[5]:
```
[audioSignal, samplingFreq] = audioread('data/roy.wav');
soundsc(audioSignal, samplingFreq)
size(audioSignal)
```

```
ans =

       20480           1
```

1.2 Basic operations

1.2.1 Matrix addition and subtraction

You add/subtract two matrices $\boldsymbol{A} \in \mathbb{C}^{N\times M}$ and $\boldsymbol{B} \in \mathbb{C}^{N\times M}$ by doing adding/subtracting all entries elementwise, i.e.,

$$\boldsymbol{A} \pm \boldsymbol{B} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix} \pm \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1M} \\ b_{21} & b_{22} & \cdots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & \cdots & b_{NM} \end{bmatrix} \tag{1}$$

$$= \begin{bmatrix} a_{11} \pm b_{11} & a_{12} \pm b_{12} & \cdots & a_{1M} \pm b_{1M} \\ a_{21} \pm b_{21} & a_{22} \pm b_{22} & \cdots & a_{2M} \pm b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} \pm b_{N1} & a_{N2} \pm b_{N2} & \cdots & a_{NM} \pm b_{NM} \end{bmatrix} \tag{2}$$

[6]:
```
% define a new matrix B to the negative of A
B = -A;
Z = A+B
```

```
Z =

     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

Some basic rules of matrix additions and subtractions are

$$\boldsymbol{A} + (\boldsymbol{B} + \boldsymbol{C}) = (\boldsymbol{A} + \boldsymbol{B}) + \boldsymbol{C} \tag{3}$$

$$\boldsymbol{A} + \boldsymbol{B} = \boldsymbol{B} + \boldsymbol{A} \tag{4}$$

$$\boldsymbol{A} - \boldsymbol{A} = \boldsymbol{0} \tag{5}$$

where $\mathbf{0}$ is the zero-matrix given by

$$\mathbf{0} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} .$$

Therefore, you can treat matrices as normal scalars when you add and subtract them.

1.2.2 Matrix multiplication

If we multiply a matrix $\boldsymbol{A} \in \mathbb{R}^{N \times M}$ by a scalar c, we obtain

$$c\boldsymbol{A} = c \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix} = \begin{bmatrix} ca_{11} & ca_{12} & \cdots & ca_{1M} \\ ca_{21} & ca_{22} & \cdots & ca_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ ca_{N1} & ca_{N2} & \cdots & ca_{NM} \end{bmatrix} .$$

Thus, we simply multiply every entry in $\boldsymbol{A}$ with c.

Some basic rules for multiplying a scalar with a matrix are

$$\begin{aligned} c\boldsymbol{A} &= \boldsymbol{A}c && (6) \\ (c_1 c_2)\boldsymbol{A} &= c_1(c_2\boldsymbol{A}) = c_2(c_1\boldsymbol{A}) && (7) \\ 1\boldsymbol{A} &= \boldsymbol{A} && (8) \\ (c_1 \pm c_2)\boldsymbol{A} &= c_1\boldsymbol{A} \pm c_2\boldsymbol{A} && (9) \\ c(\boldsymbol{A} \pm \boldsymbol{B}) &= c\boldsymbol{A} \pm c\boldsymbol{B} . && (10) \end{aligned}$$

Active break: writing a phasor of finite duration as a vector We would like to write

$$x(n) = A\mathrm{e}^{j(\omega_0 n + \phi)}$$

for $n = 0, 1, \ldots, N-1$ in matrix form.

First, we write $x(n)$ as

$$x(n) = A\mathrm{e}^{j\phi}\mathrm{e}^{j\omega_0 n} .$$

Let us now define $c = A\mathrm{e}^{j\phi}$ and

$$\boldsymbol{x} = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} \qquad \boldsymbol{z}(\omega_0) = \begin{bmatrix} 1 \\ \mathrm{e}^{j\omega_0} \\ \vdots \\ \mathrm{e}^{j\omega_0(N-1)} \end{bmatrix} . \tag{11}$$

Then,

$$\boldsymbol{x} = c\boldsymbol{z}(\omega_0) .$$

You multiply two matrices $\boldsymbol{A} \in \mathbb{R}^{N \times K}$ and $\boldsymbol{B} \in \mathbb{R}^{K \times M}$ by

$$\boldsymbol{AB} = \begin{bmatrix} a_{11} & \cdots & a_{1K} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NK} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1M} \\ \vdots & \ddots & \vdots \\ b_{K1} & \cdots & b_{KM} \end{bmatrix} \tag{12}$$

$$= \begin{bmatrix} \sum_{k=1}^{K} a_{1k} b_{k1} & \cdots & \sum_{k=1}^{K} a_{1k} b_{kM} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{K} a_{Nk} b_{k1} & \cdots & \sum_{k=1}^{K} a_{Nk} b_{kM} \end{bmatrix}. \tag{13}$$

Note that

- matrix multiplication is **not** elementwise multiplication
- dimensions have to fit, i.e., the number of columns in $\boldsymbol{A}$ and the number of rows in $\boldsymbol{B}$ **must** be the same.

A very important matrix is the **identity matrix** which is a **square** $N \times N$ matrix given by

$$\boldsymbol{I}_N = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

The identity matrix corresponds to a '1' in the scalar case since

$$\boldsymbol{I}_N \boldsymbol{A} = \boldsymbol{A} \tag{14}$$

$$\boldsymbol{A} \boldsymbol{I}_M = \boldsymbol{A} \tag{15}$$

when $\boldsymbol{A} \in \mathbb{R}^{N \times M}$.

[7]:
```
A = [ 1, 2;
     -2, 1];
B = [ 2, 1;
      3, 2];
% construct the identity matrix
I = eye(2);
% matrix multiplication
A*B
I*A*I
% elementwise multiplication
A.*B
```

```
ans =

     8     5
```

```
    -1     0


ans =

     1     2
    -2     1


ans =

     2     2
    -6     2
```

Some basic rules for multiplying matrices (assuming that the dimensions fit) are

$$(c\boldsymbol{A})\boldsymbol{B} = \boldsymbol{A}(c\boldsymbol{B}) = c(\boldsymbol{A}\boldsymbol{B}) \tag{16}$$
$$\boldsymbol{A}(\boldsymbol{B}+\boldsymbol{C}) = \boldsymbol{A}\boldsymbol{B} + \boldsymbol{A}\boldsymbol{C} \tag{17}$$
$$(\boldsymbol{A}+\boldsymbol{B})\boldsymbol{C} = \boldsymbol{A}\boldsymbol{C} + \boldsymbol{B}\boldsymbol{C} \tag{18}$$
$$(\boldsymbol{A}\boldsymbol{B})\boldsymbol{C} = \boldsymbol{A}(\boldsymbol{B}\boldsymbol{C}) \tag{19}$$
$$\boldsymbol{A}\boldsymbol{B} \neq \boldsymbol{B}\boldsymbol{A}\ . \tag{20}$$

1.2.3 (Hermitian) transpose of a matrix

The transpose $(\cdot)^T$ of a matrix $\boldsymbol{A} \in \mathbb{C}^{N\times M}$ flips the row and column indices, i.e.,

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix} \tag{21}$$

$$\boldsymbol{A}^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{N1} \\ a_{12} & a_{22} & \cdots & a_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1M} & a_{2M} & \cdots & a_{NM} \end{bmatrix}\ . \tag{22}$$

The **Hermitian** transpose $(\cdot)^H$ of a matrix $\boldsymbol{A} \in \mathbb{C}^{N\times M}$ takes the transpose **and** the complex conjugate of all elements, i.e.,

$$\boldsymbol{A}^H = \left(\boldsymbol{A}^T\right)^* = \begin{bmatrix} a_{11}^* & a_{21}^* & \cdots & a_{N1}^* \\ a_{12}^* & a_{22}^* & \cdots & a_{N2}^* \\ \vdots & \vdots & \ddots & \vdots \\ a_{1M}^* & a_{2M}^* & \cdots & a_{NM}^* \end{bmatrix}\ . \tag{23}$$

Note that

- taking the transpose and Hermitian transpose are equivalent if all elements in $\boldsymbol{A}$ are real-valued

[8]:
```
A = [11, 12;
     21, 22;
     31, 32;];
Z = A+1i*A
% compute the transpose and Hermitian transpose of Z
Z.'
Z'
```

```
Z =

  11.0000 +11.0000i  12.0000 +12.0000i
  21.0000 +21.0000i  22.0000 +22.0000i
  31.0000 +31.0000i  32.0000 +32.0000i


ans =

  11.0000 +11.0000i  21.0000 +21.0000i  31.0000 +31.0000i
  12.0000 +12.0000i  22.0000 +22.0000i  32.0000 +32.0000i


ans =

  11.0000 -11.0000i  21.0000 -21.0000i  31.0000 -31.0000i
  12.0000 -12.0000i  22.0000 -22.0000i  32.0000 -32.0000i
```

Some basic rules for the Hermitian/transpose of a matrix are

$$(\boldsymbol{A}^T)^* = \boldsymbol{A}^H \tag{24}$$
$$(\boldsymbol{A}^H)^H = \boldsymbol{A} \tag{25}$$
$$(\boldsymbol{A}+\boldsymbol{B})^H = \boldsymbol{A}^H + \boldsymbol{B}^H \tag{26}$$
$$(c\boldsymbol{A})^H = c^*\boldsymbol{A}^H \tag{27}$$
$$(\boldsymbol{A}\boldsymbol{B})^H = \boldsymbol{B}^H\boldsymbol{A}^H \tag{28}$$
$$(\boldsymbol{A}^H)^T = (\boldsymbol{A}^T)^H = \boldsymbol{A}^* \ . \tag{29}$$

1.2.4 Matrix Inversion

An $N \times N$ matrix $\boldsymbol{A}$ is invertible, if another $N \times N$ matrix $\boldsymbol{B}$ exists such that

$$\boldsymbol{A}\boldsymbol{B} = \boldsymbol{B}\boldsymbol{A} = \boldsymbol{I}_N \ .$$

Note that

- $\boldsymbol{B}$ is the **inverse** of $\boldsymbol{A}$ and is, therefore, often denoted as $\boldsymbol{A}^{-1}$
- only **square matrices** can have an inverse (general matrices might have a pseudo-inverse)
- if $\boldsymbol{A}$ has **full rank**, then it has an inverse. A matrix has full rank if, e.g.,
  - all its columns are linearly independent
  - all eigenvalues are non-zero

[9]:
```
% create a matrix with random numbers (generated from a normal distribution)
A = randn(4,4);
% compute the inverse
iA = inv(A)
% check if it is the inverse (AB must be the indentity matrix)
A*iA
```

```
iA =

    0.3639    0.0816   -0.5806   -0.7618
    0.3303   -0.5651   -0.3999   -0.0516
   -0.0567    0.0400    0.1883    0.4436
    1.2434   -0.0094   -0.3227   -1.6010


ans =

    1.0000    0.0000    0.0000         0
    0.0000    1.0000   -0.0000   -0.0000
   -0.0000    0.0000    1.0000         0
   -0.0000   -0.0000    0.0000    1.0000
```

Some basic rules for the matrix inverse are

$$(\boldsymbol{A}^{-1})^{-1} = \boldsymbol{A} \tag{30}$$
$$(\boldsymbol{A}\boldsymbol{B})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1} \tag{31}$$
$$(\boldsymbol{A}^{H})^{-1} = (\boldsymbol{A}^{-1})^{H} \tag{32}$$
$$(c\boldsymbol{A})^{-1} = c^{-1}\boldsymbol{A}^{-1} \tag{33}$$
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - cb}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} . \tag{34}$$

To be true, the above rules require that the inverses exist. For the last rule, this means that $ad \neq cb$.

Example: the DFT as a matrix operation The N-point discrete Fourier transform (DFT) of a signal $x(n)$ can be written as

$$X(k) = \sum_{n=0}^{N-1} x(n)\mathrm{e}^{-j2\pi kn/N}$$

for $k = 0, 1, \ldots, N-1$. We would like to

1. write $X(k)$ as multiplication of a data vector $\boldsymbol{x}$ and a phasor vector $\boldsymbol{f}_k$
2. write $X(k)$ for $k = 0, 1, \ldots, N-1$ as a matrix-vector product

1. If we define

$$\boldsymbol{x} = \begin{bmatrix} x(0) & x(1) & \cdots & x(N-1) \end{bmatrix}^T \tag{35}$$

$$\boldsymbol{f}_k = \begin{bmatrix} 1 & \mathrm{e}^{-j2\pi k/N} & \cdots & \mathrm{e}^{-j2\pi k(N-1)/N} \end{bmatrix}^T , \tag{36}$$

we can write $X(k)$ as the inner product

$$X(k) = \boldsymbol{f}_k^T \boldsymbol{x} .$$

2. If we define the matrix

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{f}_0^T \\ \boldsymbol{f}_1^T \\ \vdots \\ \boldsymbol{f}_{N-1}^T \end{bmatrix} ,$$

we get

$$\boldsymbol{X} = \boldsymbol{F}\boldsymbol{x}$$

where

$$\boldsymbol{X} = \begin{bmatrix} X(0) & X(1) & \cdots & X(N-1) \end{bmatrix}^T .$$

[10]:
```
N = 4;
x = randn(N,1);
F = dftmtx(N);
X1 = F*x
X2 = fft(x)
```

```
X1 =

   4.1918 + 0.0000i
  -1.5332 - 0.0725i
  -1.6220 + 0.0000i
  -1.5332 + 0.0725i


X2 =

   4.1918 + 0.0000i
```

```
-1.5332 - 0.0725i
-1.6220 + 0.0000i
-1.5332 + 0.0725i
```

1.3 Structured matrices

Matrices with structure are extremely important since such as structure - can say something about the properties of the matrix - can be exploited in fast algorithm

1.3.1 Symmetric and Hermitian matrix

A **symmetric** matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ satisfies that

$$\boldsymbol{A}^T = \boldsymbol{A} \ .$$

That is, the entries in $\boldsymbol{A}$ all satisfy

$$a_{nm} = a_{mn} \ .$$

A **Hermitian** matrix $\boldsymbol{Z} \in \mathbb{C}^{N \times N}$ satisfies that

$$\boldsymbol{Z}^H = \boldsymbol{Z} \ .$$

That is, the entries in $\boldsymbol{Z}$ satisfy

$$z_{nm} = z_{mn}^* \ .$$

Note that a symmetric/Hermitian matrix is always a square matrix.

1.3.2 Toeplitz matrix

A **Toeplitz** matrix is characterised by that all diagonals of the matrix. As an example, a square Toeplitz matrix $\boldsymbol{T} \in \mathbb{R}^{N \times N}$ can be written as

$$\boldsymbol{T} = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \cdots & \cdots & t_{-(N-1)} \\ t_1 & t_0 & t_{-1} & \ddots & & \vdots \\ t_2 & t_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & t_{-1} & t_{-2} \\ \vdots & & \ddots & t_1 & t_0 & t_{-1} \\ t_{N-1} & \cdots & \cdots & t_2 & t_1 & t_0 \end{bmatrix}$$

Note that

- a Toeplitz matrix does not have to be square
- is completely specified from the elements in the first column and row

[11]:
```
% we specify a Toeplitz matrix via the first column and row of the matrix
T1 = toeplitz([0;1;2],[0,-1,-2,-3])
% if only the first argument is given, a Hermitian Toeplitz matrix is given
T2 = toeplitz([0,1+1i*2])
```

```
T1 =

     0    -1    -2    -3
     1     0    -1    -2
     2     1     0    -1


T2 =

   0.0000 + 0.0000i   1.0000 + 2.0000i
   1.0000 - 2.0000i   0.0000 + 0.0000i
```

1.3.3 Circulant matrix

A **circulant** matrix $\boldsymbol{C} \in \mathbb{R}^{N\times N}$ is a special Toeplitz matrix in which the entries in a column is down-shifted by one compared to the previous column, i.e.,

$$\boldsymbol{C} = \begin{bmatrix} c_0 & c_{N-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{N-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{N-2} & & \ddots & \ddots & c_{N-1} \\ c_{N-1} & c_{N-2} & \cdots & c_1 & c_0 \end{bmatrix}$$

Note that

- $\boldsymbol{C}^T$ is also a **circulant** matrix
- problems involving circulant matrices can usually be solved extremely efficiently using **an FFT** algorithm (more on this later)

[12]:
```
% create a circulant matrix. Note that you specify the first row in MATLAB - not␣
↪column.
% Therefore, the output is transposed
C = gallery('circul',[0;1;2;3]).'
```

```
C =

     0     3     2     1
     1     0     3     2
     2     1     0     3
```

```
3     2     1     0
```

1.4 Bonus: The eigenvalue decomposition

A square matrix $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ matrix has an **eigenvalue decomposition** (EVD) if it can be written as

$$\boldsymbol{A}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Lambda}$$

or, equivalently,

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}$$

where

- $\boldsymbol{U} \in \mathbb{C}^{N\times N}$ is a full rank matrix containing the **eigenvectors** as column vectors
- $\boldsymbol{\Lambda} \in \mathbb{C}^{N\times N}$ is a diagonal matrix containing the **eigenvalues**.

Note that not all square matrices have an EVD. Matrices without an EVD are said to be **defective**.

The EVD is useful for many things such as computing matrix powers since

$$\begin{aligned}
\boldsymbol{A} &= \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1} && (37)\\
\boldsymbol{A}^2 &= \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1} = \boldsymbol{U}\boldsymbol{\Lambda}^2\boldsymbol{U}^{-1} && (38)\\
\boldsymbol{A}^n &= \boldsymbol{U}\boldsymbol{\Lambda}^n\boldsymbol{U}^{-1} && (39)
\end{aligned}$$

[13]:
```
% compute a random matrix
A = randn(3,3);
% compute the EVD
[U, L] = eig(A)
```

```
U =

   0.6464 + 0.0000i   0.6464 + 0.0000i  -0.3540 + 0.0000i
  -0.1335 + 0.5998i  -0.1335 - 0.5998i  -0.2115 + 0.0000i
   0.4488 - 0.0556i   0.4488 + 0.0556i   0.9110 + 0.0000i


L =

   0.8397 + 1.4503i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.8397 - 1.4503i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.2251 + 0.0000i
```

1.4.1 The EVD of special matrices

- **Symmetric matrices**: All symmetric matrices have an eigenvalue decomposition. Moreover, all eigenvalues are **real-valued** and $\boldsymbol{U}$ is an orthogonal matrix, i.e.,

  $$\boldsymbol{U}^{-1} = \boldsymbol{U}^T$$

  so that

  $$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \ .$$

- **Circulant matrices**: All circulant matrices have an eigenvalue decomposition. Moreover,

  $$\boldsymbol{U} = \boldsymbol{F}^{-1} = N^{-1}\boldsymbol{F}^H \tag{40}$$
  $$\boldsymbol{\Lambda} = \operatorname{diag}(\boldsymbol{F}\boldsymbol{c}) \tag{41}$$

  where
  - $\boldsymbol{F}$ is the **DFT matrix** whose $(k+1, n+1)$'th element is given by

    $$[\boldsymbol{F}]_{k+1,n+1} = \mathrm{e}^{-j2\pi nk/N}$$

    for $n, k = 0, 2, \ldots, N-1$.
  - $\boldsymbol{c}$ is the first column of $\boldsymbol{C}$.

Consequently, the EVD of a circulant matrix $\boldsymbol{C} \in \mathbb{C}^{N\times N}$ is

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1} = N^{-1}\boldsymbol{F}^H \operatorname{diag}(\boldsymbol{F}\boldsymbol{c})\boldsymbol{F} \ .$$

We say that the circulant matrix is **diagonalised** by the DFT. Since the DFT can be computed efficiently using an FFT, this enables the use of fast algorithms in many situations.