SIS Scuola Interdipartimentale delle **Scienze**, dell'**Ingegneria** e della **Salute**

DiST Dipartimento di Scienze e Tecnologie

L. Magistrale in IA (ML&BD)

Scientific Computing (part 2 – 6 credits)

prof. Mariarosaria Rizzardi

Centro Direzionale di Napoli – Bldg. C4
room: n. 423 – North Side, 4th floor
phone: 081 547 6545
email: mariarosaria.rizzardi@uniparthenope.it

Contents

- Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT).
- Idea of FFT Algorithm.
- Main properties of the DFT and applications.

Discrete Fourier Transform (DFT) and its Inverse

DEFINITION: $\underline{F} = DFT(\underline{f}) = \Omega_N \cdot \underline{f}$ matrix form of DFT

$$F_k = \sum_{j=0}^{N-1} f_j \, e^{-\frac{2\pi i}{N} kj} = \sum_{j=0}^{N-1} f_j \, \omega_N^{kj}, \quad k=0,1,\ldots,N-1$$ scalar form of DFT

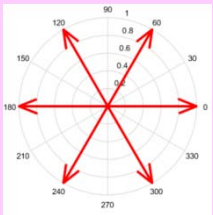
where Ω_N is the square matrix $\Omega_N = \left(\omega_N^{kj}\right)_{k,j=0,1,\ldots,N-1}$ whose elements are the powers of $\omega_N = e^{-i\frac{2\pi}{N}}$

(ω_N is a primitive* N^{th} root of unity)

(* see SC2_01b)

* A primitive N^{th} root of unity z is a N^{th} root of unity such that the powers $\{(z)^k\}_{k=0,\ldots,N-1}$ provide all the N^{th} roots of unity: $\zeta^N - 1 = 0$

$$\zeta = \sqrt[N]{1} = \sqrt[N]{[\rho = 1,\ \theta = 0]} = \left[1, \frac{2\pi}{N} k\right]_{k=0,1,\ldots,N-1} = \left\{e^{ik\frac{2\pi}{N}}\right\}_{k=0,1,\ldots,N-1}$$



$N=6$

Inverse Discrete Fourier Transform (IDFT)

DEFINITION: $\underline{f} = IDFT(\underline{F}) = \Omega_N^{-1} \underline{F}$ matrix form of IDFT

$$f_j = \frac{1}{N} \sum_{k=0}^{N-1} F_k e^{+\frac{2\pi i}{N} kj}, \quad j=0,1,\ldots,N-1$$ scalar form of IDFT

where Ω_N^{-1} is given by:

complex conjugate matrix (since Ω_N is symmetric)

(simple to compute) $$\Omega_N^{-1} = \frac{1}{N}\left(\Omega_N\right)^H = \frac{1}{N}\overline{\Omega}_N = \frac{1}{N}\left(\omega_N^{-kj}\right)_{k,j=0,1,\ldots,N-1}$$

Examples

$$\underline{\mathbf{F}} = \mathrm{DFT}\left(\underline{\mathbf{f}}\right) = \Omega_N \underline{\mathbf{f}}$$

$$\Omega_N = \left[\left(\omega_N\right)^{kj}\right]_{k,j=0,1,\ldots,N-1} \quad \text{dove} \quad \omega_N = e^{-i\frac{2\pi}{N}}$$

$i^0 = 1$
$i^1 = i$
$i^2 = -1$
$i^3 = -i$

these 4 values are repeated continuously for the periodicity of i^p

N=2 $\omega_2 = e^{-\pi i} = -1$

$$\Omega_2 = \left[\left(-1\right)^{hk}\right]_{h,k=0,1} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

N=4 $\omega_4 = e^{-i\frac{\pi}{2}} = -i$

$$\Omega_4 = \left(-i^{hk}\right)_{h,k=0,1,2,3} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & (-i)^2 & (-i)^3 \\ 1 & (-i)^2 & (-i)^4 & (-i)^6 \\ 1 & (-i)^3 & (-i)^6 & (-i)^9 \end{pmatrix}$$

Ω_N is symmetric

ω_N is a primitive N[th] root of 1

Ω_N contains, as elements, all the N^{th} roots of unity

```
syms f0 f1; f=[f0;f1];
N=2; w=exp(sym(-2*pi*i/N))
w =
-1
k=0:N-1; W=w.^(k'*k); disp(W)
[ 1,  1]
[ 1, -1]
F=W*f
F =
 f0 + f1
 f0 - f1
```

ω_N

DFT

```
N=4; w=exp(sym(-2*pi*i/N))
w =
-i
k=0:N-1; W=w.^(k'*k)
W =
[ 1,  1,  1,  1]
[ 1, -i, -1,  i]
[ 1, -1,  1, -1]
[ 1,  i, -1, -i]
```

ω_N

Main property of Ω_N: its inverse matrix can be found immediately!

... without computing an inverse matrix

```
N=…; w=exp(sym(-2*pi*i/N)); k=0:N-1; W=w.^(k'*k);
WW=simplify(W*W), D=simplify(W'*W)
```

N=2

```
WW =
[2, 0]
[0, 2]
D =
[2, 0]
[0, 2]
```

N=3

```
WW =
[3, 0, 0]
[0, 0, 3]
[0, 3, 0]
D =
[3, 0, 0]
[0, 3, 0]
[0, 0, 3]
```

N=4

```
WW =
[4, 0, 0, 0]
[0, 0, 0, 4]
[0, 0, 4, 0]
[0, 4, 0, 0]
D =
[4, 0, 0, 0]
[0, 4, 0, 0]
[0, 0, 4, 0]
[0, 0, 0, 4]
```

N=5

```
WW =
[5, 0, 0, 0, 0]
[0, 0, 0, 0, 5]
[0, 0, 0, 5, 0]
[0, 0, 5, 0, 0]
[0, 5, 0, 0, 0]
D =
[5, 0, 0, 0, 0]
[0, 5, 0, 0, 0]
[0, 0, 5, 0, 0]
[0, 0, 0, 5, 0]
[0, 0, 0, 0, 5]
```

$$\Omega_N^{-1} = \frac{1}{N}\left(\Omega_N\right)^{\mathrm{H}} = \frac{1}{N}\bar{\Omega}_N = \frac{1}{N}\left(\omega_N^{-kj}\right)_{k,j=0,1,\ldots,N-1}$$

H stays for "complex conjugate transpose" of the matrix

since the matrix is symmetric, $(\Omega_N)^{\mathrm{H}}$ is only the complex conjugate matrix

Examples

$$\underline{\mathbf{f}} = \mathrm{IDFT}\left(\underline{\mathbf{F}}\right) = \Omega_N^{-1}\underline{\mathbf{F}}$$

$$\Omega_N^{-1} = \frac{1}{N}\left(\Omega_N\right)^{\mathrm{H}} = \frac{1}{N}\overline{\Omega}_N = \frac{1}{N}\left(\omega_N^{-kj}\right)_{k,j=0,1,\ldots,N-1}$$

N=2 $\omega_2 = e^{-\pi i} = -1$

$$\Omega_2 = \left[\left(-1\right)^{hk}\right]_{h,k=0,1} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\Omega_2^{-1} = \frac{1}{2}\left[\left(-1\right)^{hk}\right]_{h,k=0,1} = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

the complex conjugate of a real number is iiself

N=4 $\omega_4 = e^{-i\frac{\pi}{2}} = -i$

$$\Omega_4 = \left(-i^{hk}\right)_{h,k=0,1,2,3} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}$$

$$\Omega_4^{-1} = \frac{1}{4}\left[\left(\overline{-i}\right)^{hk}\right]_{h,k=0,1,2,3} = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

complex conjugate $\overline{-i} = +i$

```
syms f0 f1; f=[f0;f1];
N=2; w=exp(sym(-2*pi*i/N));
k=0:N-1; W=w.^(k'*k);
disp([W   N*inv(W)])
[ 1,  1,     1,  1]
[ 1, -1,     1, -1]
F=W*f; disp(1/2*W*F)  <-- IDFT
 f0
 f1
```

```
syms f0 f1 f2 f3; f=[f0;f1;f2;f3];
N=4; w=exp(sym(-2*pi*i/N));
k=0:N-1; W=w.^(k'*k);
disp([conj(W)          N*inv(W)])
[ 1,   1,  1,   1,    1,   1,  1,   1]
[ 1,  1i, -1, -1i,    1,  1i, -1, -1i]
[ 1,  -1,  1,  -1,    1,  -1,  1,  -1]
[ 1, -1i, -1,  1i,    1, -1i, -1,  1i]
F=W*f; all(simplify(1/N*conj(W)*F == f))
ans = logical
      1
```

Periodicity of the DFT_N

$$F_k = \sum_{j=0}^{N-1} f_j \, e^{-\frac{2\pi i}{N} kj} = \sum_{j=0}^{N-1} f_j \, \omega_N^{kj}, \qquad k = 0,1,\ldots,N-1$$

$$F_{k\pm hN} = \sum_{j=0}^{N-1} f_j \, e^{-\frac{2\pi i}{N}(k\pm hN)j} = \sum_{j=0}^{N-1} f_j \, e^{-\frac{2\pi i}{N} kj} \, e^{-\frac{2\pi i}{\cancel{N}}(\pm h\cancel{N})j} = \sum_{j=0}^{N-1} f_j \, e^{-\frac{2\pi i}{N} kj} \, e^{\mp 2\pi i h} =$$

$$= \sum_{j=0}^{N-1} f_j \, e^{-\frac{2\pi i}{N} kj} = F_k$$

$e^{\mp 2\pi i h} = 1$

⇒ The DFT_N is periodic with period N

$$\underline{\mathbf{f}} = (f_0, f_1, f_2, \ldots, f_{N-1})^T \Rightarrow$$

$$\underline{\mathbf{F}} = (\ldots, F_0, F_1, F_2, \ldots, F_{N-1}, F_0, F_1, F_2, \ldots, F_{N-1}, F_0, F_1, F_2, \ldots, F_{N-1}, \ldots)^T$$

DFT and IDFT in MATLAB

In *MATLAB* a **DFT** is computed by **`fft(…)`** and an **IDFT** is computed by **`ifft(…)`**:

```
f=[-2 -1 0 1 2]';
F=fft(f)
F =         0
   -2.5000 + 3.4410i
   -2.5000 + 0.8123i
   -2.5000 - 0.8123i
   -2.5000 - 3.4410i
ifft(F)
ans =
   -2.0000
   -1.0000
    0.0000
    1.0000
    2.0000
```

```
disp(k'*k)
     0     0     0     0
     0     1     2     3
     0     2     4     6
     0     3     6     9
```

```
disp(mod(k'*k,N))
     0     0     0     0
     0     1     2     3
     0     2     0     2
     0     3     2     1
```

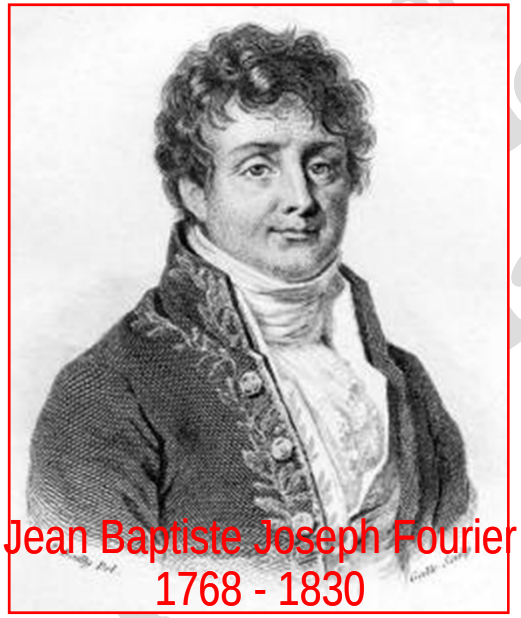
```
N=4;  f=[1:N]';
k=0:N-1; w=-i;
W4a=w.^(k'*k);
W4b=w.^mod(k'*k,N);
disp([W4a*f    fft(f)    W4b*f])
      10+0i     10+0i     10+0i
      -2+2i     -2+2i     -2+2i
      -2+0i     -2+0i     -2+0i
      -2-2i     -2-2i     -2-2i
```

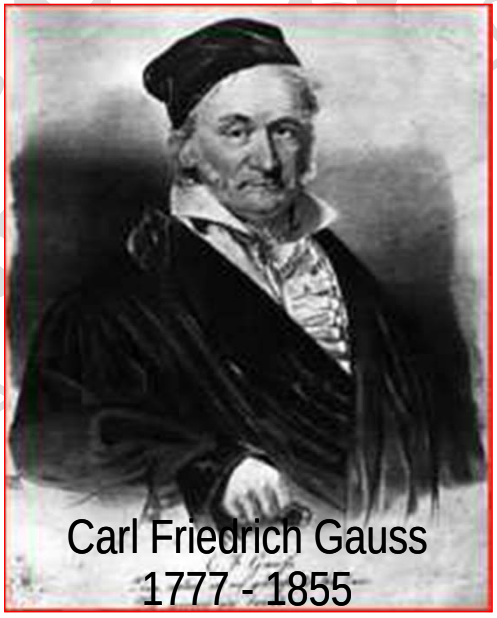
equal

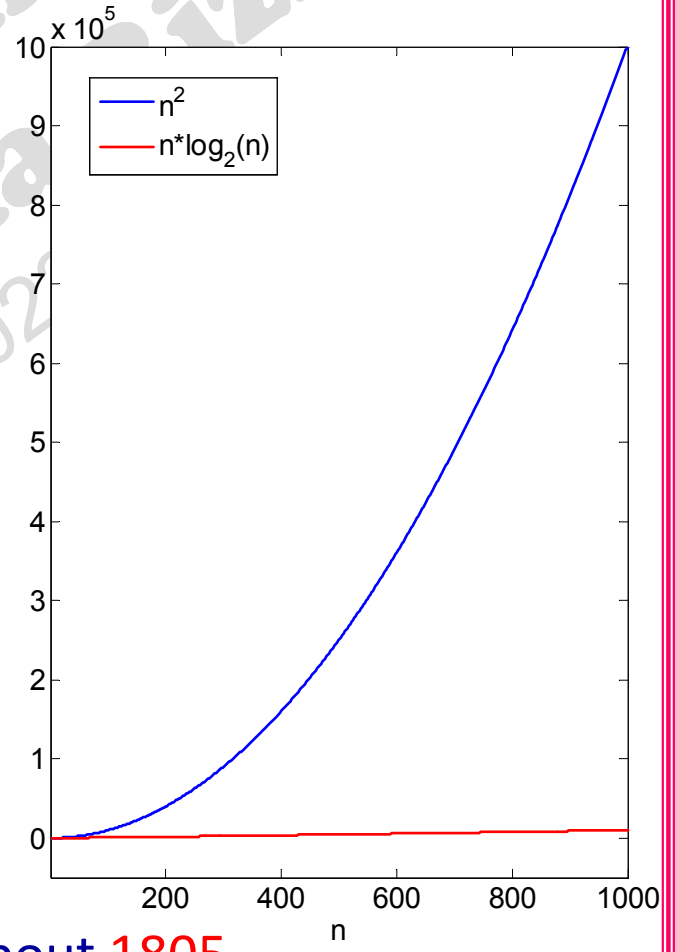
Fast Fourier Transform
FFT Algorithm (*Cooley-Tuckey*, 1965)

It reduced the computational complexity of a DFT_N from N^2 (Mat-Vec product) to $N \log_2 N$, allowing wide use of Fourier-based mathematical methods in the numerical environment.

There is currently a family of FFT algorithms: each of them is applicable to particular data and architectures. All mathematical software libraries offer one or more FFT routines (**CUDA cuFFT**, **C/C++ FFTW**).


Jean Baptiste Joseph Fourier
1768 - 1830


Carl Friedrich Gauss
1777 - 1855



This algorithm was already known to Gauss in about 1805.

FFT: Divide and Conquer algorithm

Idea for N=4 (power of 2)

```
N=2; w2=exp(sym(-2*pi*i/N)); k2=(0:N-1)'; W2=w2.^(k2*k2');   Ω2

%% Divide et Impera (Divide and Conquer)
% 1) decomposition into 2 problems of size N/2
p=[f0;f2]; % even-index components
d=[f1;f3]; % odd-index components

% 2) resolution of 2 subproblems of size N/2
DFTp=W2*p; % DFT(2)
DFTd=W2*d; % DFT(2)

% 3) combination of the previous results
G = [DFTp + (w4.^k2).*DFTd;
     DFTp - (w4.^k2).*DFTd];
```

Θ(N)

2T(N/2)

Θ(N)

```
%% compare G e F4
N=4; w4=exp(sym(-2*pi*i/N)); k4=(0:N-1)'; W4=w4.^(k4*k4');
syms f0 f1 f2 f3; f=[f0;f1;f2;f3]; F4=W4*f; % DFT(4)      Ω4
all(G == F4) % all equal components?
ans =
  logical
   1
```

(N=2^p the best case) $T(N)=2T(N/2)+\Theta(N)$ (recurrent formula)

solution ⇒ $T(N)=\Theta(N\log_2 N)$ computational complexity

More on the DFT matrix Ω_N

```
N=4; w=exp(sym(-2i*pi/N));
k=0:N-1; W=w.^mod(k'*k,N);
disp(cond(W))
1
disp(norm(W)/sqrt(N))
1
disp((W/sqrt(N))^4)
[1, 0, 0, 0]
[0, 1, 0, 0]
[0, 0, 1, 0]
[0, 0, 0, 1]
```

well-conditioned

the matrix leaves the length of a vector unchanged

Ω_N

```
N=8; w=exp(sym(-2i*pi/N));
k=0:N-1; W=w.^mod(k'*k,N);
disp(cond(W))
1
disp(norm(W)/sqrt(N))
1
all(all(round((W/sqrt(N))^4) == eye(N)))
ans =
  logical
   1
```

```
disp(roots(charpoly(W/sqrt(N))))
 -1
  1
  1
 -1i
```

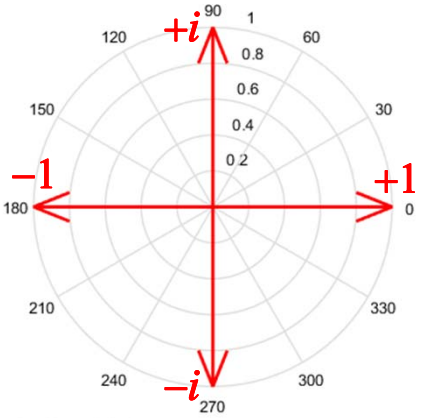
$\nu(+1)=2$
$\nu(-1)=1$
$\nu(-i)=1$

```
format short; disp(eig(double(W)/sqrt(N))); format short g
   1.0000 - 0.0000i    1
  -1.0000 - 0.0000i   -1
   1.0000 + 0.0000i    1
   0.0000 - 1.0000i   -i
  -1.0000 - 0.0000i   -1
  -0.0000 + 1.0000i    i
   1.0000 + 0.0000i    1
   0.0000 - 1.0000i   -i
```

$\nu(+1)=3$
$\nu(-1)=2$
$\nu(+i)=1$
$\nu(-i)=2$

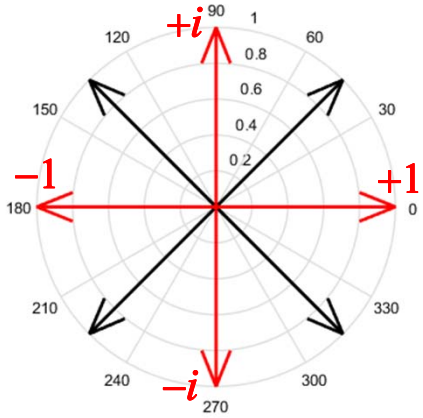
algebraic multiplicity

The eigenvalues of Ω_N are the quartic roots of unity: $-1, +1, -i, +i$ (also with multiplicity > 1)



N=4

N^{th} roots of unity



N=8

Since the matrix Ω_N is symmetric, why are its eigenvalues complex?

Main Properties of the DFT

Let $\mathscr{F}_N$ be the map which transforms a vector $\underline{\mathbf{f}} \in \mathbb{C}^N$ into its DFT vector:

$$\mathscr{F}_N : \underline{\mathbf{f}} \in \mathbb{C}^N \longrightarrow \mathscr{F}_N[\underline{\mathbf{f}}] = \underline{\mathbf{F}} \in \mathbb{C}^N$$

- $\mathscr{F}_N$ is an *invertible linear map*.
- If $\underline{\mathbf{f}}$ is *real*, then $\underline{\mathbf{F}}$ is *hermitian symmetric*, i.e. $\mathbf{F}_j = \overline{\mathbf{F}_{(N-j) \bmod N}}$, and conversely if $\underline{\mathbf{f}}$ is *symmetric hermitian*, then $\underline{\mathbf{F}}$ is *real*.
- If $\underline{\mathbf{f}}$ is *imaginary*, then $\underline{\mathbf{F}}$ is *hermitian anti-symmetric*, i.e. $\mathbf{F}_j = -\overline{\mathbf{F}_{(N-j) \bmod N}}$ and conversely if $\underline{\mathbf{f}}$ is *anti-symmetric hermitian*, then $\underline{\mathbf{F}}$ is *imaginary*.
- *Circular shift*: let $\underline{\mathbf{f}}^{[\pm h]}$ be the vector obtained from $\underline{\mathbf{f}}$ by shifting its components of $\pm h$ places, $h \in \mathbb{N}$, i.e. $\mathbf{f}_j^{[\pm h]} = \mathbf{f}_{(j \pm h) \bmod N}$, then

each components $\mathbf{F}_k$ is rotated by $2\pi/\mathrm{N}(\mp h)k$ $\quad \mathbf{F}^{[\pm h]}{}_k = e^{2\pi i/\mathrm{N}(\mp h)k}\, \mathbf{F}_k$ where $\underline{\mathbf{F}} = \mathscr{F}_N[\underline{\mathbf{f}}]$ and $\underline{\mathbf{F}}^{[\pm h]} = \mathscr{F}_N[\underline{\mathbf{f}}^{[\pm h]}]$.

- *Circular convolution*: if $\boxed{.*}$ denotes the *Hadamard product* and $\boxed{\circledast}$ the *circular convolution* of two vectors, then

$$\mathscr{F}_N[\underline{\mathbf{f}} \boxed{.*} \underline{\mathbf{g}}] = \mathscr{F}_N[\underline{\mathbf{f}}] \circledast \mathscr{F}_N[\underline{\mathbf{g}}] \quad \text{and} \quad \boxed{\mathscr{F}_N[\underline{\mathbf{f}} \circledast \underline{\mathbf{g}}] = \mathscr{F}_N[\underline{\mathbf{f}}] \boxed{.*} \mathscr{F}_N[\underline{\mathbf{g}}]}$$

more important

- *Parseval Equality*: $\left\|\underline{\mathbf{F}}\right\|_2^2 = N \left\|\underline{\mathbf{f}}\right\|_2^2$

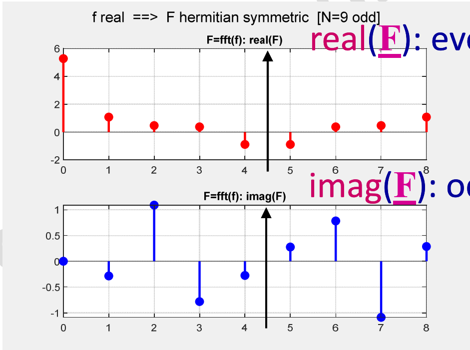
DFT Properties: examples

f real ⇒ **F** hermitian symmetric i.e. real(**F**) is even and imag(**F**) is odd

$$\mathbf{F}_j = \overline{\mathbf{F}_{(N-j) \bmod N}} \qquad j=0,\ldots,N-1$$

```
f=rand(9,1); F=fft(f)
F =
   3.1201 +      0i        real
  0.32241 -   0.92876i
  0.50997 -    0.1572i
   0.2268 -   0.31177i
  0.72486 +    1.0463i
  0.72486 -    1.0463i
   0.2268 +   0.31177i
  0.50997 +    0.1572i
  0.32241 +   0.92876i
```

complex conjugate




```
f=rand(8,1); F=fft(f)
F =
   4.3162 +      0i        real
  0.49473 -   0.67916i
   1.0416 +   0.45693i
 -0.13001 -   0.98218i
 -0.61102 +      0i        real
 -0.13001 +   0.98218i
   1.0416 -   0.45693i
  0.49473 +   0.67916i
```

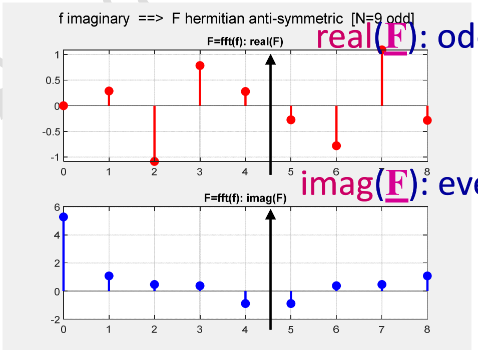
complex conjugate

f imaginary ⇒ **F** hermitian anti-symmetric i.e. real(**F**) is odd and imag(**F**) is even

$$\mathbf{F}_j = -\overline{\mathbf{F}_{(N-j) \bmod N}} \qquad j=0,\ldots,N-1$$

```
f=1i*rand(9,1); F=fft(f)
F =
        0 +    5.2737i     imaginary
  0.28598 +    1.0773i
  -1.0881 +  0.468825i
  0.78202 +  0.37306i
  0.27534 -  0.88918i
 -0.27534 -  0.88918i
 -0.78202 +  0.37306i
   1.0881 +  0.468825i
 -0.28598 +    1.0773i
```

opposite complex conjugate




```
f=1i*rand(8,1); F=fft(f)
F =
        0 +    4.3162i     imaginary
  0.67916 +   0.49473i
 -0.45693 +    1.0416i
  0.98218 -   0.13001i
        0 -   0.61102i     imaginary
 -0.98218 -   0.13001i
  0.45693 +    1.0416i
 -0.67916 +   0.49473i
```

opposite complex conjugate

Circular shift property: example

$$\mathbf{f}_j^{[\pm h]} = \mathbf{f}_{(j\pm h) \bmod N} \Rightarrow \underline{\mathbf{F}}^{[\pm h]}{}_k = e^{2\pi i/\mathrm{N}(\mp h)k}\, \mathbf{F}_k$$

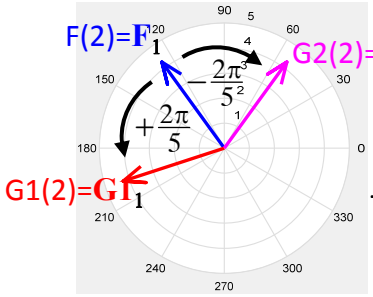
```
f=(1:5)'; N=numel(f); K=(0:N-1)'; g1=circshift(f,[-1 0]); g2=circshift(f,[+1 0]);
ColNames1={'k index','g1=circshift(f,[-1 0])','f vector', 'g2=circshift(f,[+1 0])'};
Tab1=table(K,g1,f,g2,'VariableNames',ColNames1); disp(Tab1)
F=fft(f); G1=fft(g1); G2=fft(g2); J=2; compass(F(J),'b'); hold on; compass(G1(J),'r')
compass(G2(J),'m')
```

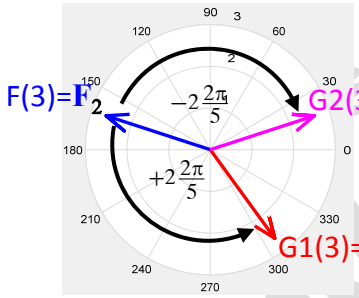
k index	g1=circshift(f,[-1 0])	f vector	g2=circshift(f,[+1 0])
0	2	1	5
1	3	2	1
2	4	3	2
3	5	4	3
4	1	5	4

-1 on rows = up by 1

+1 one rows = down by 1

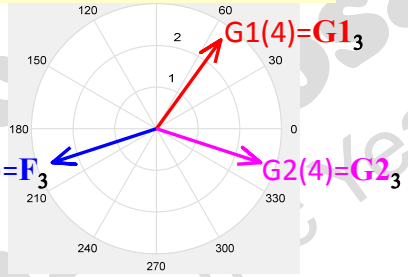
complex numbers in polar coordinates



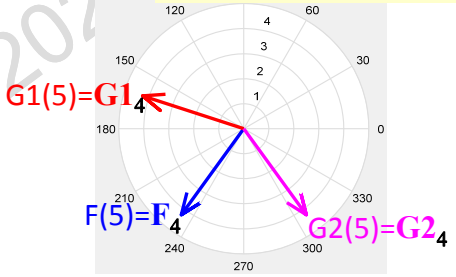


the DFT indices start from 0, but in MATLAB they start from 1

$$\mathbf{g1}_{(j-1) \bmod N} = \mathbf{f}_j \quad h=-1$$



$$\mathbf{g2}_{(j+1) \bmod N} = \mathbf{f}_j \quad h=+1$$

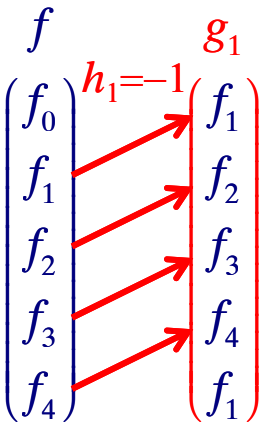



```
ColNames2={'k index','k*2*pi/N','h=-1 ==> unwrap(angle(G1)-angle(F))', 'h=+1 ==> unwrap(angle(G2)-angle(F))'};
a=2*pi/N; Tab2=table(K,K*a,unwrap(angle(G1)-angle(F)), ...
                     unwrap(angle(G2)-angle(F)),'VariableNames',ColNames2); disp(Tab2)
```

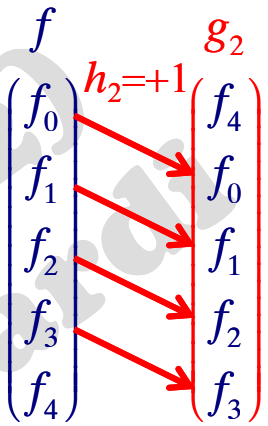
k index	k*2*pi/N	h=-1 ==> unwrap(angle(G1)-angle(F))	h=+1 ==> unwrap(angle(G2)-angle(F))
0	0	0	0
1	1.2566	1.2566	-1.2566
2	2.5133	2.5133	-2.5133
3	3.7699	3.7699	-3.7699
4	5.0265	5.0265	-5.0265

$\theta = -2\pi hk/\mathrm{N}$ $\theta = -2\pi hk/\mathrm{N}$

Circular shift property: example



$$\mathbf{f}_j^{[\pm h]} = \mathbf{f}_{(j\pm h) \bmod N} \Rightarrow \underline{\mathbf{F}}^{[\pm h]}{}_k = e^{2\pi i/\mathrm{N}(\mp h)k}\, \mathbf{F}_k$$




```
N=5; K=(0:N-1)'; f=(1:5)';
h1=-1; g1=circshift(f,[h1 0]);
h2=+1; g2=circshift(f,[h2 0]);
F=fft(f); G1=fft(g1); G2=fft(g2);
fprintf('\ncompare arguments of DFT(f) to arguments of DFT(g1) [in radians]\n')
disp([angle(F(K+1).*exp(-2i*pi/N*h1*K)) angle(G1(K+1))])
compare arguments of DFT(f) to arguments of DFT(g1) [in radians]
            0            0
      -2.8274      -2.8274
     -0.94248     -0.94248        equal
      0.94248      0.94248
       2.8274       2.8274

fprintf('\ncompare arguments of DFT(f) to arguments of DFT(g2) [in radians]\n')
disp([angle(F(K+1).*exp(-2i*pi/N*h2*K)) angle(G2(K+1))])
compare arguments of DFT(f) to arguments of DFT(g2) [in radians]
            0            0
      0.94248      0.94248
      0.31416      0.31416        equal
     -0.31416     -0.31416
     -0.94248     -0.94248
```

Circular convolution of two vectors

Cyclic or circular convolution
or modulo-N circ. convolution:

$$z = x \circledast y: \quad z_j = \sum_{h=0}^{N-1} x_h y_{[j-h]_{\bmod N}}, \quad j = 0,\ldots,N-1$$

where the sequences $\{x_h\}$ and $\{y_k\}$ are assumed periodic and of equal length N; $\{z_j\}$ will also be periodic and of length N.

For instance, from $\{x_h\}_{h=0,1,2,3}$ and $\{y_k\}_{k=0,1,2} = \{y_0, y_1, y_2, \boxed{y_3=0}\}$ (zero-pad), the element z_2 is computes as

$$z_2 = \sum \begin{array}{cccccccccccc} \ldots & -5 & -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 & \ldots \\ \ldots & x_3 & x_0 & x_1 & x_2 & x_3 & x_0 & x_1 & x_2 & x_3 & x_0 & \ldots \\ & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \\ \ldots & y_3=0 & y_2 & y_1 & y_0 & y_3=0 & y_2 & y_1 & y_0 & y_3=0 & y_2 & \ldots \end{array} \rightarrow \leftarrow$$

The following steps are executed:

- the second sequence is reversed $\{y_k\}_k$ (←);
- its element y_0 is aligned to the element x_j, of the first sequence having index equal to that of the component z_j to be computed;
- the aligned components of the two sequences are multiplied;
- these products are summed together.

```
x=[1 2 -1 1]; y=[1 2 3];
z=cconv(x,y)
z = 1     4     6     5    -1     3
```

```
y=[y 0]; % zero-padding
Z2=x(0+1)*y(2+1)+x(1+1)*y(1+1)+x(2+1)*y(0+1)+x(3+1)*y(3+1);
disp([Z2 z(2+1)])
     6     6
```

since in MATLAB indices start from 1

Convolution in MATLAB: examples

in Signal Processing Toolbox

MATLAB conv(): linear convolution and cconv(): cyclic convolution

$$z = x * y: \quad z_j = \sum_{h=0}^{m-1} x_h y_{j-h}, \quad j=0,\ldots,n+m-2$$

$$z = x \circledast y: \quad z_j = \sum_{h=0}^{N-1} x_h y_{[j-h]_{\bmod N}}, \quad j = 0,\ldots,N-1$$

```
x=[1 2 -1]; y=[1 2 3];
```

{ }: cell array

```
c=conv(x,y)
c = 1     4     6     4    -3
c=conv(x,y,'same')
c = 4     6     4
c=conv(x,y,'valid')
c = 6
c=conv(x,y,'full') % default
c = 1     4     6     4    -3
```

equal

```
cc=cconv(x,y)
cc = 1     4     6     4    -3
disp(sum(cc))
12
disp({cconv(x,y,1);cconv(x,y,2);cconv(x,y,3)})
    {[   12]}  ⇒  sum=12
    {[  4 8]}  ⇒  sum=12
    {[5 1 6]}  ⇒  sum=12
```

lenght of the output vector

Convolution properties of DFT_N

2

$$\mathcal{F}_N[\underline{f} \circledast \underline{g}] = \mathcal{F}_N[\underline{f}] \;.*\; \mathcal{F}_N[\underline{g}]$$ ⇐ more important

```
f=[3 0 4]'; g=[3 1 2]'; N=numel(f); % equal length
disp(max(abs(fft(cconv(f,g,N)) - fft(f).*fft(g))))
4.4409e-16
```

1

$$\mathcal{F}_N[\underline{f} .* \underline{g}] = \mathcal{F}_N[\underline{f}] \circledast \mathcal{F}_N[\underline{g}]$$

```
f=[3 0 4]'; g=[3 1 2]'; N=numel(f); % equal length
disp(max(abs(fft(f.*g) - cconv(fft(f),fft(g),N)/N)))
8.8818e-16
```

What is the computational cost for computing the convolution $\circledast$ of two vectors with N components ?

$$u = v \circledast w$$

$$u_j = \sum_{k=0}^{N-1} v_k w_{(j-k) \bmod N}, \qquad j = 0, 1, 2, \ldots, N-1$$

$\mathcal{O}(N^2)$

Efficient algorithm $\mathcal{O}(N \log_2 \lceil N \rceil) \ll \mathcal{O}(N^2)$

to compute $\underline{\mathbf{h}} = \underline{\mathbf{f}} \circledast \underline{\mathbf{g}}$ by means of DFT:

- $\underline{\mathbf{F}} = \mathbf{F}_N[\underline{\mathbf{f}}]$ and $\underline{\mathbf{G}} = \mathbf{F}_N[\underline{\mathbf{g}}]$; $\quad \mathcal{O}(2N \log_2 \lceil N \rceil)$
- $\underline{\mathbf{H}} = \underline{\mathbf{F}} .* \underline{\mathbf{G}}$; $\quad \mathcal{O}(N)$
- $\underline{\mathbf{h}} = \underline{\mathbf{f}} \circledast \underline{\mathbf{g}} = \mathbf{F}_N^{-1}[\underline{\mathbf{H}}]$. $\quad \mathcal{O}(N \log_2 \lceil N \rceil)$

Application:
multiply two integers by convolution in ALU

$403 \times 213 = 85839$

$f = 3 \cdot 10^0 + 0 \cdot 10^1 + 4 \cdot 10^2$
$ff = 3 \cdot 10^0 + 0 \cdot 10^1 + 4 \cdot 10^2 + 0 \cdot 10^3 + 0 \cdot 10^4$ (zero-padding)

```
f=[3 0 4]'; g=[3 1 2]';
ff=[f;0;0]; gg=[g;0;0]; % zero-padding
F=fft(ff);  G=fft(gg);
h=ifft(F.*G);
```

```
disp([cconv(f,g)  h])
     9                9
     3                3
    18=8 (reaminder) 18
  5 ← 4 ←(carry over) 4
     8                8
```

The Discrete Fourier Transform (DFT) is the numerical tool to approximate the coefficients of a Fourier Series (FS) and to approximate some samples of a Fourier Transform (FT), mathematical tools widely used in applications precisely by virtue of the existence of several “fast” algorithms to compute a DFT.