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

DiST Dipartimento di Scienze e Tecnologie

Laurea Magistrale in STN

Applicazioni di Calcolo Scientifico e Laboratorio di ACS

(12 cfu)

prof. Mariarosaria Rizzardi

Centro Direzionale di Napoli – Isola C4
stanza: n. 423 – Lato Nord, 4° piano
tel.: 081 547 6545
email: mariarosaria.rizzardi@uniparthenope.it

ACS parte 2: ACS_11a

Argomenti trattati

- **Trasformata di Fourier Discreta (DFT) e Trasformata Inversa (IDFT).**
- **Cenni sull'algoritmo FFT.**
- **Proprietà della DFT ed applicazioni.**

Cos'è una DFT ?

DFT sta per

Discrete Fourier Transform

cioè

Trasformata Discreta di Fourier

IDFT sta per

Inverse Discrete Fourier Transform

cioè

Trasformata Discreta Inversa di Fourier

Discrete Fourier Transform (DFT)

DEFINIZIONE: $\underline{\mathbf{F}}$ è la DFT di $\underline{\mathbf{f}}$ $\qquad \underline{\mathbf{F}} = \mathbf{DFT}(\underline{\mathbf{f}})$

Input: $\underline{\mathbf{f}} = (f_0, f_1, f_2, \ldots, f_{N-1})^T$ vettore reale o complesso ($\underline{\mathbf{f}} \in \mathbb{C}^N$)

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

$$\underline{\mathbf{F}} = \mathbf{DFT}(\underline{\mathbf{f}}) = \Omega_N \underline{\mathbf{f}}$$ forma matriciale DFT

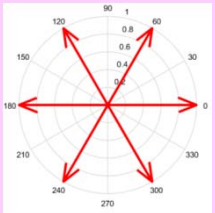
dove Ω_N è la matrice quadrata $\Omega_N = \left(\omega_N^{kj}\right)_{k,j=0,1,\ldots,N-1}$ i cui elementi sono le potenze di $\omega_N = e^{-i\frac{2\pi}{N}}$

(ω_N: radice N sima dell'unità primitiva*)

(* vedi: Parte 1 04x: Brevi note sui numeri complessi)

* Una radice N sima dell'unità primitiva z è una radice N sima di 1 tale che le potenze $\{(z)^k\}_{k=0,\ldots,N-1}$ danno tutte le radici N sime di 1 (ζ):

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



dal prodotto matrice-vettore si ha:

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

Esempi

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

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

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

questi 4 valori si ripetono per la periodicità di i^p

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

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

Ω_N è simmetrica

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 è una radice primitiva

gli elementi di Ω_N sono le N radici N^{sime} dell'unità

```
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

Principale proprietà di Ω_N: il calcolo della matrice inversa è immediato!

... e quindi non va calcolata l'inversa di una matrice

```
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 sta per "complessa coniugata della matrice trasposta"

poiché la matrice è simmetrica, $(\Omega_N)^{\mathrm{H}}$ si riduce alla sola complessa coniugata

Richiami

Inverse Discrete Fourier Transform (IDFT)

DEFINIZIONE: $\underline{\mathbf{f}}$ è la IDFT di $\underline{\mathbf{F}}$ $\quad \underline{\mathbf{F}} = \mathrm{IDFT}(\underline{\mathbf{f}})$

Input: $\underline{\mathbf{F}} = (F_0, F_1, F_2, \ldots, F_{N-1})^T$ vettore reale o complesso ($\underline{\mathbf{F}} \in \mathbb{C}^N$)

Output: $\underline{\mathbf{f}} = (f_0, f_1, f_2, \ldots, f_{N-1})^T$ ($\underline{\mathbf{f}} \in \mathbb{C}^N$)

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

forma matriciale IDFT

dove Ω_N^{-1} è l'inversa della matrice Ω_N ed è data da:

(molto facile da calcolare)

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

matrice complessa coniugata
(poiché Ω_N è simmetrica)

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

$$\bar{\omega}_N = e^{+i\frac{2\pi}{N}}$$

dal prodotto matrice-vettore:

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

forma scalare IDFT

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

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

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

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

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

il complesso coniugato di un numero reale è il numero stesso

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

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

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

complesso coniugato $\overline{-i} = +i$

```
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
```

Periodicità della DFT$_N$

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

Se si cerca di calcolare più di N componenti di una DFT$_N$

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

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

⇒ La **DFT$_N$** è periodica di periodo N

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

$$\underline{\mathbf{F}} = (\ldots, F_0, F_1, \ldots, F_{N-2}, F_{N-1}, F_0, F_1, F_2, \ldots, F_{N-1}, F_0, F_1, F_2, \ldots, F_{N-1}, \ldots)^T$$

indici …, $-N$, $1-N$, …, -2, -1, 0, 1, 2, …, $N-1$, N, $N+1$, $N+2$, …, $2N-1$, …

DFT e IDFT in MATLAB

In *MATLAB* una **DFT** si calcola mediante **fft(…)** ed una **IDFT** si calcola mediante **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
```

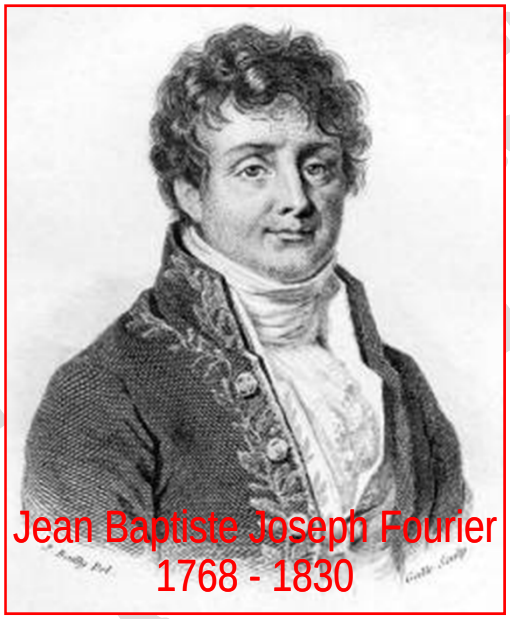
uguali!

Richiami

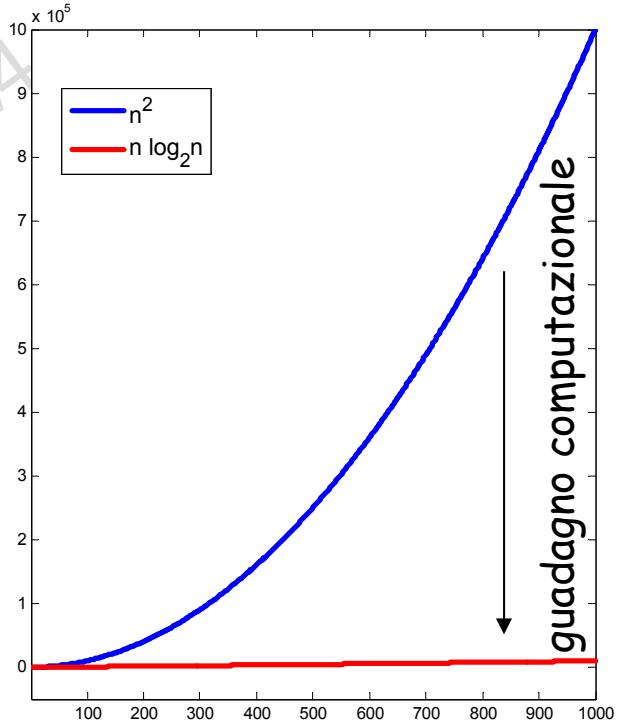
Algoritmo FFT (*Cooley-Tuckey*, 1965) Fast Fourier Transform

Ha ridotto la complessità computazionale del calcolo di una **DFT** da n^2 (prodotto Mat-Vet) a $n \cdot \log_2 n$, consentendo, in ambiente numerico, un'ampia utilizzazione dei metodi matematici basati su Fourier.

Attualmente esiste una famiglia di algoritmi FFT: ciascuno applicabile a dati (ed architetture) particolari. Molte librerie di software matematico offrono una o più routine FFT (**CUDA cuFFT, C/C++ FFTW**).


Jean Baptiste Joseph Fourier
1768 - 1830


Carl Friedrich Gauss
1777 - 1855



Questo algoritmo era già noto a Gauss all'incirca nel 1805.

Idea dell'algoritmo FFT

(Divide et Impera o Divide and Conquer)

Se $N=2M$ (N pari) allora la DFT $\underline{\mathbf{F}} = \Omega_N \underline{\mathbf{f}}$ ($\underline{\mathbf{F}} \in \mathbb{C}^N$) del vettore $\underline{\mathbf{f}}$ ($\underline{\mathbf{f}} \in \mathbb{C}^N$) può essere calcolata tramite due DFT di lunghezza $N/2$.

Separando le componenti di indici pari e quelle di indici dispari del vettore $\underline{\mathbf{f}}$ si hanno i due vettori di lunghezza $N/2=M$:

$$\underline{\mathbf{p}} = (f_0, f_2, f_4, \ldots, f_{N-2}) \quad \text{e} \quad \underline{\mathbf{d}} = (f_1, f_3, f_5, \ldots, f_{N-1})$$

indici pari — indici dispari

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

si ha

$$F_k = \sum_{j=0}^{N-1} f_j \, \omega_N^{kj} = \sum_{h=0}^{M-1} f_{2h} \, \omega_N^{k(2h)} + \sum_{h=0}^{M-1} f_{2h+1} \, \omega_N^{k(2h+1)}$$

$$\Downarrow$$

$$F_k = \sum_{h=0}^{M-1} p_h \left(\omega_N^2\right)^{kh} + \omega_N^k \sum_{h=0}^{M-1} d_h \left(\omega_N^2\right)^{kh} \qquad k = 0, 1, 2, \ldots, N-1$$

Si suddivida il vettore $\underline{\mathbf{F}}$ nei due vettori:

$$\underline{\mathbf{G}} = (\underbrace{F_0, F_1, F_2, \ldots, F_{M-1}}_{1^a \text{ metà}}) \qquad \text{e} \qquad \underline{\mathbf{H}} = (\underbrace{F_M, F_{M+1}, F_{M+2}, \ldots, F_{N-1}}_{2^a \text{ metà}})$$

ne segue che per $k = 0, 1, 2, \ldots, M-1$

$$G_k = \sum_{h=0}^{M-1} p_h \left(\omega_N^2\right)^{kh} + \omega_N^k \sum_{h=0}^{M-1} d_h \left(\omega_N^2\right)^{kh}$$

$$H_k = \sum_{h=0}^{M-1} p_h \left(\omega_N^2\right)^{(k+M)h} + \omega_N^{k+M} \sum_{h=0}^{M-1} d_h \left(\omega_N^2\right)^{(k+M)h}$$

Dall'essere $\omega_N^{k+M} = e^{-\frac{2\pi i}{N}(k+M)} = \cdots = -e^{-\frac{2\pi i}{N}k} = -\omega_N^k$

si ha

$$G_k = \sum_{h=0}^{M-1} p_h \left(\omega_N^2\right)^{kh} + \omega_N^k \sum_{h=0}^{M-1} d_h \left(\omega_N^2\right)^{kh}$$

$$H_k = \sum_{h=0}^{M-1} p_h \left(\omega_N^2\right)^{kh} - \omega_N^k \sum_{h=0}^{M-1} d_h \left(\omega_N^2\right)^{kh}$$

Se $\omega_N = e^{-\frac{2\pi i}{N}}$ è una radice N^{sima} dell'unità primitiva, cos'è ω_N^2 ?

$$\omega_N^2 = \left(e^{-\frac{2\pi i}{N}}\right)^2 = \cdots = e^{-\frac{2\pi i}{M}} = \omega_M$$

è una radice M^{sima} dell'unità primitiva

Quindi si è ottenuto per $k = 0, 1, 2, \ldots, M-1$ $\quad M = N/2$

$$G_k = \sum_{h=0}^{M-1} p_h\, \omega_M^{kh} + \omega_N^k \sum_{h=0}^{M-1} d_h\, \omega_M^{kh}$$

$$H_k = \sum_{h=0}^{M-1} p_h\, \omega_M^{kh} - \omega_N^k \sum_{h=0}^{M-1} d_h\, \omega_M^{kh}$$

2 DFT di lunghezza M

Pertanto il calcolo di una DFT di lunghezza N è stato ricondotto a due DFT di lunghezza $N/2$. Da qui l'algoritmo ricorsivo di tipo "divide et impera": (N=2^p caso migliore) compless. comp. $T(N) = 2T(N/2) + \Theta(N)$

Esercizio: implementare questa versione ricorsiva dell'algoritmo in MATLAB confrontandone i tempi con la funzione **fft()**.

Ancora sulla matrice Ω_N della DFT

```
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]
```

ben-condizionata

la matrice lascia immutata la lunghezza di un vettore

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
```

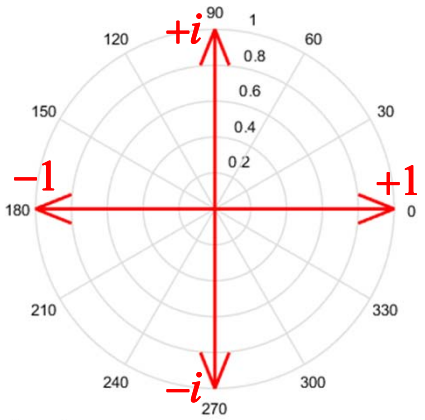
$\nu(+1)=2$
$\nu(-1)=1$
$\nu(-i)=1$

```
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
```

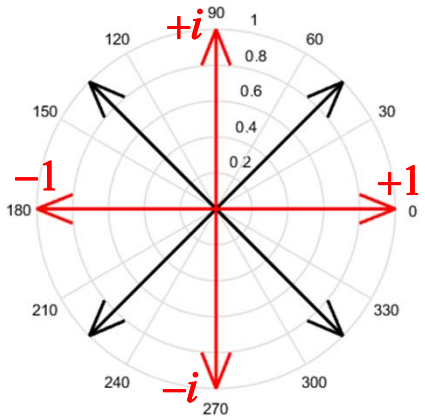
$\nu(+1)=3$
$\nu(-1)=2$
$\nu(+i)=1$
$\nu(-i)=2$

molteplicità algebrica

Gli autovalori di Ω_N sono le radici quarte di 1: $-1, +1, -i, +i$ (anche con molteplicità > 1)



radici N^{sime} dell'unità



$N=4$ $N=8$

Poiché la matrice Ω_N è simmetrica, perché i suoi autovalori sono complessi?

Principali proprietà della DFT

Sia $\mathscr{F}_N$ la trasformazione che associa ad un vettore $\underline{\mathbf{f}} \in \mathbb{C}^N$ il suo vettore DFT:

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

- $\mathscr{F}_N$ è una *trasformazione lineare invertibile*.
- Se $\underline{\mathbf{f}}$ è *reale*, allora $\underline{\mathbf{F}}$ è *hermitiano simmetrico*, cioè $\mathbf{F}_j = \overline{\mathbf{F}_{(N-j) \bmod N}}$, e viceversa se $\underline{\mathbf{f}}$ è *hermitiano simmetrico*, allora $\underline{\mathbf{F}}$ è *reale*.
- Se $\underline{\mathbf{f}}$ è *immaginario*, allora $\underline{\mathbf{F}}$ è *hermitiano antisimmetrico*, cioè $\mathbf{F}_j = -\overline{\mathbf{F}_{(N-j) \bmod N}}$ e viceversa se $\underline{\mathbf{f}}$ è *hermitiano antisimmetrico*, allora $\underline{\mathbf{F}}$ è *immaginario*.
- *Shift Circolare*: sia $\underline{\mathbf{f}}^{[\pm h]}$ il vettore ottenuto da $\underline{\mathbf{f}}$ traslando le sue componenti di $\pm h$ posti, $h \in \mathbb{N}$, cioè $\mathbf{f}_j^{[\pm h]} = \mathbf{f}_{(j \pm h) \bmod N}$, allora per la sua DFT risulta:

ogni componente $\mathbf{F}_k$ è ruotata di $2\pi/\mathrm{N}(\mp h)k$

$$\mathbf{F}^{[\pm h]}{}_k = e^{2\pi i/\mathrm{N}(\mp h)k}\, \mathbf{F}_k \quad \text{dove} \quad \underline{\mathbf{F}} = \mathscr{F}_N[\underline{\mathbf{f}}] \quad \text{e} \quad \underline{\mathbf{F}}^{[\pm h]} = \mathscr{F}_N[\underline{\mathbf{f}}^{[\pm h]}].$$

- *Convoluzione Circolare*: se $\boxed{.*}$ denota il *prodotto di Hadamard* e $\boxed{\circledast}$ la *convoluzione circolare* di due vettori, allora

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

- *Uguaglianza di Parseval* (*Teor. di Plancherel*): $\|\underline{\mathbf{F}}\|_2^2 = N \|\underline{\mathbf{f}}\|_2^2$ più importante

Proprietà della DFT: esempi

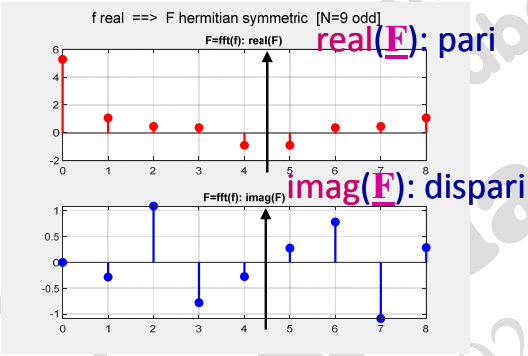
f reale $\Rightarrow$ **F** hermitiano simmetrico cioè real(**F**) è pari e imag(**F**) è dispari

$$F_j = \overline{F_{(N-j) \bmod N}} \qquad j=0,\ldots,N-1$$

```
f=rand(9,1); F=fft(f)
F =
   3.1201 +    reale   0i
  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
```

complessi coniugati

N dispari



real(**F**): pari

imag(**F**): dispari

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

complessi coniugati

N pari

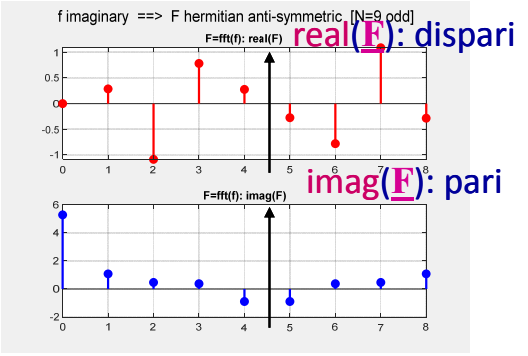
f immaginario $\Rightarrow$ **F** hermitiano antisimmetrico cioè real(**F**) è dispari e imag(**F**) è pari

$$F_j = -\overline{F_{(N-j) \bmod N}} \qquad j=0,\ldots,N-1$$

```
f=1i*rand(9,1); F=fft(f)
F =
immaginario 0 +    5.2737i
  0.28598 +    1.0773i
  -1.0881 +    0.46825i
  0.78202 +    0.37306i
  0.27534 -    0.88918i
 -0.27534 -    0.88918i
 -0.78202 +    0.37306i
   1.0881 +    0.46825i
 -0.28598 +    1.0773i
```

complessi coniugati opposti

N dispari



real(**F**): dispari

imag(**F**): pari

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

complessi coniugati opposti

N pari

Proprietà dello shift circolare: esempio

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

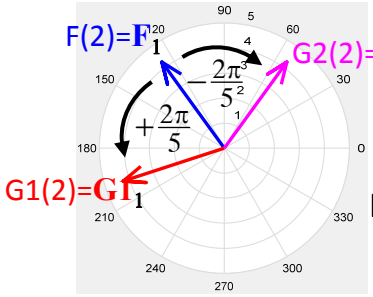
```
f=(1:5)'; N=numel(f); K=(0:N-1)'; g1=circshift(f,[-1 0]); g2=circshift(f,[+1 0]);
ColNames1={'k index','g1=circshift(f,[-1 0])','f vector','g2=circshift(f,[+1 0])'};
Tab1=table(K,g1,f,g2,'VariableNames',ColNames1); disp(Tab1)
F=fft(f); G1=fft(g1); G2=fft(g2); J=2; compass(F(J),'b'); hold on; compass(G1(J),'r')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

indice k	g1=circshift(f,[-1 0])	vettore f	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 sulle righe = sale di 1

+1 sulle righe = scende di 1

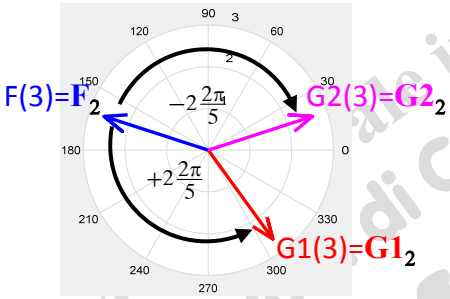
numeri complessi in coordinate polari

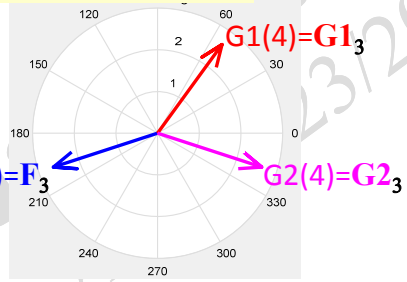


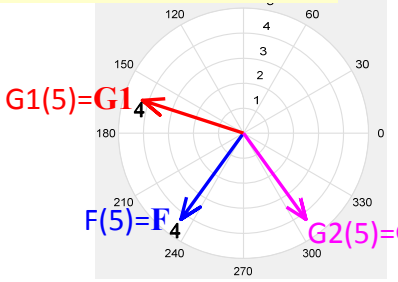
Nella DFT gli indici partono da 0, in MATLAB partono da 1

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

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





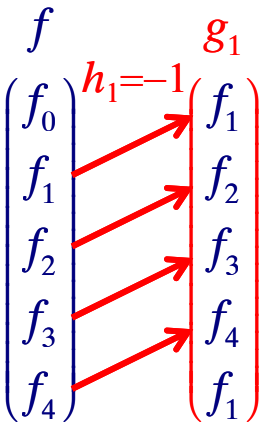



```
ColNames2={'k index','k*2*pi/N','h=-1 ==> unwrap(angle(G1)-angle(F))', 'h=+1 ==> unwrap(angle(G2)-angle(F))'};
a=2*pi/N; Tab2=table(K,K*a,unwrap(angle(G1)-angle(F)), ...
                     unwrap(angle(G2)-angle(F)),'VariableNames',ColNames2); disp(Tab2)
```

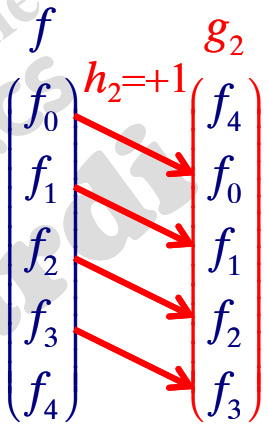
indice k	k*2*pi/N	h=-1 ==> unwrap(angle(G1)-angle(F))	h=+1 ==> unwrap(angle(G2)-angle(F))
0	0	0	0
1	1.2566	1.2566	-1.2566
2	2.5133	2.5133	-2.5133
3	3.7699	3.7699	-3.7699
4	5.0265	5.0265	-5.0265

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

Proprietà dello shift circolare: esempio



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




```
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        uguali
      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        uguali
     -0.31416     -0.31416
     -0.94248     -0.94248
```

Convoluzione circolare di due vettori

Convoluzione ciclica o circolare
o convoluzione circ. modulo-N :

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

dove le sequenze $\{x_h\}$ e $\{y_k\}$ sono assunte periodiche (cicliche) e di eguale lunghezza N; anche $\{z_j\}$ sarà periodica e di lunghezza N.

Per esempio, da $\{x_h\}_{h=0,1,2,3}$ e $\{y_k\}_{k=0,1,2}=\{y_0, y_1, y_2, \boxed{y_3=0}\}$, l'elemento z_2 è calcolato come

zero-pad

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

Vengono eseguiti i seguenti passi:

- la seconda sequenza è invertita $\{y_k\}_k$ (←);
- il suo elemento y_0 è allineato all'elemento x_j della prima sequenza con indice uguale a quello della componente z_j da calcolare;
- le componenti allineate delle due sequenze sono moltiplicate;
- questi prodotti sono sommati tra loro.

```
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
```

perché gli indici in MATLAB partono da 1

Convoluzione in MATLAB: esempi

nel Signal Processing Toolbox

MATLAB **conv()**: convoluzione lineare e **cconv()**: convoluzione ciclica

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

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

```
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
```

uguali

```
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]}  ⇒  somma=12
    {[  4 8]}  ⇒  somma=12
    {[5 1 6]}  ⇒  somma=12
```

lunghezza del vettore di output

Proprietà della convoluzione della DFT_N

2

$$\mathscr{F}_N[\underline{f} \circledast \underline{g}] = \mathscr{F}_N[\underline{f}] \,.*\, \mathscr{F}_N[\underline{g}]$$ ⇐ più importante

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

1

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

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

Applicazione della proprietà del prodotto di convoluzione

Il prodotto di convoluzione di due vettori può descrivere operazioni quali il prodotto di due polinomi ed il prodotto di due numeri naturali.

Prodotto di polinomi: esempio

Siano $p_1(x)$ e $p_2(x)$ due polinomi di grado al più 2:

$$p_1(x) = a_0 + a_1 x + a_2 x^2 \quad \text{e} \quad p_2(x) = b_0 + b_1 x + b_2 x^2 \qquad \text{3 coeff.}$$

Il polinomio prodotto $q(x) = p_1(x) \cdot p_2(x)$ è un polinomio di grado al più 4:

$$q(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 \qquad \text{5 coeff.}$$

dove i coefficienti c_k si ottengono tramite il prodotto di convoluzione

$$c = (a_0, a_1, a_2, 0, 0) \circledast (b_0, b_1, b_2, 0, 0)$$

5 coeff.

cioè

$$c_0 = a_0 b_0, \; c_1 = a_0 b_1 + a_1 b_0, \; c_2 = a_0 b_2 + a_1 b_1 + a_2 b_0, \; \ldots$$

Prodotto di numeri naturali

Analogamente per calcolare $403 \times 213 = 85839$ si esegue

$$(3,0,4,0,0) \circledast (3,1,2,0,0) = (9,3,18,4,8) \rightarrow (9,3,8,5,8)$$

$3\cdot10^0 + 0\cdot10^1 + 4\cdot10^2 + 0\cdot10^3 + 0\cdot10^4$

$3\cdot10^0 + 1\cdot10^1 + 2\cdot10^2 + 0\cdot10^3 + 0\cdot10^4$

$9\cdot10^0 + 3\cdot10^1 + 18\cdot10^2 + 4\cdot10^3 + 8\cdot10^4$

resto, riporto

$\cdots + 8\cdot10^2 + 5\cdot10^3 + \cdots$

operando nel risultato gli eventuali riporti di cifra:

```
f=[3 0 4]'; g=[3 1 2]';
c=conv(f,g); cc=cconv(f,g);
disp([c  cc])
```

```
     9     9
     3     3
    18    18
     4     4
     8     8
```

c ha lunghezza $2n-1$

```
f=[3 0 4]; g=[3 1 2]; c=conv(f,g)
c =
     9     3    18     4     8
mod(c,10)     % resti mod.10 (resti della divisione per 10)
ans =
     9     3     8     4     8
floor(c/10)  % quozienti della divisione per 10 (riporti)
ans =
     0     0     1     0     0
circshift(floor(c/10),[0 1])
ans =
     0     0     0     1     0
mod(c,10) + circshift(floor(c/10),[0 1])
ans =
     9     3     8     5     8
fliplr(mod(c,10)+circshift(floor(c/10),[0 1]))
ans =
     8     5     8     3     9
```

Quante operazioni sono richieste per calcolare la convoluzione $\circledast$ di due vettori di N componenti ?

$$u = v \circledast w$$

convoluzione di due vettori

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

$$T(N) = O(N^2)$$

Avendo a disposizione un algoritmo "veloce" per calcolare la **DFT** (***Algoritmo FFT***: $O(N \cdot \log_2 N)$), per ottenere efficientemente il prodotto di polinomi o di interi nella ALU, come prodotto di convoluzione si procede come segue. Si calcola:

1. $\underline{\mathbf{F}} = \mathscr{F}_N[\underline{\mathbf{f}}]$ e $\underline{\mathbf{G}} = \mathscr{F}_N[\underline{\mathbf{g}}]$; — 2 DFT (mediante FFT)
2. $\underline{\mathbf{H}} = \underline{\mathbf{F}} \,.*\, \underline{\mathbf{G}}$; — prodotto di Hadamard (componente x componente)
3. $\mathbf{h} = \mathbf{f} \circledast \mathbf{g} = \mathscr{F}_N^{-1}[\underline{\mathbf{H}}]$. — 1 IDFT (mediante IFFT)

per calcolare il prodotto di convoluzione $\underline{h}=\underline{f}\circledast\underline{g}$ mediante DFT (in MATLAB `fft(...)`):

1. $\underline{F}=\mathscr{F}_N[\underline{f}]$ e $\underline{G}=\mathscr{F}_N[\underline{g}]$;
2. $\underline{H}=\underline{F}$ `.*` $\underline{G}$;
3. $\underline{h}=\underline{f}\circledast\underline{g}=\mathscr{F}_N^{-1}[\underline{H}]$.

più efficiente del prodotto mat-vet

$\mathcal{O}(2N\log_2\lceil N\rceil)$
$\mathcal{O}(N)$
$\mathcal{O}(2N\log_2\lceil N\rceil)$

$403\times213=85839$

$f=3\cdot10^0+0\cdot10^1+4\cdot10^2$
$ff=3\cdot10^0+0\cdot10^1+4\cdot10^2+0\cdot10^3+0\cdot10^4$

```
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=F.*G;
h=ifft(H);
[conv(f,g) h]
ans =
```

9	9
3	3
18 = 8 + resto	18
+ riporto 4 → 5	4
8	8

$403\times213=85839$

UGUALI!

```
f=rand(3,1); g=rand(5,1);
ff=[f;zeros(numel(g)-1,1)];
gg=[g;zeros(numel(f)-1,1)];
F=fft(ff);  G=fft(gg);
h=ifft(F.*G);
[conv(f,g) h]
ans =
```

0.74415	0.74415
1.3425	1.3425
0.76824	0.76824
0.39555	0.39555
0.71021	0.71021
0.53073	0.53073
0.069447	0.069447

UGUALI!

La Trasformata Discreta di Fourier (DFT) è lo strumento numerico per approssimare i coefficienti della Serie di Fourier (FS) e per approssimare alcuni campioni della Trasformata di Fourier (FT), strumenti matematici molto utilizzati nelle applicazioni proprio in virtù dell'esistenza di vari algoritmi "veloci" per il calcolo di una DFT.