UNIVERSITÀ DEGLI STUDI DI TRIESTE

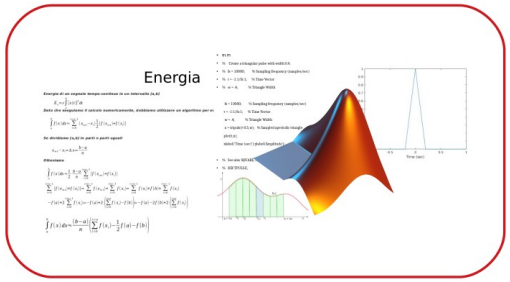
Programmazione Informatica

Ingegneria Civile e Ambientale
Ingegneria Navale

Programmazione - Matlab 3 – Funzioni, ciclo for

Antonio Fiumara
antonio.fiumara@dia.units.it

A. A. 2025-26



UNIVERSITÀ DEGLI STUDI DI TRIESTE

Struttura del Corso

Gli argomenti trattati nel corso saranno:

- Lezione Teoria 1 – Introduzione, presentazione del corso, codifica dell'informazione
- Lezione Teoria 2 – Rappresentazione dell'informazione in un computer
- Lezione Teoria 3 – Algebra booleana, introduzione e applicazioni nella programmazione
- Lezione Teoria 4 – Architettura dei calcolatori. HW, SW, Sistemi Operativi
- Lezione Teoria 5 – Algoritmi e strutture dati
- Lezione Teoria 6 – Implementazione di algoritmi notevoli (ordinamento , ricerca, ecc)

- Lezione Programmazione - Matlab 1 – Introduzione, installazione, comandi di base, matrici
- Lezione Programmazione - Matlab 2 – Input/Output (I/O), strutture di controllo
- **Lezione Programmazione - Matlab 3 – Funzioni, ciclo for**
- Lezione Programmazione - Matlab 4 – Sistemi Lineari, Algebra lineare, Analisi matematica
- Lezione Programmazione - Matlab 5 – Funzioni nel dominio del tempo e della frequenza
- Lezione Programmazione - Matlab 6 – Analisi ed elaborazione dei segnali
- Lezione Programmazione - Matlab 7 – Ripasso e approfondimenti
- Esonero esame Programmazione Matlab (3/12/25)

Introduzione

Funzioni di output
Funzioni definite dal programmatore
Ciclo for

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Esercizio

Scrivere uno script Matlab che acquisisca un vettore di numeri non nulli e calcoli la media aritmetica e la media geometrica degli elementi del vettore.

- Il vettore deve “terminare” con un valore nullo. Tale elemento nullo indica la fine dei dati e non deve essere incluso nel calcolo delle medie

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Esercizio

Scrivere uno script Matlab che acquisisca un vettore di numeri non nulli e calcoli la media aritmetica e la media geometrica degli elementi del vettore

rive/media.m

```
clear all
a=input('inserisci un vettore')
k=1
s=0
while a(k)~=0
    s=s+a(k)
     k = k + 1 % Increment k to check the next element
      a(k)~=0
end
s=s/(k-1)
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Esercizio

Scrivere uno script Matlab che acquisisca un vettore di numeri non nulli e calcoli la media aritmetica e la media geometrica degli elementi del vettore

)rive/media_for.m

```
clear all
a=input('inserisci un vettore')
k=1
s=0
for k=1:length (a)-1
    s=s+a(k)

end
s=s/k
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Ciclo for

Ciclo per ripetere un blocco di istruzioni un numero di volte specificato (e predeterminato)

- Sintassi

  for indice = valori

  Istruzioni

  .....

  end

- Valori presenta una delle seguenti forme:
- **inizio:fine** incrementa la variabile **indice** da inizio a fine di 1 e ripete l'esecuzione di istruzioni finché indice non è maggiore di finel.
- **inizio:step:fine** incrementa **indice** del valore di **step** a ciascuna iterazione o decrementa **indice** se **step** è negativo.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Funzioni

Le **funzioni** sono programmi che svolgono delle azioni definite, possono restituire dei risultati e possono elaborare dati forniti dall'esterno (parametri)

In Matlab sono disponili diverse tipologie d funzioni

- Funzioni di libreria incorporate in Matlab
- Funzioni di libreria disponibili in toolbox specializzati (da acquistare a parte)
- Funzioni definite dall'utente (programmatore)
  - funzione **anonima**
  - funzione definita in uno script Matlab utilizzando una sintassi specifica

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Funzioni anonime

La funzione anonima viene definita quando si ritiene più opportuno all'interno di uno script e non necessita di essere salvata in un file .m separato:

nome_funzione = @(x1,...,xN)(istruzione)

dove x1,...,xN sono gli N argomenti in input, (istruzione) è il corpo della funzione e dev'essere costituito da un'istruzione sola.

L'output della funzione anonima coincide quindi con il valore dell'istruzione stessa: non è quindi possibile assegnare più di un output (in maniera diretta). L'output può ovviamente essere di qualsiasi tipo

```
% Definizione
f = @(x) (1+x-x.^2) ./ (x+7*x.^2) ;

% Utilizzo (scalare)
y = ( f(1) + f(2) ) / f(3) ;

% Utilizzo (vettoriale)
v_x = linspace( 0 , 1 , 1000 ) ;
v_y = f(v_x) ;
```

```
% Definizione
f = @(x,y) sqrt( x .^ 2 + y .^ 2 ) ;

% Utilizzo (scalare)
z = f(1,2) / f(2,1) ;

% Utilizzo (matriciale)
v_x = linspace( 0 , 1 , 1000 ) ;
v_y = linspace( 0 , 1 , 1000 )' ;
v_z = f(v_x,v_y) ;
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Funzioni .m

La funzione è definita in uno script .m avente come nome (nome lo file) lo stesso nome della funzione

- N.B. il nome della funzione segue le stesse regole del nome di una variabile
- La sintassi è la seguete

function [output1, output2, …] = *nomeFunzione*(input1, input2, ...)

Corpo della funzione

……

output1 = risultato1;

output2 = risultato2;

end

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Esercizio

Determinare un'approssimazione dello zero della funzione $f(x) = \sin(2x) - x$ mediante il metodo della bisezione a partire dall'intervallo $[\pi/4, \pi/2]$.

- Definire f (x) mediante una funzione anonima.
- Con il metodo della bisezione si suddivide iterativamente un intervallo [a, b] in due sotto intervalli e si prosegue iterativamente scegliendo il sotto intervallo per il quale f (x) cambia segno. Le iterazioni terminano quando |b-a| è inferiore ad un valore (errore) prefissato.

Esercizio

```
% Definizione della funzione anonima
f = @(x) sin(2*x) - x ;

% Intervallo di partenza
ab    = [ pi/4 pi/2 ] ;

% Valori della funzione agli estremi dell'intervallo
f_ab = f( ab ) ;

% Bisezione con 50 iterazioni
for i = 1 : 50
  xm = ( ab(1) + ab(2) ) / 2 ; % punto medio dell'intervallo
  f_xm = f( xm ) ;             % f( punto medio )
  % Caso di funzione nulla nel punto medio: uscita
  if f_xm == 0
    ab = [ xm xm ] ; % assegnazione intervallo puntiforme
    f_ab = [ 0 0 ] ; % assegnazione valori nulli agli estremi
    break ;          % uscita dal ciclo for
  else
    % Nel caso di funzione non nulla nel punto medio, bisogna decidere quale
    % semi-intervallo scegliere
    if sign( f_ab(1) ) ~= sign( f_xm ) % f cambia segno nell'intervallo sx
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Esercizio

```
      % Teniamo il semi-intervallo sinistro
      ab(2)   = xm ;
      f_ab(2) = f_xm ;
    else
      % Teniamo il semi-intervallo destro
      ab(1)   = xm ;
      f_ab(1) = f_xm ;
    end
  end
end

% Output risultati
x = ( ab(1) + ab(2) ) / 2 ; % punto medio
f_x = f( x ) ;              % f( punto medio )
disp( x ) ;
disp( f_x ) ;
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Scopo delle varabili e visibilità

- Quando una funzione viene richiamata in uno script o da prompt, MATLAB ricerca i relativi file funzione .m in una lista di percorsi indicati nella variabile **path**, oltre che nell'attuale percorso di lavoro (*pwd*).
- Per chiarezza organizzativa della propria aerea di lavoro o quando i file funzione sono numerosi può essere comodo organizzare i file in sottocartelle del percorso di lavoro, che vanno aggiunte al path.
- L'aggiunta al path di tutte le sottocartelle viene eseguita tramite

  addpath (genpath (pwd)) ;

  che andrà messo in testa nello script principale.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Scopo delle varabili e visibilità

Le variabili/matrici di tutti i tipi e funzioni anonime, sono definiti in una zona di memoria detta **workspace**, accessibile dal prompt in maniera diretta (richiamando o definendo una variabile o una funzione anonima) oppure attraverso i comandi **who/whos, workspace**.

- Tutte le variabili/funzioni anonime definite nel workspace sono quindi visibili in questo contesto.
- Una funzione definita in un file funzione .m non lavora nel base workspace ma definisce uno spazio diverso e separato. Gli elementi definiti all'interno di una funzione come questa sono detti locali poichè sono visibili solo all'interno della funzione stessa.
- Una variabile definita dentro una funzione, per esempio, è una variabile locale e non è visibile nel workspace
- Allo stesso modo, una variabile definita nel workspace non è visibile all'interno della funzione. L'unico modo per scambiare dati con una funzione è farlo attraverso gli argomenti x1,...,xM in input e y1,...,yN in output.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Come scrivere una funzione

- Per scrivere una funzione, apriamo un nuovo script. Scriviamo per esempio una funzione, che chiameremo **fattoriale**, per calcolare il fattoriale di un intero utilizzando un ciclo **for**:

fattoriale.m

```
function m = fattoriale( n )
  m = 1 ;          % m è una variabile locale (l'output della funzione)
  for i = 2 : n    % i è una variabile locale
    m = m * i ;
  end
end
```

- Salveremo quindi lo script con lo stesso nome utilizzato per la funzione, quindi **fattoriale.m**.
- La funzione così definita può essere richiamata dovunque, al prompt o all'interno di altre funzioni:

```
>> fattoriale(10)
ans =
     3628800
```

coefficiente_taylor.m

```
function y = coefficiente_taylor( x )
  y = 1 / fattoriale( x ) ;
end
```

- **m,n,i** sono quindi variabili locali, visibili (e quindi utilizzabili) solo all'interno della funzione **fattoriale**. Sono quindi libero di usare **m,n,i** come nomi per altre variabili all'esterno della funzione **fattoriale**, per esempio nel base workspace o in un'altra funzione, senza preoccuparmi di eventuali conflitti.

Esercizio

- Scrivere una funzione che prenda in input due numeri n e b e fornisca in output il quoziente q ed il resto r della divisione intera di n per b:

$$n = q \cdot b + r$$

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Esercizio

- Scrivere una funzione che prenda in input due numeri n e b e fornisca in output il quoziente q ed il resto r della divisione intera di n per b:

$$n = q \cdot b + r$$

- Si hanno due numeri in output: si può scegliere di usare due argomenti in output, uno per ogni numero:

quoziente_resto.m

```
function [ q , r ] = quoziente_resto( n , b )
  r = mod( n , b ) ;
  q = ( n - r ) / b ;
end
```

oppure si possono organizzare i due numeri in unico vettore in output:

quoziente_resto_vettoriale.m

```
function qr_vettore  = quoziente_resto_vettoriale( n , b )
  qr_vettore = [ 0 0 ] ; % Preparazione vettore output
  r = mod( n , b ) ;
  qr_vettore(2) = r ;              % Assegnazione nel vettore output
  qr_vettore(1) = ( n - r ) / b ; % Assegnazione nel vettore output
end
```

Funzioni ricorsive

Le funzioni **ricorsive**, sono funzioni che **chiamano** se stesse. Ogni nuova chiamata alla funzione definisce uno spazio diverso per gli elementi locali (variabili, funzioni, ecc.).

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Funzioni ricorsive - esempio

Scrivere una funzione che calcoli il fattoriale di un numero n in maniera ricorsiva:

$$n! = n(n-1)(n-2)\cdot\cdot\cdot 2\cdot 1 = n(n-1)!$$

fattoriale_ricorsivo.m

```
function fatt = fattoriale_ricorsivo( n )
  if n > 1
    fatt = n * fattoriale_ricorsivo( n-1 ) ;
  else
    fatt = 1 ;
  end
end
```

- Nel caso della precedente funzione fattoriale_ricorsivo, ogni sua chiamata ricorsiva predispone delle nuove variabili locali n e fatt diverse dalle chiamate precedenti, anche se il nome è evidentemente lo stesso.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Funzioni ricorsive - esercizio

- Scrivere una funzione che prenda in input un vettore di coefficienti $\mathbf{a} = (a_0, a_1, \ldots, a_n)$ ed un vettore $\mathbf{x}$ e calcoli in maniera ricorsiva il polinomio

$$\begin{aligned} p(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \\ &= a_0 + x(a_1 + x(\cdots a_{n-1} + x(a_n))), \end{aligned}$$

  sfruttando quindi l'ultima forma, per tutti i valori di x del vettore $\mathbf{x}$.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Funzioni ricorsive - esercizio

- Scrivere una funzione che prenda in input un vettore di coefficienti $\mathbf{a} = (a_0, a_1, \ldots, a_n)$ ed un vettore $\mathbf{x}$ e calcoli in maniera ricorsiva il polinomio

$$\begin{aligned} p(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \\ &= a_0 + x(a_1 + x(\cdots a_{n-1} + x(a_n))), \end{aligned}$$

sfruttando quindi l'ultima forma, per tutti i valori di x del vettore $\mathbf{x}$.

polinomio_ricorsivo.m

```
function p = polinomio_ricorsivo(a, x, k)
  n = length(a) - 1 ;
  if k < n
    p = a(k+1) + x .* polinomio_ricorsivo(a, x, k+1) ;
  else
    p = a(n+1) + 0*x ;
  end
end
```

- La chiamata alla funzione dovrà quindi inizare da $k = 0$:

```
p_x = polinomio_ricorsivo(coefficienti, vettore_x, 0) ;
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Funzioni ricorsive - esercizio

- Scrivere una funzione che prenda in input un intero N e calcoli in maniera vettoriale la seguente somma:

$$e_N = \sum_{i=0}^{N} \frac{1}{i!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{N!}$$

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Funzioni ricorsive - esercizio

- Scrivere una funzione che prenda in input un intero N e calcoli in maniera vettoriale la seguente somma:

$$e_N = \sum_{i=0}^{N} \frac{1}{i!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{N!}$$

somma_eN.m

```
function eN = somma_eN(N)
  i = 1 : N ;
  fattoriale_i = [ 1 cumprod( i ) ] ;
  eN = sum( 1 ./ fattoriale_i ) ;
end
```

- Si visualizzi poi la differenza $\Delta = e - e_N$ per $N = 5, 10, 15, 20$:

UNIVERSITÀ DEGLI STUDI DI TRIESTE

input

- I dati in input e output in uno script possono essere forniti mediante opportune funzioni di input/output (I/O), che permettono:
- input e output formattati nel prompt;
- output grafici (plot);
- lettura o scrittura su file.
- La funzione **input** può essere utilizzata per acquisire un valore da tastiera attraverso il prompt. Il valore inserito può essere di qualsiasi tipo visto finora (numerico, logico, testuale), scalare o matriciale, anche in forma di espressione complessa:
- **nome_variabile** = **input**(messaggio)
- che mostra il testo riportato nel messaggio e assegna il valore inserito da
- tastiera, dopo aver premuto il tasto Invio, nella variabile **nome_variabile**:

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Input

```
>> H = input( 'Inserire altezza [metri]: ' ) ;
>> Inserire altezza [m]: 1.85
>> H
    1.8500

>> v = input( 'Inserire voti [/30]: ' ) ;
>> Inserire voti [/30]: [ 28 25 18 33 30 ]
>> v
    28    25    18    33    30
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

disp , fprintf

- La funzione **disp** può essere utilizzata per visualizzare nel prompt una variabile di qualsiasi tipo, ed equivale a richiamare una variabile direttamente nel prompt:

```
>> v = [ 1.5 2.6 3.7 ] ;
>> disp( v )
    1.5000    2.6000    3.7000
```

```
>> A = [ 1 2 3   ; ...
         4 5 6.5 ] ;
>> disp( A )
    1.0000    2.0000    3.0000
    4.0000    5.0000    6.5000
```

- La funzione **disp** non permette un output formattato, che è invece possibile ottenere con la funzione **fprintf**:

  `fprintf( formato , x1 , ... , xN )`

  dove `x1 , ... , xN` sono le `N` variabili da visualizzare e **formato** specifica la formattazione da dare all'output mediante gli operatori di formattazione:

  `%(flags)(n_caratteri).(precisione)(conversione)`

Tipo	(conversione)	Dettagli
intero	d	base 10
intero senza segno	u, o, x, X	basi 10 (u), 8 (o), 16 (x,X)
non intero	f, e, E	posizionale (f), esponenziale (e,E)
testo	c, s	caratteri (c) e stringhe (s)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

fprintf

- (conversione), esempi:

```
>> fprintf( '%d' , 99 ) ;
99
>> fprintf( '%o' , 63 ) ;
77
>> fprintf( '%X' , 255 ) ;
FF
```

```
>> fprintf( '%f' , pi ) ;
3.141593
>> fprintf( '%e' , pi ) ;
3.141593e+00
>> fprintf( '%c' , 'Testo' ) ;
Testo
```

- (precisione): numero di cifre decimali dopo il punto decimale.

```
>> fprintf( '%.1f' , 1.79612 ) ;
1.8
>> fprintf( '%.15' , pi ) ;
3.141592653589793
>> fprintf( '%.30' , 1+1/3 ) ;
1.333333333333333259318465024990
```

```
>> fprintf( '%.2e'  , pi*6.042e23 ) ;
1.90e+24
>> fprintf( '%.15e' , pi*6.042e23 ) ;
1.898150281298953e+24>>
```

- (n_caratteri): numero minimo di caratteri totali da impiegare.

```
>> fprintf( '%1d' , 12345 ) ;
12345
>> fprintf( '%8d' , 1 ) ;
       1
>> fprintf( '%8d' , 12345 ) ;
   12345
```

```
>> fprintf( '%12f' , pi ) ;
    3.141593
>> fprintf( '%12e' , pi ) ;
3.141593e+00
>> fprintf( '%14e' , pi ) ;
  3.141593e+00
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

fprintf

(flags): ulteriori possibilità di formattazione; “-” per allineare a sinistra e non a destra (default), “+” per visualizzare sempre il segno anche nel caso di numeri positivi, “0” per riempire gli spazi a sinistra con 0.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

fprintf

- **Caratteri speciale**

Carattere	Forma	Risultato
apice singolo	''	'
percentuale	%%	%
barra rovesciata	\\	\
nuova linea	\n	a capo

```
>> fprintf( 'Pi greco con %d cifre decimali si scrive %.10f\n' , 10 , pi ) ;
Pi greco con 10 cifre decimali si scrive 3.1415926536
>>

>> r = 1.5 ;
>> fprintf( 'L''area del cerchio di raggio %.2f vale %.2f\n' , r , pi*r^2) ;
L'area del cerchio di raggio 1.50 vale 7.07
>>
```

- **Output formattato di vettori**

```
>> v = 10 .^ (0:5) ;
>> fprintf( 'v è un vettore lungo %d e vale (' , length(v) ) ; ...
   fprintf( '%d,' , v ) ; ...
   fprintf( ')\n' ) ;
v è un vettore lungo 6 e vale (1,10,100,1000,10000,100000,)
>>
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Tipi di dato alfanumerici: char e string

- Per dati in forma testuale si possono impiegare due tipi diversi:
  - **char**: singoli caratteri, definiti tra apici **'(carattere)'**;
  - **string**: stringhe = sequenze di caratteri, definite tra doppi apici **"(stringa)"**.

```
>> car = 'A' ;
>> class( car )
ans =
     'char'
```

```
>> str = "Università" ;
>> class( str )
ans =
     'string'
```

- Si possono definire vettori di tipo **char** o **string** concatenando opportunamente in riga o in colonna elementi di quel tipo, esattamente come per i tipi numerici:

```
>> car = [ 'U' 'n' 'i' 'v' ] ;
>> car = 'Univ'
car =
     'Univ'
>> size( car )
ans =
     1     4
>> [ car ; car ]
ans =
     2×4 char array
     'Univ'
     'Univ'
```

```
>> str = "Università" ;
>> size( str )
ans =
     1     1
>> [ str ; str ]
     2×1 string array
     "Università"
     "Università"
```

$\mathcal{NB}$: un vettore di **char** non corrisponde ad un tipo **string**.

- La codifica impiegata è UTF-8, corrispondente alla codifica ASCII per i primi 128 caratteri.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

sprintf

- Come per i tipi numerici, il nome del tipo può essere impiegato come funzione per effettuare la conversione verso altri tipi:

```
>> car = 'Univ' ;
>> str_car = string( car )
str_car =
     "Univ"
>> num_car = int16( car )
num_car =
  1×4 int16 row vector
    85   110   105   118
```

```
>> str = "Univ" ;
>> car_str = char( str )
car_str =
    'Univ'
>> num_str = int16( str )
 Error using int16
 Conversion to int16 from string is
 not possible.
>> num_str = int16( char( str ) )
num_str =
1×4 int16 row vector
85   110   105   118
```

- La funzione **sprintf** esegue le stesse operazioni di formattazione della funzione **fprintf**, ma l'output viene indirizzato in un vettore di **char** o in una stringa di tipo **string**, a seconda dei delimitatori impiegati per il formato:

```
>> car = sprintf( 'Pi = %.5f' , pi )
car =
    'Pi = 3.14159'
>> class( car )
ans =
     'char'
```

```
>> str = sprintf( "Pi = %.5f" , pi )
str =
     "Pi = 3.14159"
>> class( str )
ans =
     'string'
```

- Si possono combinare concatenazioni di vettori **char**, definiti direttamente o tramite **sprintf**, per definire un testo formattato complesso.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Esercizio

Scrivere uno script che, acquisita da tastiera una sequenza di caratteri, renda maiuscole tutte le lettere minuscole (non accentate). I codici ASCII delle lettere minuscole vanno da 97 (’a’) a 122 (’z’). La distanza tra una lettera maiuscola e la relativa lettera minuscola è sempre 32

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Esercizio

Scrivere uno script che, acquisita da tastiera una sequenza di caratteri, renda maiuscole tutte le lettere minuscole (non accentate). I codici ASCII delle lettere minuscole vanno da 97 (’a’) a 122 (’z’). La distanza tra una lettera maiuscola e la relativa lettera minuscola è sempre 32

```
% Input da tastiera
caratteri = input( 'Inserire un vettore di caratteri: ' ) ;

% Conversione in char nel caso che l'input sia di tipo string
caratteri = char( caratteri ) ;

% Vettore Booleano dei soli caratteri minuscoli
minuscoli = ( 97 <= caratteri ) & ( caratteri <= 122 ) ;

% Sostituzione con i relativi caratteri maiuscoli
caratteri( minuscoli ) = caratteri( minuscoli ) - 32 ;

% Output
disp( caratteri ) ;
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Energia

Energia di un segnale tempo-continuo in un intervallo [a,b]

$$E_x = \varepsilon \int_a^b |x(t)|^2 dt$$

Dato che eseguiamo il calcolo numericamente, dobbiamo utilizzare un algoritmo per e:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{i=n-1} (x_{i+1}-x_i)\frac{1}{2}[f(x_{i+1})+f(x_i)]$$

Se dividiamo [a,b] in parti n parti uguali

$$x_{i+1}-x_i = \Delta x = \frac{b-a}{n}$$

Otteniamo

$$\int_a^b f(x)dx \approx \frac{1}{2}\cdot\frac{b-a}{n}\sum_{i=0}^{i=n-1}[f(x_{i+1})+f(x_i)]$$

$$\sum_{i=0}^{i=n-1}[f(x_{i+1})+f(x_i)] = \sum_{i=0}^{i=n-1} f(x_{i+1}) + \sum_{i=0}^{i=n-1} f(x_i) = \sum_{i=0}^{i=n-1} f(x_i) + f(b) + \sum_{i=0}^{i=n-1} f(x_i)$$

$$-f(a)+2\sum_{i=0}^{i=n-1} f(x_i) = -f(a)+2\left(\sum_{i=0}^{i=n} f(x_i) - f(b)\right) = -f(a)-2f(b)+2\left(\sum_{i=0}^{i=n} f(x_i)\right)$$

$$\int_a^b f(x)dx \approx \frac{(b-a)}{n}\left(\sum_{i=0}^{i=n} f(x_i) - \frac{1}{2}f(a) - f(b)\right)$$

- tri.m
- % Create a triangular pulse with width 0.4.
- % fs = 10000; % Sampling frequency (samples/sec)
- % t = -1:1/fs:1; % Time Vector
- % w = .4; % Triangle Width

```
fs = 10000;      % Sampling frequency (samples/sec)
t = -1:1/fs:1;   % Time Vector
w = .4;          % Triangle Width
x = tripuls(t-0.5,w);  % Sampled aperiodic triangle
plot(t,x);
xlabel('Time (sec)');ylabel('Amplitude')
```

- % See also SQUARE, S
- % RECTPULSE,