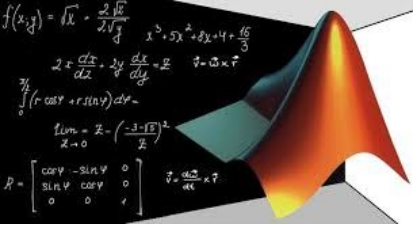


MATLAB® & SIMULINK®

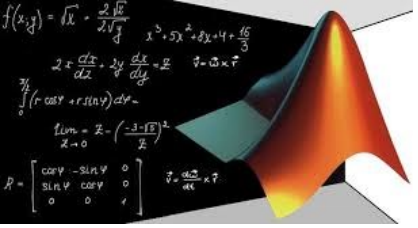
Matlab

Utilizzo del software più idoneo per accompagnare gli studenti del corso di Ingegneria Elettrica per la e-mobility nel raggiungere i propri ambiziosi traguardi

dipartimento di ingegneria unipa

Prof. Fabio Viola

LEAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

Command window
Barra di comando

Script

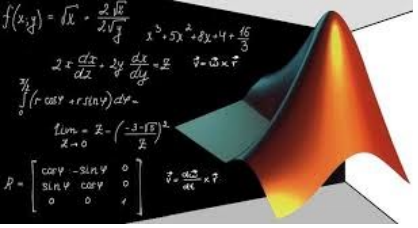
Function

Ci occupiamo adesso dell'utilizzo degli script, copioni.

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

Script
Per script si intende il più semplice programma di Matlab, basato su una sequenza di comandi prefissati. E' possibile eseguire uno script dalla riga di comando:

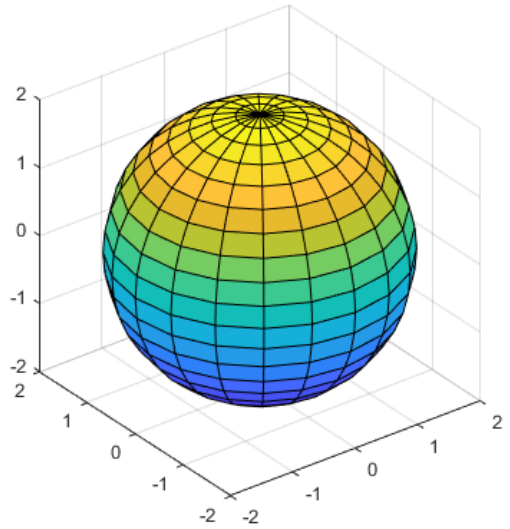
>> edit sfera

Si aprirà una nuova finestra nella quale inserire le istruzioni:

```
[x,y,z] = sphere;
r = 2;
surf(x*r,y*r,z*r) axis equal
```

Terminate le istruzioni si procede al salvataggio del file e dalla barra di comando è possibile digitare sfera ed ottenere il grafico di figura

Notare come il raggio r = 2 crea questa sfera, per esercitarsi provare con raggi differenti

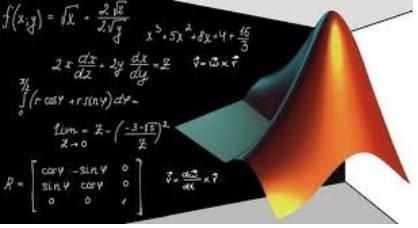


YouTube https://youtu.be/57b35wXEMdA

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

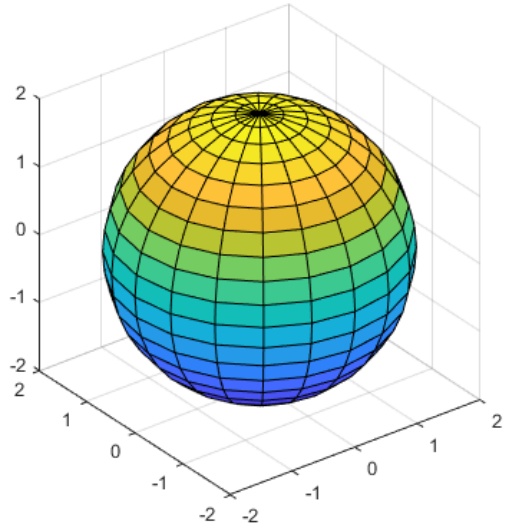
Ciclo if

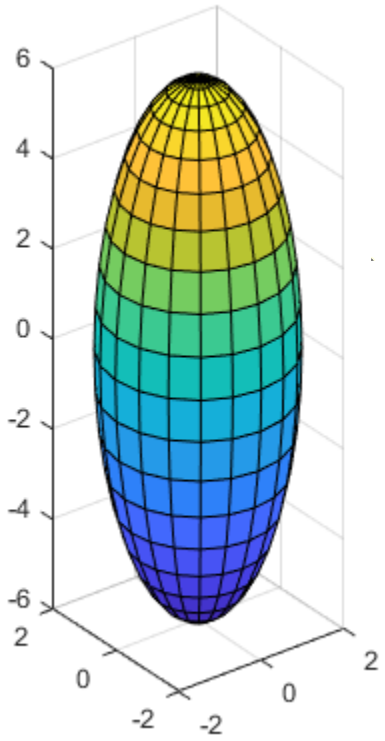
Il costrutto if-else è una **istruzione condizionale,** permette cioè di eseguire **istruzioni o blocchi codice** a seconda del verificarsi di una condizione.
Ad esempio vogliamo rappresentare una sfera, un ellissoide o un cilindro, in base al valore M= 1,2 o 3. Scegliamo prima di creare lo script un valore per M:

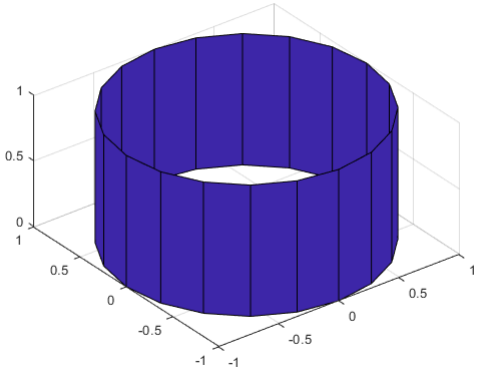
```
>> M=2;
>> edit disegna
```

Si aprirà una nuova finestra nella quale inserire le istruzioni:

```
if M == 1
   [x,y,z] = sphere;
elseif M ==2
  [x,y,z] = cylinder;
elseif M == 3
   [x,y,z] = ellipsoid(0,0,0,2,2,6);
else
% non fare nulla
end
surf(x,y,z)
axis equal
```





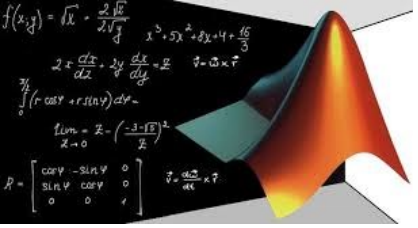


YouTube https://youtu.be/kQfGxsClEI8

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Script

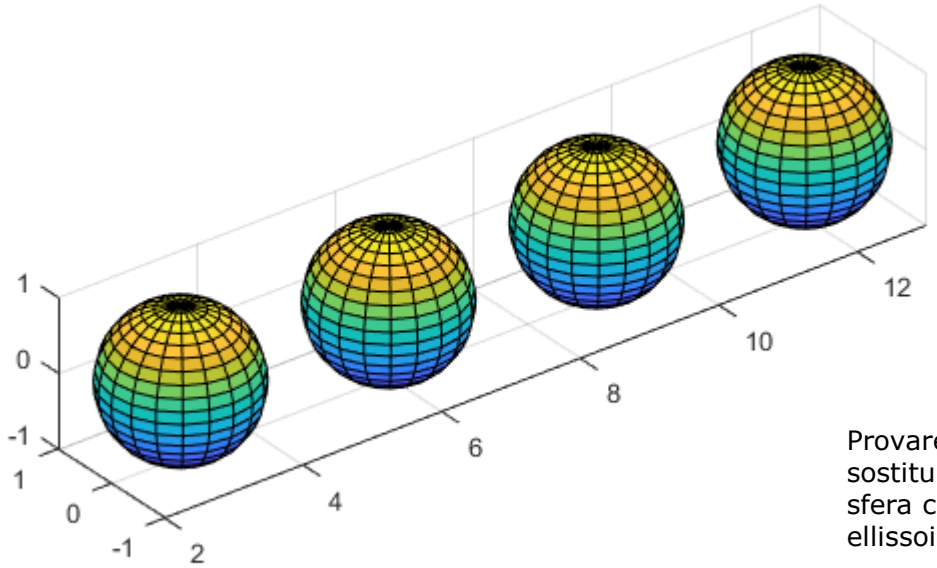
MATLAB® & SIMULINK®

Ciclo for
Nei linguaggi di programmazione il ciclo for è una struttura di controllo iterativa che determina l'esecuzione di una porzione di programma ripetuta per un certo numero noto di volte.
Vogliamo ad esempio rappresentare un numero M di sfere.

>> edit sfere

Si aprirà una nuova finestra nella quale inserire le istruzioni:

```
M = 4;
distanza = 3;
[x,y,z] = sphere;
for i = 1 : M
  surf(x+i*distanza,y,z);
  hold on;
end
axis equal
```



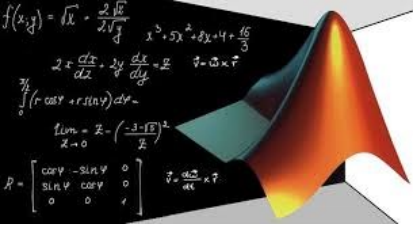
Provare a sostituire la sfera con un ellissoide

YouTube https://youtu.be/5tT27nEHCAs

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

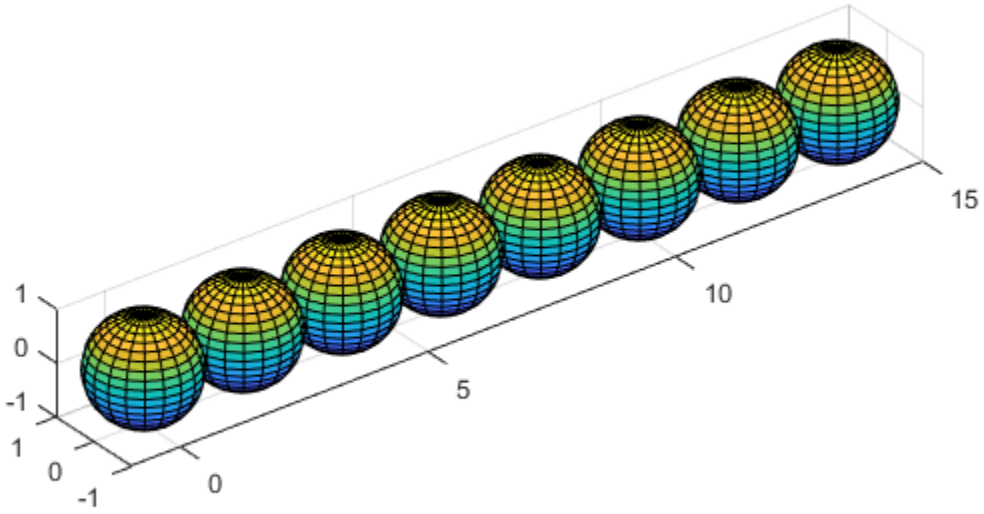
Ciclo while

Come con il for anche con il costrutto **while** possiamo definire dei **cicli.** Il programma eseguirà un'istruzione o un blocco di istruzioni finché una certa condizione resta verificata. Ad esempio riempiamo un box con delle sfere fino a quando non occupano tutto lo spazio.

>> edit scatola

Si aprirà una nuova finestra nella quale inserire le istruzioni:

```
capienza = 8;
[x,y,z] = sphere;
n=0;
while capienza > 0
  surf(x+n*2,y,z);
  hold on;
  capienza=capienza-1;
  n= n+1;
end
axis equal;
```

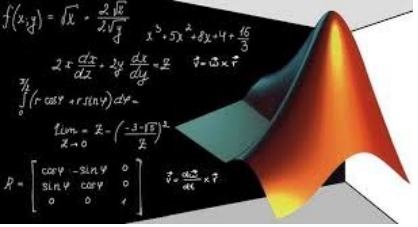


YouTube https://youtu.be/Z4DnbIqto80

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

Problema
Si vuole impostare l'analisi di un sistema di equazioni lineari che risolva il problema A x = b, con A matrice che traduce le relazioni tra le incognite x e b insieme dei termini noti

```
>> A=[1 2 3; 4 5 6; 7 8 0]
>> b=[12 ; 33; 36]
>> x=inv(A)*b
x = 4.0000 1.0000 2.0000
```

L'operazione è formalmente corretta ma è numericamente onerosa e lenta.

La risoluzione del sistema si ottiene in MatLab usando il simbolo di divisione a sx backslash \
>>x=A\b soluzione del sistema Ax=b (x=inv(A)*b)

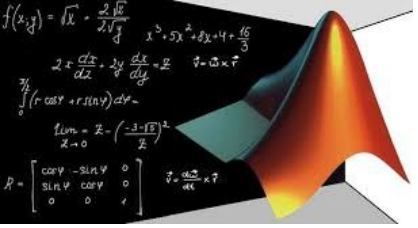
```
>> A=[1 2 3; 4 5 6; 7 8 0]
>> b=[12 ; 33; 36]
>> x=A\b
x = 4.0000 1.0000 2.0000
```

L'operatore backslash \ usa algoritmi differenti per trattare diversi tipi di matrici:
- Permutazioni di matrici triangolari.
- Matrici simmetriche e definite positive.
- Sistemi rettangolari sovradeterminati.
- Sistemi rettangolari sottodeterminati.

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

Sistemi diagonali

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} -1 \\ 6 \\ -15 \end{bmatrix}$$

equivale

$$\begin{aligned} x_1 &= -1 \\ 3x_2 &= 6 \\ 5x_3 &= -15 \end{aligned}$$

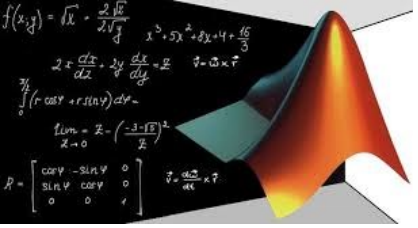
e contemporaneamente

$$x_1 = -1 \qquad x_2 = \frac{6}{3} = 2 \qquad x_3 = \frac{-15}{5} = -3$$

dipartimento di ingegneria unipa

Prof. Fabio Viola

LEAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

Sistemi triangolari

$$A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

equivale

$$\begin{aligned} -2x_1 + x_2 + 2x_3 &= 9 \\ 3x_2 + -2x_3 &= -1 \\ 4x_3 &= 8 \end{aligned}$$

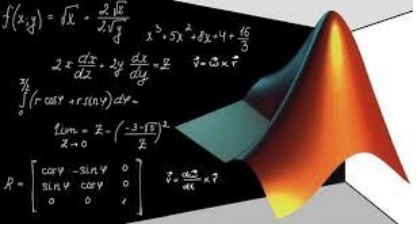
ed in successione

$$x_3 = \frac{8}{4} = 2 \quad \Rightarrow \quad x_2 = \frac{1}{3}(-1 + 2x_3) = \frac{3}{3} = 1 \quad \Rightarrow \quad x_1 = \frac{1}{-2}(9 - x_2 - 2x_3) = \frac{4}{-2} = -2$$

dipartimento di ingegneria unipa

Prof. Fabio Viola

LEAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

Sistemi triangolari

$$A^{(0)} = A \qquad b^{(0)} = b$$

$$A^{(0)} = \begin{pmatrix} 2 & 6 & 4 \\ 1 & 0 & 2 \\ 2 & 2 & 1 \end{pmatrix} \qquad b^{(0)} = \begin{pmatrix} 10 \\ -1 \\ 5 \end{pmatrix}$$

$$A^{(1)} = \begin{pmatrix} 2 & 6 & 4 \\ 0 & \times & \times \\ 0 & \times & \times \end{pmatrix} \quad b^{(1)} = \begin{pmatrix} 10 \\ \times \\ \times \end{pmatrix} \qquad A^{(1)} = \begin{pmatrix} 2 & 6 & 4 \\ 0 & -3 & 0 \\ 0 & -4 & -3 \end{pmatrix} \quad b^{(1)} = \begin{pmatrix} 10 \\ -6 \\ -5 \end{pmatrix}$$

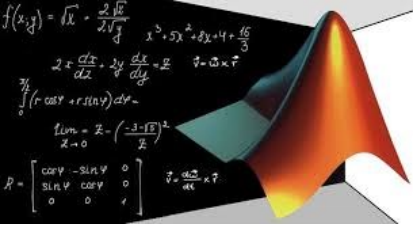
$$A^{(2)} = \begin{pmatrix} 2 & 6 & 4 \\ 0 & -3 & 0 \\ 0 & 0 & -3 \end{pmatrix} \quad y = b^{(2)} = \begin{pmatrix} 10 \\ -6 \\ 3 \end{pmatrix}$$

$$x = (1, 2, -1)$$

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Script

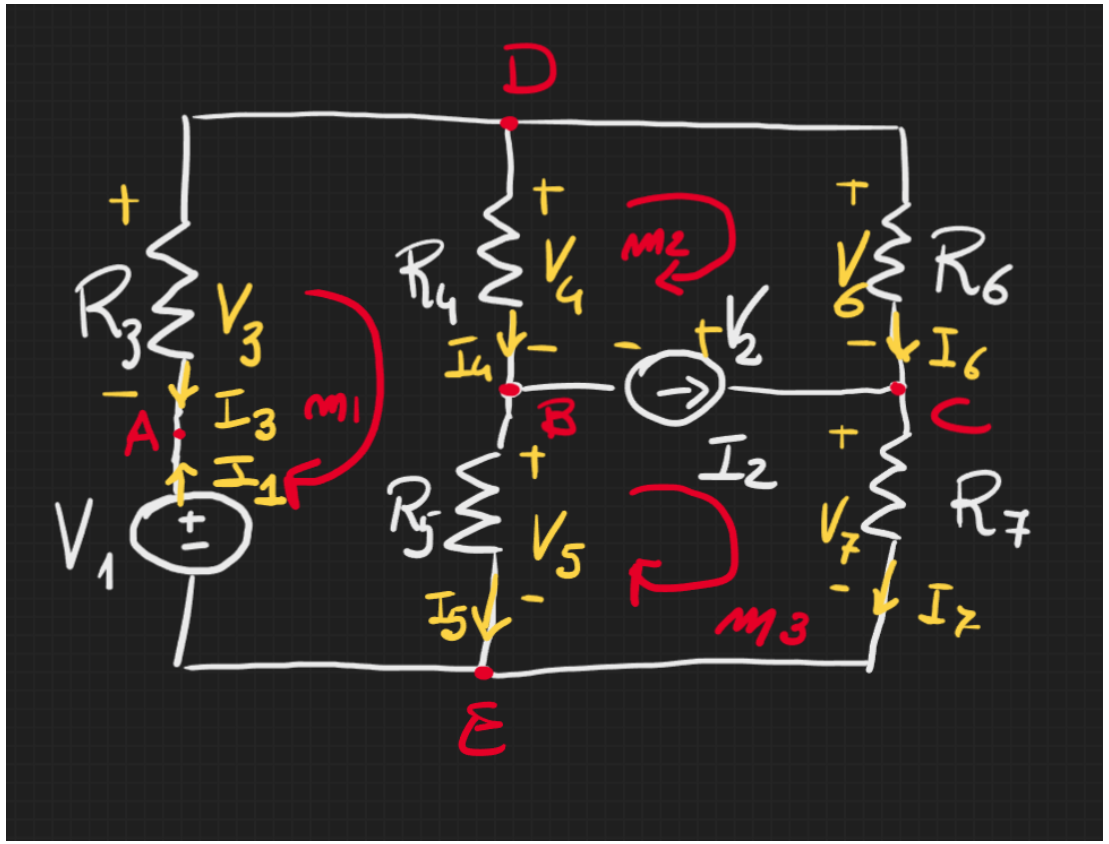
MATLAB® & SIMULINK®

Problema
Scrivere uno script che risolva la rete di figura con:
V1 =5 V, I2 = 10 mA, R3 = 100 Ω, R4 = 1 kΩ W, R5 = 3 kΩ W, R6 = 1 kΩ = e R7 = 1 kΩ

Legge di Kirchhoff ai nodi
A: I1+I3 = 0
B: I5-I4+I2 = 0
C: -I2+I6+I7 = 0
D: I3+I4+I6 = 0
E: si trascura

Legge di Kirchhoff agli anelli
m1: -V1 –V3 +V4 +V5 = 0
m2: -V4+V6 +V2 = 0
m3: -V5 –V2 +V7 = 0

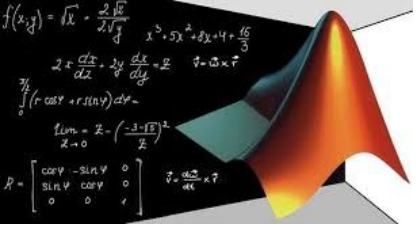
Legge di Ohm
V3 = R3I3
V4 = R4I4
V5 = R5I5
V6 = R6I6
V7 = R7I7



dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



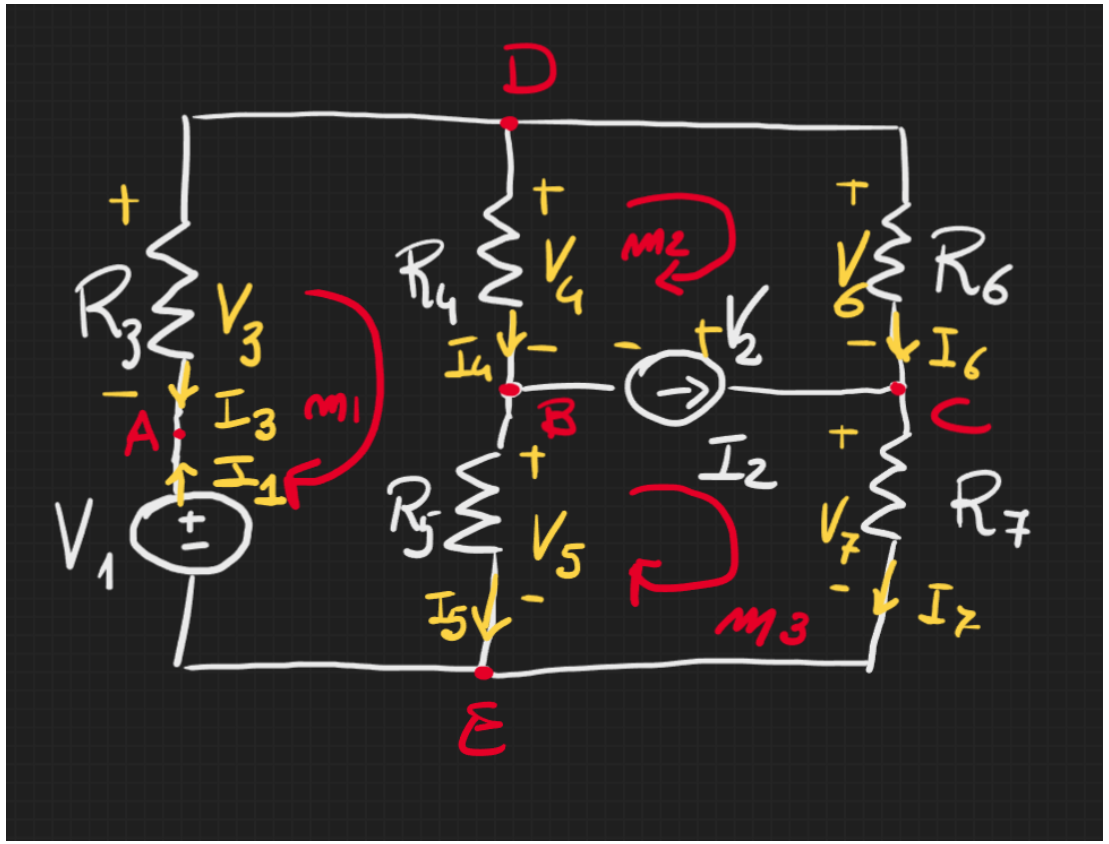
Script

MATLAB® & SIMULINK®

Incognite
X = [V2 V3 V4 V5 V6 V7 I1 I3 I4 I5 I6 I7]’

$$A=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & R3 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & R4 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & R5 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & R6 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & R6 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 \\ -I2 \\ I2 \\ 0 \\ V1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

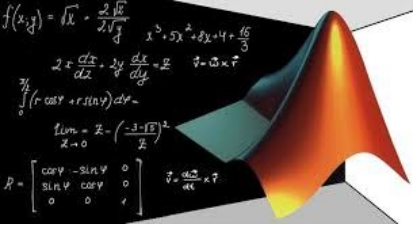
Essendo la matrice fortemente sparsa, nella sua compilazione conviene definire una matrice di zeri per poi inserire gli elementi non nulli



dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

>> edit circuito

```
clear all; clc;
V1=5; I2=-0.01; % generatori
R3=100; R4=1000; R5=3000; R6=1000; R7=1000;
% resistenze
% incognite sono: X=[V2 V3 V4 V5 V6 V7 I1 I3 I4 I5
I6 I7]
A=zeros(12,12);Y=zeros(12,1); % creazione matrici
vuote
% riempimento posti non nulli di A
A(1,7)=1;A(1,8)=1;
A(2,9)=-1;A(2,10)=1;
A(3,11)=-1;A(3,12)=1;
A(4,8)=1;A(4,9)=1;A(4,11)=1;
A(5,2)=-1;A(5,3)=1;A(5,4)=1;
A(6,1)=1;A(6,3)=-1;A(6,5)=1;
A(7,1)=-1;A(7,4)=-1;A(7,6)=1;
A(8,2)=-1;A(8,8)=R3;
A(9,3)=-1;A(9,9)=R4;
A(10,4)=-1;A(10,10)=R5;
A(11,5)=-1;A(11,11)=R6;
A(12,6)=-1;A(12,12)=R7;
% riempimento del vettore dei termini noti Y
Y(2)=-I2;Y(3)=I2;Y(5)=V1;
% utilizzo della Gauss elimination per AX=Y
X=A\Y
```

```
X =

  -13.7209
   -0.1163
   -6.2791
   11.1628
    7.4419
   -2.5581
    0.0012
   -0.0012
   -0.0063
    0.0037
    0.0074
   -0.0026
```

>> edit potenza

```
V=[V1 X(1:6)']; % creo vettore delle tenioni aggiungendo il generatore alle incognite
I=[-X(7) -I2 X(8:12)']; % creo il vettore delle correnti aggiungendo il generatore di
corrente, le correnti dei generatori cambiano segno
P=V.*I; % calcolo potenza
disp('elemento tensione Corrente Potenza')
disp(' Numero (volt) (A) (W)')
K=1:7;
Tabella=[K; V; I; P]; % tabella riassuntiva
fprintf(' %i %-9.4f %-9.4f %-9.4f \n',Tabella)
PotTot = sum(P); %bilancio tra potenza generata (negativa) e assorbita (positiva)
fprintf('potenza totale assorbita = %9.4f W \n',PotTot)
```

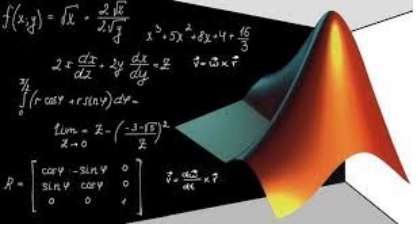
Command Window

```
>> potenza
elemento tensione Corrente Potenza
 Numero (volt) (A) (W)
 1 5.0000    -0.0012   -0.0058
 2 -13.7209  0.0100    -0.1372
 3 -0.1163   -0.0012   0.0001
 4 -6.2791   -0.0063   0.0394
 5 11.1628   0.0037    0.0415
 6 7.4419    0.0074    0.0554
 7 -2.5581   -0.0026   0.0065
potenza totale assorbita =    0.0000 W
fx >>
```

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication

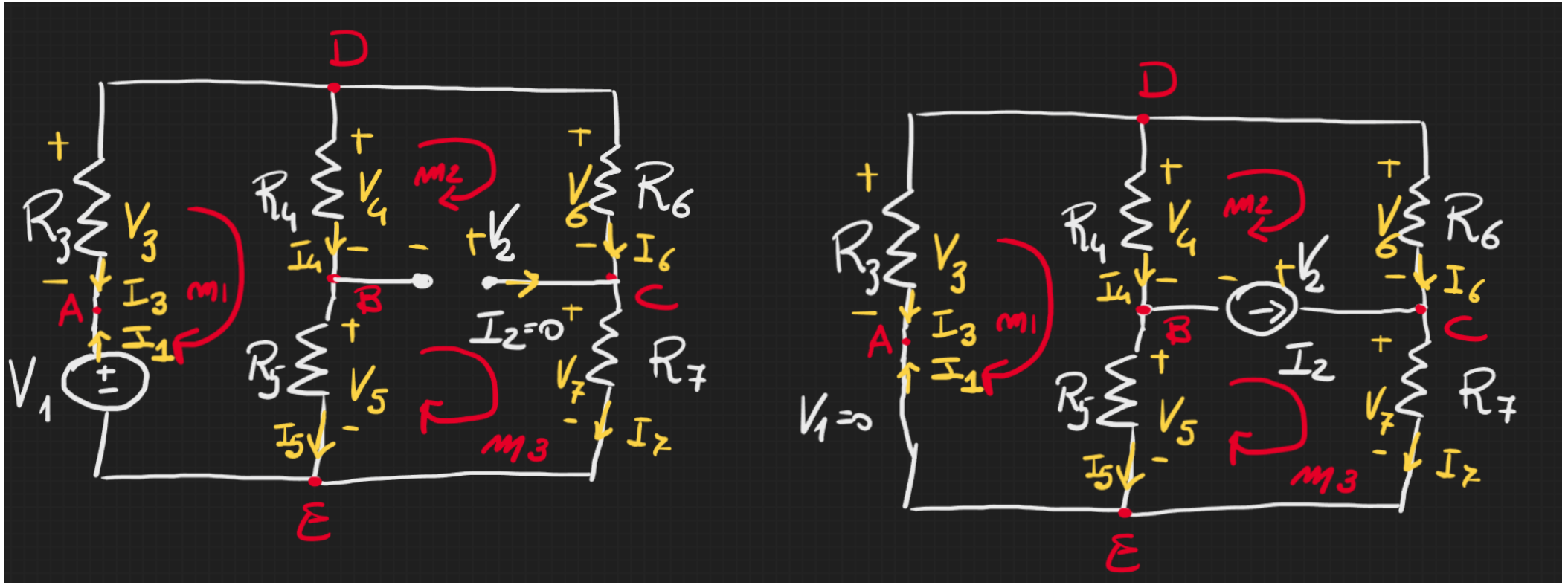


Script

MATLAB® & SIMULINK®

Applicazione del metodo di sovrapposizione degli effetti

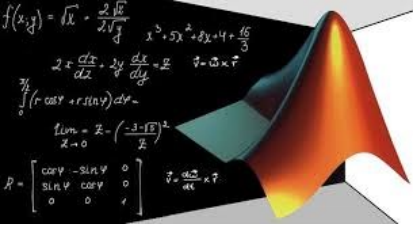
Il metodo consiste nel risolvere più circuiti, ognuno con un generatore attivo per volta, negando l'effetto degli altri: al generatore di corrente si sostituisce un circuito aperto ed al generatore di tensione si sostituisce un corto circuito.



dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

Applicazione del metodo di sovrapposizione

Incognite

X = [V2 V3 V4 V5 V6 V7 I1 I3 I4 I5 I6 I7]'

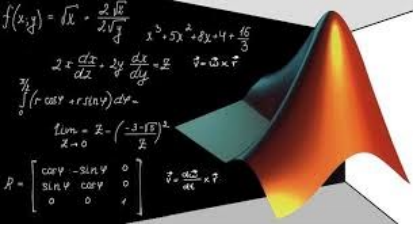
$$A=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & R3 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & R4 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & R5 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & R6 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & R6 \end{bmatrix}$$

$$Y1=\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ V1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad Y2=\begin{bmatrix} 0 \\ -I2 \\ I2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

Applicazione del metodo di sovrapposizione

>> edit circuito_sovrap

```
clear all; clc;
V1=5; I2=-0.01; % generatori
R3=100; R4=1000; R5=3000; R6=1000; R7=1000; % resistenze
% incognite sono: X=[V2 V3 V4 V5 V6 V7 I1 I3 I4 I5 I6 I7]
A=zeros(12,12);Y1=zeros(12,1);Y2=zeros(12,1); % creazione matrici vuote
% riempimento posti non nulli di A
A(1,7)=1;A(1,8)=1;
A(2,9)=-1;A(2,10)=1;
A(3,11)=-1;A(3,12)=1;
A(4,8)=1;A(4,9)=1;A(4,11)=1;
A(5,2)=-1;A(5,3)=1;A(5,4)=1;
A(6,1)=1;A(6,3)=-1;A(6,5)=1;
A(7,1)=-1;A(7,4)=-1;A(7,6)=1;
A(8,2)=-1;A(8,8)=R3;
A(9,3)=-1;A(9,9)=R4;
A(10,4)=-1;A(10,10)=R5;
A(11,5)=-1;A(11,11)=R6;
A(12,6)=-1;A(12,12)=R7;
% riempimento dei vettori dei termini noti per Y1 eY2
Y2(2)=-I2;Y2(3)=I2;Y1(5)=V1;
% utilizzo della Gauss elimination per AX=Y
X1=A\Y1; X2=A\Y2;
X1+X2
```

```
ans =

  -13.7209
   -0.1163
   -6.2791
   11.1628
    7.4419
   -2.5581
    0.0012
   -0.0012
   -0.0063
    0.0037
    0.0074
   -0.0026
```

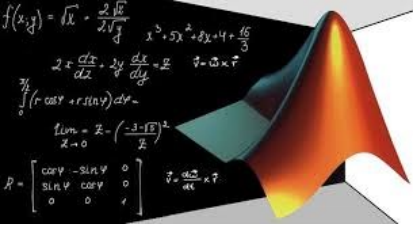
Incognite
X = [V2 V3 V4 V5 V6 V7 I1 I3 I4 I5 I6 I7]'

Osservazioni
La matrice A non subisce modifiche poiché è priva delle sollecitazioni, si modificano solamente le matrici di ingresso Y1 e Y2

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

Metodo dei potenziali di nodo
Si devono esprimere le correnti dei resistori come differenza dei potenziali sui nodi:

$$I_3 = \frac{V_D - V_A}{R_3} \quad I_4 = \frac{V_D - V_B}{R_4} \quad I_5 = \frac{V_B - 0}{R_5} \quad I_6 = \frac{V_D - V_C}{R_6} \quad I_7 = \frac{V_C - 0}{R_7}$$

Si considera 0 il potenziale di VE
Le equazioni di Kirchhoff ai nodi restituiscono:

A: $V_A - V_E = V_1$

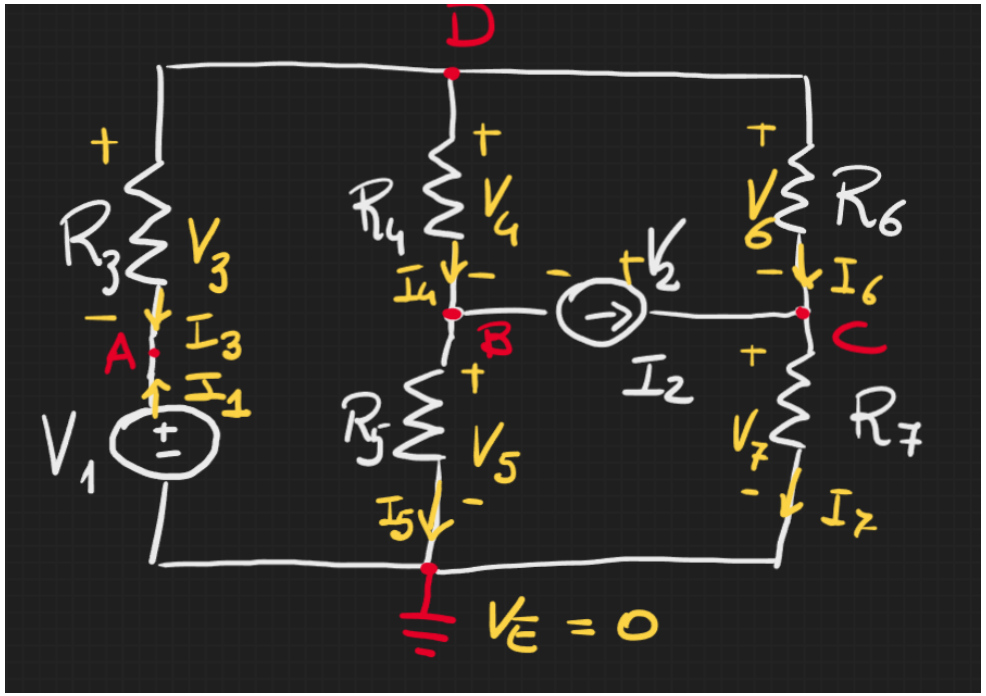
B: $\frac{V_B - V_D}{R_4} + I_2 + \frac{V_B}{R_5} = 0$

C: $-I_2 + \frac{V_C - V_D}{R_6} + \frac{V_C}{R_7} = 0$

D: $\frac{V_D - V_A}{R_3} + \frac{V_D - V_B}{R_4} + \frac{V_D - V_C}{R_6} = 0$

Incognite
V = [VB VC VD]'

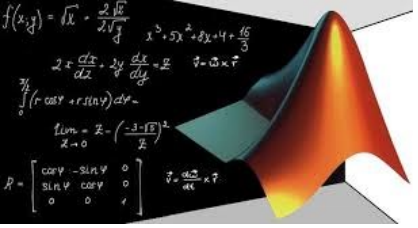
$$G = \begin{bmatrix} \frac{1}{R_4} + \frac{1}{R_5} & 0 & -\frac{1}{R_4} \\ 0 & \frac{1}{R_6} + \frac{1}{R_7} & -\frac{1}{R_6} \\ -\frac{1}{R_4} & -\frac{1}{R_6} & \frac{1}{R_3} + \frac{1}{R_4} + \frac{1}{R_6} \end{bmatrix} \qquad I = \begin{bmatrix} -I_2 \\ I_2 \\ \frac{V_2}{R_3} \end{bmatrix}$$



dipartimento di ingegneria unipa

Prof. Fabio Viola

LEAP
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

Metodo dei potenziali di nodo
Si devono esprimere le correnti dei resistori come differenza dei potenziali sui nodi:

$$I_3 = \frac{V_D - V_A}{R_3} \quad I_4 = \frac{V_D - V_B}{R_4} \quad I_5 = \frac{V_B - 0}{R_5} \quad I_6 = \frac{V_D - V_C}{R_6} \quad I_7 = \frac{V_C - 0}{R_7}$$

Si considera 0 il potenziale di VE
Le equazioni di Kirchhoff ai nodi restituiscono:

A: $V_A - V_E = V_1$

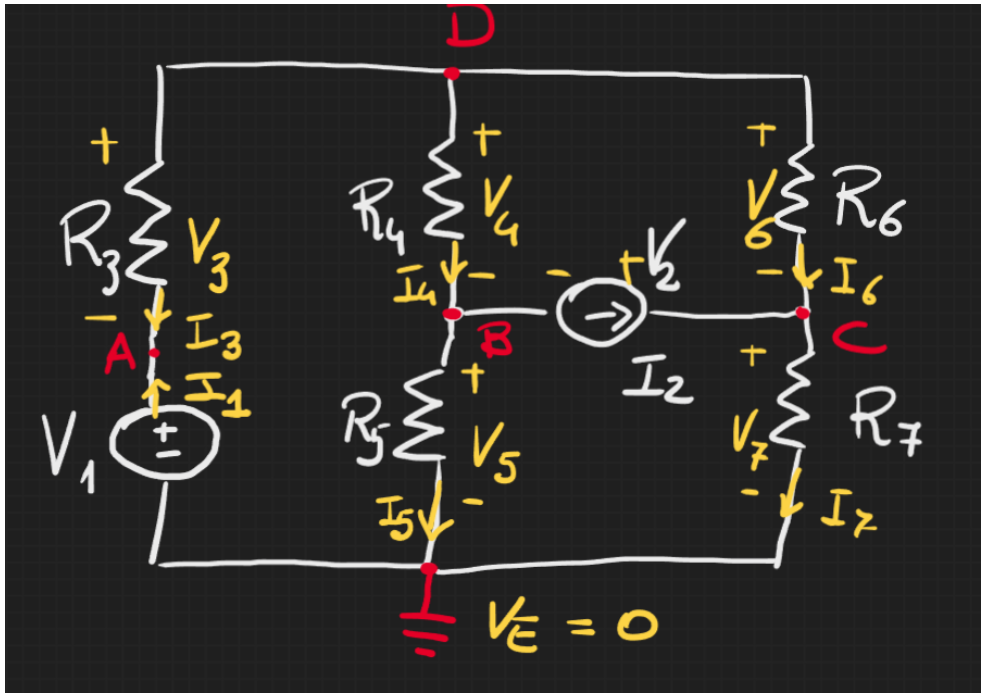
B: $\frac{V_B - V_D}{R_4} + I_2 + \frac{V_B}{R_5} = 0$

C: $-I_2 + \frac{V_C - V_D}{R_6} + \frac{V_C}{R_7} = 0$

D: $\frac{V_D - V_A}{R_3} + \frac{V_D - V_B}{R_4} + \frac{V_D - V_C}{R_6} = 0$

Incognite
V = [VB VC VD]'

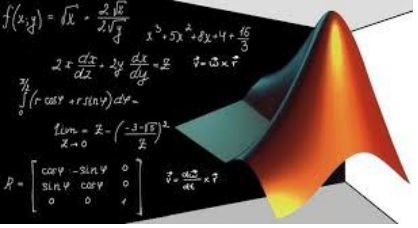
$$G = \begin{bmatrix} \frac{1}{R_4} + \frac{1}{R_5} & 0 & -\frac{1}{R_4} \\ 0 & \frac{1}{R_6} + \frac{1}{R_7} & -\frac{1}{R_6} \\ -\frac{1}{R_4} & -\frac{1}{R_6} & \frac{1}{R_3} + \frac{1}{R_4} + \frac{1}{R_6} \end{bmatrix} \qquad I = \begin{bmatrix} -I_2 \\ I_2 \\ \frac{V_2}{R_3} \end{bmatrix}$$



dipartimento di ingegneria unipa

Prof. Fabio Viola

LEAP
Laboratory of Electrical APplication



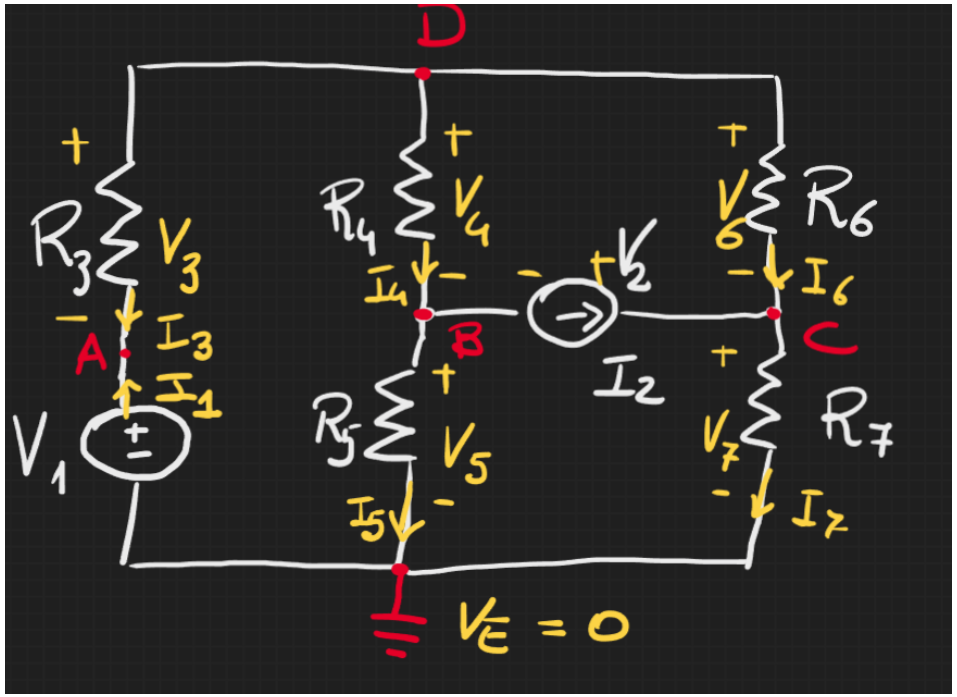
Script

MATLAB® & SIMULINK®

```
>> edit circuito_pot
```

```
clear all; clc;
% metodo dei nodi
V1=5; I2=-0.01; % sorgenti di tensione e corrente
R3=100; R4=1000; R5=3000; R6=1000; R7=1000; % resistenze
% vettore potenziali incogniti V=[VB VC VD]'
G=zeros(3,3);I=zeros(3,1); % allocamento spazio
% riempimento degli elementi non nulli di G, matrice delle conduttanze
G(1,1)=1/R4 + 1/R5;G(1,3)=-1/R4;
G(2,2)=1/R6 + 1/R7;G(2,3)=-1/R6;
G(3,1)=-1/R4;G(3,2)=-1/R6;G(3,3)=1/R3 + 1/R4 + 1/R6;
% termini noti
I(1)=-I2;I(2)=I2;I(3)=V1/R3;
% determinazione di V
V=G\I; % backslash per utilizzare la Gauss elimination
V' % stampa a schermo
VA=V1;VB=V(1);VC=V(2);VD=V(3);
V2=VC-VB
V3=VD-VA
V4=VD-VB
V5=VB
V6=VD-VC
V7 = VC
```

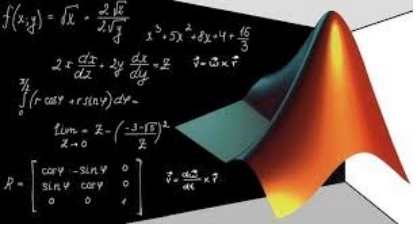
```
V2 =

  -13.7209

V3 =

   -0.1163

V4 =

   -6.2791

V5 =

   11.1628

V6 =

    7.4419

V7 =

   -2.5581
```



dipartimento di ingegneria unipa

Prof. Fabio Viola

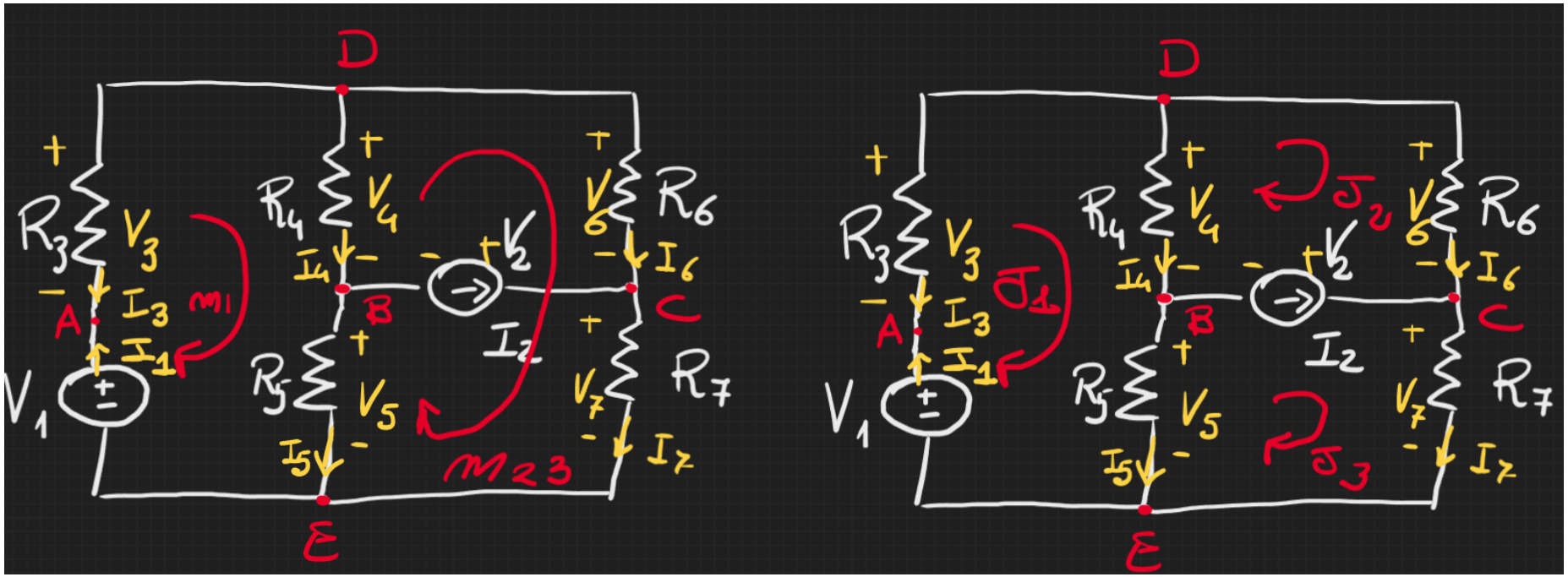
Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

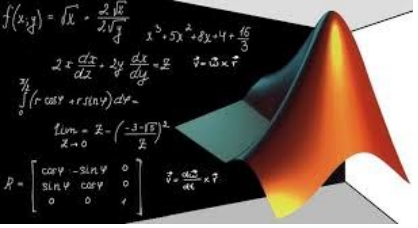
metodo delle correnti di maglia
Nel circuito è presente un generatore di corrente, e la teoria classica del metodo delle correnti di anello non può essere direttamente applicata. Si procede ad un allargamento di anello a maglia



dipartimento di ingegneria unipa

Prof. Fabio Viola

Laboratory of Electrical APplication



Script

MATLAB® & SIMULINK®

metodo delle correnti di maglia
Maglia 1:

$$-V_1 + R_3 J_1 + R_4(J_1 - J_2) + R_5(J_1 - J_3) = 0$$

Maglia 23

$$R_4(J_2 - J_1) + R_6 J_2 + R_7 J_3 + R_5(J_3 - J_1) = 0$$

Generatore di corrente:

$$I_2 = J_3 - J_2$$

rimaneggiate

$$-V_1 + R_3 J_1 + R_4(J_1 - J_2) + R_5(J_1 - I_2 - J_2) = 0$$

Maglia 23

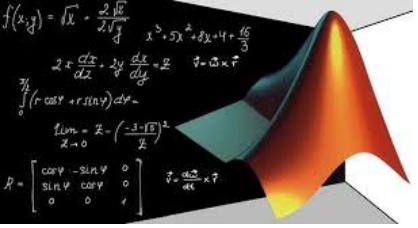
$$R_4(J_2 - J_1) + R_6 J_2 + R_7(I_2 + J_2) + R_5(I_2 + J_2 - J_1) = 0$$

Definendo la matrice delle resistenze

$$R = \begin{bmatrix} R_3 + R_4 + R_5 & -R_4 - R_5 \\ -R_4 - R_5 & R_4 + R_5 + R_6 + R_7 \end{bmatrix}$$

Ed il vettore dei termini noti

$$V = \begin{bmatrix} V_1 + R_5 I_2 \\ -R_5 I_2 - -R_7 I_2 \end{bmatrix}$$



Script

MATLAB® & SIMULINK®

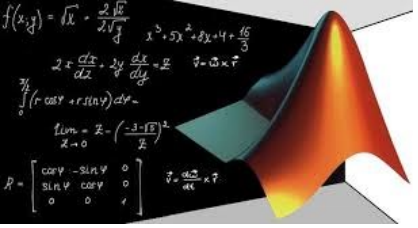
```
clear all; clc;
% analisi alle maglie
V1=5; I2=-0.01; % generatori di corrente
R3=100; R4=1000; R5=3000; R6=1000; R7=1000; % resistenze
% vettore delle correnti di anello [J1 J2 J3]'
R=zeros(2,2); V=zeros(2,1); % allocazione spazio
% inserimento dei termini non nulli
R(1,1)=R3+R4+R5;R(1,2)=-R4-R5;
R(2,1)=-R4-R5;R(2,2)=R4+R6+R5+R7;
% si può anche verificare che R e G sono l'iversa l'ula dell'altra (metodo
% dei nodi)
% vettore delle tensioni
V(1)=V1+R5*I2;V(2)=-R7*I2-R5*I2;
% utilizzo della eliminazione di Gauss
J=R\V; % la backslash implementa la Gauss elimination
J' % show the unknown mesh currents i in the command window
J1=J(1);J2=J(2);J3=I2+J(2);
I1=J1;
I3=-J1;
I4=J1-J2;
I5=J1-J3;
I6=J2;
I7=J3;
V2= R4*I4-R6*I6
V3=R3*I3
V4=R4*I4
V5 = R5*I5
V6= R6*I6
V7 = R7*I7
```

```
V2 =

  -13.7209


V3 =

   -0.1163


V4 =

   -6.2791


V5 =

   11.1628


V6 =

    7.4419


V7 =

   -2.5581
```

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Function

MATLAB® & SIMULINK®

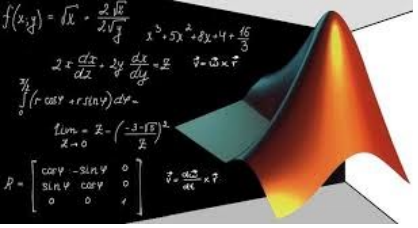
Command window
Barra di comando

Script

Function

La programmazione può avvenire in modo differente:

1) Tramite barra di comando, eseguendo dei passaggi che abbiamo noi in mente
2) Su script (copione) eseguendo istruzioni da noi inserite in una sequenza
3) Tramite funzioni che possono essere richiamate all'occorrenza



Function

MATLAB® & SIMULINK®

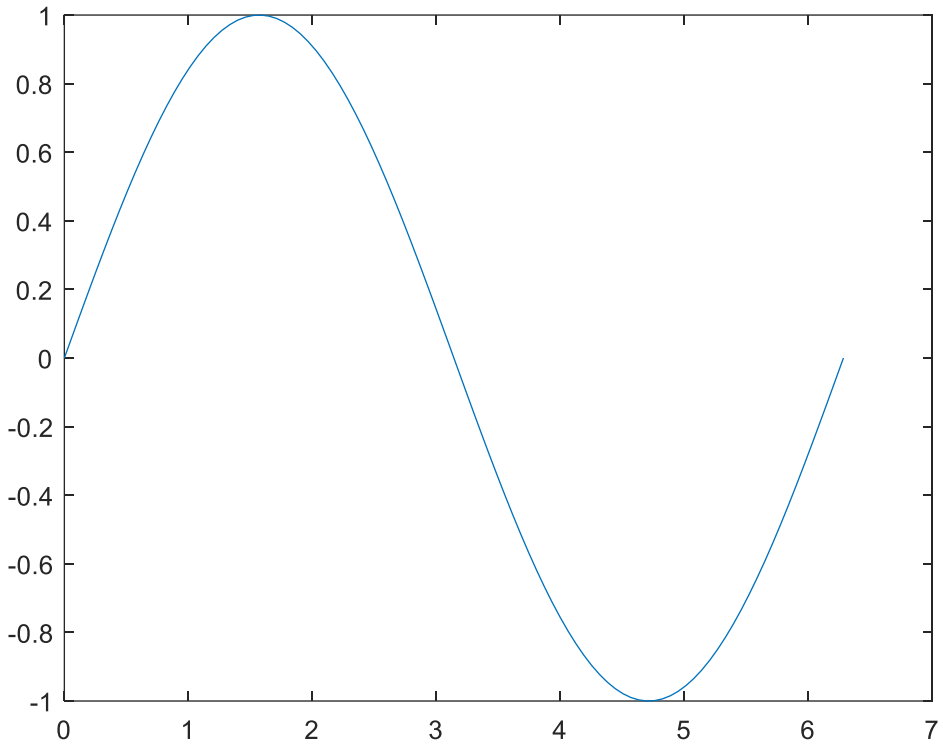
Funzioni elementari

In Matlab sono predefinite varie funzioni elementari, alcune delle quali sono riportate qui di seguito

funzione esponenziale: exp(x)
logaritmo naturale: log(x)
radice quadrata (square root): sqrt(x)
seno: sin(x)
coseno: cos(x)
tangente: tan(x)
arcoseno: asin(x)
arcocoseno: acos(x)
arcotangente: atan(x)
seno ipebolico: sinh(x)
coseno iperbolico: cosh(x)
tangente iperbolica: tanh(x)
arcoseno iperbolico: asinh(x)
arcocoseno iperbolico: acosh(x)
funzione segno: sign(x)
funzione pavimento: floor(x)
funzione soffitto: ceil(x)

Dato un vettore x in entrata, ognuna di queste funzioni restituisce il vettore delle immagini in uscita.

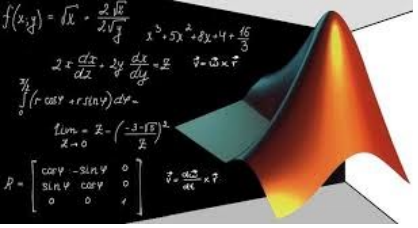
```
>> x=linspace (0, 2*pi, 100);
x=linspace (0, 2*pi, 100);
>> y=sin(x);
>> plot(x,y) >> y=sin(x);
>> plot(x,y)
```



dipartimento di ingegneria unipa

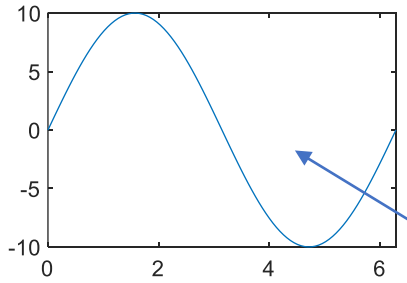
Prof. Fabio Viola

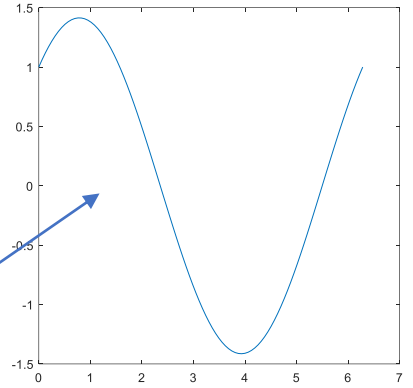
LAP
Laboratory of Electrical APplication



Function

MATLAB® & SIMULINK®

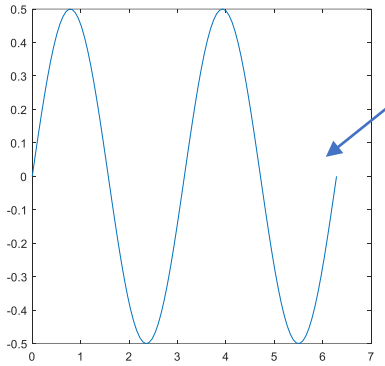


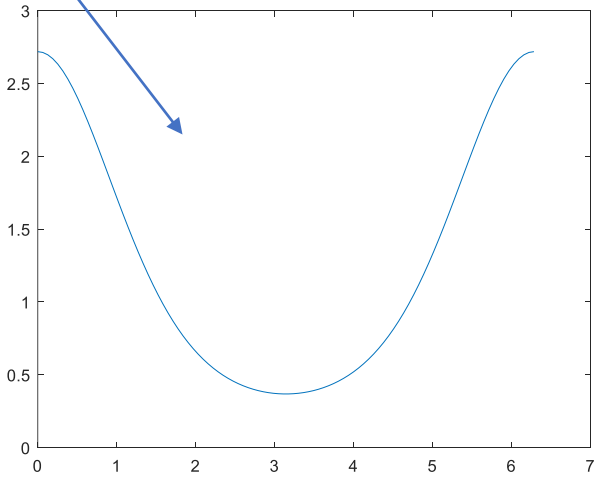


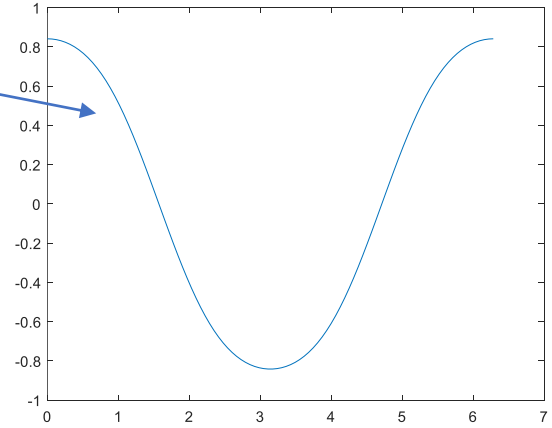
Operazioni tra funzioni

Sono previste le operazioni elementari tra funzioni:

- prodotto di uno scalare per una funzione: y = 10*sin(x);
- Somma e differenza di funzioni (devono avere stesso dominio) y =sin(x)+ cos(x);
- Prodotto di funzioni: sin(x).*cos(x);
- Composizione di funzioni: y= sin(cos(x));
- Funzioni esponenziali: y= exp(cos(x))





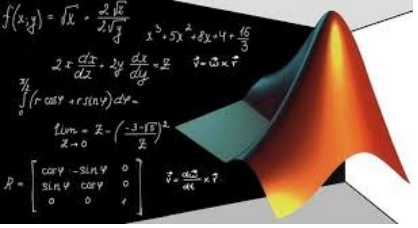


dipartimento di ingegneria unipa

Prof. Fabio Viola

LEAP

Laboratory of Electrical APplication



Function

MATLAB® & SIMULINK®

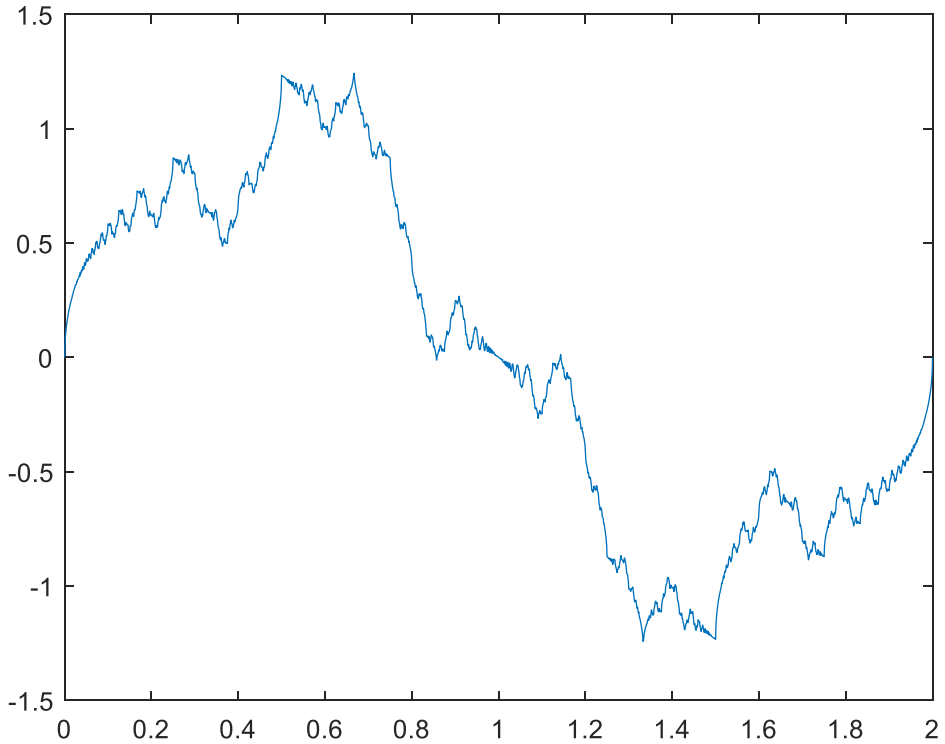
Funzioni definite dall'utente

In Matlab si ha la possibilità di definire nuove funzioni mediante gli m-file. Un m-file è un file, con estensione m, che contiene una successione di comandi che possono essere interpretati ed eseguiti da Matlab. Gli m-file che definiscono una funzione sono chiamati file di funzione. In questi file tutte le variabili sono locali (contrariamente a quanto accade nei file script).

```
>> edit riemannfx
```

```
function y = riemannfz( n, x )
t = 0;
for k = 1 : n
t = t + sin( k^2*pi*x ) / k^2;
end
y = t
```

```
>> x= 0:1/1000:2;
>> y =riemannfz(1000,x);
>> plot(x,y)
```



Osservazioni

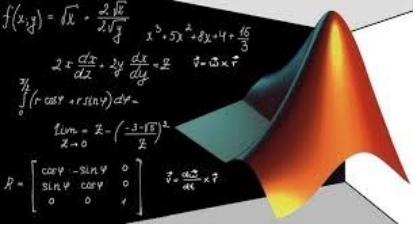
La funzione ha due ingressi:

- n numero di campioni
- x vettore dei campioni di dimensione n

dipartimento di ingegneria unipa

Prof. Fabio Viola

LEAP
Laboratory of Electrical APplication



Function

MATLAB® & SIMULINK®

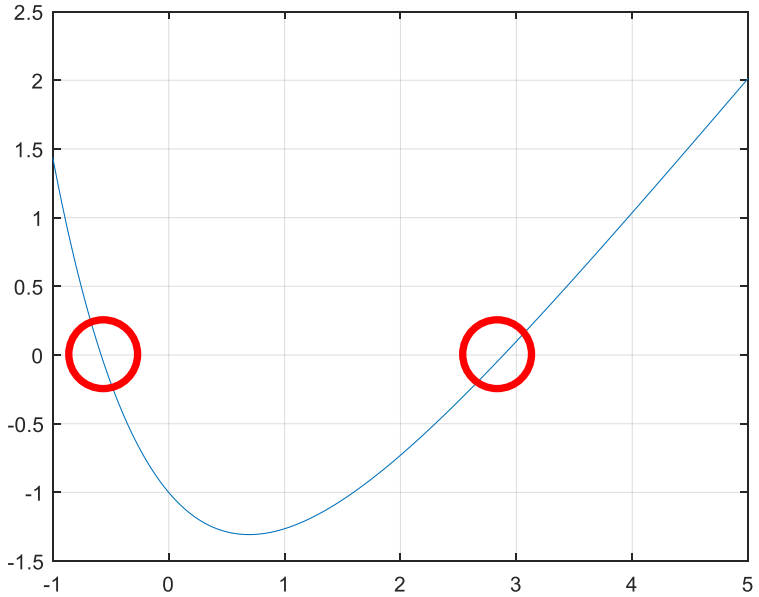
Zeri della funzione

Per trovare gli zeri di una funzione si può utilizzare il comando fzero, che ha la sintassi fzero(expr, x0), dove expr è la stringa che definisce la funzione. Questo comando restituisce un valore prossimo ad x0 in cui la funzione si annulla. Tuttavia, questo comando determina solo i punti in cui la funzione attraversa l'asse delle ascisse; quindi non determina i punti in cui la funzione è tangente all'asse x.

```
>> x = -1 : 0.001 : 5;
>> y=x+2*exp(-x)-3;
>> plot(x,y), grid on
```

Esaminando il grafico ottenuto si vede che f presenta due zeri in prossimità di x = -0.5 e di x = 2.8. Utilizziamo il comando fzero per ottenere un risultato più preciso, riscrivendo la funzione e precisando l'intorno della ricerca (-0,5):

```
>> fzero('x+2*exp(-x)-3',-0.5)

ans =

   -0.5831
```

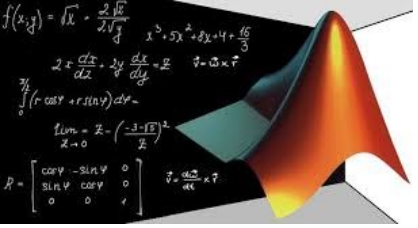


dipartimento di ingegneria unipa

Prof. Fabio Viola

LEAP

Laboratory of Electrical APplication



Function

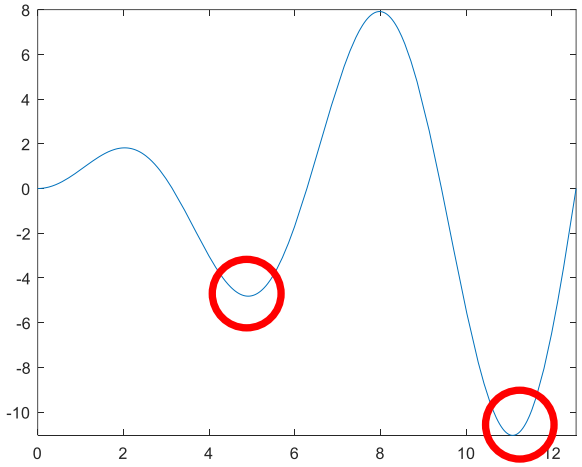
MATLAB® & SIMULINK®

Ricerca dei minimi
Per trovare i minimi di una funzione si può utilizzare il comando fminbnd, che ha la sintassi fminbnd(expr, x1, x2), dove expr è la stringa che definisce la funzione. Questo comando restituisce un valore che rende minima la funzione nell'untervallo [x1, x2]. Occorre però utilizzare questo comando con cautela. L'algoritmo sottostante questo comando inizia col cercare il primo punto di minimo in corrispondenza del quale la pendenza della funzione è nulla. Se ne trova uno allora termina, altrimenti esamina i valori che la funzione assume agli estremi dell'intervallo. Quindi il comando fminbnd da una risposta sbagliata quando la funzione possiede più punti di minimo a tangente orizzontale ed il primo minimo non è assoluto, oppure quando la funzione possiede almeno un punto di minimo a tangente orizzontale, ma assume il suo valore minimo in un estremo dell'intervallo. Conviene sempre disegnare il grafico della funzione per controllare che i risultati siano corretti
Consideriamo la funzione f(x) = x sin x sull'intervallo [0, 4 p]. Allora utilizzando il comando fminbnd si ha

```
>>fminbnd('x*sin(x)',0,4*pi)
ans =
4.9132
```

dal quale si vede che la funzione in x = 4.9132 presenta solo un minimo locale e che assume invece il suo valore minimo nell'intervallo [10, 12]. A questo punto, quindi, possiamo determinare il vero minimo della funzione

```
>>fminbnd('x*sin(x)',10,12)
ans =
11.0856
```

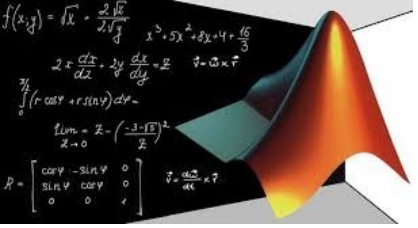


osservazione
Se la funzione avesse avuto dominio [0, 10] il minimo sarebbe stato in 10, non rilevato dalla funzione.:
>>10*sin(10) ans = -5.4402
fminbnd('x*sin(x)',0,10) ans = 4.9132 4,9132*sin(4.9132) = -4,8145

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Function

MATLAB® & SIMULINK®

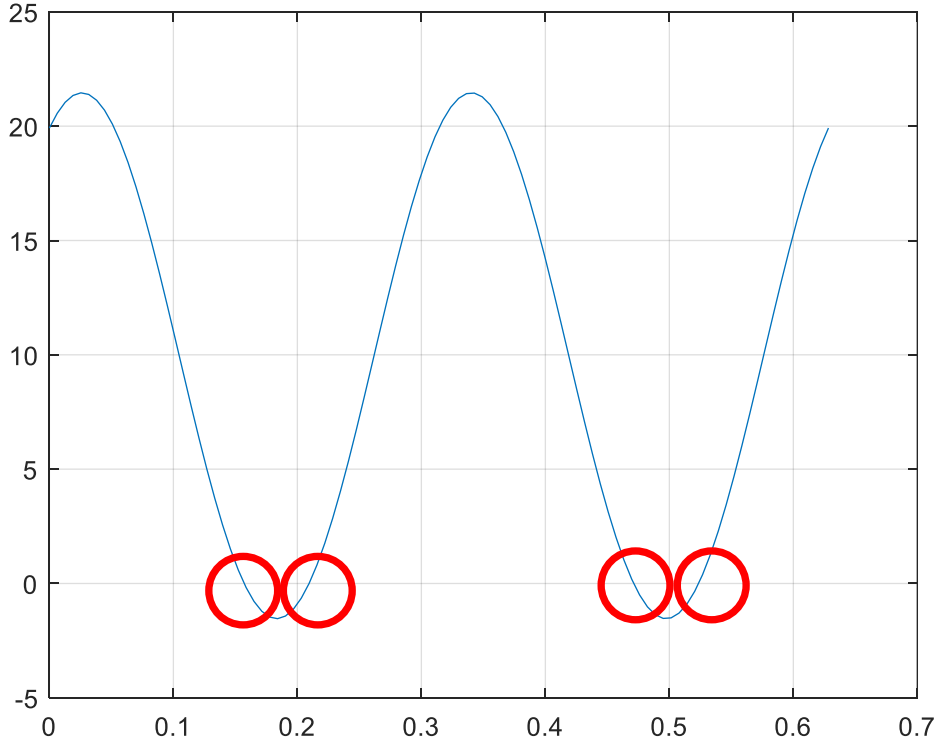
Problemi
Rappresentare le seguenti funzioni con x=linspace(0, 2*pi/10,100) , con 10 pari alla pulsazione angolare

1) $y1(x) = 23\cos(10x)$

2) $y2(x) = 1 * cos(10x - \pi/6)$

3) $y3(x) = 23\cos(10x) * 1\cos(10x - \pi/6)$

Rappresentare le stesse funzioni nello stesso grafico
Per le funzioni utilizzando il comando grid on, evidenziare i passaggi per lo zero ed individuarli esattamente con fzero

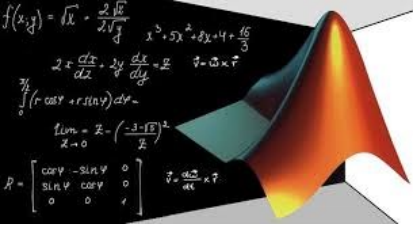


Osservazioni
Se si utilizza una lettera diversa da x, ad esempio t, la funzione fzero non funziona.
Per trovare i punti di zero che sono ravvicinati, bisogna fornire un punto di lavoro a sinistra di entrambi per il più piccolo, e a destra di entrambi per il più grande

dipartimento di ingegneria unipa

Prof. Fabio Viola

LAP
Laboratory of Electrical APplication



Function

MATLAB® & SIMULINK®

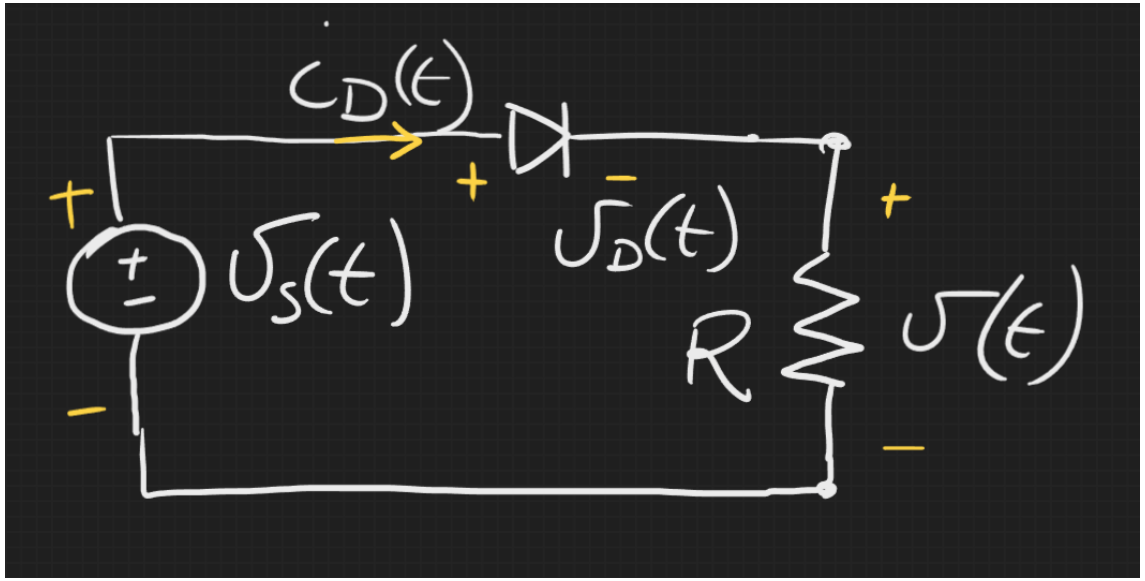
Circuiti non lineari: Diodo
L'analisi tenuta fino ad adesso è basata sul comportamento lineare della rete, per la quale vale il principio di sovrapposizione degli effetti. In presenza di elementi non lineari come il diodo tale proprietà è negata. Bisogna procedere con tecniche non lineari per la risoluzione delle reti.

La legge di Kirchhoff per le tensioni afferma:
$-v_s(t) + v_D(t) + Ri_D(t) = 0$

Con corrente sul diodo:
$i_D(t) = I_S\left(e^{v_D(t)/V_T} - 1\right)$

Che porta ad avere un problema non lineare:
$-v_s(t) + v_D(t) + RI_S\left(e^{v_D(t)/V_T} - 1\right) = 0$

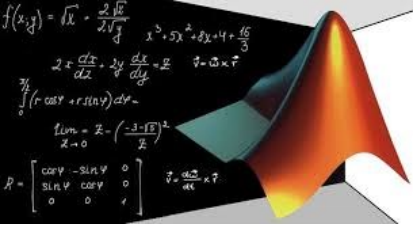
Da risolvere con tecnica iterativa.
Si utilizzano uno script per definire il problema e comandare l'algoritmo iterativo ed una funzione, che costituisce l'algoritmo iterativo



dipartimento di ingegneria unipa

Prof. Fabio Viola

LEAP
Laboratory of Electrical APplication



Function

MATLAB® & SIMULINK®

Circuiti non lineari: Diodo

Si fissa
$v_s(t)$=1*sin $(2\pi t)$
Con frequenza f pari ad 1 Hz
T0 periodo 1 secondo
R = 330 Ohm
N = 256 numero di punti temporali
Si utilizza la funzione fsolve per trovare l'insieme di valori x= $v_D(t)$ che risolvono KVL:

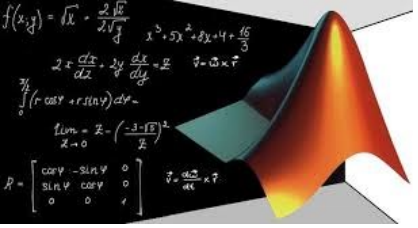
$$KVL = v_s(t) + x(t) + RI_S\left(e^{x(t)/V_T} - 1\right) = 0$$

```
clear all; clc
% circuito non lineare con diodo
global vs R % definisco vs e R variabili globali, note anche nelle funzioni richiamate
f = 1; w = 2*pi*f; T0 = 1/f; % parametri della sorgente
N = 256; T = T0/N; % campioni e step temporale
n = 0:N-1; t = n*T; %istanti di valutazione
vs = sin(w*t); % funzione sinusoidale in ingresso
R = 330; % resistenza
vd_init = zeros(1,N); % possibili valori nulli per vd
% applicazione di risolutore non lineare per gli N valori incogniti
[vd,F_val,exit_flag] = fsolve(@KVL,vd_init);
if exit_flag == 1
F_mag = norm(F_val); % norma euclidea
v = vs-vd; % tensione in uscita sul resistore
plot(t,v,'r'); hold on; plot(t,vs,'k'); plot(t,vd,'b');
grid on; xlabel('tempo - sec'); ylabel('volt')
title('Analisi del comportamento non lineare del Diodo')
else
disp('non sono in grado di risolvere.')
end
legend('tensione in uscita' , 'tensione in ingresso', 'tensione sul diodo')
```

dipartimento di ingegneria unipa

Prof. Fabio Viola

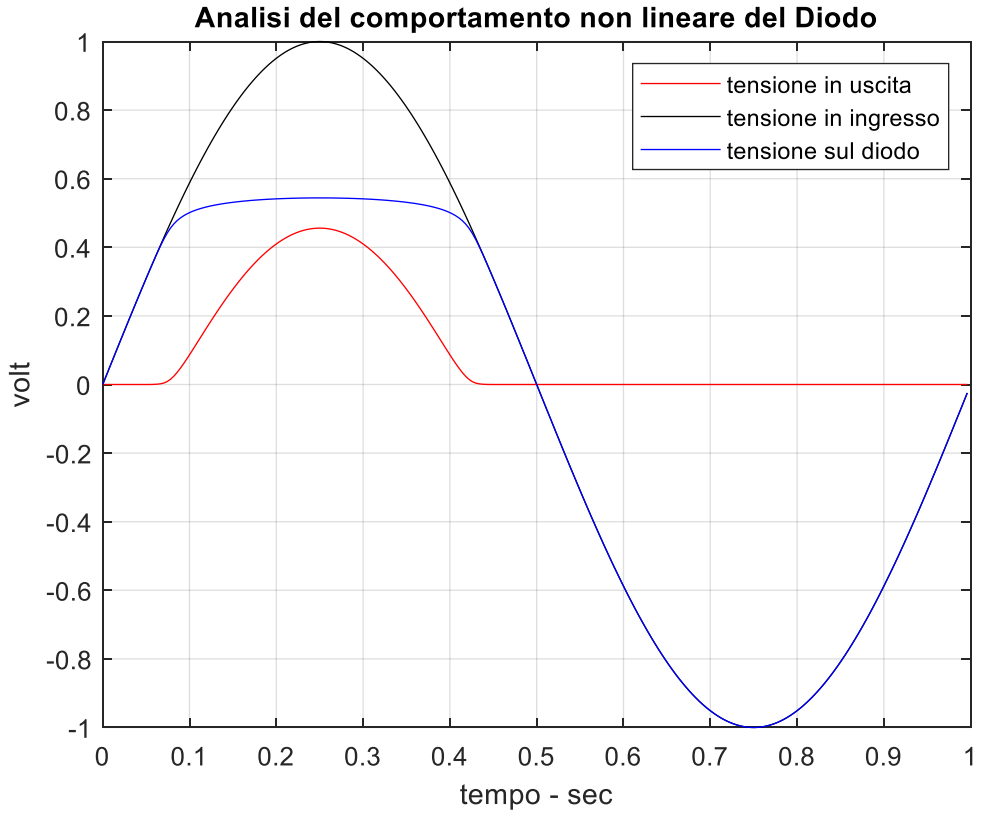
LAP
Laboratory of Electrical APplication



Function

MATLAB® & SIMULINK®

```
function F = KVL(x)
% evaluate the KVL equation
global vs R
I_S = 1e-12; % corrente di saturazione
V_T = 25.85e-3; % tensione a 300 Kelvin (kT/q)
% k di Boltazmann T temperatura, q carica elettrone)
id = I_S*(exp(x/V_T)-1); % currente di diodo
F = -vs + x + R*id; % equazione KVL negli N punti
end
```



dipartimento di ingegneria unipa

Prof. Fabio Viola

Laboratory of Electrical APplication