

# INTRODUZIONE A MATLAB e SIMULINK

*Corso di Sistemi di Controllo Digitale*

## Organizzazione delle Lezioni

- INTRODUZIONE A MATLAB
- DEFINIZIONE DI VARIABILI, MATRICI E VETTORI
- FUNZIONI ELEMENTARI PER SCALARI E MATRICI
- POLINOMI
- VISUALIZZAZIONE DI GRAFICI
- NOTE GENERALI SU MATLAB (help, istruzione di ambiente...)
- ALCUNE ISTRUZIONI PER APPLICAZIONI DI CONTROLLO
- SIMULINK

22/01/2012

1

## Introduzione

MATLAB (MATrix LABoratory) è un linguaggio di programmazione per applicazioni scientifiche (elaborazione numerica dei segnali, progetto di simulatori, sintesi di sistemi di controllo, ecc.)

MATLAB è un interprete di comandi. I comandi possono essere forniti interattivamente o contenuti in files su disco (M-files)

Comprende un vasto set di funzioni predefinite e numerose librerie (toolbox) per svariate applicazioni

Le potenzialità di MATLAB possono essere facilmente estese (è semplice creare nuovi toolbox)

E' possibile convertire un programma MATLAB in codice C e C++ in modo automatico

22/01/2012

2

## Variabili ed Espressioni

All'avvio di MATLAB appare il prompt ">"

Vi sono due tipi di istruzioni:

assegnamenti "» *variabile = espressione*"

valutazione di espressioni "» *espressione*"

La valutazione di un'*espressione* genera una matrice che viene assegnata alla *variabile* indicata. Quando nell'istruzione non si specifica la *variabile* a cui assegnare il risultato, la valutazione dell'espressione viene assegnata alla variabile di sistema "**ans**".

Se un'espressione non termina con ";" il risultato della sua valutazione viene mostrato anche sullo schermo.

22/01/2012

3

## Variabili ed Espressioni

Esempio:

```
» 8+2          » b = 6+a;
ans =          » b
    10         b =
» a = 5*ans    56
a =
    50
```

In MATLAB le variabili non devono essere dichiarate. La dichiarazione coincide con il primo assegnamento.

MATLAB è case-sensitive.

22/01/2012

4

## Matrici

Una matrice può essere definita con la sintassi seguente:

```
» A = [7 8; 8.9 7; 9 8]      » B = [1 2 3
A =                          4 5 6]
    7.0000  8.0000          B =
    8.9000  7.0000          1  2  3
    9.0000  8.0000          4  5  6
```

uno spazio o una virgola delimitano gli elementi di una stessa riga

un punto e virgola o un cambio di riga indicano la fine di una riga

Sono presenti funzioni predefinite per la generazione di particolari matrici:

“zeros(n,m)” matrice di zeri

“ones(n,m)” matrice di uni

“eye(n,m)” matrice identità

“rand(n,m)” matrice di numeri casuali

“diag([a11, a22, a33, ..., aNN])” matrice diagonale

22/01/2012

5

## Matrici

Per accedere agli elementi di una matrice:

```
» A = [7 8; 8.9 7; 9 8]      » A(2,:)
A =                          ans =
    7.0000  8.0000          8.9000  7.0000
    8.9000  7.0000          » A(:,1)
    9.0000  8.0000          ans =
» A(1,2)                7.0000
ans =                   8.9000
    8                    9.0000
```

A(n,m) estrae l'elemento (n,m) della matrice A

A(n,:) estrae l'n-esima riga della matrice A

A(:,m) estrae l'm-esima colonna della matrice A

22/01/2012

6

## Vettori

Un vettore può essere creato con la stessa sintassi utilizzata per le matrici oppure con le istruzioni:

```
» x = 1:6
x =
    1    2    3    4    5    6

» x = 0.5:0.1:0.9
x =
    0.5000  0.6000  0.7000  0.8000
    0.9000

» x = linspace(-1,1,4)
x =
   -1.0000  -0.3333  0.3333  1.0000
```

a : [step :] b crea un vettore riga di estremi a e b. Il parametro opzionale step indica l'intervallo tra ciascun elemento del vettore

linspace(a,b,N) crea un vettore riga di estremi a e b, costituito da N punti equispaziati

22/01/2012

7

## Funzioni Elementari per Scalari

Gli operatori aritmetici presenti in MATLAB sono:

+ (somma), - (differenza), \* (prodotto), / (quoziente), ^ (elevamento a potenza)

Funzioni matematiche elementari:

abs valore assoluto di un numero complesso  
angle fase di un numero complesso espressa in rad/s  
conj complesso coniugato  
exp esponenziale in base e  
real, imag parte reale e parte immaginaria di un numero complesso  
log, log10 logaritmo naturale ed in base 10  
sqrt radice quadrata

Funzioni trigonometriche

sin, cos seno, coseno  
tan tangente  
asin, acos arco seno, arco coseno  
atanarco tangente

Le variabili *i* e *j* sono predefinite uguali alla radice quadrata di -1

## Funzioni Elementari per Scalari

Altre costanti predefinite:

pi pigreco  
Inf infinito  
NaN Not a Number (generata da 0/0, o Inf/Inf)

## Funzioni Elementari per Matrici

Gli operatori elementari sono:

+ , - , \* , / , \ , .\* , ./ , .^

L'operazione di somma o di sottrazione è definita tra matrici aventi le stesse dimensioni. Se uno dei due operandi è uno scalare, esso viene sommato o sottratto a tutti gli elementi della matrice.

$X = B/A$  è la soluzione dell'equazione  $X*A = B$

$X = A \setminus B$  è la soluzione dell'equazione  $A*X = B$

.\*, ./ e .^ effettuano le corrispondenti operazioni sui singoli elementi delle matrici coinvolte.

Le funzioni matematiche elementari e trigonometriche, quando applicate alle matrici, si riferiscono ai singoli elementi della matrice

Principali operazioni matriciali:

Matrice trasposta  $A'$   
Matrice inversa  $\text{inv}(A)$

## Funzioni Elementari per Matrici

Determinante	$\text{det}(A)$
Autovalori	$\text{eig}(A)$
Polinomio caratteristico	$\text{poly}(A)$
Rango	$\text{rank}(A)$
Dimensioni	$\text{size}(A)$

## Polinomi

Un polinomio è rappresentato da un vettore riga che ne contiene i coefficienti in ordine decrescente delle potenze del polinomio medesimo. Il polinomio  $3s^3 + 2s + 8$  si rappresenta come:

```
» pol = [3 0 2 8]
pol =
    3     0     2     8
```

Per ottenere gli zeri di un polinomio:

```
» roots(pol)
ans =
    0.6136 + 1.3403i
    0.6136 - 1.3403i
   -1.2273
```

Per valutare un polinomio in un punto:

```
» polyval(pol, 1)
ans =
    13
```

## Visualizzazione di Grafici

La funzione **plot** produce grafici bidimensionali e può essere chiamata con diverse modalità.

E' possibile rappresentare sullo stesso grafico più curve, impostare una griglia, impostare delle etichette per gli assi, ecc...

Esempio:

```
» Tempo = [-30:0.1:30];
» Tau = -5;
» y = exp(Tempo/Tau);
» z = 300*sin(Tempo*pi/4);
» plot(Tempo,y)
» plot(Tempo,y,'r',Tempo,z,'y'),grid,title('ESERCIZIO')
» plot(Tempo,y,'r',Tempo,z,'y:'),grid,title('ESERCIZIO')
```

Altre funzioni che consentono la visualizzazione di grafici 2-D, sono:

loglog	grafico logaritmico in x e y
semilogx	grafico logaritmico in x e lineare in y
semilogy	grafico logaritmico in y e lineare in x

## Visualizzazione di Grafici

bar	grafico a barre
stairs	grafico a scala
mesh	grafico 3D

Altre funzioni di utilità sono  
title, xlabel, ylabel, grid  
axis, hold, clf, shg, subplot

## Altre Istruzioni Utili

help	richiama l'help in linea
help comando	visualizza l'help relativo al comando indicato
who/whos	elencano le variabili in uso
dir	elenca i files contenuti nel direttorio corrente
clear all	elimina tutte le variabili della sessione corrente
clear var1 var2	elimina le variabili var1 e var2

## Descrizione di un Sistema Lineare

- Matlab consente di descrivere sistemi lineari a tempo continuo, a segnali campionati e a tempo discreto.
- Nel caso di sistemi a segnali campionati occorre precisare la variabile Ts (tempo di campionamento) . Per i sistemi a tempo discreto si pone Ts = 1.
- Per descrivere un sistema dato in termini di funzione di trasferimento, occorre:
  - sistema a tempo continuo:
 

```
>> sistema = tf(num,den);
```

 dove *num* e *den* sono vettori con cui si esprimono i polinomi numeratore e denominatore in potenze decrescenti di s.  
Questo comando darà in output l'espressione della funzione di trasferimento del sistema a tempo continuo.
  - sistema a tempo discreto:
 

```
>> sistema = tf(num,den,1);
```

 dove *num* e *den* sono vettori con cui si esprimono i polinomi numeratore e denominatore in potenze decrescenti di z e 1 indica che il tempo di campionamento Ts è 1.  
Questo comando darà in output l'espressione della funzione di trasferimento del sistema a tempo discreto.

22/01/2012

16

## Descrizione di un Sistema Lineare

- Supponiamo, per esempio, di voler descrivere un sistema dinamico a tempo continuo con funzione di trasferimento

$$G(s) = \frac{s + 1/3}{s^2 - 1/12s - 1/12}$$

- Con i comandi: 

```
>>num = [1 1/3];
```

```
>>den = [1 -1/12 -1/12];
```

 si definiscono i polinomi a numeratore e denominatore.  
Con il comando: 

```
>> sistema = tf(num, den)
```

 si ottiene in output

Transfer function:

s + 0.3333

-----  
s^2 - 0.08333 s - 0.08333

22/01/2012

17

## Descrizione di un Sistema Lineare

- Per definire, invece, il sistema a tempo discreto con funzione di trasferimento

$$G(z) = \frac{z + 1/3}{z^2 - 1/12z - 1/12}$$

resteranno invariati i polinomi che descrivono il numeratore ed il denominatore usati in precedenza. Ciò che varia è il comando finale per definire il sistema, che ora diventa: 

```
>> sistema = tf(num, den, 1)
```

.

- Tale comando produce come output:

Transfer function:

z + 0.3333

-----  
z^2 - 0.08333 z - 0.08333

22/01/2012

18

## Istruzioni utili per il Tracciamento delle Risposte di un Sistema Lineare

Dopo aver descritto il sistema lineare in esame nel modo opportuno, a seconda che sia a tempo discreto o continuo, Matlab offre alcune funzioni per valutare la risposta di tale sistema a diversi tipi di ingresso.

1. Risposta impulsiva: con il comando 

```
>> impulse(sistema, t)
```

 Matlab calcola e grafica la risposta all'impulso del sistema; *t* è il vettore che definisce il tempo di durata della simulazione.
2. Risposta a scalino: con il comando 

```
>> step(sistema, t)
```

 Matlab calcola e grafica la risposta allo scalino del sistema; *t* è il vettore che definisce il tempo di durata della simulazione.
3. Risposta ad un ingresso generico: con il comando 

```
>> lsim(num, den, u, t)
```

 Matlab calcola e grafica la risposta del sistema all'ingresso generico, *u*; *t* è il vettore che definisce il tempo di durata della simulazione.

22/01/2012

19

## Istruzioni utili per il Tracciamento dei Diagrammi della Risposta in Frequenza

- Matlab offre anche la possibilità di analizzare sistemi dinamici lineari a tempo continuo e discreto nel dominio della frequenza:
  - Diagramma di Bode:** con il comando `>> bode (sistema)` Matlab visualizza automaticamente il diagramma di modulo e fase della funzione di trasferimento del sistema in esame.
  - Diagramma di Nyquist:** con il comando `>> nyquist (sistema,w)` Matlab traccia il diagramma di Nyquist della funzione di trasferimento del sistema in esame in corrispondenza delle pulsazioni specificate dal vettore w.
- Spesso è utile, per analizzare le caratteristiche di un sistema lineare a tempo continuo o discreto, analizzarne il luogo delle radici.
- Con il comando `>> rlocus (sistema)` Matlab traccia il luogo delle radici del sistema in esame.

22/01/2012

20

## Diagrammi di Bode per Sistemi a Tempo Discreto

- Vediamo ora in dettaglio come ottenere il diagramma di Bode di un sistema a tempo discreto. Ad esempio: per rappresentare in Matlab la funzione di trasferimento

$$G(z) = \frac{z - \frac{1}{2}}{z^2 - \frac{5}{6}z + \frac{1}{6}}$$

si useranno i comandi: `>> num = [1 -1/2];`  
`>> den = [1 -5/6 1/6];`  
`>> sistema = tf(num,den,1);`

ottenendo in output: *Transfer function:*  

$$\frac{z - 0.5}{z^2 - 0.8333 z + 0.1667}$$

22/01/2012

21

## Diagrammi di Bode per Sistemi a Tempo Discreto

- Con il comando `>> bode (sistema)`

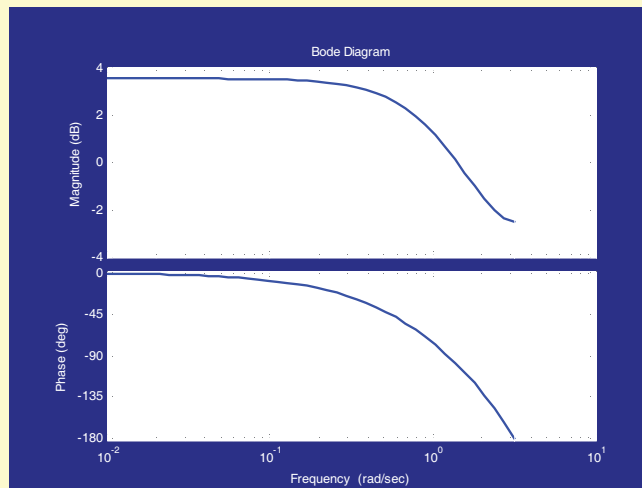
si ottiene il diagramma di Bode di modulo e fase per il sistema a tempo discreto, calcolato come

$$G(e^{j\omega})$$

Per l'esempio si ottiene il grafico mostrato.

Si noti che il diagramma è tracciato fino alla pulsazione  $\pi$ .

La ragione è che  $G(e^{j\omega})$  è una funzione periodica di periodo  $2\pi$ .



22/01/2012

## Diagrammi di Bode per Sistemi a tempo Discreto: Filtro Passa - Tutto

- Supponiamo ora di voler tracciare il diagramma di Bode di un filtro passa-tutto con funzione di trasferimento:

$$G(z) = \frac{z + 4}{z + 1/4}$$

- Per definire il sistema a tempo discreto avente  $G(z)$  come funzione di trasferimento e tracciare il relativo diagramma di Bode si usano i comandi:

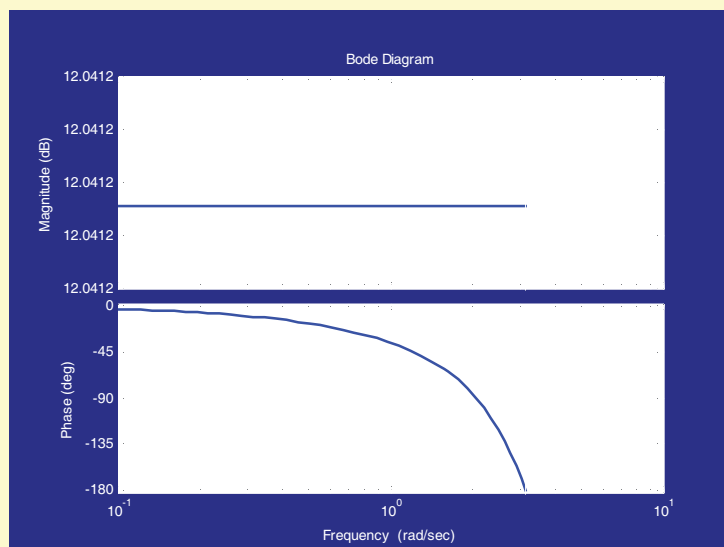
```
>> num = [1 4];
>> den = [1 1/4];
>> sistema = tf(num,den,1);
>> bode (sistema)
```

22/01/2012

23

## Diagrammi di Bode per Sistemi a Tempo Discreto: Filtro Passa - Tutto

La figura mostra il diagramma di Bode di modulo e fase del filtro passa tutto precedentemente introdotto.



22/01/2012

## Risposta ad un Ingresso Generico di Sistemi a Tempo Discreto: Filtro Passa - Tutto

- Per completare l'esempio di analisi del filtro passa – tutto introdotto precedentemente, valutiamo la risposta di tale sistema ad un segnale di ingresso descritto come un'onda quadra.
- Il primo passo da compiere è creare il vettore di ingresso desiderato:
  - Il comando: `>> [u,t] = gensig('square',10,40,1)` consente di creare il vettore degli ingressi  $u$  che, in questo caso, è un'onda quadra ('square'), di periodo 10, durata 40 e "tempo di campionamento"  $T_s = 1$  (coerentemente col fatto che il sistema da simulare è a tempo discreto). Il vettore  $t$  è il vettore dei tempi che si userà per la simulazione.
  - Utilizzando la funzione `lsim` precedentemente introdotta, è possibile simulare e visualizzare su un grafico la risposta del sistema a tale ingresso.
  - Il comando `>> lsim(sistema,u,t)` consente di simulare la risposta del sistema che descrive il filtro passa – tutto all'onda quadra in ingresso  $u$ .

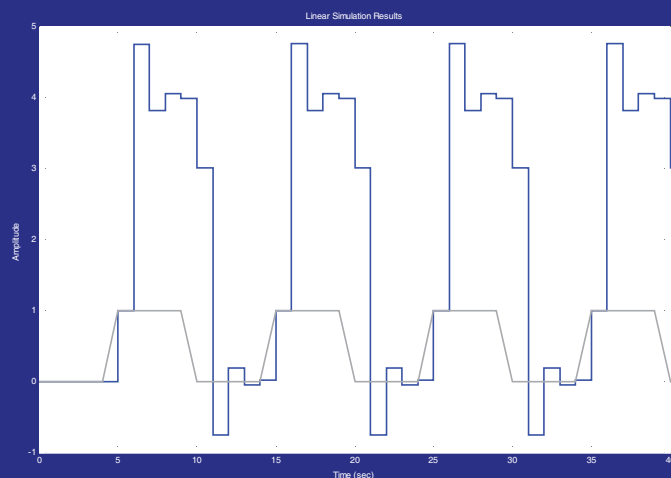
22/01/2012

25

## Risposta ad un Ingresso Generico di Sistemi a Tempo Discreto: Filtro Passa - Tutto

La figura mostra la risposta del filtro passa tutto all'ingresso  $u$ .

Il segnale in ingresso è visualizzato in colore grigio, mentre l'uscita in blu.



22/01/2012


## Simulink

- Simulink è un ambiente grafico che consente di descrivere e simulare sistemi complessi. La simulazione di un sistema dinamico con Simulink consiste in due fasi fondamentali:
  1. La prima fase consiste nella creazione di un modello grafico del sistema da simulare, che può consistere nell'interconnessione di blocchi continui e discreti, lineari e non lineari. Questo modello descriverà le relazioni matematiche esistenti tra gli ingressi e le uscite del sistema.
  2. La seconda fase consiste nell'utilizzare l'ambiente Simulink per simulare il comportamento del sistema durante una sua evoluzione temporale su un arco di tempo stabilito dall'utente. Di fatto Simulink utilizza le informazioni contenute nel modello grafico, che l'utente deve specificare nella fase di descrizione, per generare le equazioni dinamiche del modello e risolvere numericamente il problema di simulazione in esame.

22/01/2012

27

# Simulink

- Simulink interagisce direttamente con il Workspace di Matlab. I modelli descritti, pertanto, potranno contenere variabili contenute nel Workspace della sessione corrente.
- Allo stesso modo, i risultati della fase di simulazione possono essere passati direttamente al Workspace sotto forma di nuove variabili, pronte per essere analizzate.
- Per aprire l'ambiente Simulink si può, alternativamente, digitare `>> simulink` nel prompt, oppure cliccare sull'icona  presente nel toolbar di Matlab.

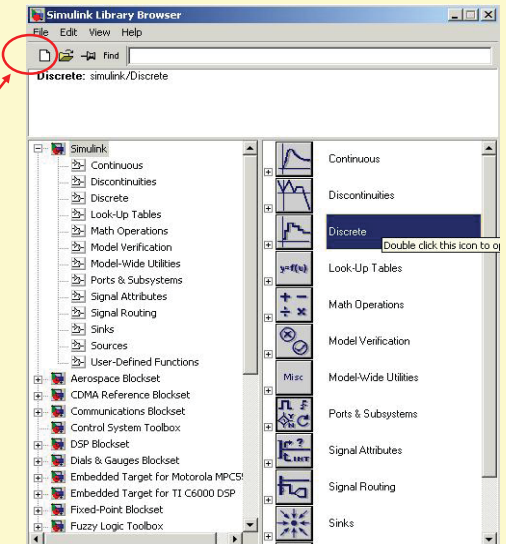
22/01/2012

28

# Simulink

Una volta aperto Simulink è disponibile la libreria dei modelli mostrata in figura, che consente di scegliere i blocchi predefiniti per la descrizione del sistema desiderato.

Cliccando su questa icona si apre un nuovo file in cui disegnare il modello, trascinando nel file i modelli base che sono contenuti nelle varie librerie.



22/01/2012

29

## Simulink: Esempio

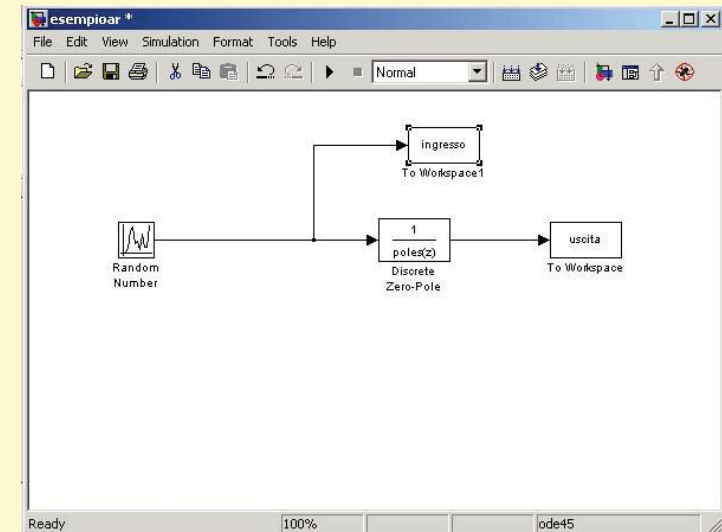
- Supponiamo di voler simulare con Simulink un sistema AR(3), asintoticamente stabile, i cui poli siano  $z_i = 0.2, 0.5, 0.3$ ,  $i=1,2,3$  alimentato da un rumore bianco  $e(t) \approx WN(0,1)$
- Per far questo dovremo utilizzare, per descrivere il segnale di ingresso il blocco *Random Number* della libreria *Sources*. Per la descrizione della funzione di trasferimento useremo il blocco *Discrete Zero-Pole* della libreria *Discrete*, che consente di descrivere la funzione di trasferimento specificandone poli e zeri.
- Infine, per poter analizzare l'output ottenuto via simulazione, utilizzeremo due blocchi *To Workspace* della libreria *Sinks*, che consentono di avere disponibili nel Workspace i risultati della simulazione per fare le analisi desiderate.

22/01/2012

30

## Simulink: Esempio

Trascinando nel file i blocchi descritti precedentemente ed interconnettendoli opportunamente, si ottiene il modello in figura. Il passo successivo da compiere, prima di eseguire la simulazione, è quello di specificare i parametri dei vari blocchi perché descrivano il sistema desiderato.



22/01/2012

31



# Simulink: Esempio

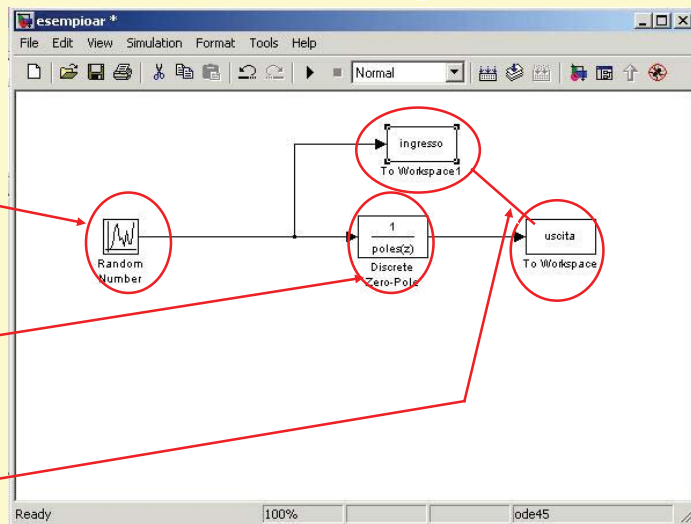
Facendo doppio clic sui blocchi, si aprono le finestre di dialogo in cui inserire i vari parametri

Specificare media e varianza del segnale di ingresso.

Specificare zeri e i poli della funzione di trasferimento

Specificare il nome della variabile in cui salvare i dati ed il formato in cui salvarla

22/01/2012

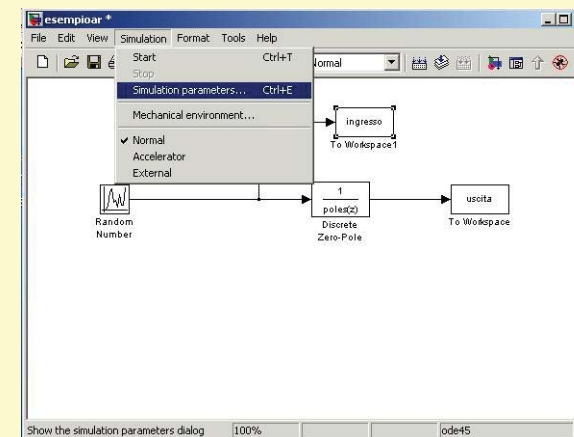


32

# Simulink: Esempio

- Prima di lanciare la simulazione, bisogna fare un ultimo passo: fissare i parametri della simulazione stessa.

Selezionando la voce **Simulation** del menu a tendina, compaiono le voci che si vedono in figura. Per impostare i parametri della simulazione selezionare la voce **Simulation Parameters**.



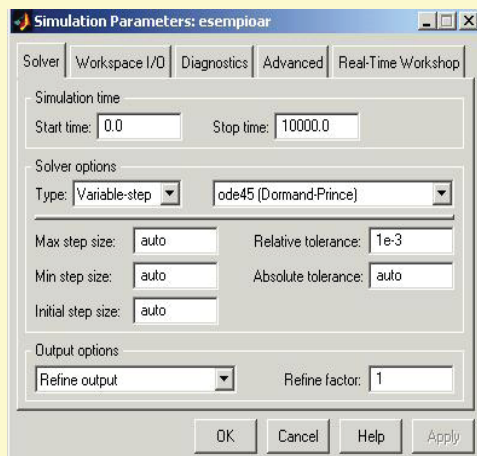
22/01/2012

33

# Simulink: Esempio

In questa finestra l'utente deve specificare


- gli istanti di inizio e fine simulazione;
- Il metodo numerico da utilizzare per la soluzione delle equazioni (se non ci sono esigenze particolari va bene quello di default);
- I parametri del solutore numerico (nella maggior parte dei casi vanno bene quelli di default).

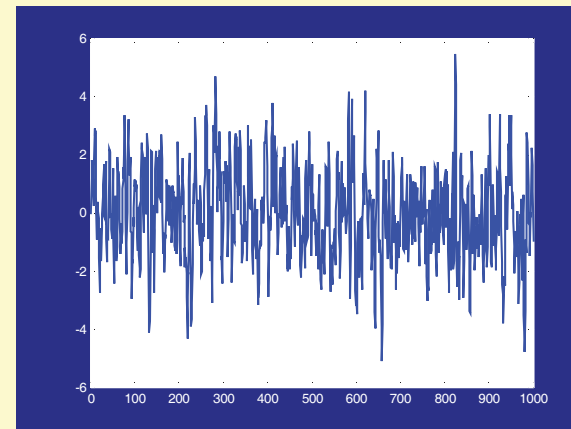


22/01/2012

34

# Simulink: Esempio

- A questo punto, cliccando sull'icona  del toolbar di Simulink si lancia la simulazione. Quando questa termina, digitando nel Workspace, ad esempio, il comando `>>plot(uscita)`, si ottiene il grafico che mostra l'andamento nel tempo dell'uscita del processo ottenuta via simulazione.



22/01/2012

35