

1 SCILAB

Scilab è un prodotto free sviluppato dall'INRIA per applicazioni di controllo di sistema e signals processing.

E' composto di tre parti distinte:

- un interprete
- librerie di funzioni (Scilab procedure)
- librerie di routines in C o Fortran

Una caratteristica chiave di Scilab risulta essere la semplicità di manipolazione delle matrici: operazioni basilari sulle matrici , come concatenazione, estrazione o trasposizione vengono realizzate come semplici operazioni quali addizioni e sottrazioni.

Anche polinomi e matrici polinomiali sono definite e la sintassi usata per manipolare questi oggetti è la stessa di quella usata per vettori costanti e matrici numeriche.

Scilab fornisce una varietà di primitive per l'analisi di sistemi non-lineari. Scicos toolbox permette la definizione grafica e la simulazione di complesse interconnessioni di sistemi ibridi.

Infine, Scilab è facilmente interfacciabile con funzioni Fortran o C.

Dopo aver lanciato il programma è possibile editare i comandi dal prompt o lanciare il file in cui sono stati precedentemente scritti i comandi d'interesse (vedi Fig. 1). Affinchè il file venga eseguito correttamente è necessario spostarsi nella directory dove si trova il file, come mostrato nella figura seguente.

Per eseguirlo basta selezionare “ Exec” da “File” e poi scegliere il file da eseguire.

Scilab è fornito di un HELP soddisfacente. Per avere informazioni su qualsiasi funzione chiamare l'Help (come in Fig. 2) e nel campo “Aprop” digitare la funzione d'interesse.

Si può notare che mano a mano che si digitano le lettere nella finestra sopra vengono visualizzate tutte le voci che iniziano con le lettere che stiamo digitando . Ciò è utile quando non si conosce esattamente il nome della funzione che realizza l'operazione di nostro interesse.

Si può anche scegliere un argomento generale (es. Graphic Library) e vedere le funzioni relative.

Selezionata la voce cliccare su “SHOW” e verrà visualizzata una pagina contenente tutte le informazioni necessarie.

Un manuale on-line e alcuni esempi possono essere trovati ai seguenti indirizzi:

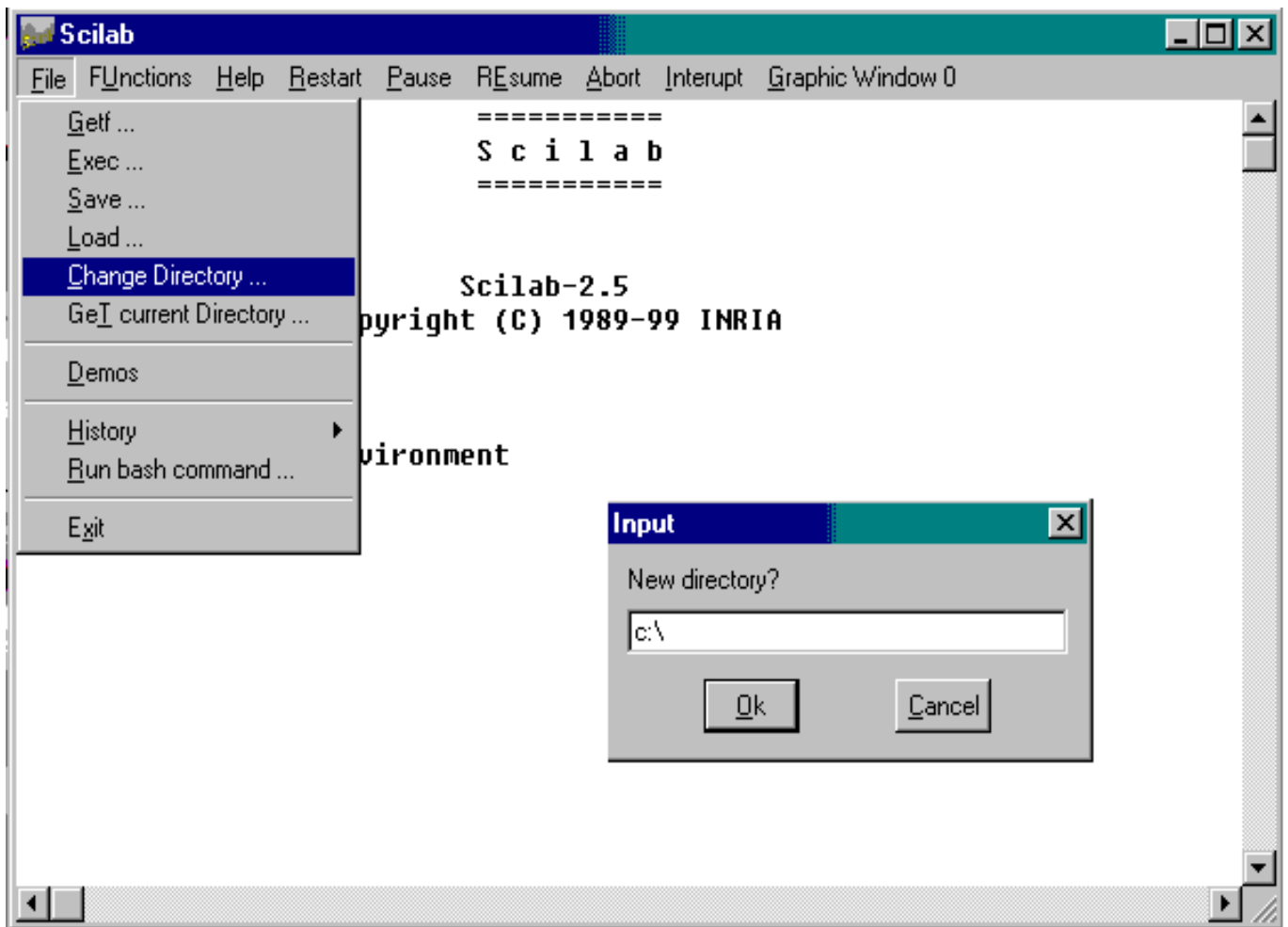


Figura 1: Scelta della Directory di lavoro

- <http://www-rocq.inria.fr/scilab/doc/intro/intro.html>
- <http://poincare.dma.unifi.it/scilab/demos.html>

2 AMBIENTE

Scilab viene caricato con una serie di variabili e primitive. Il comando *who* visualizza la lista delle variabili già definite

Le variabili possono essere salvate in un file esterno utilizzando *save* e queste variabili possono essere caricate in Scilab utilizzando il comando *load*.

Si noti che dopo il comando *clear < nome_variabibile >* la viaribile non esiste più nell'ambiente in uso e con il comando *clear* senza alcuna variabile si cancellano tutte le viariabili presenti nell'ambiente.

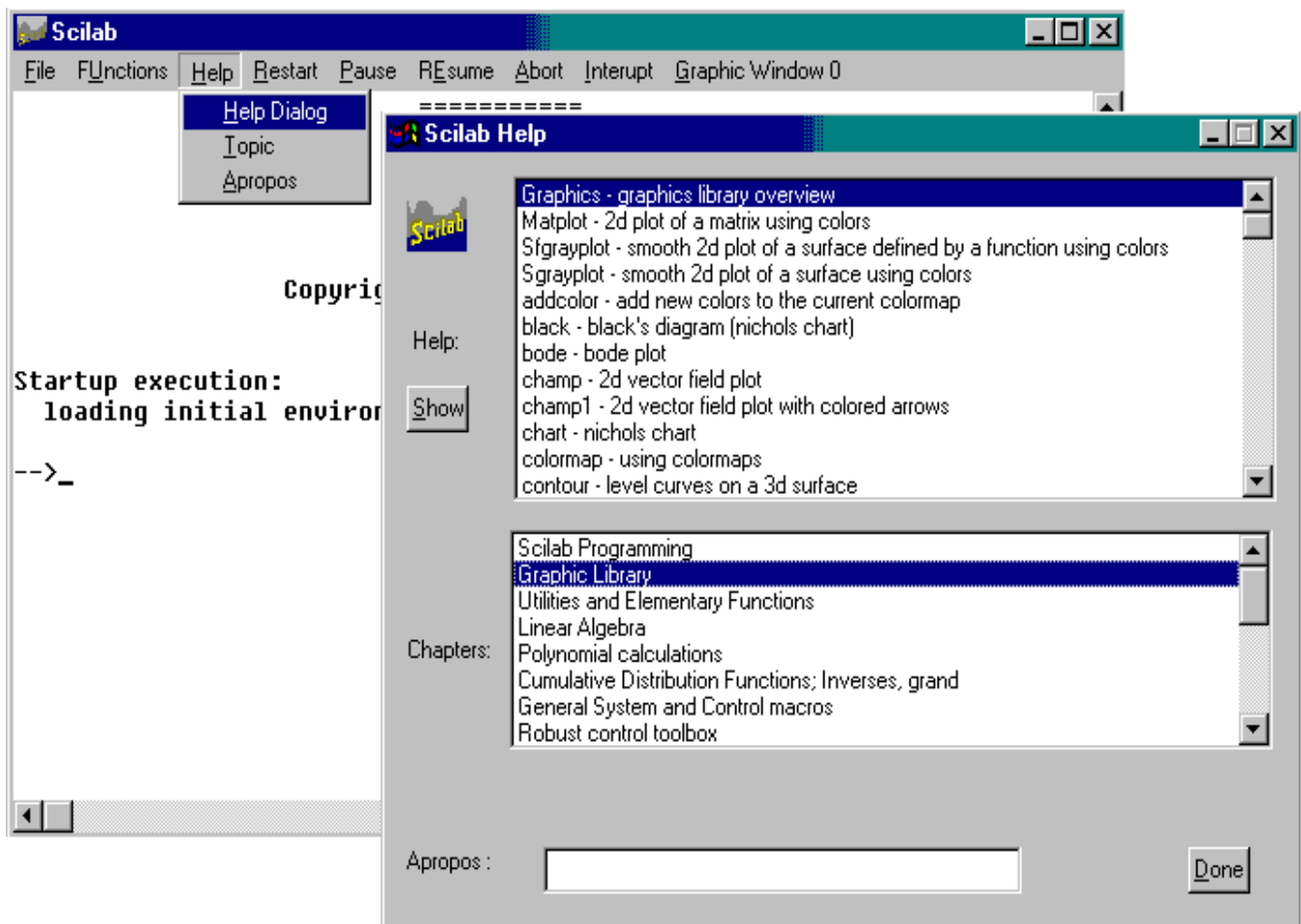


Figura 2: Help

3 MATEMATICA

3.1 SCALARI

Assegnazione di un valore ad una variabile:

$$a = 1;$$

Con il simbolo '==' viene restituito un booleano, senza che venga assegnato il valore:

$$a == 1$$

$$ans = T$$

Per gli scalari sono definite tutte le operazioni.

3.2 POLINOMI

Per definire un polinomio si fa uso della funzione 'poly':

$$p = \text{poly}([0 \ 2 \ 3], 'x');$$

definisce un polinomio con variabile letteraria x e radici in zero, due e tre, con il coefficiente del termine di grado massimo sempre pari ad uno.

Se non si conoscono le radici allora si definisce la variabile letteraria di nostro interesse (ad es. 'x') come un polinomio con radice in zero:

$$x = \text{poly}(0, 'x');$$

Si può ora scrivere qualsiasi polinomio in x nel seguente modo:

$$p = 2 + 3 * x + x^2$$

Per conoscere le radici del polinomio p appena dichiarato si utilizza la funzione 'roots':

$$[r] = \text{roots}(p)$$

$$r =$$

$$-1.$$

$$-2.$$

sono possibili tutte le normali operazioni fra polinomi: divisioni, moltiplicazioni, addizioni e sottrazioni come fra scalari

$$r = x/p$$

$$r = \frac{x}{2 + 3x + x^2}$$

$$r = x * p$$

$$r = 2x + 3x^2 + x^3$$

3.3 MATRICI

DEFINIZIONE DI UN VETTORE

$$\text{vet} = [0 : 1 : 10]$$

Definisce un vettore con elementi da 0 a 10 con passo 1

$$vet = 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10$$

Nella seguente tabella vengono riportate le possibili operazioni sulle matrici:

[]	matrix definition, concatenation
;	row separator
()	extraction m=a(k)
()	insertion: a(k)=m
'	transpose
+	addition
-	subtraction
*	multiplication
\	left division
/	right division
^	exponent
.*	elementwise multiplication
.\	elementwise left division
./	elementwise right division
.\^	elementwise exponent
.*.	kroncker product
./.	kroncker right division
.\.	kroncker left division

DEFINIZIONE DI UNA MATRICE

$$A = [1\ 2\ 3; 4\ 7\ 9; 18\ 3\ 0]$$

$$A =$$

$$1.2.3.$$

$$4.7.9.$$

$$18.3.0.$$

DETERMINANTE

$$det(A);$$

INVERSA

$$invr(A);$$

oppure

inv(*A*);

ESTRAZIONE DELLA PRIMA RIGA

A(1,:)

ans = 1.2.3.

ESTRAZIONE DELL'ULTIMA COLONNA

A(:,\$)

ans =

3.

9.

0.

4 GRAFICI

Per visualizzare tutte le funzioni utilizzabili per la visualizzazione di un grafico nella finestra dell'Help evidenziare la voce “ Graphic Library”. In generale le funzioni utilizzate sono le seguenti, per ognuna vanno specificati il vettore delle x (ascisse) e quello delle y (ordinate):

- *plot*(*x*,*y*)
- *plot2d*(*x*,*y*)
- *plot2d1*(*x*,*y*) //assi logaritmici

Per aprire una seconda finestra grafica basterà dare i comandi

xset('window',2);*xselect*(2);

che rispettivamente istanziano una seconda finestra e la selezionano. Per tornare alla prima finestra (che si apre automaticamente con l'uso dei comandi di *plot* visti prima) basterà impartire il comando

xselect(1)

Invece, l'uso della funzione *xbasc*() permette di “pulire” l'eventuale finestra-grafico già aperta evitando così la sovrapposizione di più grafici.

5 FUNZIONI

Le funzioni sono insiemi di comandi che vengono eseguiti in un nuovo ambiente isolando così le variabili delle funzioni da quelle iniziali. Esse possono essere create e eseguite in diversi modi. Alle funzioni possono essere passati dei parametri; esse possono contenere cicli *while*, *for*..., possono essere richiamate da altre funzioni...

Per definire una funzione direttamente dal prompt di Scilab si usa il comando *def f*

Esempio

```
def f('x' = foo(y)', 'if y > 0 then, x = 1; else, x = -1; end')
```

```
foo(5)
```

```
ans = 1.
```

```
foo(-3)
```

```
ans = -1.
```

E' però più comodo usare un editor di testo e caricare la funzione in Scilab con *getf('filename')*. La prima riga del file dev essere : *function[y1, ..., yn] = nome_funzione(x1, ..., xk)* dove le *yi* sono gli output e le *xi* sono le variabili di input. Ovviamente, il nome della funzione che dovrà essere utilizzato è quello definito come *nome_funzione* e non il nome del file. Un file, peraltro, può contenere più definizioni di funzioni.

- CICLI

in Scilab esistono due tipi di cicli : *for* e *while*.

- FOR

```
x = 1; for k = 1 : 4, x = x * k, end
```

```
x = 1.
```

```
x = 2.
```

```
x = 6.
```

```
x = 24.
```

- WHILE

Il ciclo *while* esegue ripetutamente una sequenza di comandi fino a quando la condizione non è soddisfatta

```
x = 1; while x < 14, x = 2 * x, end
```

```

x=2.
x=4.
x=8.
x=16.

```

- ISTRUZIONI CONDIZIONALI

Si possono utilizzare due tipi di istruzioni condizionali : “IF-THEN-ELSE” e “SELECT-CASE”

– IF THEN ELSE

```
x=1;
```

if x > 0 then, y = -x, else, y = x, end

```
y= - 1.
```

```
x=-1;
```

if x > 0 then, y = -x, else, y = x, end

```
y= - 1.
```

– SELECT CASE

```
x=-1
```

```
x =- 1.
```

select x, case 1, y = x + 5, case - 1, y = sqrt(x), end

```
y= i
```

6 CONTROL TOOLBOX

Scilab è provvisto di molteplici funzioni utili nell’analisi dei segnali

DEFINIZIONE DI UN SISTEMA LINEARE (Funzione di trasferimento)

```
s = poly(0,'s')
```

```
sl = syslin('c',1/(s * s + 0.2 * s + 1))
```

sl è un istema lineare continuo (la lettera *c* sta ad indicare proprio che si tratta di un sistema continuo; per il caso discreto si deve specificare *d*)

$$sl = \frac{1}{s * s + 0.2 * s + 1}$$

Del sistema lineare possono essere visualizzati il diagramma di BODE, di NYQUIST e di NICHOLS utilizzando le seguenti funzioni:

- `bode(sl)`
- `nyquist(sl)`
- `chart(list(1,0,2,3))`
- `black(sl,0.01,100)`

Gli ultimi due servono per graficare la carta di Nichols e il diagramma rispettivamente. Tutte le funzioni graficano direttamente i rispettivi diagrammi, come si può vedere dalla Fig. 3:

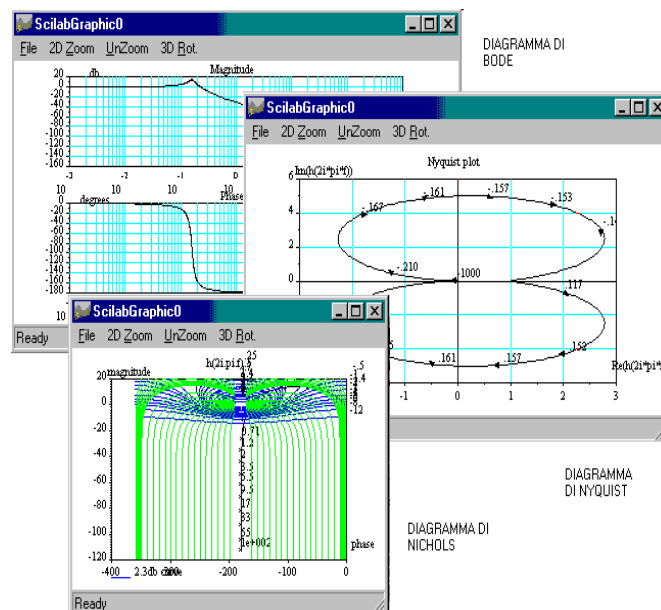


Figura 3: Diagrammi di Bode Nyquist Nichols

Per gli esempi che seguono definiamo un vettore degli istanti di tempo :

```
instants=0:0.05:20;
```

vettore degli istanti di tempo

- risposta al gradino


```
y = csim('step', instants, sl);
```
- risposta al gradino traslato


```
def f('in' = u(t)', if t < 3 then in = 0; else in = 1; end');
y1 = csim(u, instants, sl);
plot2d(instants', y1');
```

- Risposta impulsiva

```
yi = csim('imp', instants, sl);
xbase();
plot2d(instants', yi');
```

- Discretization

```
dt = 0.05;
ld = dscr(tf2ss(sl), 0.05);
```

Dove sl è il sistema nel tempo continuo

EQUAZIONI DIFFERENZIALI

E' possibile risolvere sistemi di equazioni differenziali del tipo

$$\dot{y} = f(t, y), y(t_0) = y_0$$

dove y può essere anche un vettore, mediante l'uso della funzione *ode*;

$$y = ode(y_0, t_0, t, f)$$

dove:

- y_0 : vettore che esprime le condizioni iniziali
- t_0 : istante iniziale
- t : vettore degli istanti di tempo per i quali è valutato y
- f : funzione esterna di cui è stata definita la sintassi

Per maggiori informazioni vedere l'Help sulla voce *ode*

Esempio

$$\dot{y} = y^2 - y \sin(t) + \cos(t), y(0) = 0$$

$$def f([ydot] = f(t, y)', ydot = y^2 - y * \sin(t) + \cos(t)')$$

f è il nome della funzione ; t ed y sono le variabili di input di f , e viene restituito $ydot$ che contiene le derivate;

$$y_0 = 0; t_0 = 0; t = 0 : 0.1 : \%pi;$$

$$y = ode(y_0, t_0, t, f)$$

$$plot(t, y)$$

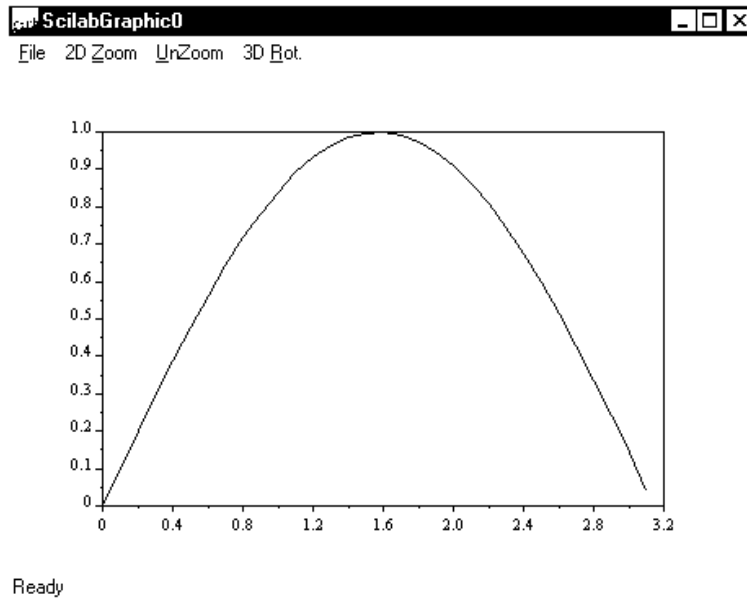


Figura 4: Grafico della funzione calcolata con *ode*

7 INTERFACCIA CON PROGRAMMI C E FORTRAN

Scilab può interfacciarsi con programmi scritti in C o in Fortran. Naturalmente la prima cosa da fare è un link a questi programmi. Per ciò si usa il comando *link* seguito dal comando *fort* che trasmette le variabili definite in scilab (matrici, stringhe..) ai programmi linkati e trasforma in variabili di Scilab gli output di tali funzioni.

Il comando *link('path/pgm.o','pgm',flag)* connette il programma già compilato *pgm* a Scilab. Qui *pgm.o* è un object file che si trova nella directory *path* e *pgm* è il punto d'ingresso (program name) nel file *pgm.o*. Il valore di *flag* deve essere settato a 'C' per un programma in C e a 'F' per il Fortran. ('F' è di default e può essere omissso). Se l'operazione di link è OK Scilab ritorna un intero *n* associato a questo programma. Per eliminare il link digitare *ulink(n)*. Il comando *c_link('pgm')* ritorna true se *pgm* correttamente linkato a Scilab e false altrimenti. Per maggiori informazioni vedere l' help di *link* e di *fort*.

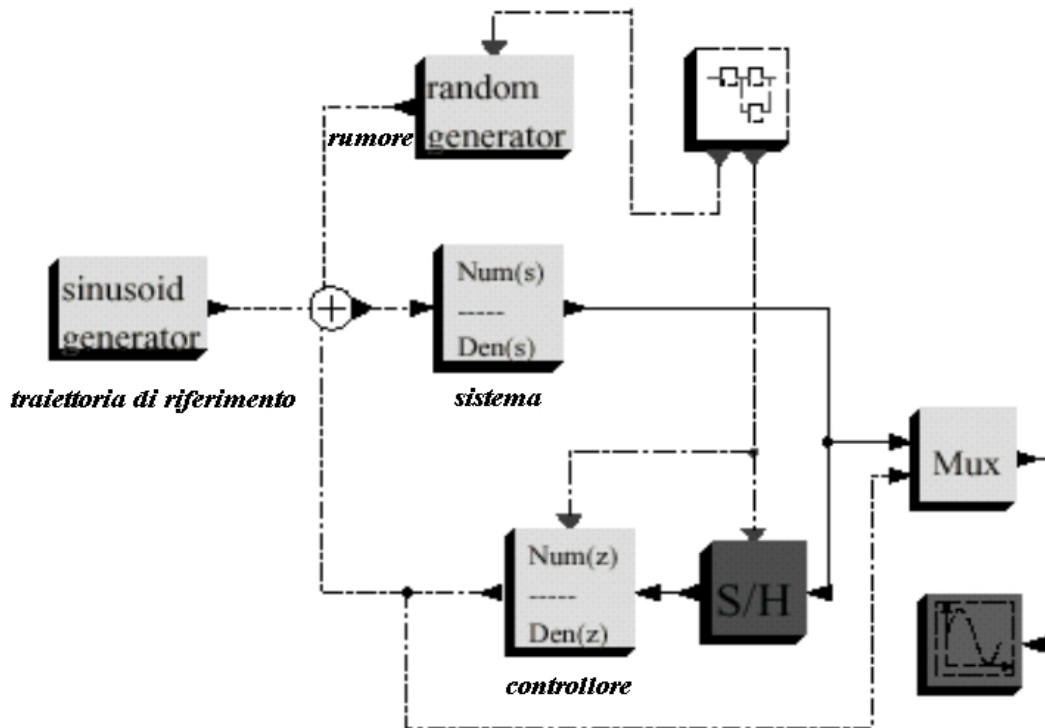


Figura 5: Esempio di sistema simulato con *Scicos*

8 SCICOS

8.1 INTRODUZIONE

SCICOS (Scilab Connected Object Simulator) è una libreria di Scilab che permette di simulare sistemi dinamici, sia continui che discreti. Si avvale di un editor grafico, che permette di costruire modelli tramite l'interconnessione di *blocchi* (che rappresentino funzioni base o definite dall'utente). Ogni blocco agisce come un *sub-sistema*: è attivato da un segnale di *input* e fornisce come *output* un segnale dipendente dall'ingresso e dal suo stato. E' possibile visualizzare i vari output connettendoli con blocchi appositi ("*MScope*" block).

In Scicos è associato con ogni segnale un insieme di indici di tempo, detti istanti di attivazione, in cui evolve il segnale. Per istanti diversi da quelli di attivazione, segnali Scicos rimangono costanti (come si può vedere dalla figura). L'insieme degli istanti di attivazione è l'unione di intervalli di tempo e di punti isolati detti eventi. I segnali sono generati da blocchi pilotati da segnali di attivazione. Un segnale di attivazione porta il blocco ad adeguare il suo output in funzione dell'input e del suo stato interno. Il segnale di output potrà a sua volta essere usato per alimentare altri blocchi. I blocchi

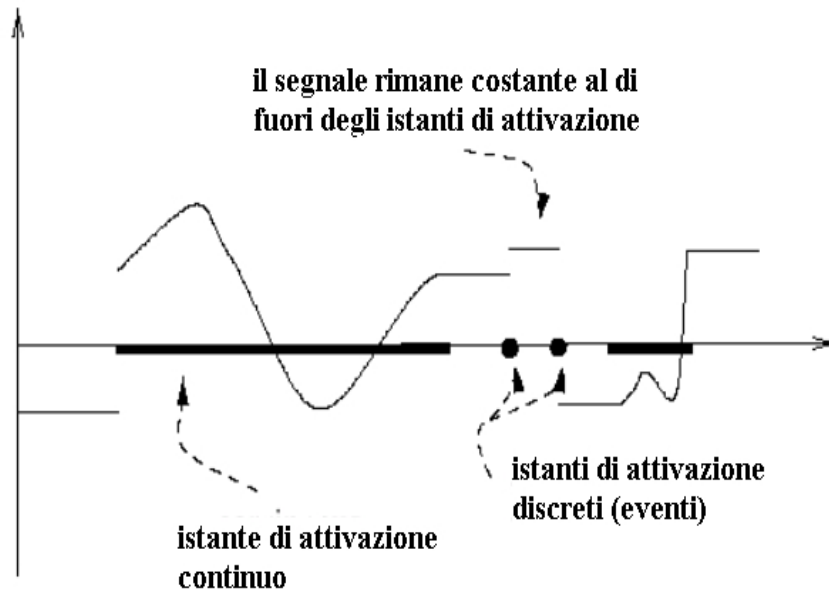


Figura 6: tempo continuo e tempo discreto - gli istanti di attivazione

sono attivati da segnali di attivazione che sono immessi dall' input di attivazione, nella parte alta del blocco. Un blocco che non abbia porte per input di attivazione è da considerarsi sempre attivo oppure eredita i suoi istanti di attivazione dall'unione degli istanti di attivazione dei suoi segnali in input. Le porte in basso nei blocchi sono le porte di output di attivazione. I segnali in uscita sono segnali di attivazione generati dal blocco. Ad esempio, il blocco "clock" genera un segnale di attivazione costituito da un treno di eventi regolarmente distribuiti nel tempo. Se colleghiamo quest'output con la porta input di attivazione di un blocco "scope" (MScope), questo specifica a quali istanti di tempo gli effettivi input dello scope saranno visualizzati.

8.2 L'EDITOR DI SCICOS

In Scicos i sistemi sono modellati dall'interconnessione tra blocchi e sub-sistemi; i blocchi possono essere trovati in varie *palette* o essere definiti dall'utente. Scicos ha un'interfaccia grafica di facile uso per la creazione di diagrammi. Per avviare l'editor ed aprire la finestra principale è necessario digitare "scicos()"; da Scilab. Costruire un modello in Scicos porterà allora tipicamente a:

- aprire alcune palette (dall' edit menu)
- copiare i blocchi dalle palette alla finestra di Scicos (selezionando copy dal menu edit, cliccando poi sul componente desiderato ed infine sulla finestra principale)
- commettere le porte di input e output (selezionando link dal menu edit e cliccando poi le porte d output e di input)

Si noti che per far sì che un link si generi da un altro link si deve selezionare il tasto link dal menu di edit, selezionare un link esistente, nel punto in cui si deve biforcare e poi la destinazione del link (ad esempio la porta input di un blocco). Si può interrompere il processo di creazione di un link ed eliminare l'ultimo link creato cliccando sul tasto destro del mouse. E' oltretutto opportuno piazzare in qualsiasi diagramma Scicos almeno un blocco scope o "scrivi su file", per poter visualizzare o salvare gli esiti della simulazione. E' possibile vedere alcuni esempi tra i demo di Scicos.

8.3 ADATTARE I PARAMETRI

I parametri dei blocchi possono essere modificati aprendo le finestre degli stessi, usando il pulsante "*open/set*". Si potranno così modificare alcuni parametri, usando espressioni Scilab. Tali espressioni saranno memorizzati simbolicamente e poi valutate. Il contesto del diagramma può essere editato cliccando il tasto Context, ed eventualmente valutato con il tasto "*Eval*". Ciò è necessario solo se le modifiche al contesto includono cambiamenti nei valori di parametri usati precedentemente.

8.4 SIMULAZIONE

Quando viene premuto il tasto run nel menu simulate il programma viene (se è opportuno) compilato e comincia la simulazione. Questa potrà essere interrotta premendo il tasto stop nella finestra principale di Scicos. Un file Scicos, salvato nel formato *.cos, non necessiterà di essere compilato quando sarà caricato; il file salvato contiene i risultati della simulazione. Utilizzando la funzione *scicosim* è possibile estrarre solo i dati desiderati riguardo una simulazione, e avviare una simulazione senza entrare in ambiente Scicos.

8.5 ALTRE FUNZIONALITA'

Sono previste dall'editor altre funzionalità:

- salvare e caricare programmi Scicos (in vari formati)
- ingrandire e cambiare punto di vista
- cambiare l'aspetto ed i colori dei blocchi
- inserire del testo nei diagrammi
- stampare ed esportare i diagrammi di Scicos

Il tasto "*help*" permette di ottenere aiuti in vari modi; si può cliccare *help* e poi il blocco o il bottone di interesse, per avere il relativo manuale. Un'altra importante caratteristica di Scicos è la possibilità

di creare sub-sistemi. Infatti non sarebbe auspicabile costituire un sistema complesso con centinaia di componenti nello stesso diagramma. Per questo, Scicos prevede la possibilità di raggruppare insieme dei blocchi, definendo dei super-blocchi. Ogni super-blocco si comporta come un blocco qualsiasi, ma può avere al suo interno un numero indeterminato di blocchi o anche di super-blocchi.

8.6 BLOCCHI BASE

I blocchi di base disponibili in Scicos sono di 3 tipi: "Regular Basic Block", "Zero Crossing Basic Blocks" e "Synchro Basic Blocks". Ogni blocco può avere due tipi di porte di input e di output: porte per input regolari, per input di attivazione, per output regolari, per output di attivazione. Ogni tipo di input o di output deve essere collegato con un link adeguato (regolare o di attivazione). Le porte di input di attivazione sono poste in cima ad ogni blocco, quelle di output di attivazione sul fondo.

- Regular Basic Block: può rappresentare variabili continue x o discrete z
- Zero Crossing Basic Blocks : generano output solo se almeno un input regolare cambia segno. L'evento generato può dipendere dalla combinazione degli eventi che hanno cambiato segno. Per esempi sugli ZBB vedi la paletta Threshold
- Synchro Basic Blocks : blocchi di sincronizzazione; possono avere più input o output di attivazione, ma hanno un solo input regolare; appena ricevono in input un segnale di attivazione generano un segnale di attivazione in output scegliendolo in base al regular input. Per esempi vedi blocco event select oppure if-then-else

8.7 DIPENDENZA DAL TEMPO ED EREDITARIETA'

Per evitare l'utilizzo esplicito di tutti i segnali di attivazione è stata aggiunta in Scicos la possibilità di ereditare l'attivazione. In particolare, non è necessario che un blocco abbia input di attivazione : questo erediterà il segnale di attivazione automaticamente dai suoi input regolari. I blocchi che sono attivi permanentemente, detti tempo dipendenti, non hanno bisogno di porte di input di attivazione (e non ereditano).

8.8 COSTRUZIONE DI BLOCCHI

Un nuovo blocco può essere costruito come *SuperBlock* semplicemente interconnettendo blocchi base e compilandolo. Si potranno così definire, come per in blocco base, le funzioni di interfaccia (che si occupi dell'interazione con l'utente) e computazionale (comportamento dinamico).

Le funzioni di interfaccia devono essere sempre scritte come funzioni Scilab. Le funzioni computazionali possono essere scritte in *C*, in *Fortran* o nel linguaggio di *Scilab*. Si possono trovare degli esempi nella cartella *< SCIDIR > /routines/scicos*.

Consultando il manuale originale sar  possibile comprendere la sintassi da rispettare per la creazione di funzioni di interfaccia e computazionali.

8.9 ESEMPI

Per costruire il proprio sistema vi si devono trascinare dalle palette i blocchi di interesse.

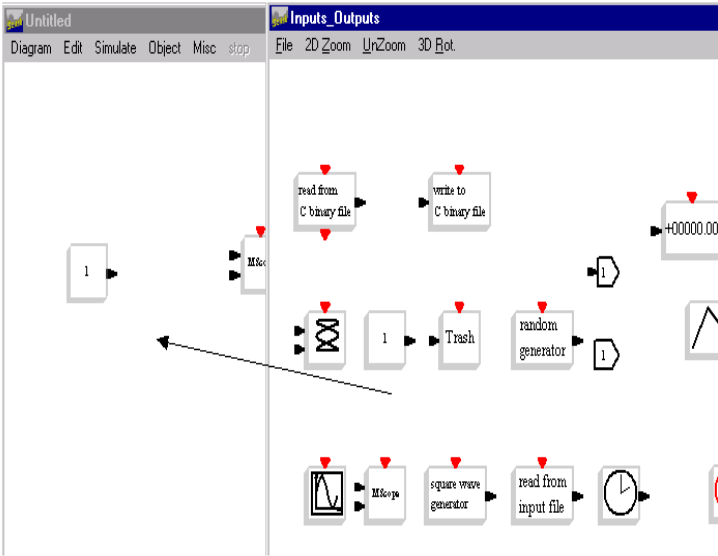


Figura 7: la paletta input-output

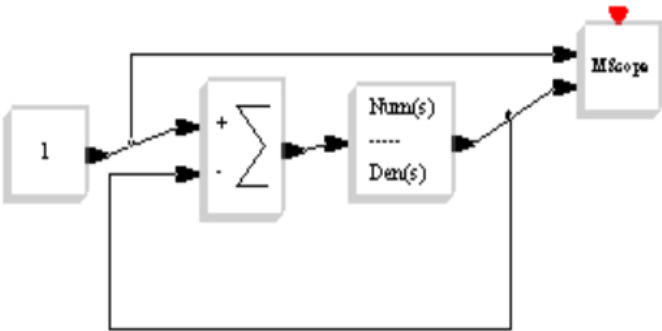


Figura 8: un semplice sistema

Sar  possibile trovare gli esempi sul sito [www. dia.uniroma3.it](http://www.dia.uniroma3.it).....INSERIRE!!!!!!

ESEMPIO 1

Per realizzare il nostro primo esempio abbiamo scelto di inserire dalle palette una sorgente di tipo sinusoidale, un sistema e uno "scope". E' stato necessario inserire un blocco "clock" per poter eseguire la simulazione.

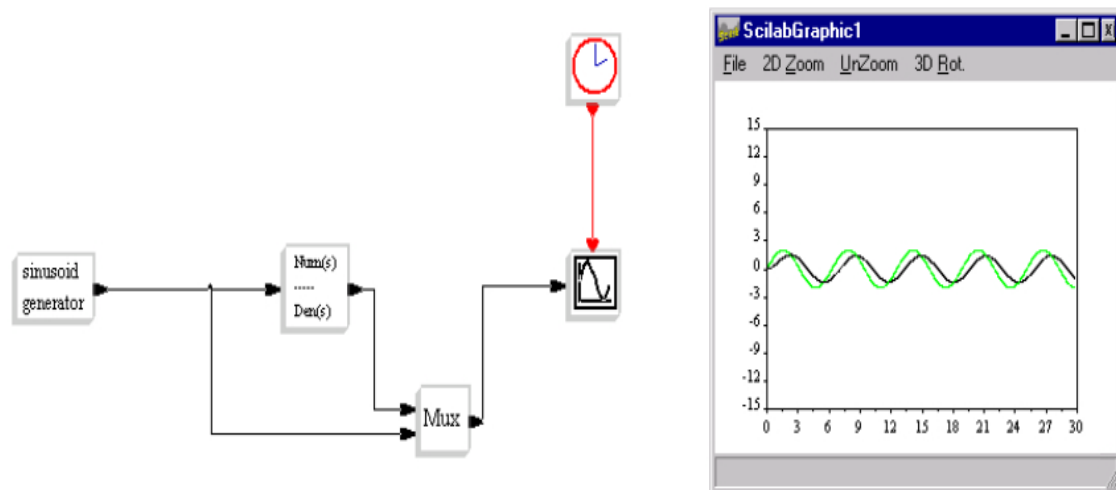


Figura 9: esempio 1

I parametri della sinusoide sono stati settati nel seguente modo:

- magnitude : 2
- frequency : 1
- phase : 0

Per quanto riguarda il clock:

- period : 0,001
- init time : 0,1
- phase : 0

Il sistema :

- num : 1
- den : s+1

ESEMPIO 2

Nel secondo esempio abbiamo applicato all'esempio precedente, ovvero sinusoide e sistema a retroazione, (nel nostro caso unitaria). Sono rimasti, ovviamente, i blocchi "scope" e "clock".

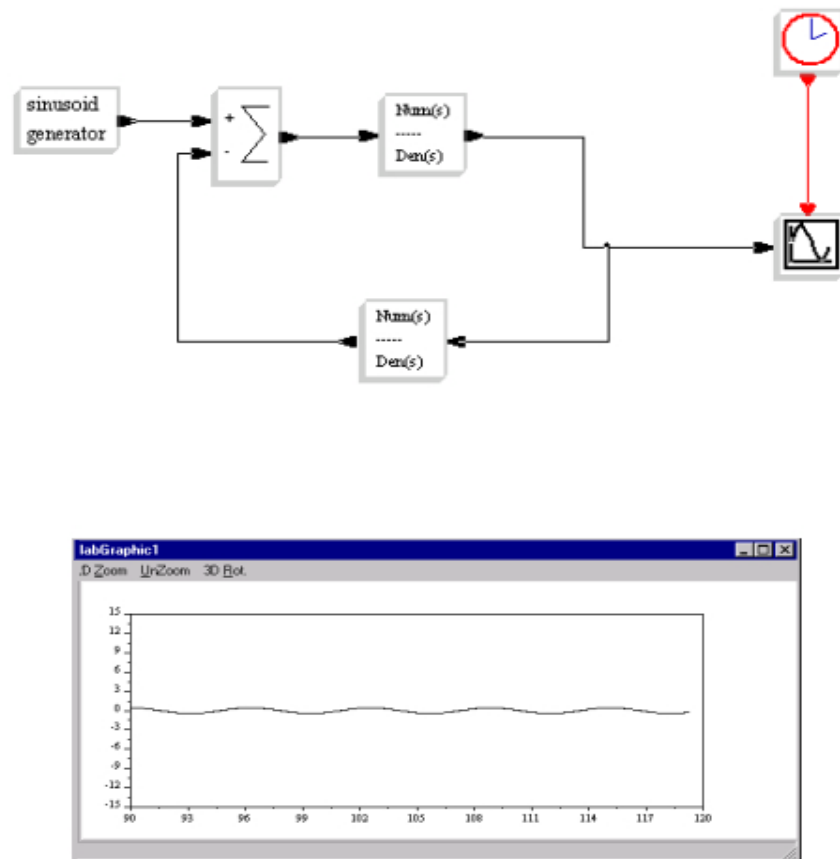


Figura 10: esempio 2

I parametri della sinusoide sono stati settati nel seguente modo:

- magnitude : 1
- frequency : 1
- phase : 0

Per quanto riguarda il clock:

- period : 0,001
- init time : 0,1

- phase : 0

Il sistema :

- num : 1
- den : s+1

La retroazione :

- num : 1
- den : 1

8.10 CONCLUSIONI

Abbiamo voluto fornire solo una breve descrizione delle potenzialità di Scicos. E' vivamente consigliato l'utilizzo dei manuali delle funzioni Scicos e dei demo Scicos. Spesso è più semplice cominciare con un demo che con un diagramma vuoto.