Esercitazione 2

Istruzioni trattate: operatori con punto prefisso (`.op`), costrutto `function`, `plot`, `linspace`, `clf`, `xgrid`, `xtitle`, `xlabel`, `ylabel`, `plot2d`, `legend`.

Nella prima parte di questa esercitazione introdurremo la nozione di *estensione banale di una funzione e di un operatore alle matrici* e mostreremo come *definire una funzione* utilizzando il costrutto `function`. Nella seconda parte mostreremo come *disegnare il grafico di una o più funzioni* utilizzando le istruzioni `plot` e `plot2d`.

Prima parte

- Estensione banale di una funzione e di un operatore alle matrici.

  Sia `f` una funzione definita su un sottoinsieme `S` di $F(2,53)$ a valori in $F(2,53)$. Per ogni matrice A ad elementi a_{ij} in `S`, sia `f(`A`)` la matrice, della stessa dimensione di A, di elemento i,j dato da `f(`a_{ij}`)`. La funzione così ottenuta è *l'estensione banale di* `f` *alle matrici.* Se non sorgono conflitti, anche l'estensione banale ha nome `f` (per un esempio di conflitto, si veda la prima parte dell'Esercizio 1).

  Analogamente, sia `op` un operatore definito su un sottoinsieme $\mathtt{S}_1 \times \mathtt{S}_2$ di $F(2,53) \times F(2,53)$, a valori in $F(2,53)$. Per ogni coppia di matrici A, B di uguale dimensione, ad elementi a_{ij} in $\mathtt{S}_1$ e, rispettivamente, b_{ij} in $\mathtt{S}_2$, sia A `op` B la matrice, della stessa dimensione di A e B, di elemento i,j dato da a_{ij} `op` b_{ij}. L'operatore così ottenuto è *l'estensione banale di* `op` *alle matrici.* Se non sorgono conflitti, anche l'estensione banale si indica con `op`, altrimenti si usa la *notazione con punto prefisso* `.op` (per un esempio di conflitto, si veda la seconda parte dell'Esercizio 1).

  Ad esempio:

```
-->A = [0,1,2;3,4,5]
 A  =

    0.    1.    2.
    3.    4.    5.

-->x = A .^2
 x  =

    0.    1.     4.
    9.    16.    25.
```

  La prima istruzione crea la variabile `A` e le assegna il valore:

$$\texttt{[0,1,2;3,4,5]} \quad \text{ovvero} \quad \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

  Un modo per ottenere una matrice di dimensione $r \times c$ ed elementi a_{ij}, in *Scilab*, è l'istruzione:

$$\texttt{[}\langle\text{riga } 1\rangle\,\texttt{;}\,\ldots\,\texttt{;}\langle\text{riga } r\rangle\texttt{]}$$

  dove $\langle\text{riga } k\rangle = a_{k1}\,\texttt{,}\,\ldots\,\texttt{,}a_{kc}$. La matrice è *costruita per righe* (è una colonna di righe).

  La seconda istruzione crea la variabile `x` e le assegna il valore in `A` della funzione `.^2`, estensione banale della funzione `^2` alle matrici.

```
-->A = (%pi/2) * [0,1,2;3,4,5]
 A  =

    0.           1.5707963    3.1415927
```

```
    4.712389    6.2831853    7.8539816

-->sin(A)
 ans  =

    0.     1.           1.225D-16
  - 1.   - 2.449D-16    1.
```

La seconda istruzione chiede a *Scilab* di calcolare il valore in `A` dell'estensione banale della funzione `sin` alle matrici. *Scilab* esegue quanto richiesto ed assegna il valore ottenuto alla "variabile di appoggio" `ans`.[1]

Si osservi che l'estensione banale di una *funzione predefinita* alle matrici è a sua volta una *funzione predefinita*.

- Definizione di una funzione.

  Un metodo per "insegnare" a *Scilab* una nuova funzione è l'uso del costrutto `function`:

  `function` ⟨variabili di uscita⟩ `=` ⟨nome⟩`(`⟨variabili di ingresso⟩`)`
  ⟨istruzioni⟩
  `endfunction`

  Si consideri l'esempio che segue:

```
function y = SqrtAbs(x)
    //
    // x, y: matrici ad elementi reali di uguale dimensione.
    //
    // y(i,j) = sqrt( | 1 - x(i,j)^2 | ).
    //
    y = sqrt(abs(1 - x .^2));
endfunction
```

  La prima riga (l'*intestazione* della funzione) dichiara che la funzione ha *una* variabile di uscita (di *nome locale* `y`), *nome* `SqrtAbs` e *una* variabile di ingresso (di *nome locale* `x`). Le cinque righe successive iniziano con il carattere `//`: questo carattere indica a *Scilab* che la parte restante della riga è un *commento*. Questi commenti hanno lo scopo di dichiarare *quale funzione si vorrebbe definire*: in questo caso l'estensione banale alle matrici della funzione:

$$y = \sqrt{|1 - x^2|}$$

  La dichiarazione fatta ha anche lo scopo di *dare all'utilizzatore la responsabilità di un uso corretto della funzione* (in questo caso, di non utilizzarla con un argomento che non sia *una matrice ad elementi reali*)[2]. Nella riga successiva si assegna il valore alla variabile di uscita `y`. Il valore è determinato dalla seguente sequenza di *funzioni predefinite*:

  (1) `x .^2`

  Calcola in `x` il valore dell'estensione banale alle matrici della funzione `^2` (la funzione predefinita corrispondente alla funzione elementare quadrato).

  (2) `1 - x .^2`

  Se t è un elemento di $F(2,53)$ ed A una matrice ad elementi a_{ij} in $F(2,53)$ allora t `-` A è la matrice ad elementi in $F(2,53)$, della stessa dimensione di A, di componente i,j definita da: t `-` a_{ij} (ovvero: $t \ominus a_{ij}$).

  (3) `abs(1 - x .^2)`

  Calcola in `1 - x .^2` il valore dell'estensione banale alle matrici della funzione valore assoluto `abs`.

[1] Il nome della variabile deriva dal termine inglese *answer* che significa *risposta*.
[2] Vedere l'Esercizio 5.

(4) `sqrt(abs(1 - x .^2))`

Calcola in `abs(1 - x .^2)` il valore dell'estensione banale alle matrici della funzione `sqrt` (la funzione predefinita corrispondente alla funzione elementare radice quadrata).

L'ultima riga dichiara la *fine della definizione* della funzione.

Inserita la definizione precedente nel file `SqrtAbs.sci`[3] utilizzando *SciNotes*, l'editor di *Scilab*,[4] si utilizza l'apposita icona di *SciNotes* per far *eseguire* a *Scilab* il contenuto del file. Questo equivale all'esecuzione del comando:

`exec('`⟨percorso del file `SqrtAbs.sci`⟩`', -1)`

Adesso è possibile utilizzare la funzione (*non* predefinita!) `SqrtAbs`:

```
-->A = [1,0,2;-1,0,-2]
 A  =

    1.    0.    2.
  - 1.    0.  - 2.

-->SqrtAbs(A)
 ans  =

    0.    1.    1.7320508
    0.    1.    1.7320508
```

La prima istruzione crea la variabile `A` e le assegna il valore:

$$\texttt{[1,0,2;-1,0,-2]} \quad \text{ovvero} \quad \begin{bmatrix} 1 & 0 & 2 \\ -1 & 0 & -2 \end{bmatrix}$$

La seconda istruzione chiede a *Scilab* di calcolare il valore della funzione `SqrtAbs` in `A`. Si osservi che dopo aver digitato, ad esempio, `S` nella riga di comando della *console*, l'uso del tasto *Tab* fa comparire una tendina nella quale *Scilab* mostra alcune *parole chiave* che iniziano con `S` (maiuscola o minuscola). Selezionando quella che interessa, *Scilab* completa automaticamente la scrittura.

Seconda parte

- Disegnare il grafico di una funzione: i comandi `plot` e `plot2d`.

  La funzione del comando `plot` è di *disegnare su un piano cartesiano la spezzata di vertici in una data sequenza di punti.* Le *coordinate dei punti* sono le componenti di due vettori di uguale dimensione. Il primo contiene le *ascisse* dei punti, il secondo le *ordinate.* Ad esempio, in risposta al comando:

  `plot([0,1,2,3,4],[1,2,4,3,5])`

  Scilab crea una *finestra grafica* nella quale disegna la spezzata di vertici nei punti $(0,1),(1,2),$ $(2,4),(3,3),(4,5)$ — Figura 1.

  La seguente sequenza di comandi produce il grafico della funzione `SqrtAbs` sull'intervallo $[-1.5;1.5]$:[5]

```
-->x = linspace(-1.5,1.5,301);

-->clf(); plot(x,SqrtAbs(x))
```

[3] I files che contengono istruzioni per *Scilab* hanno estensione `sci` oppure `sce`. Usualmente la prima si adotta quando il file contiene esclusivamente definizioni di funzioni, la seconda quando contiene almeno un'istruzione che non sia una definizione di funzione.

[4] L'editor si attiva utilizzando l'icona, presente quando è attiva la finestra della *console* di *Scilab*, del blocco note in alto a sinistra.

[5] Più correttamente: *un'approssimazione* del grafico.

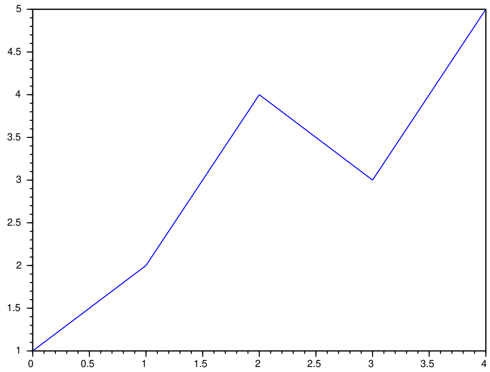

Figura 1: Grafico generato dall'istruzione `plot([0,1,2,3,4],[1,2,4,3,5])`.

Il comando `linspace(`a`,`b`,`n`)` genera il vettore *riga* le cui n componenti sono i numeri reali che suddividono l'intervallo $[a, b]$ in $n-1$ sottointervalli di uguale lunghezza:[6]

$$[\, a, a + \frac{b-a}{n-1}, a + 2\,\frac{b-a}{n-1}, \ldots, b\,]$$

Dunque il primo comando assegna ad `x` il vettore riga le cui 301 componenti sono i numeri reali che suddividono l'intervallo $[-1.5\,; 1.5]$ in 300 sottointervalli di uguale lunghezza. La seconda riga contiene altri *due* comandi. Il primo di essi, `clf()`, ha l'effetto di *cancellare* il contenuto della *finestra grafica corrente* (in questo caso l'unica presente). Il secondo, `plot(x,SqrtAbs(x))`, genera la riga `SqrtAbs(x)` che contiene le ordinate della sequenza di punti da considerare e poi produce il grafico richiesto (Figura 2).

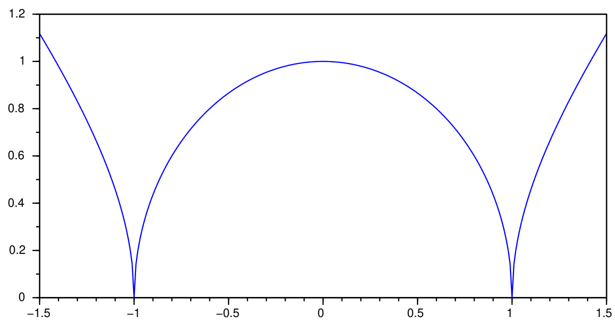

Figura 2: Grafico generato dall'istruzione `plot(x,SqrtAbs(x))`.

Si osservi che, nella seconda riga, l'uso del comando `clf` prima del comando `plot` è essenziale. Il comando `plot` disegna l'oggetto richiesto nella *finestra grafica corrente* senza cancellarne prima il contenuto.

La sequenza di comandi che segue modifica l'ultimo disegno prodotto introducendo, nell'ordine, *una griglia*, *il titolo della figura*, *l'etichetta per l'asse orizzontale* e *l'etichetta per l'asse verticale*:

```
-->xgrid()

-->xtitle('Grafico della funzione SqrtAbs')

-->xlabel('x')
```

[6]Più correttamente: le n componenti sono *un'approssimazione* dei numeri reali che suddividono l'intervallo $[a, b]$ in $n-1$ sottointervalli di uguale lunghezza.

```
-->ylabel('SqrtAbs(x)')
```

Si ottiene, alla fine, il grafico riprodotto in Figura 3.

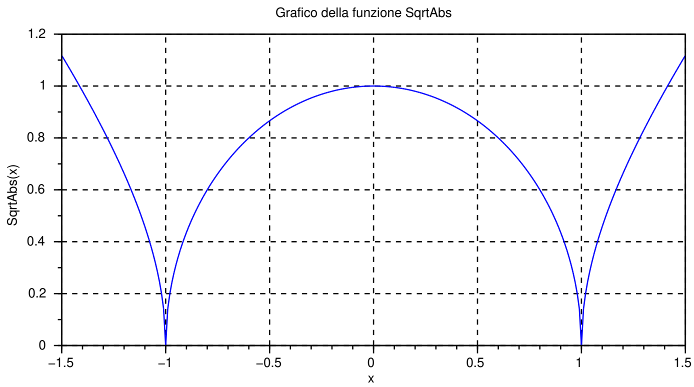

Figura 3: Grafico con griglia ed etichette.

Lo stesso disegno con il grafico *a tratteggio* (anzichè a tratto continuo) e di colore *rosso* (anziché blu) si ottiene aggiungendo agli argomenti del comando `plot` la stringa `'r--'` che specifica lo *stile* da usare per disegnare la curva:

```
plot(x,SqrtAbs(x),'r--')
```

Precisamente, `r` impone il *colore* e `--` impone il *tipo di tratteggio* della curva. Per i dettagli si rimanda alla pagina di *help* del comando `plot`.

Come si può notare dal disegno riportato nella Figura 3, *Scilab* ha adottato *scale diverse* per l'asse delle ascisse e delle ordinate (il *quadrato* $[0,1] \times [0,1]$ è rappresentato nel disegno da un *rettangolo*). Un metodo per obbligare *Scilab* ad adottare *la stessa scala* sugli assi, ed ottenere così un disegno in *scala isometrica*, è di utilizzare il comando `plot2d` con l'opzione `frameflag = 4`:

```
-->clf(); plot2d(x,SqrtAbs(x),frameflag = 4);

-->xgrid();

-->xtitle('Stesso grafico in scala isometrica');

-->xlabel('x'); ylabel('SqrtAbs(x)');
```

Si ottiene il disegno riprodotto in Figura 4. Notare che nell'ultimo disegno il grafico non è di colore blu e manca il rettangolo che circonda la porzione di piano cartesiano rappresentata. Per ottenere queste caratteristiche, occorre aggiungere due opzioni al comando `plot2d`:

```
plot2d(x,SqrtAbs(x),frameflag = 4,style = 2,axesflag = 1)
```

Per i dettagli si rimanda alla pagina di *help* del comando `plot2d`.

- Disegnare il grafico di due funzioni sullo stesso piano cartesiano.

  I comandi `plot` e `plot2d` consentono facilmente di rappresentare il grafico di *due o più funzioni* su uno stesso piano cartesiano: basta specificare *due o più sequenze* di vertici.

  Se `x` ed `y` sono due matrici ad elementi reali di dimensione $r \times c$, il comando `plot(x,y)` disegna (usando c colori differenti) le curve corrispondenti a c sequenze di r punti. Le coordinate degli r punti della k–esima sequenza sono: le componenti della k–esima *colonna* della matrice `x` (ascisse) e le componenti della k–esima *colonna* della matrice `y` (ordinate). Ad esempio, la sequenza di comandi:

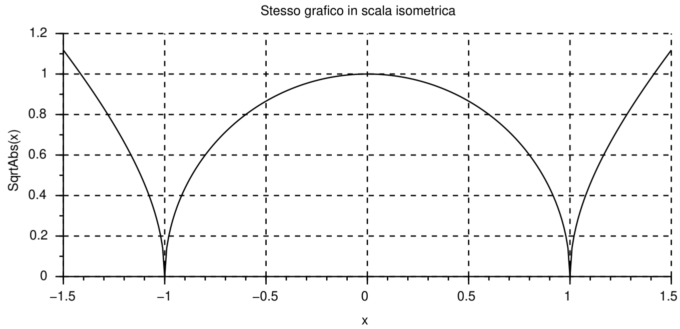

Figura 4: Grafico generato dall'istruzione `plot2d(x,SqrtAbs(x),frameflag = 4)`.

```
-->x1 = [0,1,2,3,4,5]; y1 = [8,6,7,5,6,4];

-->x2 = [0,1.5,2.5,3.5,4.5,5]; y2 = [-8,7,-6,5,-4,3];

-->x = [x1',x2']; y = [y1',y2'];

-->clf(); plot(x,y); xgrid();
```

produce il disegno riportato in Figura 5. Si osservi che se `v` è una matrice, `v'` è la *trasposta* di `v`: `x1'`, `x2'`, `y1'` e `y2'` sono *colonne*. Le matrici `x` e `y` sono *costruite per colonne* (ciascuna è una riga di colonne) utilizzando l'istruzione:

$$\texttt{[}\,\langle\text{colonna } 1\rangle\,\texttt{,}\,\ldots\,\texttt{,}\langle\text{colonna } c\rangle\,\texttt{]}$$

con $c = 2$.

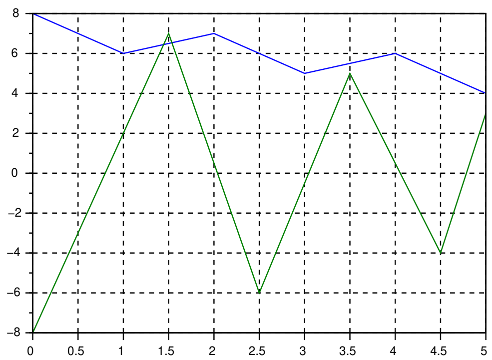

Figura 5: Due curve sullo stesso piano cartesiano.

Se, invece, `x` è una *colonna* di r numeri reali ed `y` una matrice ad elementi reali di dimensione $r \times c$, il comando `plot(x,y)` disegna ancora (usando c colori differenti) le curve corrispondenti a c sequenze di r punti. Le coordinate dei punti della k–esima sequenza sono, adesso: le componenti della *colonna* `x` (ascisse) e le componenti della k–esima *colonna* della matrice `y` (ordinate).

Allo stesso modo si specificano le sequenze dei vertici per il comando `plot2d`.

La sequenza di comandi che segue produce il disegno riprodotto in Figura 6.

```
-->x = linspace(-5,5,601)';
```

```
-->clf(); plot2d(x,[SqrtAbs(x),abs(x)],frameflag = 4);

--> xgrid(); xlabel('x'); legend('SqrtAbs(x)','|x|');
```

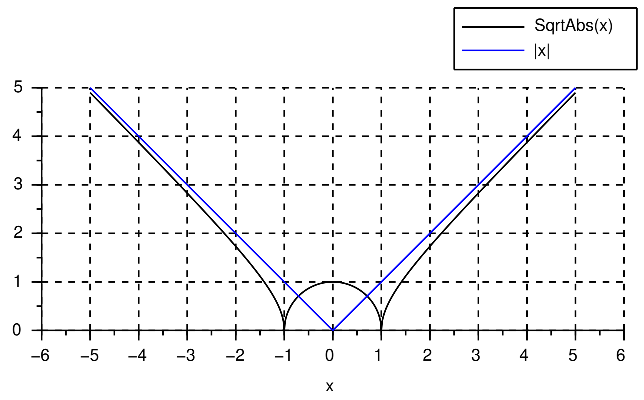

Figura 6: Disegno con etichetta e legenda.

Il comando `legend(`⟨stringa 1⟩`,...,`⟨stringa c⟩`)` associato ad un comando che disegna c curve produce un rettangolo (una *legenda*) contenente la corrispondenza tra il *colore* della k–esima curva e *stringa k*.

Esercizi

1. Discutere il dialogo seguente, che spiega perché l'estensione banale della funzione `^2` alle matrici *non può* indicarsi con `^2` e perché per indicare l'estensione banale dell'operatore `*` alle matrici occorre utilizzare la notazione con *punto prefisso*:

```
-->B = [1,0;-1,1]
 B  =

    1.    0.
  - 1.    1.

-->B ^2
 ans  =

    1.    0.
  - 2.    1.

-->B .^2
 ans  =

    1.    0.
    1.    1.

-->B * B
 ans  =

    1.    0.
  - 2.    1.

-->B .* B
 ans  =

    1.    0.
    1.    1.
```

2. Discutere il dialogo seguente:

```
-->A = [1,2,3,4,5]
 A  =

    1.    2.    3.    4.    5.

-->1 ./ A
 ans  =

    1.    0.5    0.3333333    0.25    0.2

-->A .* A
 ans  =

    1.    4.    9.    16.    25.
```

3. Se A è una matrice di dimensione $r \times c$, allora `ones(`A`)` è la matrice della stessa dimensione di A ad elementi tutti uguali a *uno* (vedere la pagina di *help* relativa alla *funzione predefinita* `ones`). Costruire un dialogo che aiuti a constatare che, detta `A` la matrice introdotta nell'Esercizio precedente, le matrici `2 + A` e `2 * ones(A) + A` sono uguali.

4. Si consideri la seguente definizione della funzione `MostraFraz`:

```
function s = MostraFraz(x)
   //
   // Restituisce la stringa s che rappresenta la scrittura
   // in base due della frazione dell'elemento x di F(2,53).
   //
   [f,e] = log2(x);
   s = '0.';
   for i=1:53,
     s = s + string(int(2*f));
     f = 2*f - int(2*f);
   end;
endfunction
```

Utilizzare *SciNotes* per inserire la definizione nel file `MostraFraz.sci` e per far eseguire a *Scilab* il contenuto del file. Utilizzare poi la funzione `MostraFraz` per determinare la scrittura posizionale in base due della frazione del predecessore di 1. Infine: verificare la correttezza della risposta.

5. Inserire la seguente definizione di funzione:

```
function y = AbsLog(x)
    //
    // x: matrice ad elementi reali POSITIVI;
    // y: matrice ad elementi reali della stessa
    //    dimensione di x.
    //
    // y(i,j) = | log( x(i,j) ) |.
    //
    y = abs(log(x));
endfunction
```

e discutere il seguente dialogo tra *Scilab* ed un utilizzatore poco attento:

```
-->AbsLog(2)
 ans  =

    0.6931472

-->AbsLog(1/2)
 ans  =

    0.6931472

-->AbsLog(-2)
 ans  =

    3.2171505
```