Complementi di Matematica e Calcolo Numerico
C.d.L Chimica Industriale A.A. 2018-2019

Laboratorio 5 - 11/04/2019

FATTORIZZAZIONE DI *CHOLESKY*

Se $A \in \mathbb{R}^{n\times n}$ è una matrice simmetrica definita positiva, allora esiste una matrice $B \in \mathbb{R}^{n\times n}$ triangolare inferiore tale che

$$A = BB^T.$$

Tale fattorizzazione è detta fattorizzazione di **Cholesky**.
Il comando `R = chol(A)` di Matlab determina la matrice $R = B^T$ di tale fattorizzazione e pertanto abbiamo $A = R^T R$.

Se A è la matrice dei coefficienti di un sistema lineare $Ax = b$ per calcolare la soluzione di tale sistema possiamo sfruttare la fattorizzazione $A = R^T R$, e determinare x risolvendo in sequenza i due sistemi triangolari

$$R^T y = b$$
$$Rx = y$$

Esempio

Verificare che la matrice

$$A = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 5 & 6 \\ 0 & 6 & 10 \end{bmatrix},$$

è simmetrica definita positiva. Calcolare con il comando `chol` di Matlab la fattorizzazione di Cholesky di A. Risolvere il sistema lineare $Ax = b$ con `b=[3;1;1]` sfruttando la fattorizzazione calcolata.

```
% Inserisco i dati
A=[ 1  -1   0; -1  5 6; 0 6 10];
b=[3;1;1];
% Controllo che A sia simmetrica e definita positiva
A'==A
eig(A)
% oppure
A(1,1)>0
det(A(1:2,1:2))>0
det(A)>0
% Calcolo Fattorizzazione di Cholesky
R=chol(A);
% Risolvo sistemi triangolari per sostituzioni
y=R'\b;
x=R\y
```

Esercizio 1

Sia A la matrice 13×13 avente i primi 13 numeri interi positivi in ordine decrecrescente sulla diagonale principale, -1 sulla prima sopra e sottodiagonale.

- Si calcoli il determinante di A.

  $\det(A) =$

- Si calcoli, usando l'apposito comando di Matlab, la fattorizzazione di Cholesky $A = R^T R$. Si calcoli il determinante di R, $\det(R) =$

- Sia b il vettore colonna unitario. Sfruttando la fattorizzazione calcolata si determini la soluzione x del sistema lineare $Ax = b$ e se ne calcoli la norma $||x||_2$. Sia y il termine noto del sistema lineare triangolare superiore che occorrerà risolvere per sostituzione all'indietro, si riporti l'ultima componente di tale vettore.

  $y(13)=$ $||x||_2=$

Programmare con Matlab: Function files

Le funzioni matlab sono porzioni di codici scritte in un file indipendente che svolgono un determinato compito e comunicano con lo spazio di lavoro solo attraverso i parametri in ingresso ed in uscita.

L'intestazione di una function Matlab ha sempre la struttura:

$$\underbrace{\texttt{function}}_{\text{parola chiave}} \underbrace{[\texttt{out1}, \texttt{out2}, \ldots]}_{\text{parametri in uscita}} = \underbrace{\texttt{nomefun}}_{\text{nome funzione}} \underbrace{(\texttt{in1}, \texttt{in2}, \ldots)}_{\text{parametri in ingresso}}$$

L'intestazione è seguita dalle istruzioni e la function terminerà con la parola chiave `return`. Prima di essa, deve essere stato assegnato un valore a ciascuno dei parametri in uscita `out1,out2,...`.

La funzione `nomefun` deve essere salvata nel file `nomefun.m`.
Att.ne! Un file può contenere un unica funzione.

Le variabili assegnate nel blocco istruzioni interno alla function sono locali, ovvero vengono cancellate dalla memoria al termine della chiamata.

Per chiamare una function, ad esempio dallo spazio di lavoro:

```
>> [value1,value2,value3]=nomefun(in1,in2,in3);
```

Una funzione può richiamare o essere richiamata da altre.

Esempio. La function `mat_diff_1d`, dato in ingresso un intero n, restituisce la matrice $A \in \mathbb{R}^{n\times n}$ seguente ed il suo determinante.

$$A = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & .. & \\ & .. & .. & -1 \\ & & -1 & 2 \end{bmatrix}$$

```
function [A det_A]=mat_diff_1d(n)
A=diag(2*ones(n,1))-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
det_A=det(A);
return;
```

Esempio di output:

```
>> [A,detA]=mat_diff_1d(5)
A =
     2    -1     0     0     0
    -1     2    -1     0     0
     0    -1     2    -1     0
     0     0    -1     2    -1
     0     0     0    -1     2
detA=
     6
```

Grafici-2D

Il più semplice comando Matlab per disegnare un grafico è :

`plot(x,y)`

dove $x = (x_1, \ldots, x_n)$ e $y = (y_1, \ldots, y_n)$ sono 2 vettori di ugual dimensione. Il comando `plot(x,y)` rappresenterà in una finestra grafica una linea che collega i punti di coordinate (x_i, y_i), $i = 1, \ldots, n$

Esempio

Disegnare il grafico della funzione $f(x) = 2\sin(x)\cos(x) + 2x$ nell'intervallo $[0, \pi/2]$

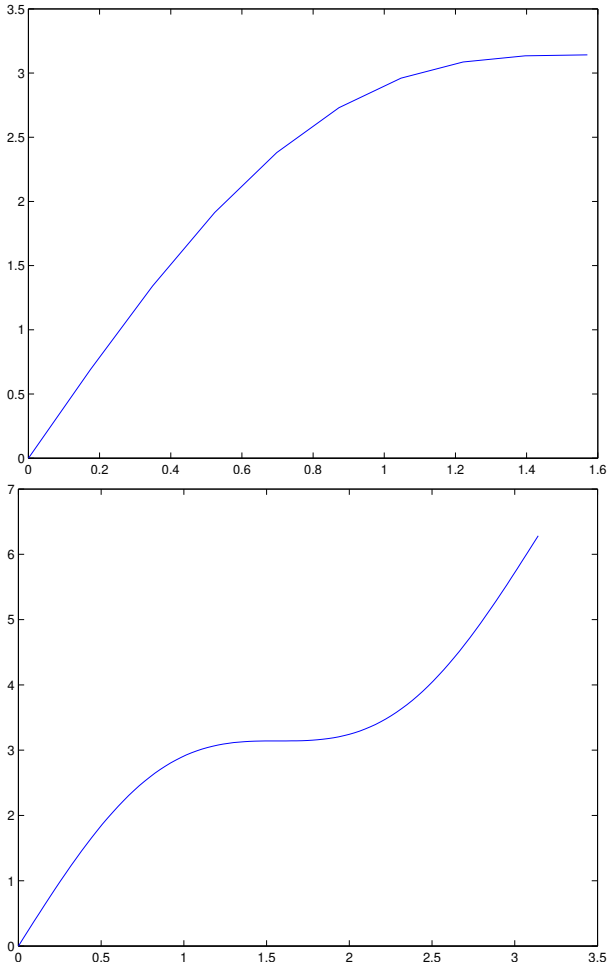
```
>> x=linspace(0, pi/2,10);
>> y=2*sin(x).*cos(x)+2*x;
>> plot(x,y)
```

Si noti che grazie alla caratteristica vettoriale di Matlab è sufficiente un solo comando per valutare f in un vettore di punti, basta infatti eseguire le operazioni componente per componente.

Se cambio il vettore delle ascisse x devo ricalcolare il vettore y contenente i valori assunti dalla funzione f nei nuovi punti prima di fare il grafico:

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

Sarà utile, per non dover manualmente ripetere la successione di istruzioni ogni volta che si cambia un parametro, memorizzare il lavoro in uno *script-file*.



L'istruzione `plot` crea una nuova finestra grafica solo se non ci sono finestre grafiche già aperte, altrimenti utilizza l'ultima finestra creata, e sovrascrive il nuovo grafico a quello creato in precedenza. Pertanto se vogliamo visualizzare nella stessa finestra grafica i grafici di due funzioni

$$f(x) = \sin(x) + x, \qquad g(x) = x^2 + \cos(x) \qquad x \in [0, \pi]$$

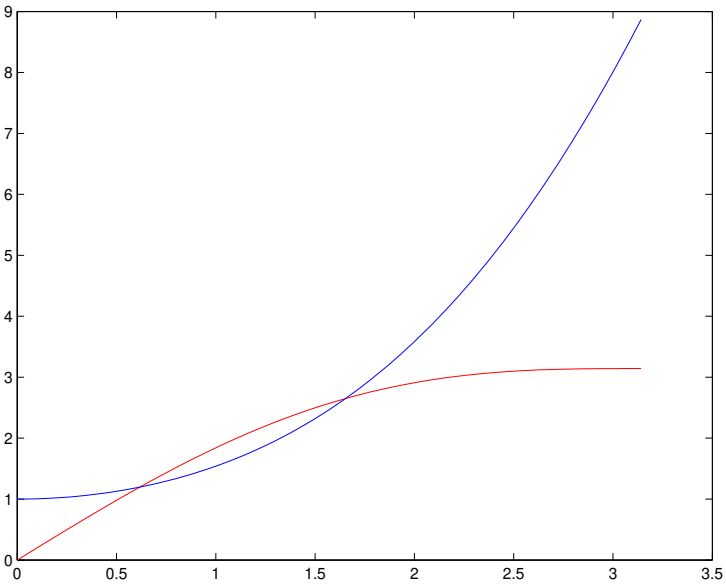
abbiamo due possibilità:

- possiamo disegnare il grafico di f e aggiungere successivamente il grafico di g specificando a Matlab di non cancellare il primo grafico tramite il comando `hold on`

```
>> x=linspace(0,pi);
>> fx=sin(x)+x;
>> plot(x,fx)
>> hold on
>> gx=x.^2+cos(x);
>> plot(x,gx)
```

- possiamo disegnare i due grafici con un solo comando `plot`

```
>> x=linspace(0,pi);
>> fx=sin(x)+x;
>> gx=x.^2+cos(x);
>> plot(x,fx,x,gx)
```



Se invece vogliamo visualizzare le due funzioni in differenti finestre grafiche possiamo utilizzare il comando `figure`. Quest'ultimo genera una nuova finestra e la numera in ordine crescente o come indicato:

```
>> x=linspace(0,pi);
>> fx=sin(x)+x;
>> gx=x.^2 +cos(x);
>> figure(1)
>> plot(x,fx)
>> figure(2)
>> plot(x,gx)
```

Il comando **plot** prevede la possibilità di scegliere il tipo di linea, o di simbolo ed il colore, con cui vengono rappresentati graficamente i dati (vedi help plot). La sintassi generale è:

```
plot(x, y,'specifiche')
```

Dove **'specifiche'** è una stringa di caratteri scelti nella lista seguente

b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
w	white	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

Il comando **doc LineSpec** mostra una lista di tutte le opzioni disponibili per specificare lo stile della linea.

Per disegnare solo i punti dati (x_i, y_i) con un asterisco senza collegarli con una linea:

```
>> plot(x,y,'*')
```

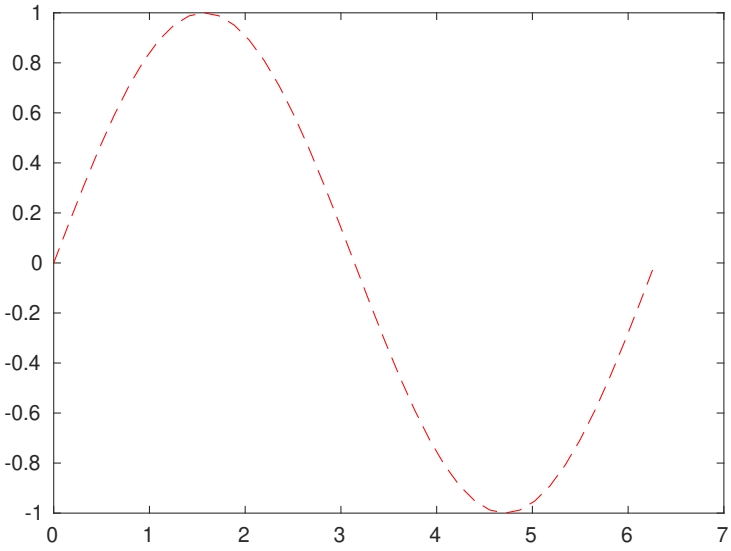
Se vogliamo disegnare ad esempio la funzione $\sin(x)$ nell'intervallo $[0, 2\,\pi]$ con una linea rossa tratteggiata

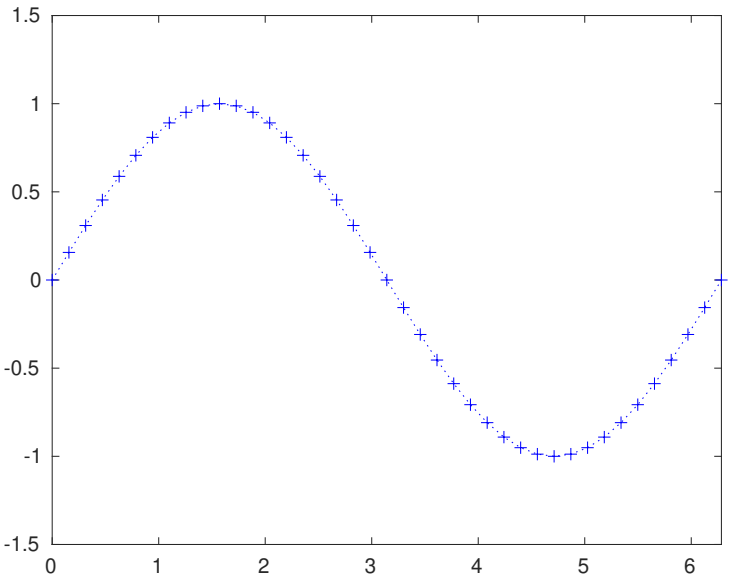
```
>> x = 0:pi/20:2*pi;
>> y=sin(x);
>> plot(x,y,'r--')
```

digitando successivamente

```
>> figure(2)
>> plot(x,y,'b+:')
>> axis([0 2*pi -1.5 1.5])
```

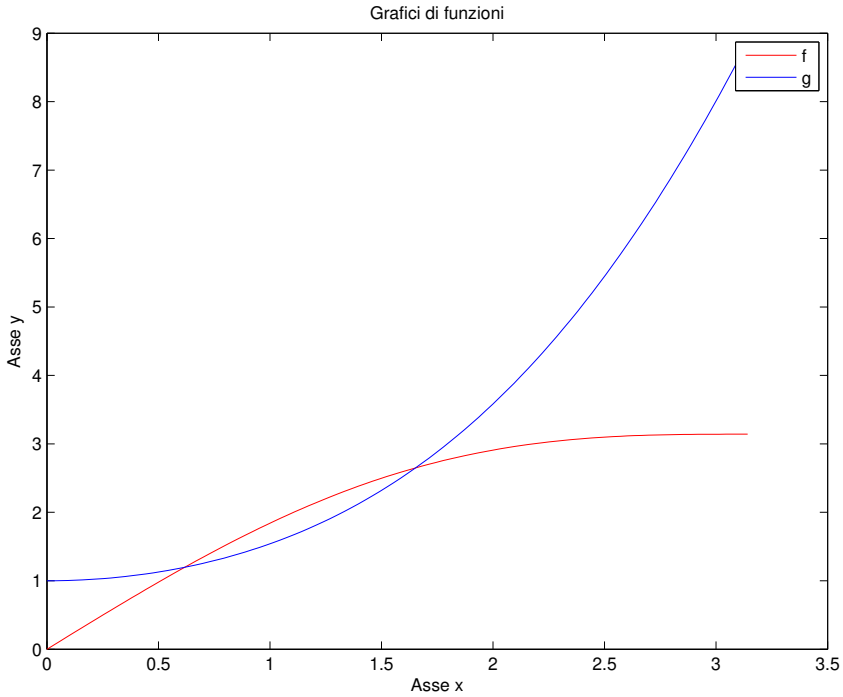
visualizzeremo in un altra finestra il grafico di $\sin(x)$ con linea punteggiata blu evidenziando con il simbolo `+` i punti (x_i, y_i) in cui la funzione è stata valutata per disegnare il grafico.
Inoltre con il comando `axis` abbiamo definito i limiti degli assi che di default vengono scelti automaticamente da Matlab in base ai dati visualizzati.


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7


1.5
1
0.5
0
-0.5
-1
-1.5
0
1
2
3
4
5
6

I comandi `title,xlabel,ylabel,legend` ci permettono di completare il nostro grafico con un titolo, delle etichette sugli assi e una legenda:

```
>> x=linspace(0,pi);
>> fx=sin(x)+x;
>> gx=x.^2 +cos(x);
>> plot(x,fx,'r', x,gx,'b')
>> title('Grafici di funzioni')
>> xlabel('Asse x')
>> ylabel('Asse y')
>> legend('f','g')
```



Il comando `grid on` inserisce una griglia sul grafico.

Esercizio 2

Scrivere uno *script-file* Matlab per disegnare il grafico delle seguenti funzioni negli intervalli indicati:

- $f(x) = \frac{2\log(x+2)}{\sqrt{x+1}} \quad x \in [1, 2]$
- $g(x) = \frac{x^2+2x+1}{x^2+1} \quad x \in [0, 5]$

con le seguenti modalità:

- disegnare due grafici distinti nella stessa finestra
- disporre i due grafici in due finestre distinte.

A tale scopo si definisca un vettore z di 50 punti equispaziati nell'intervallo di definizione di ciascuna funzione, la si valuti nei punti di z e si disegni il grafico utilizzando il comando `plot`.

Esercizio 3

Scrivere una *function* Matlab che preso in input un numero naturale $N <= 50$ disegni i grafici delle funzioni $f_n(x) = x^n$ nell'intervallo $[0, 1]$ per $n = 0, \ldots N$ nella medesima finestra grafica corredandola di titolo, ed etichette sugli assi.

Funzioni "anonime" (anonymous functions) (@)

Assegnata una funzione del tipo $f(x) = (\sin(x) + x)^2$ vogliamo valutare i valori assunti da f per diversi valori di x.
Quando l'espressione della funzione è lunga e/o complessa e la funzione deve essere valutata in istanti successivi per diversi valori delle variabili da cui dipende, è utile poter definire la funzione una volta per tutte senza dover riscrivere la sua espressione ogni volta che la si vuole valutare in punti differenti. In Matlab è possibile definire una anonymous function direttamente nello spazio di lavoro, senza ricorrere a file esterni, mediante il comando **@** (function handle).
Per esempio, definiamo:

```
>> f=@(x)(sin(x)+x).^2
```

dove nella parentesi che segue il simbolo @ elenchiamo in modo ordinato (in questo esempio solo la x) le variabili da cui la funzione dipende, mentre nell'espressione che segue scriviamo l'espressione matematica che definisce la funzione.

Attenzione: ricordarsi di utilizzare operazioni con i punti se si vuole che la funzione operi sui vettori!

Ad una funzione così definita non sono associati dei valori numerici (verificare con `whos f`). Per associarle valori numerici scriviamo, per esempio

```
>> x=0:0.01:2*pi;
```

```
>> y=f(x);
```

Tali valori numerici vengono conservati nel vettore y (verificare con `whos y`). Possiamo poi usarli, per esempio, per disegnare il grafico di f con il comando

```
>> plot(x,y)
```

È possibile definire funzioni che dipendono da più variabili o parametri

```
>> f=@(a,x) 2*x+a;
```

Attenzione: $y = f(2, 10)$ è diverso da $z = f(10, 2)$.

Esercizio 4. Dopo averla definita con @, fare un grafico della funzione:

$$f(x) = \frac{2\log(x+2)}{\sqrt{x+1}}$$

in $[1, 2]$ con linea nera tratteggiata.

Esercizio 5. Dopo aver definito la funzione $f(a, x) = a^x$ con @ come funzione di due variabili, disegnare i grafici di f in $[0, 1]$ per $a = 0.1, 1, 10$ nella stessa finestra utilizzando colori diversi e corredandola di legenda;