Metodi numerici con elementi di Programmazione

A.A. 2013-2014

Esercizi svolti in Laboratorio

Lezione del 12-11-2013

Docente: Vittoria Bruni

Email: vittoria.bruni@sbai.uniroma1.it

Ufficio: Via A. Scarpa,
Pal. B, I piano, Stanza n. 16
Tel. 06 49766648

Ricevimento: Giovedì 14.00-15.00

Il materiale didattico è disponibile sul sito http://ingaero.uniroma1.it/ nella pagina dedicata al corso Metodi Numerici con elementi di Programmazione

Per consultazione: Getting Started with MatLab – The mathworks www.mathworks.com

Esercizio

Risolvere il seguente sistema lineare

$$\begin{cases} x_2 - 2x_3 = -3 \\ -x_1 + 5x_2 + 3x_3 = -3 \\ 2x_1 - 4x_2 = 6 \end{cases}$$

dopo aver calcolato il determinante, il rango, le norme 1 e infinito e l'inversa della matrice dei coefficienti associata.

OSS:

La matrice del sistema è $A = \begin{pmatrix} 0 & 1 & -2 \\ -1 & 5 & 3 \\ 2 & -4 & 0 \end{pmatrix}$

Il termine noto è $b = \begin{pmatrix} -3 \\ -3 \\ 6 \end{pmatrix}$

Esercizio

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

>> det(A)
ans =
    18
```

Nota: *A* è non singolare!!!

Come si calcola il rango di una matrice?
Esiste una funzione MatLab predefinita che calcola il rango della matrice?

```
>> lookfor 'matrix rank'
rank              - Matrix rank.
```

```
>> help rank
 RANK   Matrix rank.
    RANK(A) provides an estimate of the number of linearly
    independent rows or columns of a matrix A.
    RANK(A,tol) is the number of singular values of A
    that are larger than tol.
    RANK(A) uses the default tol = max(size(A)) * eps(norm(A)).

    Class support for input A:
       float: double, single

    Overloaded methods:
       gf/rank
       rptcp/rank

    Reference page in Help browser
       doc rank

>> rank(A)
ans =
     3
```

Esiste una funzione MatLab predefinita che calcola norme di matrici?

```
>> lookfor norm
realmin     - Smallest positive normalized floating point number.
randn       - Normally distributed pseudorandom numbers.
condest     - 1-norm condition number estimate.
norm        - Matrix or vector norm.
normest     - Estimate the matrix 2-norm.
normest1    - Estimate of 1-norm of matrix by block 1-norm power met
sprandn     - Sparse normally distributed random matrix.
surfnorm    - Surface normals.
isonormals  - Isosurface normals.
```

```
>> help norm
 NORM   Matrix or vector norm.
    For matrices...
      NORM(X) is the 2-norm of X.
      NORM(X,2) is the same as NORM(X).
      NORM(X,1) is the 1-norm of X.
      NORM(X,inf) is the infinity norm of X.
      NORM(X,'fro') is the Frobenius norm of X.
      NORM(X,P) is available for matrix X only if P is 1, 2, inf or
   'fro'.
    For vectors...
      NORM(V,P) = sum(abs(V).^P)^(1/P).
      NORM(V) = norm(V,2).
      NORM(V,inf) = max(abs(V)).
      NORM(V,-inf) = min(abs(V)).
    See also cond, rcond, condest, normest, hypot.
    Overloaded methods:
       codistributed/norm
       mfilt.norm
       adaptfilt.norm
       idmodel/norm
       dfilt.norm
    Reference page in Help browser
       doc norm
```

```
>> norm(A,1)
ans =
    10
```

È equivalente a

```
>> max(sum(abs(A)))
ans =
    10

>> norm(A,inf)
ans =
     9
```

È equivalente a

```
>> max(sum(abs(A')))
ans =
     9
```

Esercizio: Scrivere il comando Matlab che produce lo stesso risultato della funzione norm(A,2)

```
>> b = [-3 -3 6]'; % Il vettore b deve essere un vettore
                   colonna!!!!
>> x = A\b;
>> disp(x)
     1
    -1
     1
>> IA = inv(A)
IA =
    0.6667    0.4444    0.7222
    0.3333    0.2222    0.1111
   -0.3333    0.1111    0.0556
```

Il Calcolo dell'inversa di *A* è equivalente a

```
>> E = eye(size(A))
E =
     1     0     0
     0     1     0
     0     0     1
```

```
>> IA2 = A\E   % risolve i 3 sistemi
               A * IA(:,1) = (1 0 0)'
               A * IA(:,2) = (0 1 0)'
               A * IA(:,3) = (0 0 1)'
```

```
IA2 =
    0.6667    0.4444    0.7222
    0.3333    0.2222    0.1111
   -0.3333    0.1111    0.0556


>> IA == IA2
ans =
     1     1     1
     1     1     1
     1     1     1
```

Osservazioni

Scambiando la prima e la terza riga di *A*, il rango della matrice non cambia

```
A =
     0     1    -2
    -1     5     3
     2    -4     0
>> B = A;
>> B(1,:)=A(3,:);,  B(3,:)=A(1,:);
>> disp(B)
     2    -4     0
    -1     5     3
     0     1    -2
>> rank(A)
ans =
     3
>> rank(B)
ans =
     3
```

Osservazioni

Sostituendo alla seconda riga di A, una combinazione lineare della

prima e della seconda riga, il rango non cambia

```
>> B = A;
>> B(2,:)=B(1,:)-2*B(2,:);

>> B
B =
     0      1     -2
     2     -9     -8
     2     -4      0

>> rank(B)
ans =
     3
```

Osservazioni

Sostituendo alla seconda riga di A, una combinazione lineare della prima e della terza riga, il rango cambia

(le righe della matrice non sono più linearmente indipendenti!)

```
>> B=A;
>> B(2,:)=B(1,:)-2*B(3,:);
>> B
B =
     0     1    -2
    -4     9    -2
     2    -4     0
>> rank(B)
ans =
     2
>> det(B)
ans =
     0
```

Esercizio

- Scrivere o script Matlab esercizio_lab1.m che disegni il grafico della seguente funzione

$$f_n(x) = \begin{cases} x^n + 1 & \text{per } -1 \leq x \leq 0 \\ (-1)^n |x|^n & \text{per } 0 < x \leq 1 \end{cases}$$

in corrispondenza di quattro valori distinti di *n* sulla stessa finestra grafica ma usando 4 sotto-finestre distinte.
I valori di *n* devono essere richiesti in input.
Inserire opportuni controlli e messaggi di errore nel caso in cui i valori di *n* non siano numeri interi positivi.

Soluzione

1) Determinare i parametri di input e il tipo

   n = vettore di 4 numeri interi positivi

2) Scrivere uno pseudocodice

   1. Leggi n e controlla lo input

      n deve essere un vettore di lunghezza 4 e le sue componenti devono essere numeri positivi e interi

   2. Genera il vettore x di numeri reali nell'intervallo [-1 1]
   3. Assegna alla variabile x1 gli elementi di x <=0
   4. Assegna alla variabile x2 gli elementi di x>0
   5. Dette L1 e L2 rispettivamente la lunghezza di x1 e la lunghezza di x2, per ogni componente n(i) del vettore n, con 1<=i<=4 esegui le seguenti istruzioni:
      - valuta x1(k)^n(i)+1 per 1<=k<=L1 e assegna il vettore di numeri risultante alla variabile f1
      - valuta (-1)^n(i)|x2(k)|^n(i) per 1<=k<=L2 e assegna il vettore di numeri risultante alla variabile f2
      - grafica i punti di coordinate (x1,f1) e (x2,f2) sulla i-esima finestra grafica

3) Tradurre lo pseudocodice in linguaggio MatLab

- Aprire la finestra di Editor e salvare il file con il nome esercizio_lab1.m (Controllare che il file sia salvato nella directory in cui si sta lavorando!)

- Scrivere lo help dello script

```
% Lo script esercizio_lab1 legge in input il vettore n di
% numeri interi e positivi di lunghezza 4,  valuta e grafica la
% funzione fn(x) in un vettore di punti appartenenti
% all'intervallo [-1, 1].
% La funzione fn(x) è definita come segue
%       fn(x)= x^n+1    se -1<=x<=0
%       fn(x)=(-1)^n |x|^n    se 0<x<=1
```

3) Tradurre lo pseudocodice in linguaggio MatLab

- Scrivere il codice Matlab

1. Leggi n e controlla lo input

n deve essere un vettore di lunghezza 4 e le sue componenti devono essere numeri positivi e interi

```
n = input('inserisci un vettore di 4 numeri interi e positivi ');
if (n~=round(n)) | (~all(n>=0)) | (length(n) ~=4)
   error('n non è un vettore di numeri interi e positivi di lunghezza 4!!!')
end
```

2. Genera il vettore x di numeri reali nell'intervallo [-1 1]

```
x = linspace(-1,1,500);
```

Oss: il numero di punti può essere scelto in modo arbitrario

3) Tradurre lo pseudocodice in linguaggio MatLab

3. Assegna alla variabile x1 gli elementi di x <=0

```
ind1 = find(x<=0);
x1=x(ind1);
```

4. Assegna alla variabile x2 gli elementi di x>0

```
ind2 = find(x>0);
x2=x(ind2);
```

3) Tradurre lo pseudocodice in linguaggio MatLab

5. Dette L1 e L2 rispettivamente la lunghezza di x1 e la lunghezza di x2, per ogni componente n(i) del vettore n, con 1<=i<=4 esegui le seguenti istruzioni:
   - valuta x1(k)^n(i)+1 per 1<=k<=L1 e assegna il vettore di numeri risultante alla variabile f1
   - valuta (-1)^n(i)|x2(k)|^n(i) per 1<=k<=L2 e assegna il vettore di numeri risultante alla variabile f2
   - grafica i punti di coordinate (x1,f1) e (x2,f2) sulla i-esima finestra grafica

```
figure,
for i=1:4
      f1 = x1.^n(i)+1;
      f2 = (-1)^n(i) *abs(x2).^n(i);
      subplot(2,2,i), plot(x1,f1,'b-',x2,f2,'b-')
      title(['grafico per n = ', int2str(n(i))])
end
```

3) Salvare il file

4) Andare sul Command window e digitare

>> esercizio_lab1

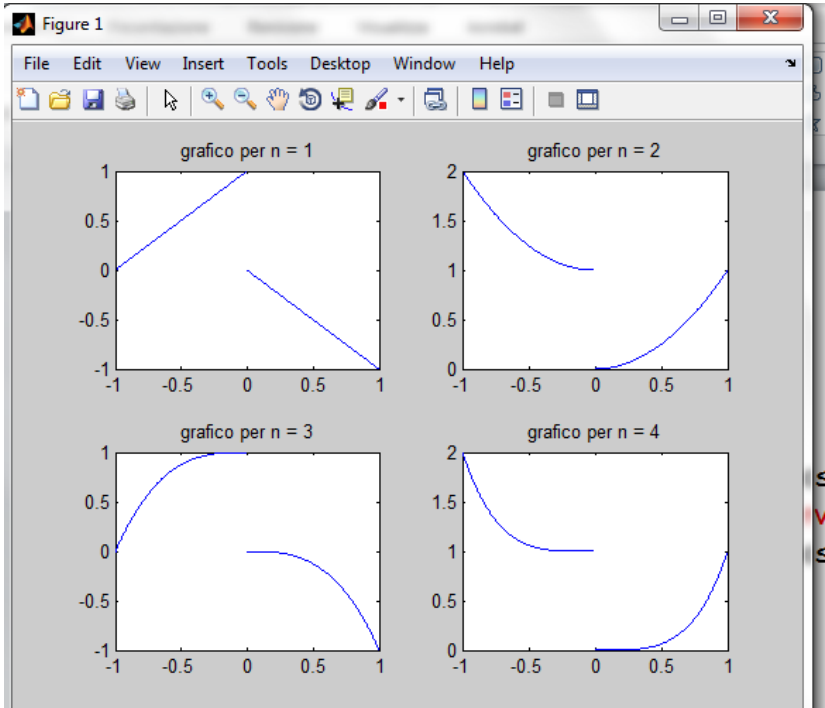
e premere il tasto invio.

Si visualizza il seguente messaggio.

>> inserisci un vettore di 4 numeri interi e positivi |

Bisogna introdurre il vettore n e premere invio

>> inserisci un vettore di 4 numeri interi e positivi [1 2 3 4]



Esercizio

Richiamare lo script esercizio_lab1 dando in input

n = [1 -2 3 4]
n = [1 2 3]
n = [0.5 2 3]
n = [1 -2 -3 4 5]

```
>> esercizio_lab1
>> inserisci un vettore di 4 numeri interi e positivi [1 -2 3 4]
??? Error using ==> esercizio_lab1 at 10
n non è un vettore di numeri interi e positivi di lunghezza 4!!!

>> esercizio_lab1
>> inserisci un vettore di 4 numeri interi e positivi [1 2 3]
??? Error using ==> esercizio_lab1 at 10
n non è un vettore di numeri interi e positivi di lunghezza 4!!!

>> esercizio_lab1
>> inserisci un vettore di 4 numeri interi e positivi [0.5 2 3]
??? Error using ==> esercizio_lab1 at 10
n non è un vettore di numeri interi e positivi di lunghezza 4!!!
```

```
>> esercizio_lab1
>> inserisci un vettore di 4 numeri interi e positivi [1 -2 -3 4 5]
??? Error using ==> esercizio_lab1 at 10
n non è un vettore di numeri interi e positivi di lunghezza 4!!!
```

Esercizio

(da fare a «casa»)

- Scrivere lo script Matlab esercizio1.m che valuti la funzione

$$f(x) = \begin{cases} (m - \frac{x^2}{m})^m, & x \in [-m, 0], \\ (\frac{x^2}{m} + m)^m, & x \in [0, m], \end{cases}$$

in punti equidistanti (di passo *h*) contenuti nell'intervallo [-*m*,*m*] e stampi in una tabella i punti in cui è stata valutata la funzione e il valore corrispondente. Lo script deve ripetere le operazioni precedenti per 6 valori distinti di *m* e disegnare le 6 funzioni su una stessa finestra grafica usando 6 sotto-finestre distinte. Etichettare opportunamente le sotto-finestre in modo da distinguere le funzioni. *m* e il passo *h* devono essere dati in input.
Salvare opportunamente i dati prodotti nel file datiscript.mat.