Prima Esercitazione: introduzione a Matlab

Esercizio 1 *Vettori e matrici in Matlab*

Siano A, B e C le seguenti matrici

$$A = \begin{pmatrix} 1 & 3 & 2 \\ -5 & 3 & 1 \\ -10 & 0 & 3 \\ 1 & 0 & -2 \end{pmatrix}, B = \begin{pmatrix} 1 & -2 & 5 \\ 6 & 1 & -1 \end{pmatrix}, C = \begin{pmatrix} 10 & -5 \\ 3 & 1 \end{pmatrix}$$

1. Calcolare le matrici AB, BA e AB^T, se possibile
2. Calcolare la matrice $D = I_2 - BB^T$, con I_2 la matrice diagonale di dimensione 2 (comando `eye`)
3. Calcolare il determinante delle matrici A, B, C, D e $E = AA^T$
4. Calcolare le inverse delle matrici A, B, C, D, E

Esercizio 2 *Vettori e matrici in Matlab*

Date le matrici

$$A = \begin{pmatrix} 1 & -1 & 7 \\ -4 & 2 & 11 \\ 8 & 0 & 3 \end{pmatrix}, B = \begin{pmatrix} 3 & -2 & -1 \\ 7 & 8 & 6 \\ 5 & 1 & 3 \end{pmatrix} \tag{1}$$

Cosa fanno le seguenti istruzioni?

`3 * A` , `A * B` , `A * inv(B)` , `cos(A)`, `exp(B )`, `C = [A B]` , `D = [A,B]` , `E = [A;B]`

Esercizio 3 *Il comando :*

Costruire col minimo numero di comandi i seguenti vettori e matrici:

1. [1 2 3 ... 19 20 19 ...2 1]
2. [0 0.1 0.2 ... 1]
3. [100 99 ... 0]
4. $\begin{pmatrix} 2 & 2 & 2 & 3 \\ 2 & 2 & 2 & 3 \\ 2 & 2 & 2 & 3 \end{pmatrix}$ $\begin{pmatrix} 2 & 0 & 0 & 1 & 2 & 3 & \cdots & 10 \\ 0 & 2 & 0 & 1 & 2 & 3 & \cdots & 10 \\ 0 & 0 & 2 & 1 & 2 & 3 & \cdots & 10 \end{pmatrix}$ $\begin{pmatrix} 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}$

Seconda Esercitazione: Matlab come linguaggio

Esercizio 1 *script in Matlab*

Costruire un m-file di tipo script che preso la matrice a costruisce la matrice $a \cdot a^T + 2I$.
Attenzione: deve calcolare il formato di a per saper quale matrice identica. Usare: comando **`size`**

Esercizio 2 *script in Matlab*

Costruire un m-file di tipo script che preso il vettore x costruisce la tabulazione di $x^2 + 1$.

Per esempio, se **`x=[ 1 2 3 4]`**, allora lo script scrive

1	2
2	5
3	10
4	17

Esercizio 3 *script in Matlab*

Costruire un m-file di tipo script che preso il vettore x lo rovescia come ordine.
Per esempio, se **`x=[ 1 2 3 4]`**, allora lo script scrive **`x=[ 4 3 2 1]`**.
Usare con attenzione il simbolo **:**

Esercizio 4 *L'algoritmo di punto fisso*

Costruire un m-file di tipo script che presa la funzione $f(x)$ e un **`t`** esegua l'algoritmo di punto fisso sulla funzione $f(x)$ partendo da **`t`**.
È bene costruire un vettore **`x`** che contenga tutti i passi dell'algoritmo.

Criterio d'arresto:

1. Come primo criterio: dopo un certo numero di passi (usando **`for...end`**)
2. Come secondo criterio: stabilito un certo valore ε=**`epsilon`** quando $\mid x(i) - x(i+1) \mid < \varepsilon$, oppure quando rilevi che l'algoritmo non converge, usando **`if...end`**

Sperimentare il programma con le seguenti funzioni o con altre che si ritiene opportuno:

$f(x) = \cos(x) \qquad f(x) = 2 - e^x \qquad f(x) = 1 - \ln(x+1) \qquad f(x) = 1 - x^3 \qquad f(x) = e^{-x^2}$

Esercizio 1bis *funzione in Matlab*

Come **Esercizio 1**, ma come funzione di **`a`**

Esercizio 2bis *funzione in Matlab*

Come **Esercizio 2**, ma come funzione di **`x`**

Esercizio 3bis *funzione in Matlab*

Come **Esercizio 3**, ma come funzione di **`x`**

Esercizio 4bis *L'algoritmo di punto fisso*

Come **Esercizio 4**, ma come funzione di **`t`** e **`epsilon`**.
Lo si potrebbe fare canche come funzione della funzione, ma non è tanto affidabile. Meglio cambiare ogni volta il file.

Terza Esercitazione: Matlab come linguaggio: i cicli

Esercizio 1 *function in Matlab* **Metodo di bisezione**

Consideriamo la funzione $f(x) = \dfrac{1}{x^2 + 1/2} - e^x$ nell'intervallo $[0, 1]$

1. Verificare usando il teorema degli zeri che esiste un punto $\alpha \in [0, 1]$ tale che $f(\alpha) = 0$. Dimostrare che tale punto è unico.
2. Determinare analiticamente il numero N di iterazioni necessarie per calcolare α con un'approssimazione di $\varepsilon = 10^{-10}$ con il metodo di bisezione.
3. Completare il seguente programma che implementa il metodo di bisezione inserendo istruzioni al posto dei puntini.

```
a = 0; b = 1; % intervallo iniziale
epsilon = 10^-10;  % tolleranza
h = b-a; x = (a+b)/2;
```

con un ciclo **for...end**

```
N=........
for index=1:N
......
```

oppure con un ciclo **while...end**

```
  k = 0;
  while (h>eps)
....
if k>1000; disp('troppe iterazioni'); end
```

La funzione **fun** deve essere scritta nel file "fun.m"

```
  function y = fun(x)
    y = 1/(x^2+0.5)-exp(x);
  end
```

Esercizio 2 *function in Matlab* **Metodo di Newton-Raphson**

Consideriamo di nuovo la funzione $f(x) = \dfrac{1}{x^2 + 1/2} - e^x$ nell'intervallo $[0, 1]$

Completare il seguente programma che implementa il metodo di Newton inserendo istruzioni al posto dei puntini. Utilizzarlo per calcolare lo zero di f con un'approssimazione di $\varepsilon = 10^{-10}$.

```
a = 0; b = 1; % intervallo iniziale
epsilon = 10^-10;  % tolleranza
x0 = 0.5; err = 1;
%
 k = 0;
 while (err > epsilon) & (k < 1000)
 k = k + 1;
..............................
```

La derivate prima della funzione **fun** deve essere scritta nel file "derfun.m"

```
 function y = derfun(x)
   y = -2*x/(x^2+0.5)^2-exp(x);
 end
```

Quarta Esercitazione: I cicli e l'algebra lineare

Esercizio 1 *function in Matlab* **Una funzione con un input e due output.**

Dato un vettore, ne trova il massimo e la sua posizione.

Per esempio, se $x = [1 \quad 2 \quad 1.5 \quad 0 \quad 5 \quad 8.1 \quad -1 \quad 5]$, la funzione deve restituire 8.1 (il massimo) e 6 (la sua posizione).

La sintassi iniziale dev'essere

```
function [m,p]=massimo(x)
```

Nota: La funzione *built-in* **`max`** fa esattamente la stessa cosa per un vettore, ma si comporta diversamente per una matrice.

Esercizio 2 *function in Matlab* **Una funzione con controllo dell'input.**

Data una matrice **quadrata** ne estrae l'antidiagonale come vettore riga

Deve quindi dare errore se l'input non è una matrice quadrata.
Quindi prima di ogni altra cosa deve avere il controllo

```
if size(a,1) \simeq  size(a,2)
    error('non va')
....
```

La funzione **`error`** arresta il programma e produce un avvertimento.

Esiste anche la funzione **`warning`** che **non** arresta il programma.
Nota: Cercare di evitare un ciclo **`for...end`**

Esercizio 3 *function in Matlab* **L'algoritmo di Gauss: caso semplice.**

Una funzione che esegua su una matrice **quadrata** (controllo) l'algoritmo di Gauss nel caso semplice, supponendo cioè che non siano necessari scambi di righe e ci sia sempre il pivot sulla diagonale.
Costruire la funzione in tre tempi:

$\boxed{1}$ La funzione esegue solo il primo ciclo dell'algoritmo di Gauss con un loop **`for...end`**:

cioè da $\begin{pmatrix} a_{11} & a_{12} & \cdots & * \\ a_{21} & a_{22} & \cdots & * \\ \cdots & \cdots & \cdots & \\ a_{n1} & a_{n2} & \cdots & * \end{pmatrix}$ arriva solo a $\begin{pmatrix} a_{11} & a_{12} & \cdots & * \\ 0 & a_{22} & \cdots & * \\ \cdots & \cdots & \cdots & \\ 0 & a_{n2} & \cdots & * \end{pmatrix}$

Eventualmente può controllare se $a_{11} = 0$ e in tal caso arrestare la funzione.

$\boxed{2}$ La funzione esegue solo il primo e il secondo ciclo dell'algoritmo di Gauss con due loop **`for...end`** uno dopo l'altro:

cioè da $\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & * \\ a_{21} & a_{22} & a_{23} & \cdots & * \\ \cdots & \cdots & \cdots & & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & * \end{pmatrix}$ arriva solo a $\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & * \\ 0 & a_{22} & a_{23} & \cdots & * \\ \cdots & \cdots & \cdots & & \\ 0 & 0 & a_{n3} & \cdots & * \end{pmatrix}$

Eventualmente può controllare, dopo il primo ciclo, se $a_{22} = 0$ e in tal caso arrestare la funzione.

$\boxed{3}$ La funzione esegue tutto l'algoritmo di Gauss sulle prime $n-1$ colonne:

Esaminando il listato del passo precedente si può capire come va eliminato il secondo loop e costruito un loop di $n-1$ passi che ingloba il primo (ci sono due *nested loops*).

Come sopra può controllare, prima di ogni ciclo, se $a_{ii} = 0$ e in tal caso arrestare la funzione.

Esercizio 4 *function in Matlab* **L'algoritmo di Gauss-Jordan nel caso semplice**

Modificare il precedente algoritmo per fargli eseguire Gauss-Jordan anziché solo Gauss, cioè

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \cdots \\ a_{21} & a_{22} & a_{23} & a_{24} & \cdots \\ a_{31} & a_{32} & a_{33} & a_{34} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \\ a_{n1} & a_{n2} & a_{n3} & a_{n4} & \cdots \end{pmatrix} \rightarrow \begin{pmatrix} a_{11} & 0 & a_{13} & a_{14} & \cdots \\ 0 & a_{22} & a_{23} & a_{24} & \cdots \\ 0 & 0 & a_{33} & a_{34} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \\ 0 & 0 & a_{n3} & a_{n4} & \cdots \end{pmatrix} \rightarrow \begin{pmatrix} a_{11} & 0 & 0 & a_{14} & \cdots \\ 0 & a_{22} & 0 & a_{24} & \cdots \\ 0 & 0 & a_{33} & a_{34} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \\ 0 & 0 & 0 & a_{n4} & \cdots \end{pmatrix}$$

Quinta Esercitazione: Algebra lineare: LU, Jacobi, Gauss-Seidel Introduzione alla grafica

Si considerino i sistemi $Ax = b$ e $A_1x = b$ seguenti:

$$A = \begin{pmatrix} 9 & -5 & 3 & 0 \\ 2 & 4 & -3 & -4 \\ -1 & 3 & 6 & -5 \\ 0 & -5 & 1 & 7 \end{pmatrix} \qquad A_1 = \begin{pmatrix} 4 & 3 & 3 & 3 \\ 4 & 5 & -5 & 0 \\ -1 & -3 & 6 & 2 \\ -1 & 1 & -2 & 4 \end{pmatrix} \qquad b = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

Esercizio 1 *Gauss in Matlab*

Risolvere i sistemi mediante gli algoritmi di Gauss *built-in* in Matlab (sia con "\", sia con **`rref`**)

Esercizio 2 *LU in Matlab*

Risolvere i sistemi mediante la fattorizzazione LU.

Esercizio 3 *function in Matlab* **Jacobi e Gauss-Seidel.**

Completare la seguente funzione **`[x,n] = jacobi(A,b,eps)`** che usa il metodo di Jacobi per calcolare la soluzione del sistema con una tolleranza di **`epsilon`**.
Il numero di iterazioni effettuate è **`n`**.

Usare come criterio di arresto successivamente

1. Il criterio del residuo. $| Ax - b | < epsilon$
2. La norma $| x_i - x_{i+1} | < epsilon$

```
 function [x,n] = jacobi(A,b,eps);
 S =
 T=;
 x=ones(size(b));
 nmax = 200; % numero massimo di iterazioni
 n = 0;
 residuo = norm(A*x-b);
% while n <= nmax   & residuo > epsilon      % primo criterio
% while n <= nmax   & abs(x-... > epsilon      % secondo criterio
  ...
```

Esercizio 4 *Grafici in Matlab*

Disegnare il grafico delle seguenti funzioni negli intervalli dati:

$f(x) = \dfrac{x^2+x+1}{x^2+1}$ $\quad x \in [-2, 2]$	$f(x) = \log(x^2 - x + 1)$ $\quad x \in [0, 2]$
$f(x) = \begin{cases} \arctan\left(\dfrac{1}{x^2+1}\right) \cdot \dfrac{4}{\pi} & x \in [-1, 0] \\ \sqrt{\mid 1 - x^2 \mid} & x \in [0, 2] \end{cases}$	$f(x) = \begin{cases} \dfrac{2x^3+1}{x} & x \in (0, 1) \\ \log(2 - x) + 3 & x \in (1, 2) \end{cases}$

Esercizio 4 *Grafici in Matlab* **Una funzione con ricerca massimi e minimi**

Consideriamo per ogni $k \in \mathbb{R}$ la funzione seguente: $f(x) = e^{\left(\frac{x^2 - x + k}{x^2+1}\right)}$

Costruire un m-file **funzione** di k che tracci il grafico di f nell'intervallo $I = [-3, 3]$ (con approssimazione di 0.05) e abbia come outupt la matrice così fatta

$$\begin{pmatrix} \text{minimo di } f(x) \text{ in } I & \text{massimo di } f(x) \text{ in } I \\ \text{punto di minimo di } f(x) \text{ in } I & \text{punto di massimo di } f(x) \text{ in } I \end{pmatrix}$$

Sesta Esercitazione: Autovalori, auotovettori, condizionamento Introduzione alla grafica in 3D

Consideriamo $\forall\, k \in \mathbb{R}$ le matrici 4×4 $A = \begin{pmatrix} 1 & 0 & k & k \\ 2 & 3 & -1 & 1 \\ k & 2 & 2 & 0 \\ k & 1 & -1 & 0 \end{pmatrix}$ $\quad B = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 3 & 1 & 0 \\ 0 & 1 & k & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}$

Esercizio 1 *Autovalori idi una matrice qualunque*

Calcolare tutti gli autovalori e tutti gli autovettori di A per $k = -4$ e $k = 0$ e verificare l'eventuale ortogonalità degli autovettori.

Esercizio 2 *Autovalori di una matrice simmetrica.*

Calcolare tutti gli autovalori e tutti gli autovettori di B per $k = 0$ e $k = -3/5$ e verificare l'eventuale ortogonalità degli autovettori.

Esercizio 3 *Condizionamento in Matlab*

1. Mediante un m-file determinare (a meno di 10^{-2}) il $k \in [0, 1]$ per cui cond(A) è massimo e disegnare il grafico di cond(A) in funzione di k.
2. Per il k trovato, risolvere:

   il sistema $Ax = b$ con $b = \begin{pmatrix} 1 \\ 4 \\ 2 \\ 1 \end{pmatrix}$ e il sistema $Ax = b_1$ con $b_1 = \begin{pmatrix} 1.1 \\ 3.9 \\ 2 \\ 1 \end{pmatrix}$ e confrontare le soluzioni
3. Mediante un m-file determinare (a meno di 10^{-2}) il $k \in [2, 4]$ per cui cond(B) è minimo e disegnare il grafico di cond(B) in funzione di k.

Esercizio 4 *Grafici in 3D in Matlab*

Eseguire il grafico in 3D le seguenti funzioni nei domini indicati mediante varie opzioni (griglia, superficie, linee di livello, colori vari, angoli di visione vari):

1	$z = x^2 - y^2$	$-2 \leq x \leq 2\,;\ -2 \leq y \leq 2$
2	$z = \sqrt{1 - x^2}$	$-1 \leq x \leq 1\,;\ -2 \leq y \leq 2$
3	$z = \sqrt{\max\{1 - x^2, 1 - y^2\}}$	$-1 \leq x \leq 1\,;\ -2 \leq y \leq 2$
4	$z = \sqrt{\max\{1 - x^2, 1 - y^2\}}$	$-2 \leq x \leq 2\,;\ -2 \leq y \leq 2$
5	$z = \sqrt{1 - x^2 - y^2\}}$	$-1 \leq x \leq 1\,;\ -1 \leq y \leq 1$
6	$z = \sqrt{1 - x^2 + y^2\}}$	$-1 \leq x \leq 1\,;\ -1 \leq y \leq 1$
7	$z = \begin{cases} 1 - x^2 & \text{se } x >= y \\ 2x - 2y & \text{se } x < y \end{cases}$	$-1 \leq x \leq 1\,;\ -1 \leq y \leq 1$

Settima Esercitazione: Interpolazione, gestione stringhe Calendario (funzione MOD)

Premessa 1: Come leggere dati da un file esterno. Da un file di testo o da una tabella Excel. Si usano i comandi **`load`** e **`xlsread`**

```
clear all; % cancella tutte le variabili
load DatiA.txt % file di testo
load DatiB.txt % file di testo
A1 = xlsread('Datix.xls') % file Excel
```

Premessa 2: Come usare le stringhe. Se la variabile **`a`** vale 5.2234 e volete scrivere su schermo

```
La variabile a vale 5.2234
```

Dato che il comando **`disp`** non può scrivere stringhe e numeri insieme, occorre trasformare il numero in stringa col comando **`num2str`** (in inglese si legge *number to string*) e concatenare le stringhe come se fossero vettori.

```
disp(['La variabile a vale ' , num2str(a)])
```

C'è anche il comando **`sprintf`** che è più flessibile, ma occorre conoscere un po' di C^{++}

Premessa 3: Per fare la divisione di numeri interi occorrono la funzione **`/`** e la funzione **`mod`**. Esempio: Dividendo 157 per 7 si ha quoziente $q = 22$ e resto $r = 3$.

```
q = fix(157/7) , r=mod(157,7)
```

Premessa 4: Il comando **`switch`**

Invece del test condizionale **`if ... end`** è conveniente in molti casi (anzi è consigliato da molti programmatori e da molte norme) il comando **`switch`**

L'esempio tipico è

```
if  a==1
    disp('bene')
elseif  a==-1;
    disp('male')
elseif  a==-2;
    disp('male')
elseif  a==0
    disp('cosi''')
else
    disp('non so')
end
```

Confrontare la leggibilità del listato sopra col seguente che fa le stesse cose

```
switch a
case 1
    disp('bene')
case {-1,-2 }
    disp('male')
case  0
    disp('cosi''')
otherwise
    disp('non so')
end
```

Esercizio 1 *Interpolazione in vari modi*

porre $x = (1, 2, ..., 10)$ e ricavare y dal file DatiA.
Indi disegnare i punti usando l'opzione **`plot (x,y,'o')`**
Successivamente:

- interpolare con una spezzata
- interpolare con una spline (leggere le istruzioni del comando **`spline`**)
- approssimare ai minimi quadrati con una retta.
- approssimare ai minimi quadrati con una parabola.

per sovrapporre i vari disegni si possono usare due opzioni:

```
plot(x,y,' ',x1,y1,' ', ...)
```

```
plot(x,y,' ')
hold on
plot(x1,y1,' ')
```

E poi **`hold off`** quando non si vuole più sovrapporre.

Esercizio 2 *Calendario: calcolare il giorno della settimana di una data.*

Creare un funzione **`calendario(g,m,a)`** che caslcola il giorno della settimana di g/m/a.

Si comincia col calcolare il giorno in cui cade il primo gennaio nell'anno a:
Si parte da un anno in cui il primo gennaio cadeva di domenica p.es. 1928.
Si calcola quanti giorni in più ci sono in questo modo:
Sia $d = a - 1928$:
Se la divisione di d per 4 è esatta si aggiunge il quoziente della divisione per 4: $d = d + q$
Altrimenti si aggiunge il quoziente +1 : $d = d + q + 1$.

Il primo gennaio comincia dell'anno a cade di d (0=domenica, 1=lunedì etc.)
Se $d > 7$ si riduce modulo 7 (comando **`mod`**).

Per calcolare il primo giorno del mese corrente si usano le seguenti tabelle: (usare il comando **`switch`**)

Anni normali	**Anni bisestili**
0 = Gen Ott	0 = Gen Apr Lug
1 = Mag	1 = Ott
2 = Ago	2 = Mag
3 = Feb Mar Nov	3 = Feb Ago
4 = Giu	4 = Mar Nov
5 = Set Dic	5 = Giu
6 = Apr Lug	6 = Set Dic

Ovvero si aggiunge il numero dato al giorno del primo gennaio (e si riduce modulo 7).

A questo punto dovrebbe essere facile calcolare qualunque giorno di qualunque anno dal 1/1/1928 al 31/12/2099 (il 2100 sarà un anno particolare, non bisestile anche se divisibile per 4).

Ottava Esercitazione: Altri tipi di grafici, equazioni differenziali

Esercizio 1 *Curve parametriche*

Disegnare il grafico delle seguenti curve piane o nello spazio espresse in forma parametrica: (comandi **`plot`** e **`plot3`**, cercando di disegnare la porzione più significativa

$$\begin{cases} x = t^2 - 1 \\ y = t\,(t^2 - 1) \end{cases} \qquad \begin{cases} x = t^2 \\ y = t^3 \end{cases} \qquad \begin{cases} x = 2\,\cos(t) \\ y = 3\,\sin(t) \end{cases} \qquad \begin{cases} x = 2\,\cosh(t) \\ y = 3\,\sinh(t) \end{cases}$$

$$\begin{cases} x = 2\,\cos(t) \\ y = 3\,\sin(t) \\ z = t \end{cases} \qquad \begin{cases} x = 2\,\cos(t) \\ y = 3\,\sin(t) \\ z = \sin(8t) \end{cases}$$

Esercizio 2 *Curve polari.*

Disegnare il grafico delle seguenti curve piane espresse in forma polare: (comando **`polar`**), cercando di disegnare la porzione più significativa

$$\varrho = \theta \qquad \varrho = cos(n\theta) \quad n \in \mathbb{N} \qquad \varrho = 1 + cos(\theta)$$

Esercizio 3 *Problema di Cauchy*

Costruire un m-file funzione che dipendente da t_0, y_0, t_1, h che calcoli la soluzione approssimata del problema di Cauchy

$$\begin{cases} y' = \mathsf{fun}(t, y) \\ y(t_0) = y_0 \end{cases}$$

usando il metodo di Eulero, nell'intervallo $[t_0, t_1]$ con passo h.

La funzione **`fun`** sarà definita in un file a parte e richiamata dal programma.

Collaudare l'm-file coi seguenti problemi differenziali

$$\begin{array}{lll} y' = \mathsf{fun}(t, y) & y(0) = 1 & \mathsf{fun}(t, y) = t^2/4 + ty + 1 \\ y' = \mathsf{fun}(t, y) & y(0) = 1 & \mathsf{fun}(t, y) = t - t\,y^2 + 1 \\ y' = \mathsf{fun}(t, y) & y(0) = 1 & \mathsf{fun}(t, y) = t^2\,y - t\,y^2 + 1 \\ y' = \mathsf{fun}(t, y) & y(-1) = 1 & \mathsf{fun}(t, y) = t^2\,y - t\,y^2 + 1 \end{array}$$

Disegnare la funzione con vari passi e in vari intervalli e confrontare i grafici ottenuti.

Usare poi la funzione predefinita in MatLab **`ode23('fun',t,y0)`** e confrontare i risultati.

Nona Esercitazione: Equazioni alle differenze e programmazione

Esercizio 1 *Equazione alle differenze*

Costruire un m-file funzione di `c,p,h` che calcoli e disegni la soluzione approssimata del problema differenziale

$$\begin{cases} y'' + c(t)y(t) = p(t) \\ y(t_a) = 0 \qquad y(t_b) = 0 \end{cases}$$

usando le differenze finite nell'intervallo $[t_a, t_b]$ con passo h.

Sia `t` l'intervallo diviso con passo h. Siano `c` e `p` le tabulazioni di $c(t)$ e $p(t)$. L'incognita è `y`.

Occorrerà risolvere un sistema lineare tridiagonale

$$\begin{cases} \dfrac{y_3 - 2y_2 + y_1}{h^2} + c_2 y_2 & = & p_2 \\ \dfrac{y_4 - 2y_3 + y_2}{h^2} + c_3 y_3 & = & p_3 \\ \dots & \dots & \dots \\ \dfrac{y_n - 2y_{n-1} + y_{n-2}}{h^2} + c_{n-1} y_{n-1} & = & p_{n-1} \end{cases}$$

È un sistema nelle incognite y_2 , ... , y_{n-1} la cui matrice è tridiagonale. La diagonale è $[c_2 h^2 - 2 , \dots , c_{n-1}h^2 - 2]$. La sopra e la sottodiagonale sono fatte di 1.

Il termine noto è $[p_2 h^2 , \dots , p_{n-1}h^2] - [y_1, 0, 0, ..., 0, y_n]$ (trasposto).

Provare inizialmente usando come `c` la matrice di -1 e come `p` la matrice [1 1 ...1 2 2 2] (circa a metà). L'intervallo $t[_a, t_b]$ non è importante (per es. può essere $[0, 1]$)

Disegnare la funzione con vari `p` e confrontare i grafici ottenuti.

Esercizio 2 *Il commesso viaggiatore*

Il problema è il seguente: Sono dati sette punti nel piano, per esempio quelli a lato che possono essere pensati come città da visitare.
Si tratta di trovare l'itinerario più breve che partendo da $(0, 0)$ passi per tutti i sette punti.

Iniziare col costruire la matrice `d` 7×7 delle distanze tra i vari punti. Il programma per trovare l'itinerario (nel peggior modo, cioè provando tutti i 5040 percorsi possibili) può essere articolato così:

x	y
3	0
2	1
0	2
5	2
4	4
1	5
6	7

```
d=...  % ---- COSTRUIRE d -----
towns=7 % numero città da visitare
w=perms(1:towns); % costruisce tutte le possibili permutazioni dei
          % numeri da 1 a towns
          % Per towns=7 è una matrice di 5040 righe.  Non scrivetela !!!
minimo=10000 % Inizializzare minimo con un numero alto
for id=1:length(w)
      dist=sqrt(x(w(id,1))^2+y(w(id,1))^2 )
                % inizializza la distanza (da (0,0) alla prima città)
      for jd=1:towns-1 % i segmenti sono towns-1
              dist=dist+d(..  ,  ..)  % inserire gli indici giusti...
      end
      if....  % vedo se la distanza è minima e, se l'ho trovata,
              % devo conservare la permutazione per la fine
              % e aggiornare il valore di minimo
          plot(....)  % magari disegnare l'itinerario trovato
      end
end
```

Nota: Tra le demo di MatLab c'è il problema del commesso viaggiatore, fatto in modo un po' più efficiente...