Introduzione agli ambienti MATLAB© e Octave

Utilizzo di Matlab/Octave ed operazioni fondamentali.
Primi rudimenti di grafica.

MATLAB e Octave

MATLAB = **MAT**rix **LAB**oratory è un ambiente di programmazione orientato al calcolo scientifico.

Octave è definito come un interprete per linguaggio di alto livello.
- hanno una buona potenzialità grafica (integrata per Matlab, basata su gnuplot per Octave)
- esistono versioni per Unix/Linux, Windows, Mac.
- i files sono portabili da una piattaforma all'altra e da matlab a octave e viceversa.

MATLAB: `www.mathworks.com`
Link alla licenza unibs: `https://www.unibs.it/it/opportunita-e-servizi/servizi/servizi-digitali/matlab`

Octave NON è la versione free di Matlab, ma è largamente compatibile con Matlab. È distribuito gratuitamente qui:
`http://www.gnu.org/software/octave/`

- Entrambi hanno **funzioni intrinseche** molto potenti (es: risoluzione di sistemi lineari, calcolo di autovalori e autovettori di una matrice);
- esistono **toolbox** o **packages** (librerie di software specifico -file scritti in linguaggio matlab/octave-):
  - Control System
  - Signal Processing
  - Statistics
  - Communications
  - ...

Prompt:
>>

Le istruzioni che vedremo sono valide sia per Matlab che per Octave.

Assegnazione di variabili scalari

```
>>a=1.54
```

- a nome della variabile (max 31 caratteri alfanumerici, il primo dei quali non deve essere un numero),
- `1.54` valore numerico assegnato alla variabile,
- Di default lettere maiuscole e minuscole sono considerate diverse sia nei comandi che nei nomi delle variabili.

Il comando

`>> a=1.54` 

produce

```
a =
    1.5400
```

`>> a=1.54;` 

non produce risposta

```
>> 1.67
```


produce

```
ans =
    1.6700
```

`ans` è il nome della variabile di default.

per visualizzare il contenuto della variabile a

```
>> a
```


produce

```
a =
    1.5400
```

per poter spezzare un'istruzione troppo lunga:
tre punti in sequenza

```
>> b=1+1/2+5/3+1/4+23/6+...
2/9+1/10;
```

Operazioni aritmetiche

^ potenza
* prodotto
/ divisione
+ somma
- differenza

Es: per calcolare $x = \dfrac{3 + 5^3 - 2/3}{4(5 + 2^4)}$ il comando da dare è:

```
>> x=(3+5^3-2/3)/(4*(5+2^4))
```

- Sono osservate le precedenze classiche dell'artimetica
- Per alterare le precedenze si utilizzano esclusivamente le parentesi **tonde**

Quali variabili sono in memoria

```
>> whos

  Name        Size                Bytes  Class

  a           1x1                     8  double array
  ans         1x1                     8  double array
  b           1x1                     8  double array
  x           1x1                     8  double array
```

Di default, Matlab/Octave lavorano con variabili in **doppia precisione** .
Ogni numero memorizzato in doppia precisione occupa 8 Bytes.
Le variabili scalari sono viste come **array** di dimensione `1x1` (una riga e una colonna).

OSS. Di default lettere maiuscole e minuscole sono considerate diverse sia nei comandi che nei nomi delle variabili.

Formato di **rappresentazione** dei numeri

```
>> c=0.456723
c =
    0.4567
```

Il numero è stato rappresentato con 5 cifre

```
>> format short e
>> c
c =
    4.5672e-01
```

Forma esponenziale con 5 cifre per la mantissa

```
>> format long e
>> c
c =
     4.567230000000000e-01
```

Forma esponenziale con 16 cifre per la mantissa

```
>> format long
>> c
c =
    0.45672300000000
```

Il numero è rappresentato con 15 cifre

Di default viene utilizzato il formato `format short`. Per tornare a questo formato di rappresentazione:

```
>> format short
```

N.B. Il formato di rappresentazione può cambiare, ma il formato di memorizzazione dei numeri è sempre lo stesso (8Bytes).

Variabili predefinite

`pi`	π
`i, j`	$\sqrt{-1}$ unità immaginaria
`NaN`	not a number
`eps`	`2.2204e-16` precisione di macchina

Il contenuto di queste variabili può essere variato con una semplice operazione di assegnazione:

```
>> pi=18
pi =
    18
```

Per riassegnare alla variabile `pi` il valore π:

```
>> clear pi
>> pi
ans =
    3.1416
```

Per cancellare il contenuto di tutte le variabili:

```
>> clear
```

Assegnazione di array

```
>> a=[1  2  3  4];
>> a=[1,2,3,4];
>> a=(1:4);
```

Modi equivalenti per generare un array 1x4, 1 riga e 4 colonne, vettore riga

```
>> a
a =
     1     2     3     4
```

```
>> b=[1;2;3;4]
b =
     1
     2
     3
     4
```

Per generare un array 4x1, 4 righe e 1 colonna, vettore colonna

```
>> c=[5 3 4; 2 4 -2]
c =
     5     3     4
     2     4    -2
```

Per generare un array 2x3, matrice 2 righe e 3 colonne

Lo spazio o la virgola separano elementi sulla stessa riga. Il punto e virgola separa le righe.

Operazione di trasposizione:

```
>> a'
ans =
     1
     2
     3
     4
```

Il vettore trasposto di `a` *viene memorizzato nella variabile* `ans`

```
>> a1=a'
```

Il vettore trasposto di `a` *viene memorizzato nella variabile* `a1`

Analogo discorso vale per la trasposizione di matrici:

```
>> c1=c'
c1 =
     5     2
     3     4
     4    -2
>> whos

  Name        Size                Bytes  Class

  a           1x4                    32  double array
  ans         4x1                    32  double array
  b           4x1                    32  double array
  c           2x3                    48  double array
  c1          3x2                    48  double array
```

```
>> a(2)
ans =
      2
```

Per accedere ad un elemento di un vettore

```
>> c(2,1)
ans =
      2
```

Per accedere ad un elemento di una matrice

```
>> d=c(1,:)
d =
      5      3      4
```

Per estrarre la prima riga di una matrice

```
>> e=c(:,1:2)
e =
      5      3
      2      4
```

Per estrarre le prime due colonne di una matrice

```
>> b(3)=5
b =
     1
     2
     5
     4
```

Per modificare un elemento di un vettore. Se non si utilizza il ";" viene visualizzato l'array completo

```
>> c(1,3)=18
c =
     5     3    18
     2     4    -2
```

Per modificare un elemento di una matrice.

Operazioni su array

+ somma di vettori o matrici (elemento per elemento)
- differenza di vettori o matrici (elemento per elemento)
* prodotto tra vettori e/o matrici (righe per colonne)

Sono le operazioni dell'algebra lineare; quindi:

- per somma e differenza: gli operandi devono avere le stesse dimensioni
- per il prodotto: la dimensione interna dei due array deve coincidere.

```
>> f=a1+b
f =
      2
      4
      8
      8
```

$$f_i = a1_i + b_i$$

$$\text{con } a1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \text{ e } b = \begin{bmatrix} 1 \\ 2 \\ 5 \\ 4 \end{bmatrix}$$

OK, d e b sono entrambi vettori colonna (4x1)

```
>> g=a-b
```

a=vettore riga (1x4)
b=vettore colonna (4x1)
L'OPERAZIONE NON HA SENSO IN ALGEBRA LINEARE, MA MATLAB LA SVOLGE, espandendo i vettori a matrici e facendo la somma di matrici

Prodotto scalare tra vettori

Se $a \in \mathbb{R}^{1\times n}$ e $b \in \mathbb{R}^{n\times 1}$

$$\mathtt{a}\cdot\mathtt{b} = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$$

Dimensioni: $(1 \times \cancel{n})(\cancel{n} \times 1) \rightarrow (1 \times 1)$
quindi il risultato è uno scalare

```
>> a*b
ans =
     36
```

$$a = [1,\ 2,\ 3,\ 4], \quad b = \begin{bmatrix} 1 \\ 2 \\ 5 \\ 4 \end{bmatrix}$$

(1x4)(4x1) -prodotto scalare- OK

```
>> a*a
Error using  *
Inner matrix dimensions must agree.
```

(1x4)(1x4) -prodotto non possibile

Prodotto di matrici

Ricordiamo che se $A \in \mathbb{R}^{n\times m}$ e $B \in \mathbb{R}^{p\times q}$, il prodotto AB è possibile solo se $m = p$ e che
se $A \in \mathbb{R}^{n\times m}$ e $B \in \mathbb{R}^{m\times q}$, allora $C \in \mathbb{R}^{n\times q}$ $((n \times \cancel{m})(\cancel{m} \times q) \rightarrow (n \times q))$
e

$$C_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}, \qquad i = 1, \ldots n,\ j = 1, \ldots, q$$

l'operazione * realizza il prodotto tra matrici:

```
A=[2 3 4; 1 -2 1];   % (2x3)
B=[3 1 4; 2 -1 0; 2 7 -1];   % (3x3)
C=A*B    % (2x3)(3x3) --> (2,3)    OK

C =
    20    27     4
     1    10     3
```

```
>> B*A
??? Error using ==> *
Inner matrix dimensions must agree.
```

(3x3)(2x3) - prodotto non possibile-

Operazioni punto

Esistono poi le operazioni "punto" che agiscono su array che abbiano le stesse dimensioni:

`.*` prodotto elemento per elemento

`./` divisione elemento per elemento

`.^` potenza elemento per elemento

```
>> b=[2;4;1;-2];
>> b2=b.*b
b2 =
     4
    16
     1
     4
```

$$(b2)_i = b_i * b_i$$

essendo $b = \begin{bmatrix} 2 \\ 4 \\ 1 \\ -2 \end{bmatrix}$

Avrei ottenuto lo stesso risultato con

```
>> b2=b.^2
```

Semplici comandi sulle matrici

```
[n,m]=size(A) % restituisce il n. di righe
              % e colonne di A
det(A)        % calcola il determinante di A (n x n)
rank(A)       % calcola il rango di A
inv(A)        % calcola l'inversa di A (n x n)
eig(A)        % calcola gli autovalori di A (n x n)
x=A\b         % risolve il sistema lineare
              % A x = b, dove A (n x n), b(n x 1)
% attenzione a non confondere
%  / (slash) con \ (backslash)
```

Funzioni matematiche e grafica

Problema 1: disegnare $f(x) = (2x - \sqrt{2})^2 \sin(2x)$ sull'intervallo $I = [-2\pi, 2\pi]$

```
% definisco la funzione con il function handle
f=@(x)(2*x-sqrt(2)).^2.*sin(2*x)
% apro una finestra grafica (che ha numero 1)
figure(1)
% fplot(f,[a,b]), [a,b] intervallo di definizione
fplot(f,[-2*pi,2*pi]);
legend('f(x)=(2x-\sqrt{2})^2\sin(2x)')
xlabel('x') % aggiungo label all'asse x
ylabel('y') % aggiungo label all'asse y
grid on % griglia
```

N.B. Il comando `fplot` valuta f in un insieme di punti non necessariamente equispaziati, scelti in maniera automatica e congiunge questi punti con dei segmenti.

Problema 2: valutare $f(x) = \sin(x^2)$ in 100 punti equispaziati $x_i \in [-2\pi, 2\pi]$ con $i = 1, \ldots, 100$ e disegnare la funzione utilizzando i punti $(x_i, f(x_i))$.

```
>> x=linspace(-2*pi,2*pi,100);
% x=linspace(a,b,n) crea un vettore riga di n elem,
% contenenti le ascisse di n punti equispaziati
% sull'intervallo chiuso [a,b]

>> f=@(x)sin(x.^2);
% definisco f
>> y=f(x);      % valuto f
```

x è un vettore, si vuole calcolare $y_i = sin(x_i^2)$ per ogni i, quindi si deve usare l'operazione ".^"

```
>> figure(2)   % apro una finestra grafica
>> plot(x,y)   % matlab disegna la spezzata che
               % congiunge i punti (x_i,y_i)
```

La sintassi del comando `plot` è:
`plot(x,y, 'color_linestyle_marker')`

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

color: `c,m,y,r,b,g,w,k`

linestyle: `-,--,:,-.,none`

marker: `+,o,*,.,x,s`

Per disegnare 2 o più coppie di vettori sullo stesso grafico, ad esempio $f(x) = sin(x^2)$ e $g(x) = (\sin(x))^2$:

```
>> g=@(x)(sin(x)).^2;
>> yg=g(x);
>> plot(x,y,'b-',x,yg,'r--');
```

bisogna ripetere: 'ascisse, ordinate, specifiche' per ogni coppia di vettori.
Le specifiche sono opzionali.

Per conoscere nel dettaglio tutte le opzioni di un comando, oppure se non ci si ricorda la sintassi del comando:

`help` *nome_comando*

```
>> help plot
```

Se non ci si ricorda il nome del comando, ma si vuole fare una ricerca per *parola_chiave* (in inglese), oppure se si cercano tutti i comandi che facciano riferimento ad una *parola_chiave*:

`lookfor` *parola_chiave*

```
>> lookfor  plot
```

Funzioni matematiche intrinseche

`sqrt(x)`	$\sqrt{x}$
`round(x)`	arrotondamento: `round(3.6)=4`
`fix(x)`	parte intera del numero: `fix(3.6)=3`
`sign(x)`	segno di `x` (vale -1, 0 o 1)
`sin(x), cos(x), tan(x)`	$\sin(x), \cos(x), \tan(x)$
`sinh(x), cosh(x), tanh(x)`	$\sinh(x), \cosh(x), \tanh(x)$
`asin(x), acos(x), atan(x)`	$\arcsin(x), \arccos(x), \arctan(x)$
`exp(x), log(x), log10(x)`	e^x, $\log_e(x)$, $\log_{10}(x)$

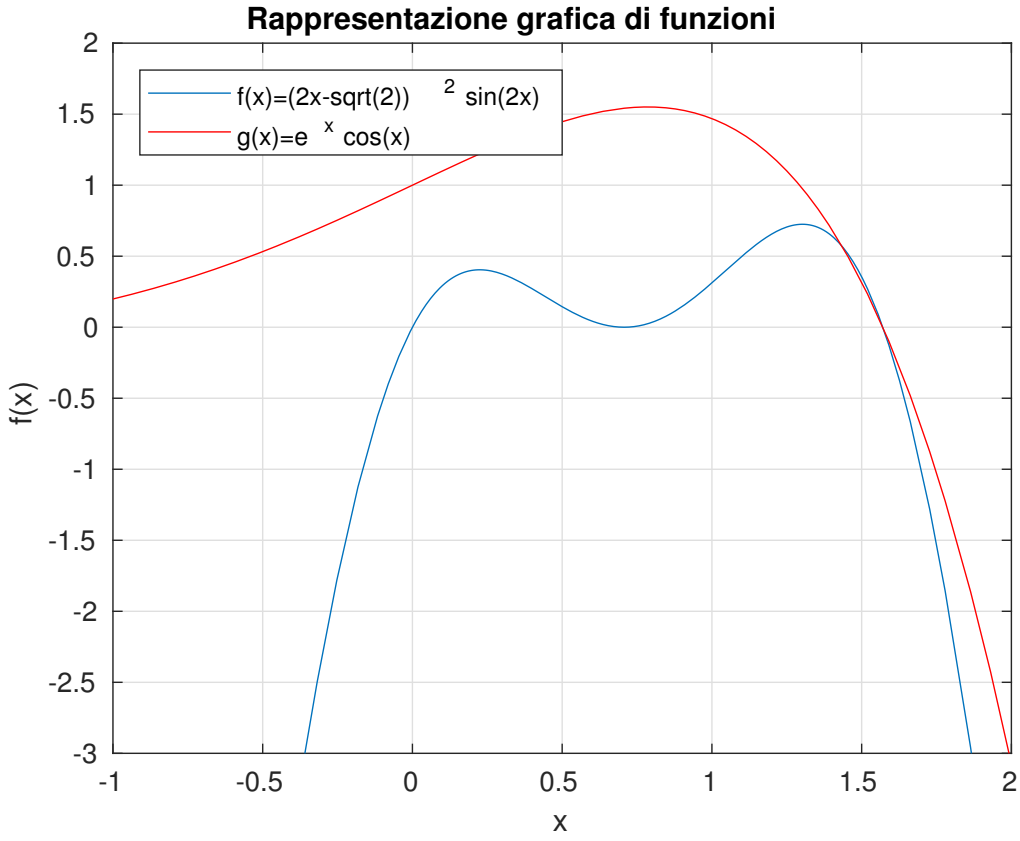
Per `z` complesso:

```
>> z=3+i*4
```

`real(z)`	parte reale di `z`
`imag(z)`	parte immaginaria di `z`
`conj(z)`	complesso coniugato di `z`
`abs(z)`	modulo di `z`

Generazione di un m-file

(Il lavoro con Octave è simile)
Problema 2: Creazione di un grafico 2D.
Disegnare $f(x) = (2x - \sqrt{2})^2 \sin(2x)$ e $g(x) = e^x \cos(x)$ sull'intervallo $I = [-1, 2]$.



Dal menù **Editor** selezionare **New**. Si apre una finestra di **Editor** in cui si possono scrivere i comandi matlab (non compare più il prompt)

```
f=@(x)(2*x-sqrt(2)).^2.*sin(2*x);
figure(1); clf
fplot(f,[-1,2])
xlabel('x'); ylabel('f(x)')
title('Rappresentazione grafica di funzioni')
hold on % mantiene il grafico fatto
g=@(x)exp(x).*cos(x);
fplot(g,[-1,2],'r')
l=legend('f(x)=(2x-sqrt(2))^2 sin(2x)','g(x)=e^x cos(x)');
set(l,'Location','Northwest')
grid on % disegna la griglia
axis([-1,2,-3,2]) % fissa il box della figura
hold off
```

- Salvare
  Per salvare il contenuto del file: dal menù dell'Editor selezionare **Save as**.
  Specificare il direttorio in cui salvare (es: `c:\tmp` o `e:\`) ed il nome per il file (es: `dis2d.m`)
  N.B. L'estensione dei file matlab è sempre **m**.

- aggiungere il path
  Dalla finestra dei comandi matlab:
  `>> addpath c:\tmp` oppure
  `>> addpath e:\`
  per dire di cercare il file in tale direttorio, quindi richiamare il file generato, dando il nome del file stesso:
  `>> dis2d`

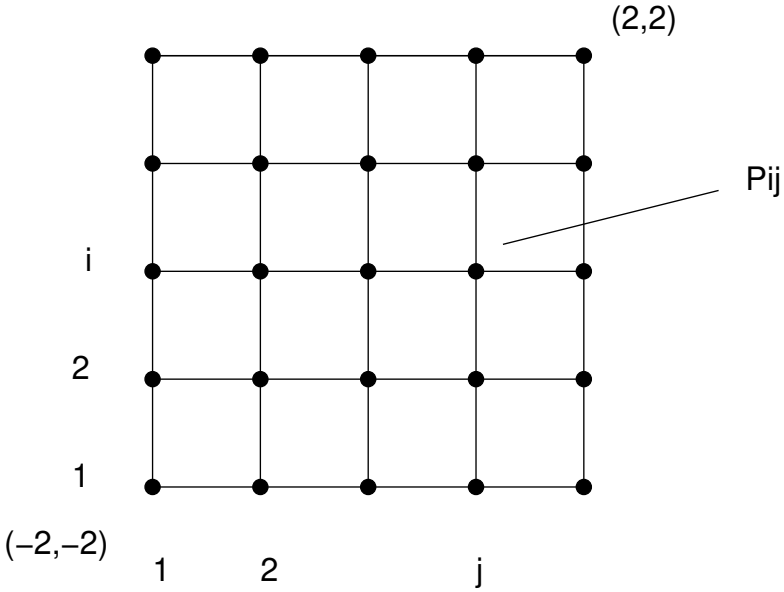
Matlab/Octave segnala errori?

1. Leggere il tipo di errore
2. Tornare nell'editor, cercare l'errore e modificare il file
3. Salvare le modifiche effettuate
4. Tornare alla finestra dei comandi Matlab e ridare il comando

```
>> dis2d
```

Grafici 3D

Problema: Rappresentare graficamente $f(x,y) = xe^{-(x^2+y^2)}$ sul dominio $\Omega = [-2,2]^2$.



Anzitutto bisogna definire una griglia su Ω.

```
>> [x,y]=meshgrid(-2:.1:2,-2:.1:2);
```

x e y sono due matrici

.

```
>> clf
```

Per pulire la figura precedente

```
>> f=@(x,y)x.*exp(-x.^2-y.^2);
>> z=f(x,y); surf(x,y,z); colorbar
```

Altri comandi di grafica 3D:

`>> mesh(x,y,z)`	*Superficie*
`>> meshc(x,y,z)`	*Superficie e countour-lines*
`>> surfc(x,y,z)`	*Superficie e countour-lines*
`>> pcolor(x,y,z)`	*Superficie colorata piatta*
`>> surf(x,y,z,gradient(z))`	*Superficie colorata secondo la grandezza di* $\partial z/\partial x$
`>> contour(x,y,z)`	*Contour-lines (linee di livello)*
`>> plot3(x,y,z)`	*Linee lungo la direzione* y *serve anche per disegnare linee in 3D*

Per creare più figure, basta anteporre al comando di disegno l'istruzione `figure(k)` dove `k` è un numero intero positivo di una figura non attiva. Es:.

```
>> mesh(x,y,z);
>> figure(2); surf(x,y,z,gradient(z));
>> figure(3); plot3(x,y,z);
```

Per passare il comando da una finestra all'altra, al fine di modificare il grafico:

```
>> figure(2)
>> colorbar
```

Se si vuole una sola finestra con più grafici:

```
>> figure(1)
>> subplot(2,2,1); mesh(x,y,z);
>> title('mesh')
>> subplot(2,2,2); surfc(x,y,z);
>> title('surfc')
>> subplot(2,2,3); plot3(x,y,z);
>> title('plot3')
>> subplot(2,2,4); surf(x,y,z,gradient(z));
>> title('surf,gradient')
```

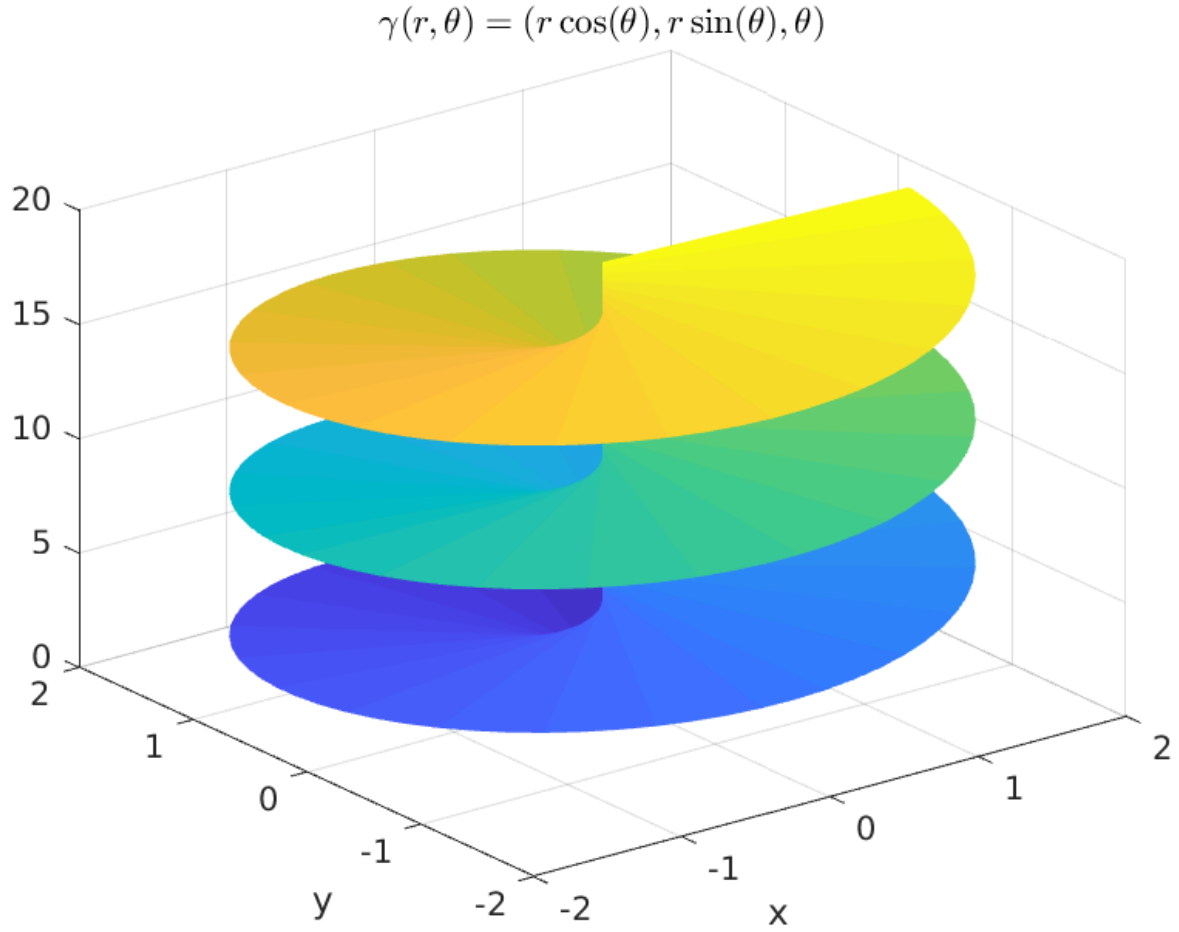
Per salvare la figura nel file 4plot.png e in 4plot.eps

```
print(1,'-dpng','4plot')
print(1,'-deps2c','4plot')
```

Disegno di una superficie attraverso le equazioni parametriche

Disegnare $\gamma(r,\theta) = (\underbrace{r\cos(\theta)}_{x}, \underbrace{r\sin(\theta)}_{y}, \underbrace{\theta}_{z})$

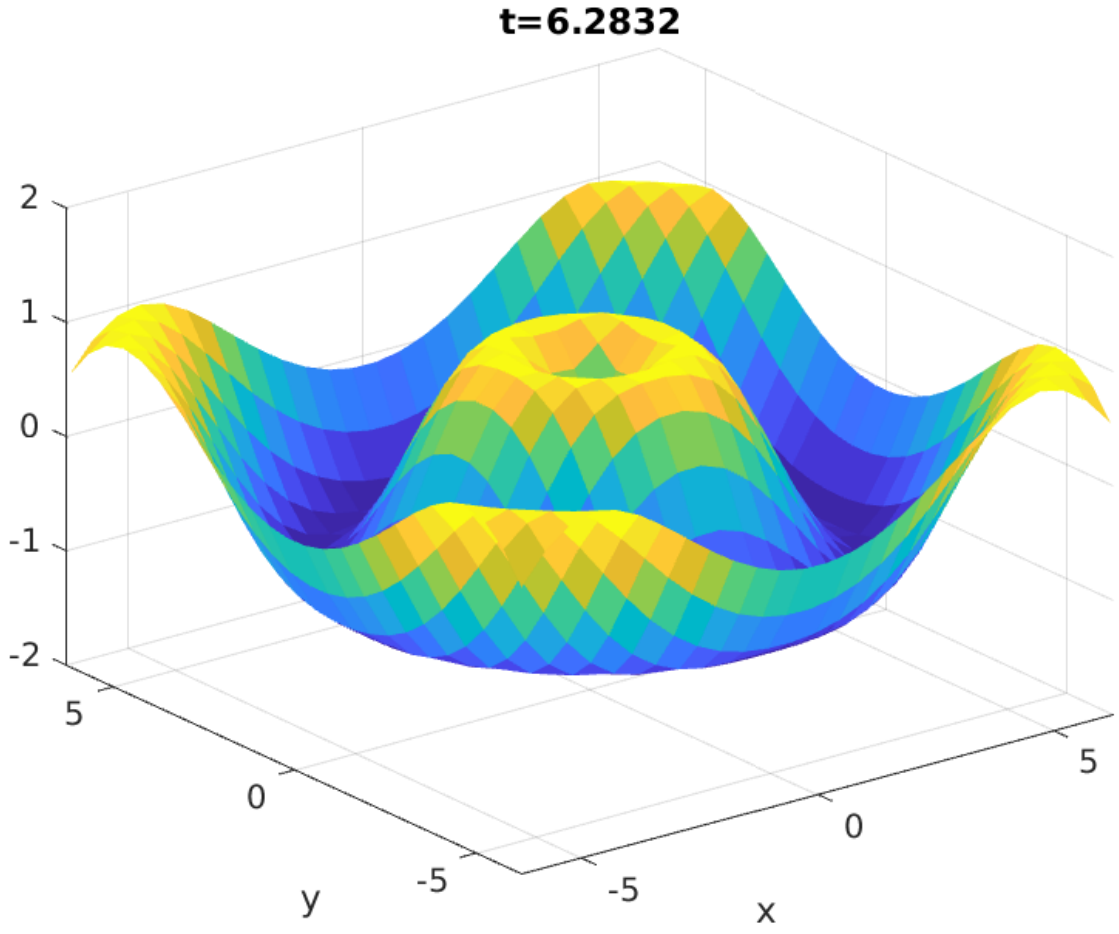
per $r \in [0,2]$ e $\theta \in [0, 6\pi]$.



Generazione di un movie

(solo in MATLAB)

Problema. Disegnare $f(x, y, t) = \sin(\sqrt{x^2 + y^2} - 2t)$ con $(x, y) \in [-2\pi, 2\pi]^2$ e $t \in [0, 2\pi]$.



Disegno di una superficie attraverso le equazioni parametriche

$$\gamma(r,\theta) = (\underset{x}{r\cos(\theta)}, \underset{y}{r\sin(\theta)}, \underset{z}{\theta})$$

per $r \in [0,2]$ e $\theta \in [0, 6\pi]$.

```
[r,theta]=meshgrid(0:.1:2,0:.1:6*pi);
x=r.*cos(theta);
y=r.*sin(theta);
z=theta;
figure;   % apre una nuova figura
          % senza cancellare quelle esistenti
s=surf(x,y,z) % disegna e genera l'oggetto 's'
s.EdgeColor='none'; % per togliere le righe nere della mesh
colormap('colorcube') % per cambiare la mappa di colori
colormap('default') % per tornare alla mappa di default
```

Generazione di un movie

(solo in MATLAB)

Problema. Disegnare $f(x, y, t) = \sin(\sqrt{x^2 + y^2} - 2t)$ con $(x, y) \in [-2\pi, 2\pi]^2$ e $t \in [0, 2\pi]$.

```
f=@(x,y,t)sin(sqrt(x.^2+y.^2)-2*t); % function handle
[x,y]=meshgrid(-2*pi:.5:2*pi); % genero la griglia
nframes=50; % numero di frame (istanti in tempo)
tt=linspace(0,2*pi,nframes); % discretizzo l'intervallo tem
Mv=struct('cdata',{},'colormap',{});
figure(1); clf
for n=1:nframes % ciclo sugli istanti temporali
  t=tt(n); z=f(x,y,t);
  s=surf(x,y,z); s.EdgeColor='none';
  axis([-2*pi 2*pi -2*pi 2*pi -2 2]);
  xlabel('x'); ylabel('y')
title(['t=',num2str(t)])
  Mv(n) = getframe; % salvo l'immagine in una struct
  pause(0.01) % fermo per 0.01 sec
end
movie(Mv,4); % riproduco il movie 4 volte
```