Lezione n° 3

Sommario:

- Sistemi con parametro
- Geometria dello spazio
  - Distanza tra due rette
  - Distanza tra punto e piano
  - Quadriche:
    - Ellissoide
    - Iperboloide ad una falda
    - Iperboloide a due falde
    - Cono circolare retto

Sistemi parametrici

Un sistema parametrico è un sistema in cui i coefficienti delle incognite e i termini noti non sono tutti numeri assegnati, ma alcuni sono, appunto, parametri.
Questi, poi, possono essere sostituiti con valori determinati per risolvere quindi il sistema stesso con coefficienti e termini numerici noti.
Noi considereremo soltanto sistemi parametrici in cui vi sia un solo parametro, denotato con *k*, eventualmente presente in più di un coefficiente e/o più di un termine noto.
In generale, ci attendiamo che attribuendo al parametro *k* un valore o un altro, il conseguente sistema numerico che si ottiene ammetta soluzioni con diversa struttura.
Il problema si divide allora in due aspetti: in primo luogo dovremo stabilire per quali valori attribuiti al parametro *k* il sistema possiede soluzioni e con quale molteplicità; quindi, nei casi di risolubilità, si dovrà procedere a calcolare effettivamente quali sono queste soluzioni.
Si intuisce dunque l'importanza di non confondere quale dei due aspetti si sta affrontando.

```
%Esercizio 1 – Risoluzione di un sistema lineare con parametro
```

Consideriamo il seguente sistema e calcoliamone le soluzioni in funzione del parametro k.

$$\begin{cases} 3x - y = 2 \\ x + 2y + 3z = k + 8 \\ (k-4)x + y = -1 \end{cases}$$

Per prima cosa, definiamo una variabile simbolica **k**, che rappresenta proprio il nostro parametro.

```
>> syms k;
```

Definiamo le matrici dei coefficienti e dei termini noti:

```
>> A = [3 -1 0;1 2 3;k-4 1 0]; B = [2;k+8;-1];
```

Calcoliamo il determinante della matrice dei coefficienti per valutare se il sistema ammette soluzioni.

```
>> a = det(A)

a =

3 - 3*k
```

Possiamo risolvere il sistema con il metodo di Cramer se e solo se il $\det(A) \neq 0$, cioè $k \neq 1$. Quindi:

```
>> X = A\B

X =

                      1/(k - 1)

            -(2*k - 5)/(k - 1)

 (k^2 + 11*k - 19)/(3*(k - 1))
```

Il sistema ammette infinite soluzioni dipendenti da `k`, il quale può assumere qualsiasi valore, tranne '1'. Se volessimo valutare la soluzione del sistema in corrispondenza di un determinato valore di `k`, ad esempio `2`, dovremmo procedere nel seguente modo.

```
>> Xk = subs(X,k,2)

Xk =

    1.0000

    1.0000

    2.3333
```

Il comando **`subs`** sostituisce in una espressione simbolica, nel nostro caso `X`, un vecchio parametro, `k`, con uno nuovo.

N.B.: sostituendo `k` con un valore numerico, la variabile su cui viene salvata la nuova soluzione non sarà più visualizzata come simbolica, ma come matrice.
Consultare l'`help` di Matlab per maggiori informazioni.

```
------------------------------clear--------------------------------
 ------------------------------clc---------------------------------
```

Calcolo della distanza di due rette

Dobbiamo rappresentare due rette nello spazio e calcolarne la distanza reciproca. Si ricorda il sistema che parametrizza una retta nello spazio:

$$\begin{cases} x_1 = x_0 + lt \\ y_1 = y_0 + mt \\ z_1 = z_0 + nt \end{cases}$$

dove la terna $(l, m, n) \neq (0,0,0)$.

Innanzitutto mi serve un vettore che rappresenta il dominio delle due rette:

```
>> t = linspace(-10,10,100); %Range di valori di dominio
```

Potevo anche fare a meno di scrivere "100" nel terzo campo perché il comando `linspace` di default genera 100 elementi equispaziati.

```
%Prima retta:          x1=t; y1=0; z1=0;
%(si ottiene per (x0,y0,z0)=(0,0,0) e (l,m,n)=(1,0,0)

>> x1 = t;

>> y1 = zeros(1,size(t,2));

>> z1 = zeros(1,size(t,2));
```

Invece di usare il comando `size(t,2)` posso usare **`length`** che mi restituisce la lunghezza maggiore di una matrice.

```
%Seconda retta:        x2=0; y2=t; z2=3/2;
%(si ottiene per (x0,y0,z0)=(0,0,3/2) e (l,m,n)=(0,1,0)

>> x2 = zeros(1,length(t));

>> y2 = t;

>> z2 = zeros(1,size(t,2))+3/2;
```

Per introdurre la grafica 3D utilizzo, tra i tanti, il comando **`plot3`**.

```
>> plot3(x1,y1,z1) %Disegno la prima retta
```

Posso abilitare una griglia per mettere in risalto l'allineamento delle curve.

```
>> grid on

>> hold on %abilito la "ritenuta" delle immagini
```

Anche il comando `plot3` ammette diverse opzioni, si consiglia di consultarne l'`help`.

```
>> plot3(x2,y2,z2,'r') %Disegno la seconda retta
```

Si dovrà tracciare una terza retta ortogonale alle precedenti; in questo caso è semplice perché la prima retta corrisponde all'asse *x* mentre la seconda è una traslazione dell'asse *y* a quota *z* pari a 3/2: la terza retta corrisponde quindi all'asse *z*.

```
%Retta ortogonale      x3=0; y3=0; z3=t;

>> x3 = zeros(1,size(t,2));

>> y3 = zeros(1,size(t,2));

>> z3 = t;

>> plot3(x3,y3,z3,'g')
```

A questo punto si vanno a confrontare le parametrizzazioni delle prime due rette con quella della terza. Ad esempio per la prima retta e la terza avrò:

(t1,0,0) = (0,0,t3) $\Rightarrow$ t1=0, t3=0 $\Rightarrow$ P1 = (0,0,0)

Per la seconda e la terza, invece:

(0,t2,3/2) = (0,0,t3) $\Rightarrow$ t2=0, t3=3/2 $\Rightarrow$ P2 = (0,0,3/2)

Ottengo così due punti

```
>> P1 = [0 0 0];

>> P2 = [0 0 3/2];
```

Marco le intersezioni trovate nel grafico utilizzando il comando **`scatter3`**, facendo attenzione ad inserire ognuna delle tre componenti del punto (x,y,z).

```
>> scatter3(0,0,0,'filled'); % l'opzione 'filled' riempie i marcatori

>> scatter3(0,0,3/2,'filled');
```

A questo punto devo calcolare la distanza fra i due punti, che coincide con la distanza fra le due rette. Si osserva che il primo punto corrisponde all'origine mentre il secondo ha le prime due componenti nulle, da cui si ricava che la distanza fra le due rette è esattamente pari a 3/2, ovvero pari al valore dell'unica componente non nulla del secondo punto.

In generale avrei dovuto procedere applicando la norma 2 sulla differenza dei due punti.

```
>> norm(P1-P2)

ans =

    1.5000
```

```
      ------------------------------clear---------------------------------
      ------------------------------clc---------------------------------
```

Calcolo della distanza fra punto e piano

Vogliamo ora calcolare la distanza fra un piano e un punto.

Ricordiamo che l'equazione generica di un piano nello spazio è:

$$\pi: ax + by + cz = 0$$

e che la distanza da un generico punto `P=(x0,y0,z0)` è data dalla seguente formula:

$$d(\pi, \mathrm{P}) = \frac{|ax_0 + by_0 + cz_0 + d|}{\sqrt{a^2 + b^2 + c^2}}$$

Consideriamo, quindi, il seguente piano:

$$3x + 4y - 2z + 1 = 0$$

ed il punto $\mathrm{P} = [1,0,20]$.

Disegno il piano: come prima cosa, genero un dominio di appartenenza utilizzando due vettori.

```
>> x = linspace(-10,10,100);
>> y = linspace(-10,10,100);
```

Per poter costruire il grafico di un piano su un dominio rettangolare $[a, b] \times [c, d]$, è necessario in primo luogo costruire una griglia di punti su cui andare a valutare la funzione sulle variabili x e y.

La tabulazione di un dominio $[a, b] \times [c, d]$ è possibile averla tramite la funzione **`meshgrid`**: essa permette di definire in due matrici le ascisse e le ordinate dei punti in cui deve essere valutata la funzione di due variabili
La `meshgrid` trasforma due vettori `x` e `y` in due matrici (in questo caso `X` e `Y`) in cui gli elementi `X(i,j)` e `Y(i,j)` sono l'ascissa e l'ordinata del punto (i,j) della griglia. In altre parole:

```
X = [x; x; x; …; x]

Y = [y y y y … y]
```

```
>> [X Y] = meshgrid(x,y); %Tabulazione di dominio [a,b]x[c,d] su griglia
                          %regolare rettangolare a maglia regolare
                          %rettangolare
```

Graficando:

```
>> plot3(X,Y,(3*X+4*Y+1)/2,'r')
```

È possibile vedere la superficie da un'altra angolazione si può usare la funzione utilizzando il comando **`view(AZ,EL)`**, dove `AZ` è l'azimut (ovvero l'angolo tra l'asse delle y e il punto di vista misurato sul piano coordinato `x-y`) e `EL` l'elevazione voluta (ovvero l'angolo tra il piano coordinato `x-y` e il punto di vista).

```
>> view(-79,-72);
```

```
>> grid on
```

Prima di procedere a disegnare il punto, devo bloccare l'immagine.

```
>> hold on
```

```
>> scatter3(1,0,20,'filled'); % Disegno il punto P
```

Scrivo l'equazione parametrica della retta passante per il punto $P = [1,0,20]$ e perpendicolare al piano, unica del fascio di rette di direzione (3,4,-2):

$$\begin{cases} x = 1 + 3t \\ y = 4t \\ z = 20 - 2t \end{cases}$$

da cui, ricavando `t`, deriva l'identità:

$$\frac{x - 1}{3} = \frac{y}{4} = \frac{z - 20}{-2}$$

Ora "scrivo" in Matlab le due variabili `y` e `z` in funzione di `x` (parametro indipendente):

```
>> x = linspace(-20,20);
```

```
>> y = 4*(x-1)/3; z = -2*(x-1)/3+20;
```

Disegnamo la retta!

```
>> plot3(x,y,z,'b');
```

```
>> view(-104,-4);
```

Trovo il punto Q intersezione tra il piano e la retta mettendo a sistema l'equazione del piano con due delle equazioni della retta. Cioè risolvo il sistema:

$$\begin{cases} 3x + 4y - 2z + 1 = 0 \\ \dfrac{x-1}{3} = \dfrac{y}{4} \\ \dfrac{x-1}{3} = \dfrac{z-20}{-2} \end{cases}$$

```
>> syms x y z
```

Il comando **`solve`** permette di risolvere un sistema di equazioni algebriche in variabile simbolica:

```
>> [xq yq zq] = solve(3*x+4*y-2*z+1,(x-1)/3-y/4,(x-1)/3+(z-20)/2,x,y,z);
```

Salvo poi la soluzione ottenuta in Q:

```
>> Q = [xq yq zq]

Q =

[ 137/29, 144/29, 508/29]
```

NOTA BENE:
Se non volessi una soluzione "simbolica" ma "reale", dovrei convertire il numero. Il comando **`double`** converte il numero nel formato comunemente usato da Matlab, cioè in un numero floating point, a "virgola mobile", composto da 64 bit divisi in Segno, Esponente e Mantissa.

```
>> Q1 = double([xq yq zq])

Q1 =

    4.7241    4.9655   17.5172
```

```
%%%%%%%%%%%%% Metodo alternativo senza l'uso del comando syms %%%%%%%%%%%%%
```

È possibile riscrivere il sistema lineare con i termini noti a secondo membro:

$$\begin{cases} 3x + 4y - 2z = -1 \\ \dfrac{x}{3} - \dfrac{y}{4} = \dfrac{1}{3} \\ \dfrac{x}{3} + \dfrac{z}{2} = 10 + \dfrac{1}{3} \end{cases}$$

Procediamo con la matrice dei coefficienti e la matrice dei termini noti:

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

>> B = [-1;1/3;10+1/3];
```

Risolviamo il sistema ottenuto:

```
>> Q = [A\B]'

Q =

    4.7241    4.9655   17.5172

                        %%%%%%%%%%%%% %%%%%%%%%%%%%
```

Una volta ricavata la soluzione, ovvero il punto di intersezione tra il piano e la retta perpendicolare ad esso e passante per P, procediamo con l'ultima fase dell'esercizio.

```
>> scatter3(Q(1),Q(2),Q(3),'filled'); %Marco la soluzione del sistema
```

Rimane da calcolare la distanza fra il punto `P` iniziale ed il punto `Q`:

```
>> P = [1 0 20];

>> d1 = norm(P-Q)

d1 =

    6.6850
```

Possiamo verificare quanto trovato utilizzando la formula per il calcolo della distanza fra un piano ed un punto.

```
>> d2 = abs(3*P(1,1)+4*P(1,2)-2*P(1,3)+1)/norm([3,4,-2])

d2 =

    6.6850
```

Il commando **`abs`** restituisce in uscita il valore assoluto del termine tra parentesi.
Come si evince, alcune strade sono più brevi di altre.

```
      ------------------------------clear---------------------------------
      ------------------------------clc---------------------------------
```

```
%Esercizio 3 - Distanza fra punto e piano – METODO ALTERNATIVO
```

Vediamo un ulteriore modo per calcolare la distanza tra il piano di equazione

$$3x + 4y - 2z + 1 = 0$$

ed il punto $P = [1,0,20]$.

```
%Soluzione

>> t = linspace(-10,10,100); %Range di dominio

>> x = t;

>> y = t;

>> [X Y] = meshgrid(x,y);

>> plot3(X,Y,(3*X+4*Y+1)/2,'r') %Disegno il piano

>> view(-79,-72); %visione di superficie personalizzata

>> grid on

>> hold on

>> scatter3(1,0,20,'filled'); %Marco il punto P

%Scrivo l'equazione della retta passante per il punto e perpendicolare al
%piano: x=1+3t, y=4t, z=20-2t

>> x = 1+3*t;

>> y = 4*t;

>> z = 20-2*t;

>> plot3(x,y,z,'b') %Disegno la retta

>> view(-104,-4);

%Risoluzione del problema attraverso l'utilizzo del sistema lineare:

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

>> B = [-1;1/3;10+1/3];

>> Q = [A\B]'; %Coordinate della soluzione

>> scatter3(Q(1),Q(2),Q(3),'filled');

%Rimane da calcolare la distanza fra il punto P iniziale ed il punto Q

>> P = [1 0 20];

>> d1 = norm(P-Q);
```

```
%Possiamo verificare quanto trovato utilizzando la formula per il calcolo
%della distanza fra un piano ed un punto
%                    d=|a*Px+b*Py+c*Pz+d|/sqrt(a^2+b^2+c^2)
%dove (Px, Py, Pz) sono le coordinate del punto P e (a, b ,c) sono i
%coefficienti dell'equazione del piano.
```

Abbiamo visto quindi che possiamo ricavare l'analogo risultato scrivendo:

```
>> d2 = abs(3*P(1,1)+4*P(1,2)-2*P(1,3)+1)/norm([3,4,-2])

d2 =

    6.6850
```

```
% --------- → Metodo alternativo con l'uso del syms
```

Dichiaro la `pi` che conterrà l'equazione del piano $3x + 4y - 2z + 1 = 0$ espressa in `xp`, `yp`, `zp`:

```
>> syms pi xp yp zp %ATTENZIONE!!! pi è già il valore P greco (π): ne
                    %sovrascrivo quindi il valore

>> pi = 3*xp+4*yp-2*zp+1 %Equazione del piano

pi =

3*xp + 4*yp - 2*zp + 1
```

Dichiaro `t` che utilizzo nella parametrizzazione della retta e che comparirà in `xr`, `yr`, `zr`

```
>> syms t xr yr zr

>> xr = 1+3*t;

>> yr = 4*t;

>> zr = 20-2*t;
```

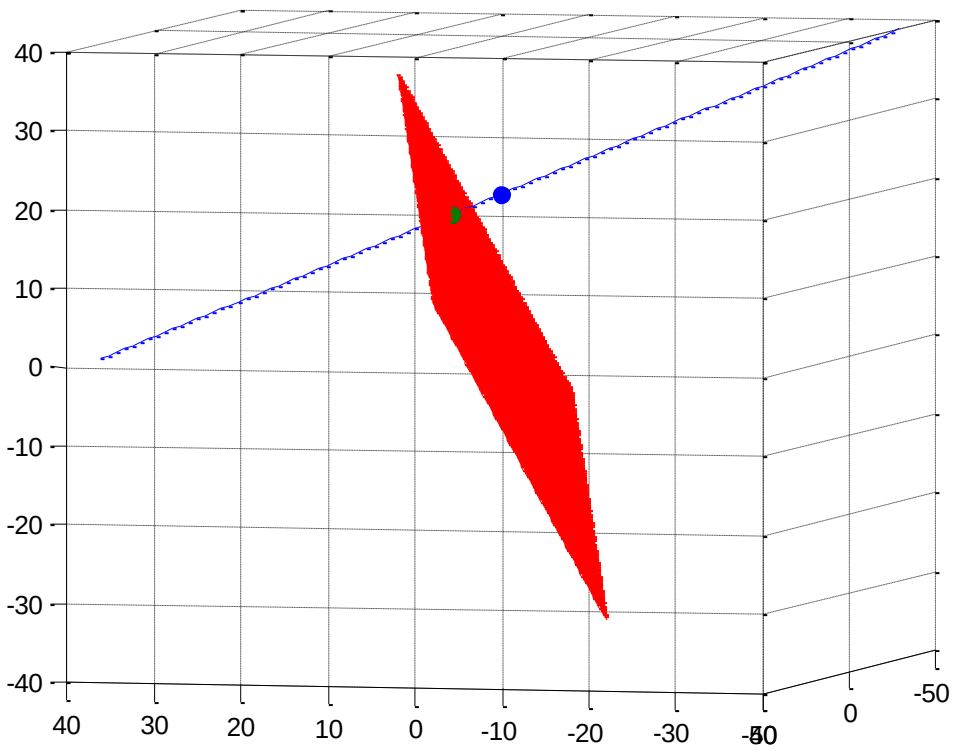
Sostituisco nell'equazione del piano la parametrizzazione della retta espressa in funzione di `t`:

```
>> pi_t = subs(pi,{xp,yp,zp},{xr,yr,zr})

pi_t =

29*t - 36
```

```
>> t = solve(pi_t,t) %Risoluzione del sistema

t =

36/29

>> double(t) %Conversione in floating point

ans =

    1.2414
```

Infine, una volta trovato il valore di `t`, vado a sostituirlo nella parametrizzazione della retta per ottenere le coordinate del punto `Q` di intersezione fra retta e piano (`xq`, `yq`, `zq`):

```
>> xq = subs(xr,t); yq = subs(yr,t); zq = subs(zr,t);

>> Q = double([xq yq zq])

Q =

    4.7241    4.9655   17.5172

>> scatter3(xq,yq,zq,'filled') %Marco il punto Q
```



```
%Rimane da calcolare la distanza fra il punto P iniziale ed il punto Q

>> d1 = norm(P-Q)

d1 =

    6.6850

%Possiamo verificare quanto trovato utilizzando la formula per il calcolo
%della distanza fra un piano ed un punto
%                    d=|a*Px+b*Py+c*Pz+d|/sqrt(a^2+b^2+c^2)
%dove (Px, Py, Pz) sono le coordinate del punto P e (a, b ,c) sono i
%coefficienti dell'equazione del piano.

>> d2 = abs(3*P(1,1)+4*P(1,2)-2*P(1,3)+1)/norm([3,4,-2]);


      ------------------------------clear---------------------------------
      ------------------------------clc-----------------------------------
```

Ellissoide

Vogliamo ora disegnare un ellissoide nello Spazio.

L'equazione cartesiana dell'ellissoide è:

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$$

Poniamo

```
>> a = 3; b = 2;
```

La variabile `c` è tralasciata in quanto posta uguale a 1 nella risoluzione.

Creo il dominio della parte superiore dell'ellissoide:

```
>> r = linspace(0,1,30);
```

Genero il dominio dell'angolo θ:

```
>> theta = linspace(0,2*pi,30);
```

Genero la tabulazione di un dominio di superficie - $[a,b]\times[c,d]$ - utilizzando `meshgrid`:

```
>> [r,theta] = meshgrid(r,theta);
```

Nella parametrizzazione `r` rappresenta la coordinata variabile delle ordinate di un arco di circonferenza: sapendo che la circonferenza ha raggio unitario, possiamo ricavare il valore delle ascisse con il teorema di Pitagora. La funzione coseno parametrizza la prima rotazione attorno a `z`.

```
>> x = a*cos(theta).*sqrt(1-r.^2);
```

Analogamente per `y`, la funzione seno parametrizza la seconda coordinata di rotazione:

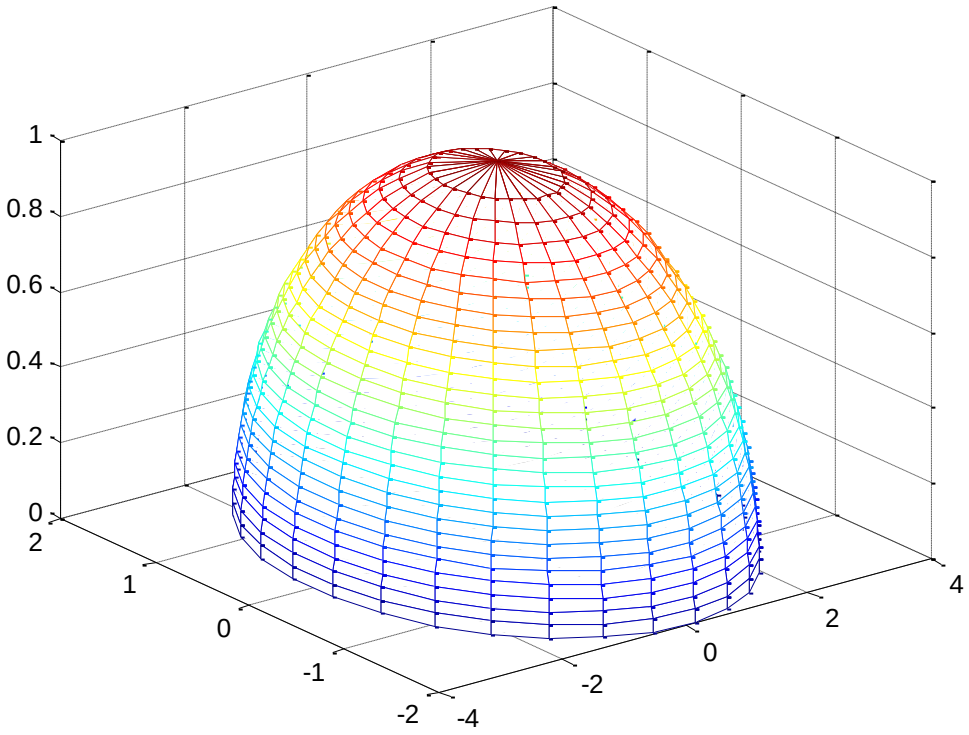
```
>> y = b*sin(theta).*sqrt(1-r.^2);
```

Come detto prima, l'asse z è definito dal vettore `r`.

```
>> z = r;
```

Infine, in grafica 3D, è sufficiente usare il comando **`mesh`** per visualizzare la superficie:

```
>> mesh(x,y,z);
```



La rappresentazione grafica di `mesh` è l'analogo in più dimensioni del comando plot. In questo caso si raccordano i punti a quattro a quattro con pezze piane (parti di piano che passano per quattro punti).

Anche per la funzione grafica mesh devo abilitare il blocco delle immagini:

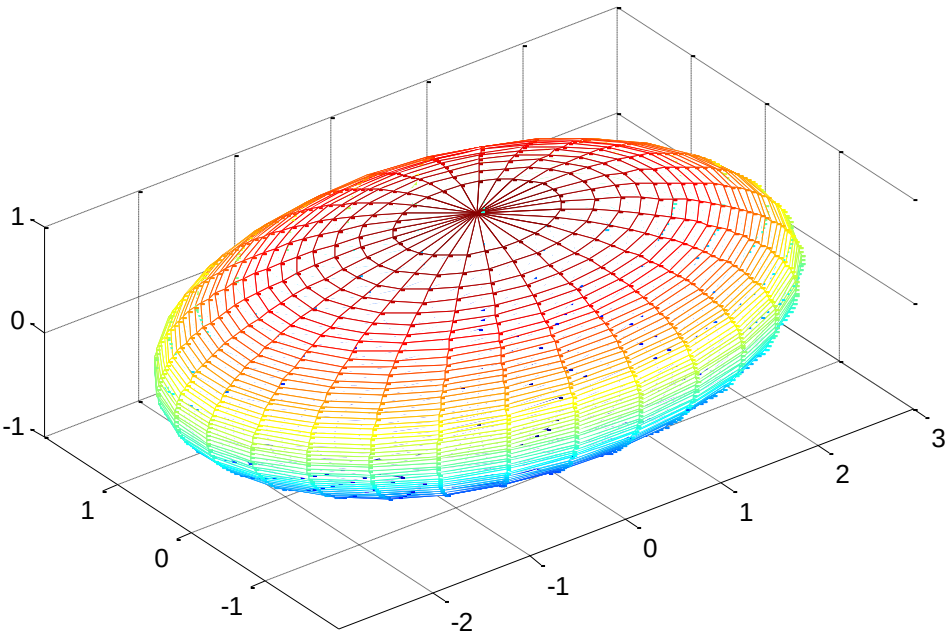
```
>> hold on
```

Si procede col disegnare la parte inferiore dell'ellissoide:

```
>> mesh(x,y,-z);
```

Alcuni comandi permettono di abbellire il grafico, come `title, xlabel, ylabel, zlabel, text,`… Posso decidere di imporre i tre assi cartesiani della stessa dimensione:

```
>> axis equal
```




```
------------------------------clear---------------------------------
------------------------------clc-----------------------------------
```

```
%METODO ALTERNATIVO PER LA RAPPRESENTAZIONE DELL'ELLISSOIDE
```

Ricordiamo che l'ellissoide ha

equazione parametrica: $\begin{cases} x = a \cdot \cos(\theta) \cdot \sin(\varphi) \\ y = b \cdot \sin(\theta) \cdot \sin(\varphi) \\ z = c \cdot \cos(\varphi) \end{cases} \qquad (\theta, \varphi) \in [0,2\pi] \times [0,\pi]$

Quindi definiamo i range di dominio di θ e φ:

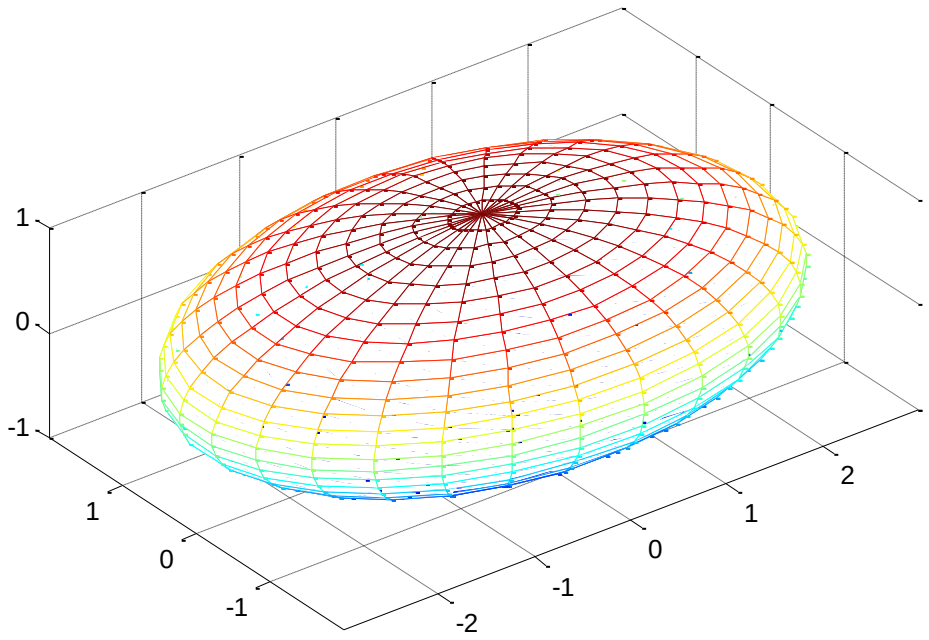
```
>> theta = linspace(0,2*pi,30);
>> phi = linspace(0, pi,30);
>> [theta,phi] = meshgrid(theta,phi);
```

Parametrizziamo:

```
>> x = 3*cos(theta).*sin(phi);
>> y = 2*sin(theta).*sin(phi);
>> z = cos(phi);
```

Grafichiamo:

```
>> mesh(x,y,z);
>> axis equal
```




```
--------------------------------clear---------------------------------
--------------------------------clc-----------------------------------
```

Iperboloide ad una falda

L'equazione cartesiana dell'iperboloide ad una falda è:

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=1$$

Tuttavia noi lavoreremo con la sua equazione parametrica:

$$\begin{cases} x = a\cdot\cosh(u)\cdot\cos(\theta) \\ y = b\cdot\cosh(u)\cdot\sin(\theta) \\ z = c\cdot\sinh(u) \end{cases} \qquad (u,\theta)\in\mathbb{R}\times[0,2\pi]$$

Pongo i coefficienti `a`, `b` e `c` pari a 1:

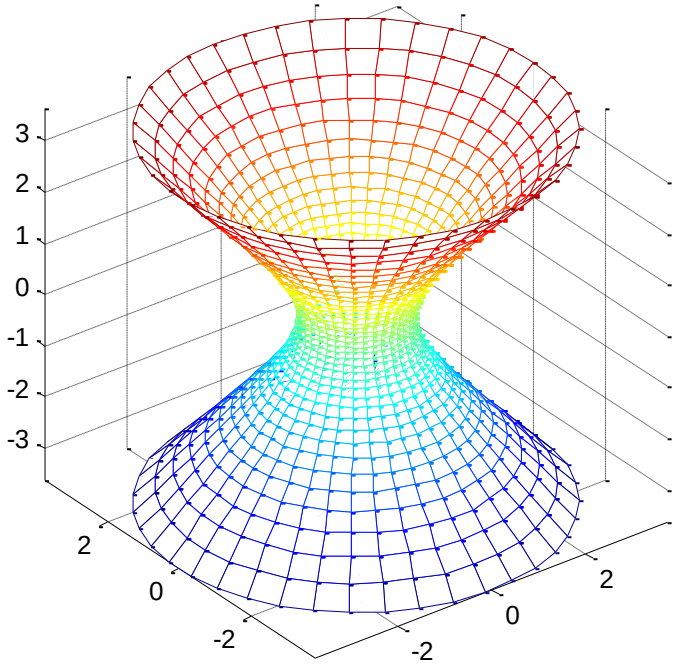
```
>> a=1; b=1; c=1;
```

Non potendolo far variare in tutto l'asse dei reali, pongo `u` nell'intervallo `[-2;2]` – e (per semplicità) chiamo `v` la variabile 'theta' – :

```
>> u = linspace(-2,2,40);
>> v = linspace(0,2*pi,40);
>> [u,v] = meshgrid(u,v);
```

Adesso pongo x, y, z secondo l'equazione parametrica:

```
>> x = a*cosh(u).*cos(v);
>> y = b*cosh(u).*sin(v);
>> z = c*sinh(u);
>> mesh(x,y,z)
>> axis equal
```




```
------------------------------clear-------------------------------
------------------------------clc---------------------------------
```

Iperboloide a due falde

L'equazione cartesiana dell'iperboloide a due falde è:

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=-1$$

Equazione parametrica per la falda nel semispazio $x > 0$:

$$\begin{cases} x = a \cdot \sinh(u) \cdot \cos(v) \\ y = b \cdot \sinh(u) \cdot \sin(v) \\ z = c \cdot \cosh(u) \end{cases} \qquad (u, v) \in \mathbb{R} \times [0,2\pi]$$

Pongo i coefficienti:

```
>> a = 1; b = 1; c = 1;
```

Pongo il dominio di `u` e `v`:

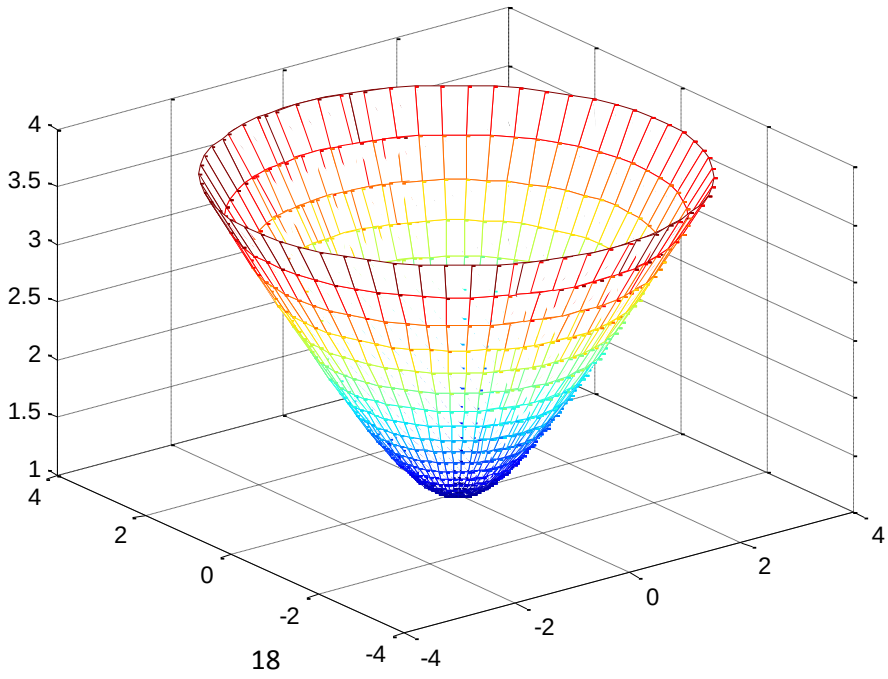
```
>> u = linspace(-2,2,40);
>> v = linspace(0,2*pi,40);
>> [u,v] = meshgrid(u,v);
```

Adesso ricavo le tre coordinate:

```
>> x = a*sinh(u).*cos(v);
>> y = b*sinh(u).*sin(v);
>> z = c*cosh(u);
```

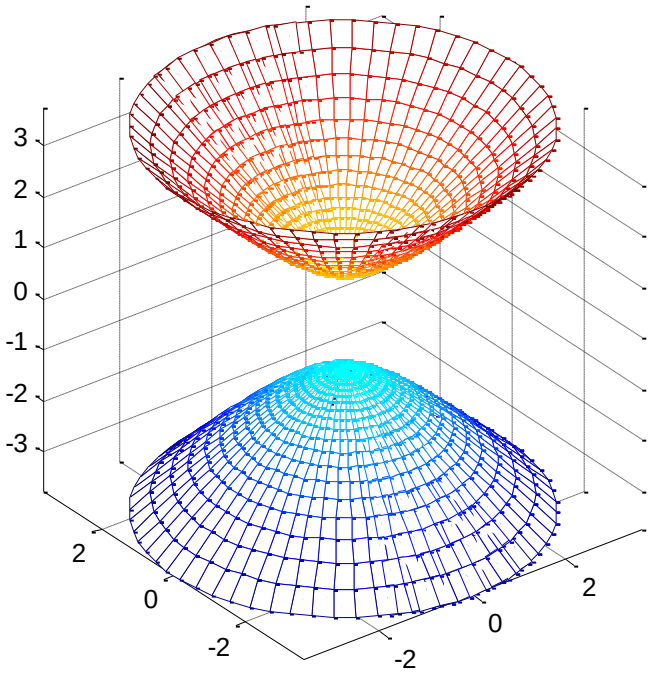
Ora disegno la parte superiore della superficie:

```
>> mesh(x,y,z);
```



Passo a disegnare quella inferiore:

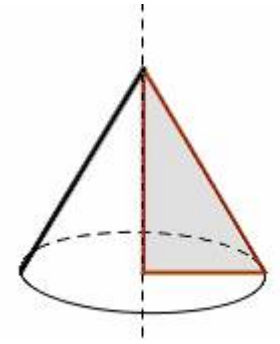
```
>> hold on

>> mesh(x,y,-z)

>> axis equal
```




```
------------------------------clear---------------------------------
------------------------------clc-----------------------------------
```

Cono circolare retto

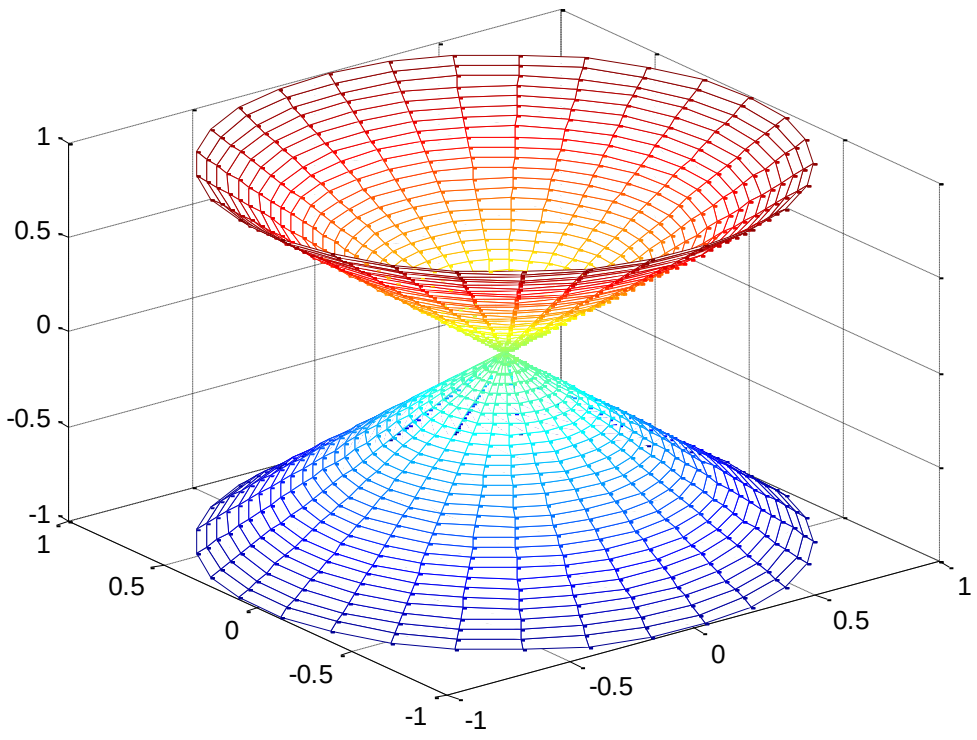
Un cono circola retto è un cono la cui altezza cade nel centro del cerchio di base.



```
>> r = linspace(0,1,30); % Range di dominio dell'altezza
>> theta = linspace(0,2*pi,30); %Range di dominio dell'angolo θ, da 0 a 2π
>> [r,theta] = meshgrid(r,theta);
```

Genere le coordinate x e y in funzione dell'altezza z(= r) e θ

```
>> x = r.*cos(theta);
>> y = r.*sin(theta);
>> z = r;
>> mesh(x,y,z); %Disegno la parte superiore del cono
>> hold on
>> mesh(x,y,-z); %Disegno la parte inferiore speculare del cono
```



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Esercizio %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Calcolare la distanza fra il piano 5x+3y-2z+1=0 ed il punto P=[-2,20,0].

%Graficare inoltre il piano, il punto e la retta perpendicolare al piano %passante per il punto.

Soluzione: d=8.2733