Esercizio 1

Iniziamo a svolgere questo mastodontico esercizio. La prima cosa che dobbiamo fare è adattare la nostra function che implementa il metodo di Eulero al caso vettoriale, e lo facciamo con la seguente funzione `eulerovec`, che restituisce il vettore dei tempi x e la matrice soluzione u:

```
function [x,u]=eulerovec(odefun,slot,init,h)
N=floor((slot(2)-slot(1))/h);
x=zeros(1,N+1);
x(1)=slot(1);
for i=2:N+1
    x(i)=x(i-1)+h;
end
u=zeros(length(init),N+1);
u(:,1)=init;
for i=2:N+1
    ff=feval(odefun, x(i-1),u(:,i-1));
    u(:,i)=u(:,i-1)+h*ff;
end
end
```

Dette ora y_i con i che varia tra 1 e 6 le variabili indicate dal testo, dobbiamo scrivere un sistema di equazioni differenziali del primo ordine da dare in pasto a `eulerovec` che sia equivalente all'equazione del testo. Naturalmente, dalla definizione del testo segue:

$$y_1' = y_4$$

$$y_2' = y_5$$

$$y_3' = y_6$$

Riscriviamo l'equazione del testo (che notiamo fornire esattamente le tre derivate che ci mancano) in funzione dei dati forniti dal testo, indicando con $\mathbf{y}$ il vettore (y_1, y_2, y_3), e con con $\mathbf{w}$ il vettore (y_4, y_5, y_6):

$$\mathbf{w}' = \mathbf{F} - \mathbf{y}\frac{\mathbf{w}^T\mathbf{w} + \mathbf{y}^T\mathbf{F}}{\mathbf{y}^T\mathbf{y}}$$

ed ora possiamo finalmente scrivere una funzione che implementi la suddetta equazione differenziale.

```
function yp=sfera(x,y)
%y vettore colonna
g=9.81;
F=[0;0;-g];
yp=[y(4);y(5);y(6);F-2*y(1:3)*((2*(y(4:6)'*y(4:6))+2*(y(1:3)'*F))./norm(2*y(1:3))^2);];
yp=yp;
end
```

Con gioia risolviamo l'equazione usando i quattro metodi. Iniziamo con Eulero:

```
function e=motosferaeulero
y0=[0;1;0;0.8;0;1.2];
tspan=[0,10];
[x,y]=eulerovec(@(x,y)sfera(x,y),tspan,y0,0.0025);
[t,u]=eulerovec(@(x,y)sfera(x,y),tspan,y0,0.00025);
plot3(y(1,:),y(2,:),y(3,:),u(1,:),u(2,:),u(3,:),'--')
title('Eulero')
e(1)=abs(y(1,length(x))^2+y(2,length(x))^2+y(3,length(x))^2-1);
e(2)=abs(u(1,length(t))^2+u(2,length(t))^2+u(3,length(t))^2-1);
end
```

Questa funzione restituisce 0.4482 e 0.0435 : dunque che $h = 0.0025$ è del tutto insufficiente per ottenere una solzuione soddisfacente, mentre il secondo valore funziona già decisamente meglio.

```
function e=motosferaRK
y0=[0;1;0;0.8;0;1.2];
tspan=[0,10];
[x,y]=RK4(@(x,y)sfera(x,y),tspan,y0,0.005);
[t,u]=RK4(@(x,y)sfera(x,y),tspan,y0,0.0005);
plot3(y(1,:),y(2,:),y(3,:),u(1,:),u(2,:),u(3,:),'--')
title('Runge-Kutta')
e(1)=abs(y(1,length(x))^2+y(2,length(x))^2+y(3,length(x))^2-1);
e(2)=abs(u(1,length(t))^2+u(2,length(t))^2+u(3,length(t))^2-1);
end
```

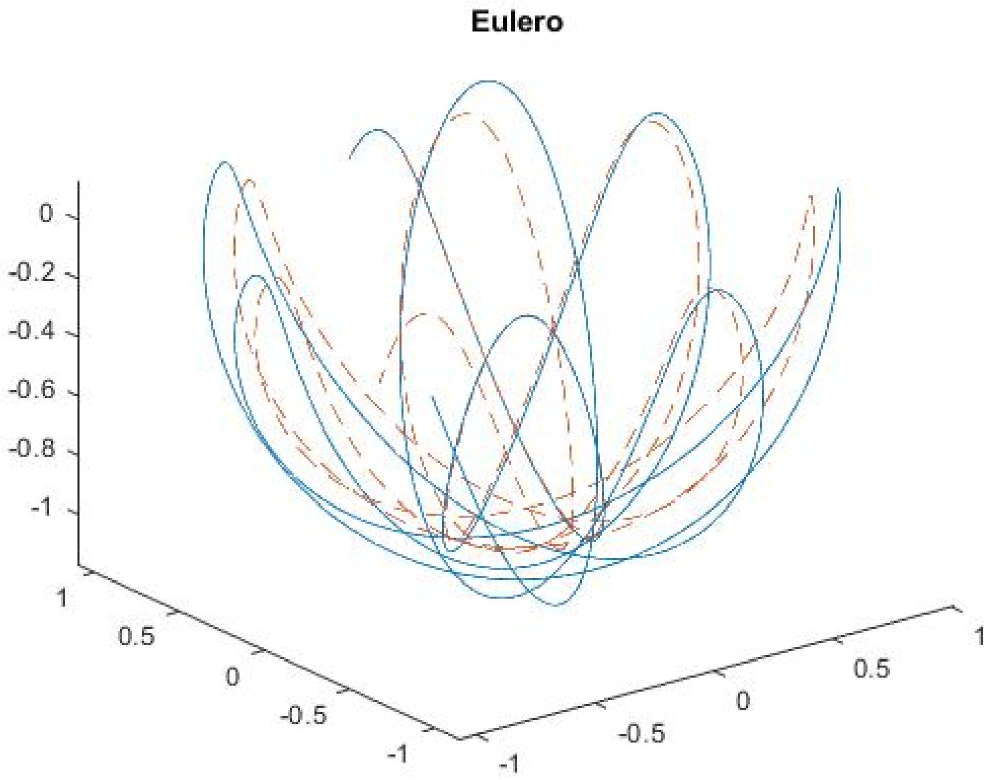
Questa restituisce errori dell'ordine di 10^{-5}, e infatti le curve sono indistinguibili ad occhio nudo..

```
function e=motosferaode23
y0=[0;1;0;0.8;0;1.2];
tspan=[0,10];
options=odeset('RelTol',1e-10);
[x,y]=ode23(@(x,y)sfera(x,y),tspan,y0);
[t,u]=ode23(@(x,y)sfera(x,y),tspan,y0,options);
plot3(y(:,1),y(:,2),y(:,3),u(:,1),u(:,2),u(:,3),'--')
title('ode23')
e(1)=abs(y(length(x),1)^2+y(length(x),2)^2+y(length(x),3)^2-1);
e(2)=abs(u(length(t),1)^2+u(length(t),2)^2+u(length(t),3)^2-1);
end
```

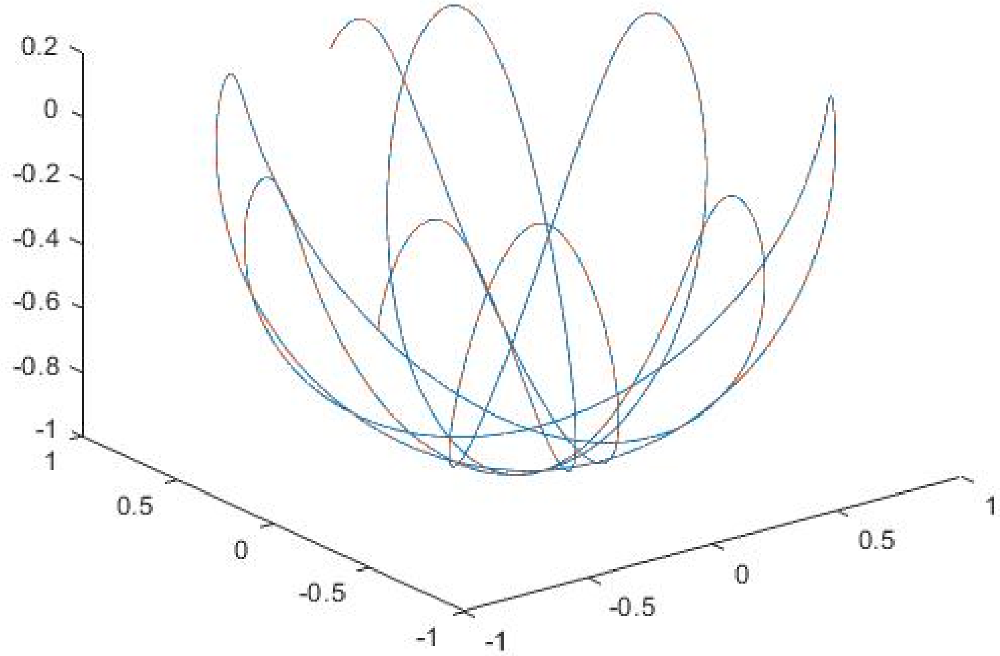
Anche questa è molto precisa, in particolare nel secondo caso (l'errore è minore di 10^{-4}).

```
function e=motosferaode45
y0=[0;1;0;0.8;0;1.2];
tspan=[0,50];
options=odeset('RelTol',1e-10);
[x,y]=ode45(@(x,y)sfera(x,y),tspan,y0);
[t,u]=ode45(@(x,y)sfera(x,y),tspan,y0,options);
plot3(y(:,1),y(:,2),y(:,3),u(:,1),u(:,2),u(:,3),'--')
title('ode45')
e(1)=abs(y(length(x),1)^2+y(length(x),2)^2+y(length(x),3)^2-1);
e(2)=abs(u(length(t),1)^2+u(length(t),2)^2+u(length(t),3)^2-1);
end
```

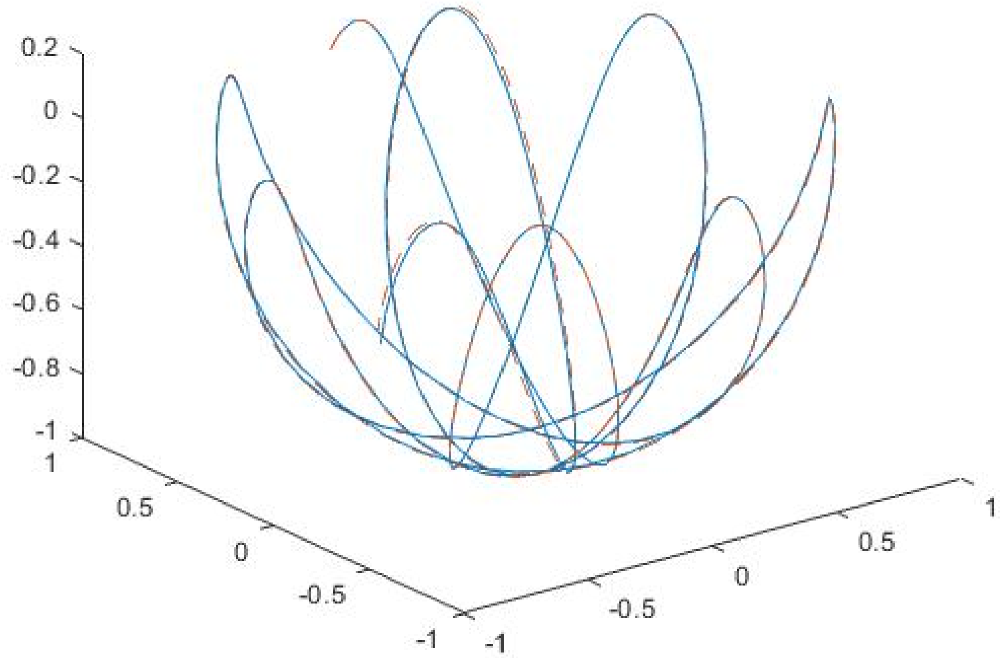
Qua invece, mentre il passo più piccolo ci fornisce una soluzione accettabile, quello più grande devia in modo devastante dalla soluzione cercata: aumentando l'intervallo di integrazione (ad esempio a 50) la superfice tangente alla soluzione non appare più per nulla sferica.

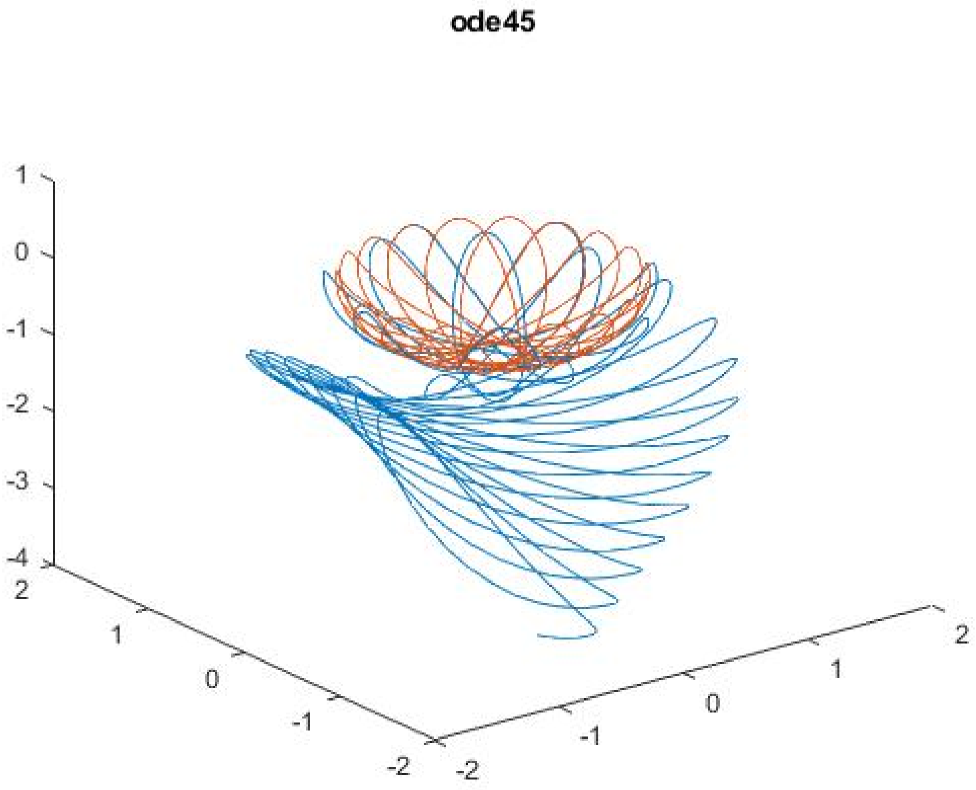

Eulero
0
-0.2
-0.4
-0.6
-0.8
-1
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

Runge-Kutta



ode23



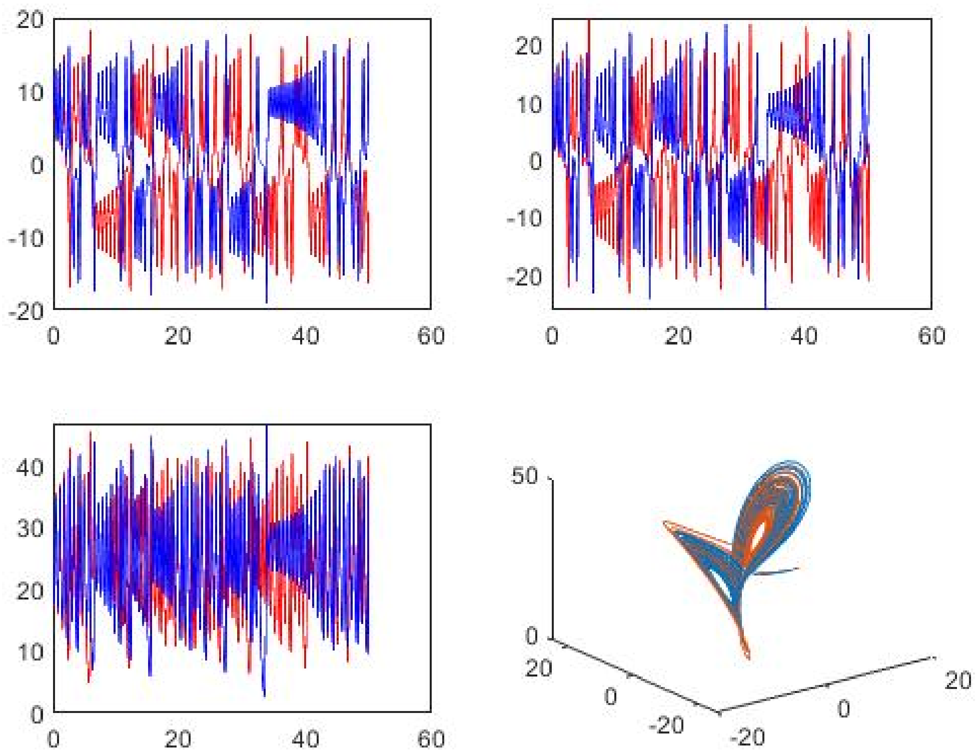

ode45
1
0
-1
-2
-3
-4
2
1
0
-1
-2
-2
-1
0
1
2

Esercizio 2

La function risolve l'equazione differenziale dell'attrattore di Lorenz con i parametri del testo e ne plotta i grafici richiesti:

```
function []=lorenz(tmax)
sigma=10;
r=28;
b=8/3;
f=@(x,y) [sigma*(y(2)-y(1));r*y(1)-y(2)-y(1)*y(3); y(1)*y(2)-b*y(3)];
[x1,y1]=ode45(f,[0,tmax],[10;0;20]);
[x2,y2]=ode45(f,[0,tmax],[11;0;20]);

subplot(2,2,1)
plot (x1,y1(:,1),'r',x2,y2(:,1),'b');

subplot(2,2,2)
plot (x1,y1(:,2),'r',x2,y2(:,2),'b');

subplot(2,2,3)
plot (x1,y1(:,3),'r',x2,y2(:,3),'b');

subplot(2,2,4)
plot3(y1(:,1),y1(:,2),y1(:,3),y2(:,1),y2(:,2),y2(:,3));
end
```

Di seguito i grafici richiesti (con $tmax = 50$), da cui si intuisce bene quanto è caotica la soluzione ottenuta.



Esercizio 3

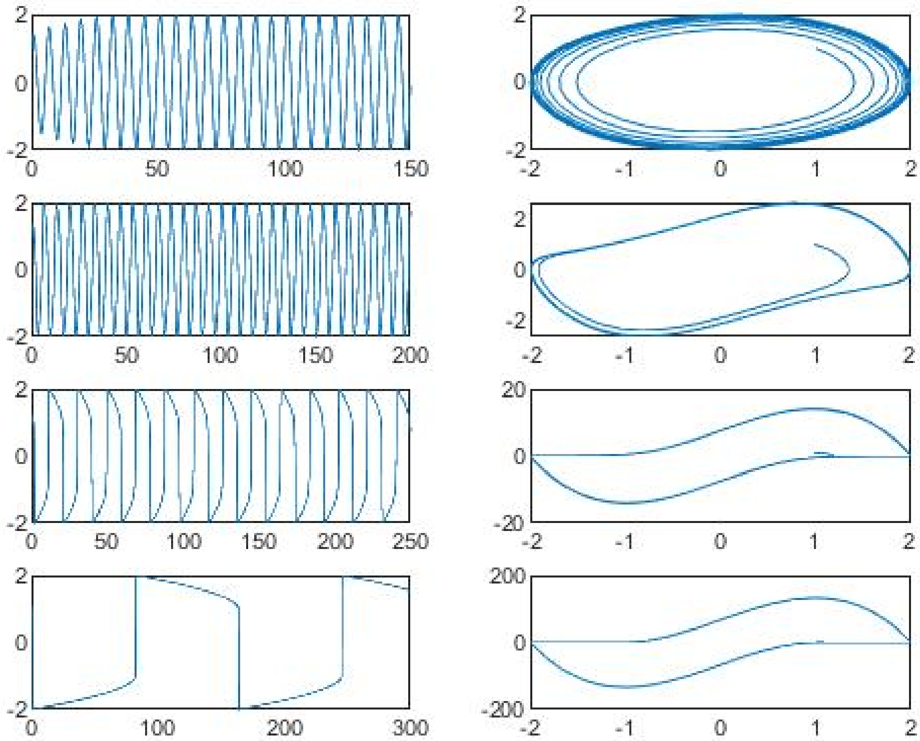
Scriviamo le funzioni che risolvono l'equazione di Van Der Pol tramite `ode45` e tramite `ode15s`. Adottiamo una dilatazione dinamica lineare dei punti della griglia, per ottenere un grafico che gestisca decentemente il fatto che anche la soluzione cresce abbastanza (per far variare μ lo abbiamo messo come variabile vettoriale):

```
function [v]=vanderpol1
%restituisce il numero di punti della griglia al variare di mu
mu=[0.1,1,10,100];
v=zeros(1,4);
for i=1:4
    f=@(t,y) [y(2);mu(i)*(1-y(1)^2)*y(2)-y(1)];
    [t,y]=ode45(f,[0 100+50*i],[1;1]);
    subplot(4,2,2*i-1)
    plot(t,y(:,1));
    subplot(4,2,2*i)
    plot(y(:,1),y(:,2));
    v(i)=length(t);
end
end
```

```
function [v]=vanderpol2
%restituisce il numero di punti della griglia al variare di mu
mu=[0.1,1,10,100];
v=zeros(1,4);
for i=1:4
    f=@(t,y) [y(2);mu(i)*(1-y(1)^2)*y(2)-y(1)];
    [t,y]=ode15s(f,[0 100+50*i],[1;1]);
    subplot(4,2,2*i-1)
    plot(t,y(:,1));
    subplot(4,2,2*i)
    plot(y(:,1),y(:,2));
    v(i)=length(t);
end
end
```

Benchè i grafici siano uguali ad occhio nudo, il numero di nodi ottenuti con `ode45` (761, 2129, 6909, 67585 per i vari μ) risulta molto maggiore rispetto a quello ottenuto con `ode15s`(886, 1935, 2577, 585), in particolare per μ elevato.
Il metodo di Eulero appare invece pessimo, con le soluzioni che oscillano in maniera spropositata e smettono di essere disegnati ma matlab già per $h < 1$.

```
function []=vanderpoleulero(h)
%h=numero di nodi della griglia
mu=[0.1,1,10,100];
for i=1:4
    y(1,1)=1; y(2,1)=1;
    for j=2:h+1
        y(1,j)=y(2,j-1)*100/(h+1)+y(1,j-1);
        y(2,j)=y(2,j-1)+100/(h+1)*mu(i)*(1-y(1,j-1)^2)*y(2,j-1)-y(1,j-1);
    end
    t=linspace(0,100,h+1);
    subplot(4,2,2*i-1), plot(t,y(1,:));
    subplot(4,2,2*i), plot(y(1,:),y(2,:));
end
end
```


2
0
-2
0
50
100
150
2
0
-2
0
50
100
150
200
2
0
-2
0
50
100
150
200
250
2
0
-2
0
100
200
300
2
0
-2
-2
-1
0
1
2
20
0
-20
200
0
-200

ode45