Rendu sur le serveur pédagogique :

Ecrire un rapport avec vos lignes de code en Matlab ou Scilab sur l'exercice Oscillateur conservatif à un degré de liberté. Répondre à toutes les questions le schémas Euler explicite, implicite et Runge Kutta.

Oscillateur conservatif linéaire à un degré de liberté

1. Solution analytique de l'équation

1.1 La solution est sous forme de $x(t) = x_0 cos(\omega_0 t + \varphi)$, $\omega_0 = \sqrt{\frac{k}{m}}$. En utilisant les conditions initiales données, le code Matlab est comme ci-dessous :

```
clear all ; close all ; clc
T0=3; w0=2*pi; w0c=w0*w0;
q0=1.; dq0=0.0; dte=0.01;
te=(0:dte:T0)';
npe=size(te,1);
qe=zeros(npe,1);
dqe=zeros(npe,1);
qe=q0*cos(w0*te)+dq0/w0*sin(w0*te) ;
dqe=-w0*q0*sin(w0*te)+dq0*cos(w0*te) ;
ddqe=-w0c*qe ;
```

1.2 E*=19.7392 ne change pas.

2. Résolution de l'équation avec un schémas d'EULER explicite

2.1 D'après (5), on obtient

$$\begin{cases} q_{i+1} = q_i + \Delta t \times \dot{q}_i \\ \dot{q}_{i+1} = \dot{q}_i + \Delta t \times \ddot{q}_i \\ \ddot{q}_{i+1} = -{\omega_0}^2 q_{i+1} \end{cases}$$

Donc,

$$\begin{vmatrix} q_{i+1} \\ \dot{q}_{i+1} \end{vmatrix} = \begin{bmatrix} 1 & \Delta t \\ -{\omega_0}^2 \Delta t & 1 \end{bmatrix} \begin{vmatrix} q_i \\ \dot{q}_i \end{vmatrix}$$

2.2 a) La programmation directe

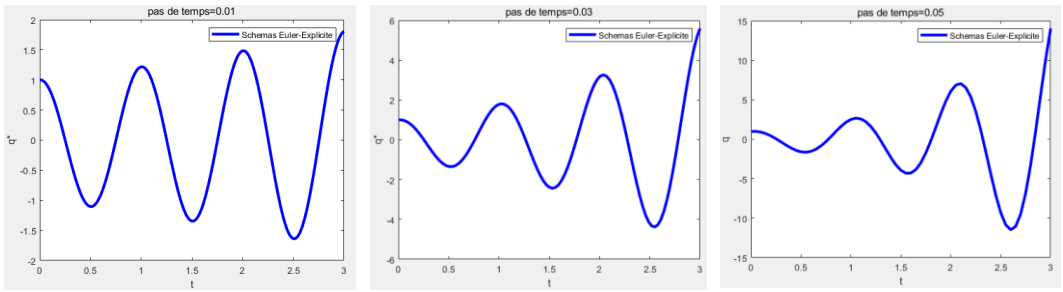
```
clear all ; close all ; clc
T0=3; w0=2*pi; w0c=w0*w0;
q0=1;dq0=0;
dt1=0.01;
t1=(0:dt1:T0)';
np1=size(t1,1);
q1=zeros(np1,1);
dq1=zeros(np1,1);
ddq1=zeros(np1,1);
q1(1)=q0;
dq1(1)=dq0;
```

```
ddq1(1)=-w0c*q1(1);
for inc=2:np1
q1(inc)=q1(inc-1)+dt1*dq1(inc-1);
dq1(inc)=dq1(inc-1)+dt1*ddq1(inc-1);
ddq1(inc)=-w0c*q1(inc);
end
plot(t1,q1,'b-','Linewidth',3);
xlabel('t') ;
ylabel('q') ;
title(sprintf("pas de temps=%.2f",dt1));
legend('Schemas Euler-Explicite') ;
```

b) Avec Matrice d'amplification

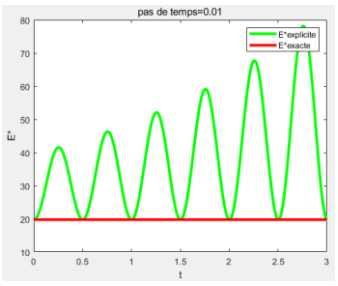
```
clear all ; close all ; clc
dt1=0.05; T0=10;
q0=1 ;dq0=0 ;
w0=2*pi; w0c=w0*w0;
q=[q0 ;dq0] ;
t1=(0:dt1:T0)';
np1=size(t1,1);
q1b=zeros(np1,1);
q1b(1)=q0;
A=[1,dt1;-w0c*dt1,1];
for inc=2:np1;
q=A*q;
q1b(inc)=q(1);
dq1b(inc)= q(2);
end;
plot(t1,q1b,'g-','Linewidth',3);
xlabel('t') ;
ylabel('q') ;
legend('Euler-Explicite-Avec Matrice') ;
```

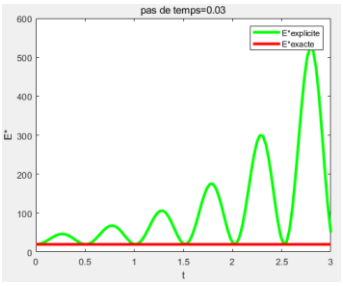
2.3 Comparer différents pas de temps

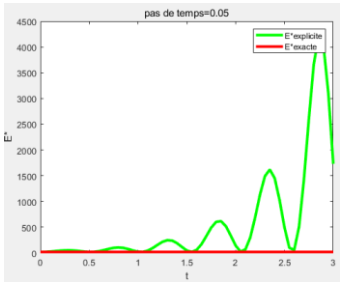


La solution obtenue est divergente, plus le pas de temps est petit, plus la divergence est lente.

2.4 Comparer E* en différents pas de temps avec E* obtenue a partir de la solution exacte.



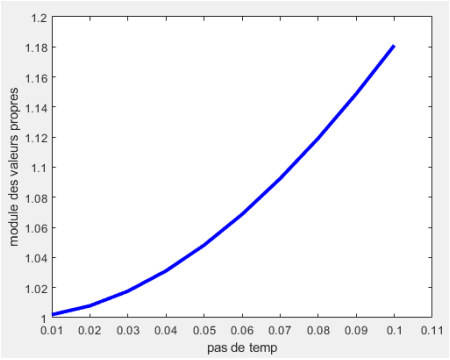




L'énergie obtenue est divergente, plus le pas de temps est petit, plus la divergence est lente.

2.5 études de la stabilité en utilisant code ci-dessous :

```
clear all ; close all ; clc
dt=(0.01:0.01:0.1);
w0=2*pi;
nmp=size(dt,2);
mo=zeros(1,nmp);
interm=zeros(2,2);
for inc=1:nmp;
A=[1,dt(inc);-1*w0*w0*dt(inc),1];
[z,d]=eig(A);
interm=abs(d);
mo(inc)=interm(1,1);
end;
plot(dt,mo,'b-','Linewidth',3);
xlabel('pas de temp');
ylabel('module des valeurs propres');
```



Comme les modules des valeurs propres sont toujours supérieur a 1, la solution numérique avec un schémas Euler explicite est inconditionnellement instable. Plus le pas de temps est grand, plus l'instabilité est grande.

3. Résolution de l' équation avec un Schéma d'Euler implicite

3.1 Sans matrice d'amplification

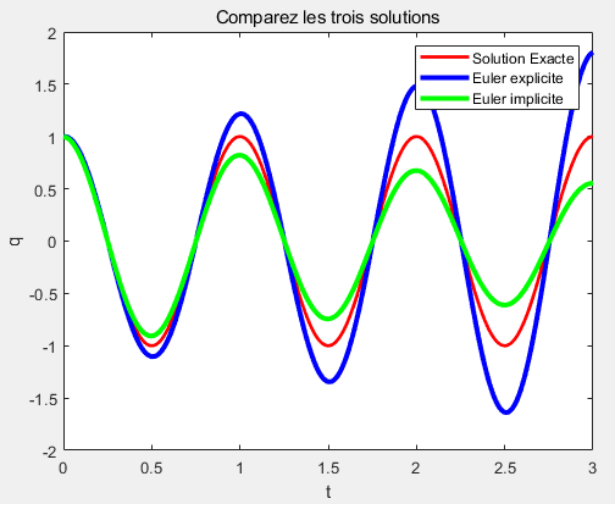
```
clear all ; close all ; clc
dt2=0.01; T0=3;
q0=1;dq0=0;
w0=2*pi; w0c=w0*w0;
t2=(0:dt2:T0)';
```

```
np2=size(t2,1);
q2=zeros(np2,1);
dq2=zeros(np2,1);
energ2=zeros(np2,1);
q2(1)=q0;
dq2(1)=dq0;
for inc=2:np2
    q2(inc)=(q2(inc-1)+dt2*dq2(inc-1))/(1+w0c*dt2*dt2);
    ddqc=-w0c*q2(inc);
    dq2(inc)=dq2(inc-1)+dt2*ddqc;
end
energ2=0.5*(dq2.*dq2+w0c*(dq2.^2));
plot(t2,q2,'b-','Linewidth',3);
```

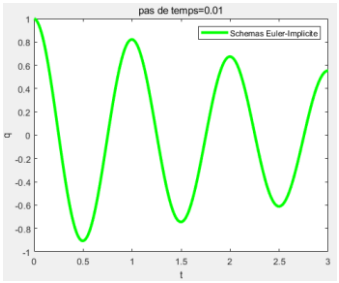
3.2 Comparer les valeurs de q fournies par les trois solutions :

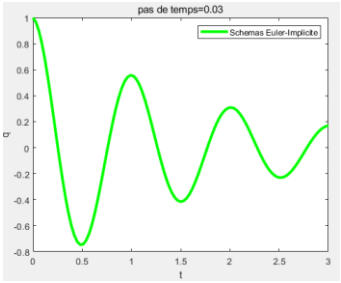
```
clear all ; close all ; clc

T0=3; w0=2*pi; w0c=w0*w0;
q0=1.; dq0=0.0;

% Solution exacte
dte=0.01;
te=(0:dte:T0)';
npe=size(te,1);
qe=zeros(npe,1);
dqe=zeros(npe,1);
energe=zeros(npe,1);
qe=q0*cos(w0*te)+dq0/w0*sin(w0*te) ;
dqe=-w0*q0*sin(w0*te)+dq0*cos(w0*te) ;
ddqe=-w0c*qe ;
energe=0.5*(dqe.*dqe+w0c*(qe.*qe));

% Euler explicite
dt1=0.01;
t1=(0:dt1:T0)';
np1=size(t1,1);
q1=zeros(np1,1);
dq1=zeros(np1,1);
ddq1=zeros(np1,1);
energ1=zeros(np1,1);
q1(1)=q0;
dq1(1)=dq0;
ddq1(1)=-w0c*q1(1);
for inc=2:np1
q1(inc)=q1(inc-1)+dt1*dq1(inc-1);
```

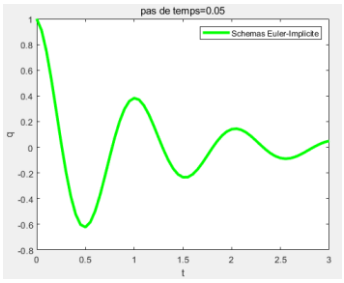
```
dq1(inc)=dq1(inc-1)+dt1*ddq1(inc-1);
ddq1(inc)=-w0c*q1(inc);
end
energ1=0.5*(dq1.*dq1+w0c*(q0.^2));

% Euler implicite
dt2=0.01;
t2=(0:dt2:T0)';
np2=size(t2,1);
q2=zeros(np2,1);
dq2=zeros(np2,1);
energ2=zeros(np2,1);
q2(1)=q0;
dq2(1)=dq0;
for inc=2:np2
    q2(inc)=(q2(inc-1)+dt2*dq2(inc-1))/(1+w0c*dt2*dt2);
    ddqc=-w0c*q2(inc);
    dq2(inc)=dq2(inc-1)+dt2*ddqc;
end
energ2=0.5*(dq2.*dq2+w0c*(dq2.^2));

%plot

plot(te,qe, '-r', 'linewidth',2);
hold on
plot(t1,q1,'b-','Linewidth',3);
plot(t2,q2,'g-','Linewidth',3);

xlabel('t') ;
ylabel('q') ;
title('Comparez les trois solutions')
legend('Solution Exacte','Euler explicite','Euler implicite') ;
```



3.3 Comparer différents pas de temps

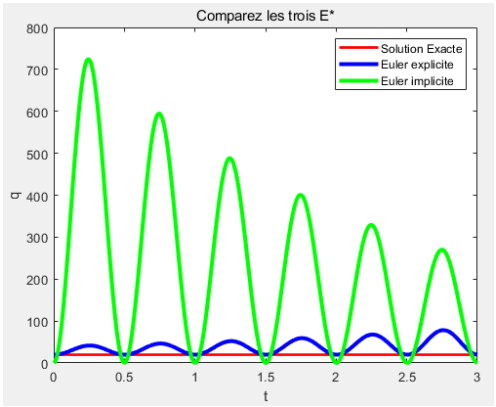


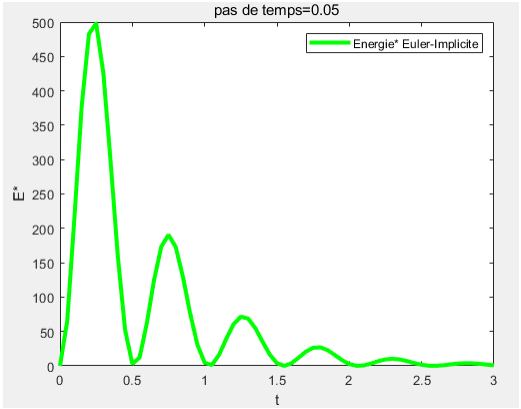




Il apparait des amortissements numériques. Plus le pas de temps est petit, plus l'atténuation des oscillations est faible.

3.4 Comparer les E* en pas de temps 0.01



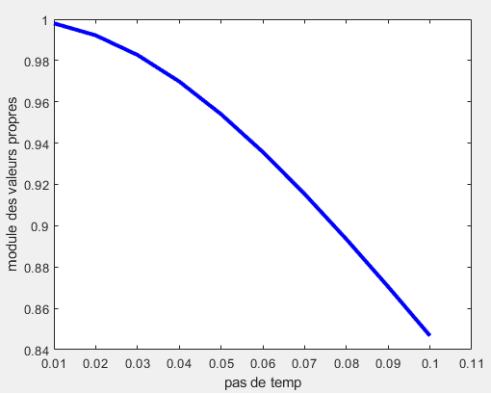


L'énergie obtenue avec schémas d'Euler Implicite est convergente.

On varie le pas de temps, et trouve que plus le pas de temps est petit, plus la convergence est lente.

3.5 Calculer numériquement les valeurs propres de la matrice d'amplification en fonction du pas de temps.

```
clear all ; close all ; clc
dt2=(0.01:0.01:0.1);
w0=2*pi;
nmp=size(dt2,2);
mo=zeros(1,nmp);
interm=zeros(2,2);
for inc=1:nmp;
A=[1,dt2(inc);-1*w0*w0*dt2(inc),1];
A=A/(1+w0*w0*dt2(inc)*dt2(inc));
[z,d]=eig(A);
interm=abs(d);
mo(inc)=interm(1,1);
end;
plot(dt2,mo,'b-','Linewidth',3);
xlabel('pas de temp');
ylabel('module des valeurs propres');
```



Comme les modules des valeurs propres sont toujours inférieur à 1, la solution numérique avec un schémas Euler implicite est inconditionnellement stable. Plus le pas de temps est grand, plus la solution numérique est stable.

4. Résolution de l'équation avec un schéma de RUNGE KUTTA

4.1 Transformation

$$\begin{vmatrix}\dot{x}(t)\\ \ddot{x}(t)\end{vmatrix} = \begin{bmatrix}0 & \Delta t\\ -{\omega_0}^2 & 1\end{bmatrix}\begin{vmatrix}x(t)\\ \dot{x}(t)\end{vmatrix}$$

4.2 Code Matlab

Définir fonction et l'enregistrer sur le nom cal_f.m

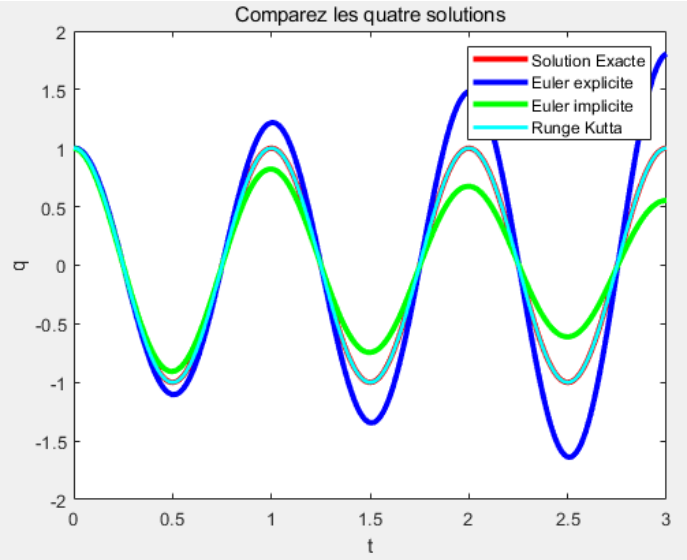
```
function [ dUc ] = cal_f(Uc,tc,w0c);
dUc=zeros(2,1);
dUc(1)=Uc(2);
dUc(2)=-w0c*Uc(1);
end
```

Main Function:

```
clear all ; close all ; clc
T0=3;w0=2*pi;w0c=w0*w0;
q0=1;dq0=0;
dt4=0.01;
t4=(0:dt4:T0)';
np4=size(t4,1);
q4=zeros(np4,1);
dq4=zeros(np4,1);
q4(1)=q0;
dq4(1)=dq0;
qj=[q0;dq0];
for i=2:np4
    tc=t4(i-1)
    xc=qj
    k1=cal_f(xc,tc,w0c)
    xc=qj+k1*dt4/2
    k2=cal_f(xc,tc+dt4/2,w0c)
    xc=qj+k2*dt4/2
    k3=cal_f(xc,tc+dt4/2,w0c)
```

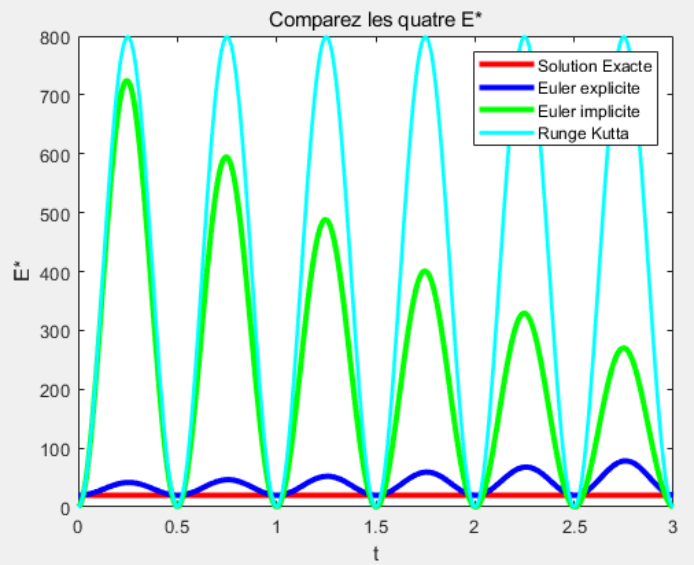
```
        xc=qj+k3*dt4
        k4=cal_f(xc,tc+dt4,w0c)
        dq=(k1+2*k2+2*k3+k4)/6
        qj=qj+dq*dt4
        q4(i)=qj(1)
        dq4(i)=qj(2)
end
plot(t4,q4,'p-','Linewidth',3);
xlabel('t') ;
ylabel('q') ;
title(sprintf("pas de temps=%.2f",dt4));
legend('Schema Runge Kutta');
```

4.3 Comparer les 4 solutions



Le schéma Runge Kutta est parfaitement coïncidente avec la solution exacte.

4.4 Comparer E*



5. Résolution de l'équation avec un schéma de NEWMARK

5.1 La méthode des trapèzes

5.1.1 Programmer

5.1.2 Comparer les cinq solutions

5.1.3 Comparer les cing E*

5.1.4 Calculer les valeurs propres pour $\Delta t \in [0,1]$

5.2 Schéma des différences finies centrées