

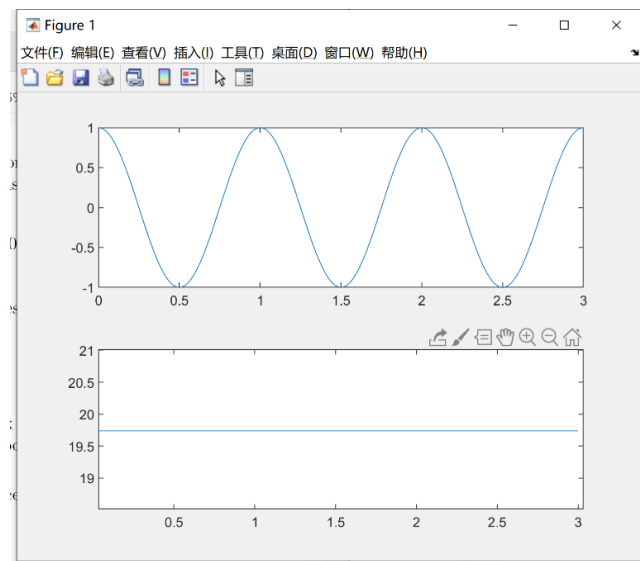
EXO 1

1.1 Code de Matlab :

```
syms q omega0;
equation = 'D2q+omega0^2*q=0';
x= dsolve(equation, 'q(0)=1', 'Dq(0)=0');
vitess = diff(x);
t = 0:0.01:3;
x= subs(x, omega0, 2*pi);
x= subs(x, t);
vitess= subs(vitess, omega0, 2*pi);
vitess= subs(vitess, t);
Etoile = (vitess.^2 + (2*pi)^2*x.^2)/2;
subplot(2,1,1);
plot(t,x);
subplot(2,1,2);
plot(t,Etoile);
```

La solution analytique de q est : $q = \cos(2\pi t)$

1.2 Avec le code de 1.1, j'obtiens que le terme E^* est une constante qui vaut environ 19.7392



2.1 On a d'abord l'équation (1) : $\ddot{q} + \omega_0^2 q = 0 \Rightarrow \ddot{q} = -\omega_0^2 q$
 Puis on peut écrire une équation différentielle :

$$\begin{pmatrix} \dot{q}_i \\ \ddot{q}_i \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{pmatrix} \begin{pmatrix} q_i \\ \dot{q}_i \end{pmatrix}$$

On a aussi la relation 5 :

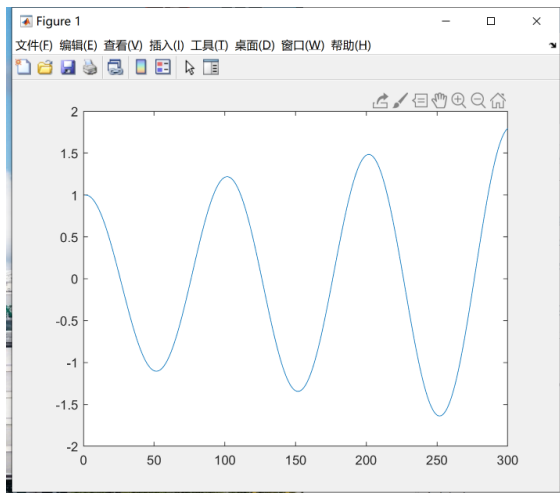
$$\begin{pmatrix} q_{i+1} \\ \dot{q}_{i+1} \\ q_{i+1} \end{pmatrix} = \begin{pmatrix} q_i \\ \dot{q}_i \\ q_i \end{pmatrix} + \Delta t \begin{pmatrix} \dot{q}_i \\ \ddot{q}_i \\ -\omega^2 q_i \end{pmatrix}$$

En fin, on peut écrire :

2.2 Je choisis la méthode 2 avec un pas de 0.01s

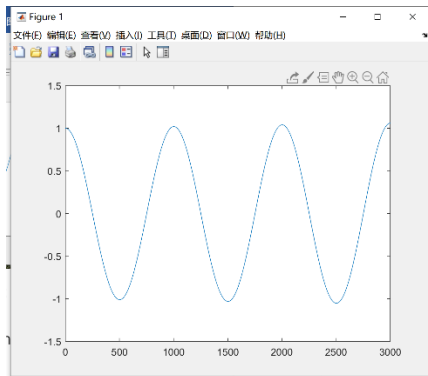
Le code de Matlab:

```
interval = 0.01;
omega = 2*pi;
m = [1,interval;-omega^2*interval,1];
res = zeros(2,300);
res(:,1) = [1;0];
for i=1:299
    res(:,i+1) = m*res(:,i);
end
plot(res(1,:))
```

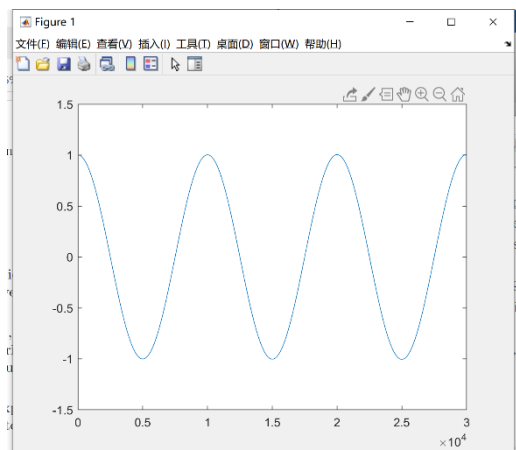


Cette image est évidemment divergente.

2.3 Je prends un pas plus petit a 0.001s :



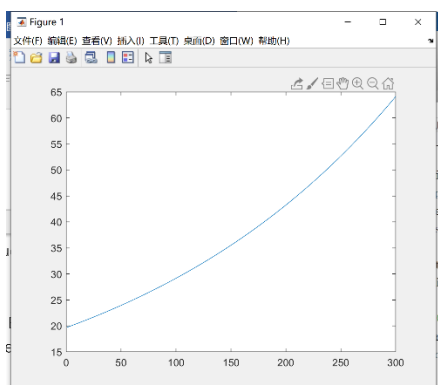
Puis, je prends un pas encore plus petit a 0.0001s :



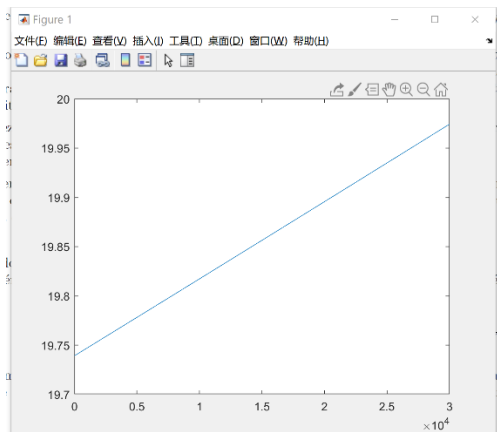
C'est évident que plus le pas de temps est petit, plus la divergence est lente.

2.4 Le terme E^* n'est plus constant. Mais je constate que E^* tend vers une valeur constante quand on met le pas plus petit.

Pour un pas de 0.01s : Ce terme varie entre (20, 65)



Pour un pas de 0.00001s : Ce terme varie entre (19.73, 19.97). Ce qui signifie que c'est plus stable.



2.5

```
>> eig(m)
ans =
    1.0000 + 0.0006i
    1.0000 - 0.0006i
```

Les valeurs propres de cette matrice sont positives, donc le résultat est divergent. Pour le rendre plus stable, il faut que les parties réelles des valeurs propres soient proches de 1.

Le code de Matlab :

```
interval = 0.0001;
steps = 3/interval;
omega = 2*pi;
m = [1, interval; -omega^2*interval, 1];
res = zeros(2, steps);
res(:, 1) = [1; 0];
for i=1:(steps-1)
    res(:, i+1) = m*res(:, i);
end
plot(res(1,:))
etoile = (res(2,:).^2 + omega^2*res(1,:).^2)/2;
plot(etoile)
```

$$\begin{aligned} 3. \quad X_{i+1} &= X_i + \Delta t X_{i+1} \Rightarrow (I - \Delta t \cdot M) X_{i+1} = X_i \\ X_{i+1} &= X_i + \Delta t \cdot M X_{i+1} \\ \text{où } X_i &= \begin{pmatrix} q_i \\ \dot{q}_i \end{pmatrix} \\ X_{i+1} &= \underline{\underline{(I - \Delta t \cdot M)^{-1} X_i}} \end{aligned}$$

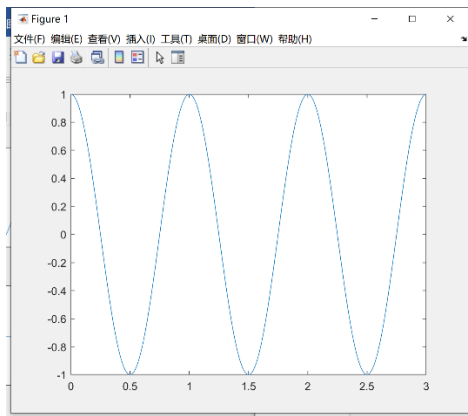
3.1 Je choisie la méthode 2.

Le code de Matlab :

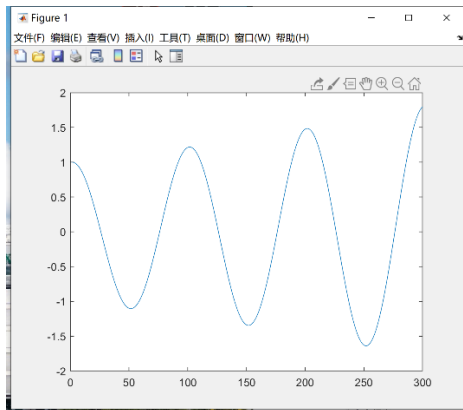
```
interval = 0.01;
steps = 3/interval;
omega = 2*pi;
m = [0, 1; -omega^2, 0];
M = inv(eye(2) - interval*m);
res = zeros(2, steps);
res(:, 1) = [1; 0];
for i=1:(steps-1)
    res(:, i+1) = M*res(:, i);
end
```

`plot(res(1,:))`

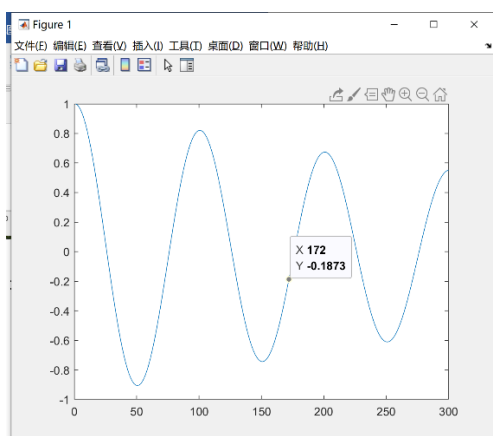
3.2 La solution exacte : Ni convergence, ni divergence



EULER explicite : Divergence

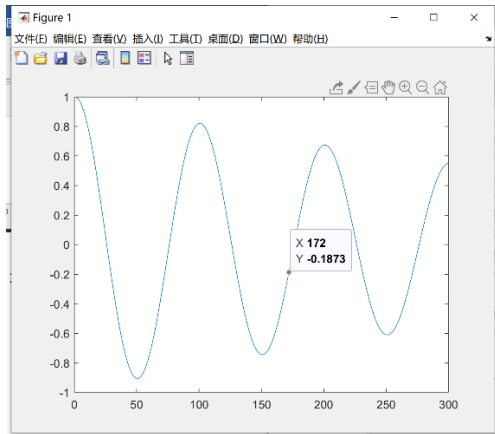


EULER implicite : Convergence

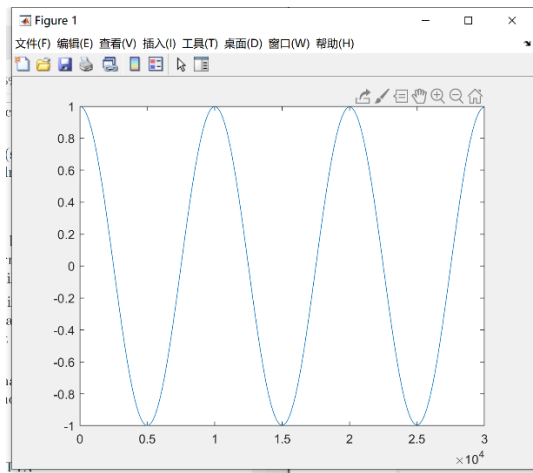


3.3

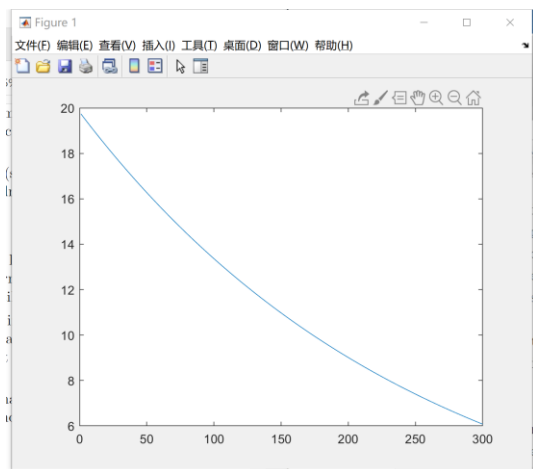
Quand le pas de temps est 0.01s : La solution est évidemment divergente.



Quand on met le pas de temps plus petit, à 0.0001s : C'est peu convergent.

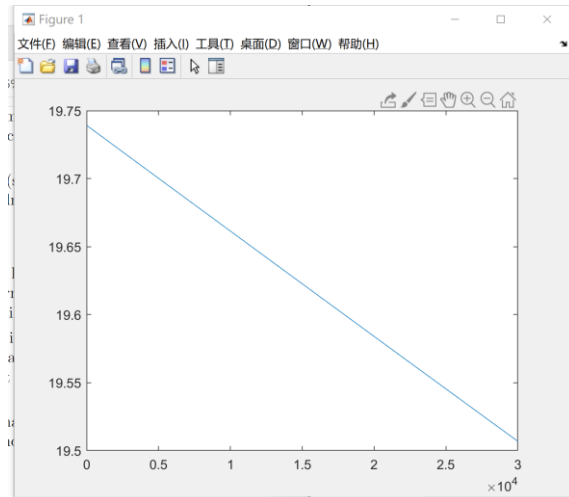


3.4 Cette fois le terme E^* est décroissant au pas de 0.01s.



Quand on met le pas de temps plus petit, E^* tend aussi vers une constante :

Figure au pas de 0.0001s



Evidemment la variation est plus petite.

3.5

```
>> eig(M)
```

```
ans =
```

```
0.9961 + 0.0626i
0.9961 - 0.0626i
```

Pour le pas de temps 0.01s

Pour rendre le résultat plus stable, il faut que les parties réelles des valeurs propres soient proches de 1.

EXO4

4.1 Je peux réécrire l'équation comme :

$$\begin{pmatrix} \ddot{q} \\ \dot{q} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix} \begin{pmatrix} q \\ \dot{q} \end{pmatrix}$$

$$\Rightarrow \dot{y}(t) = f(y(t), t)$$

$$\dot{y}(t) = M y(t)$$

4.2 Le code de Matlab :

```
interval = 0.01;
```

```
steps = 3/interval;
```

```
omega = 2*pi;
```

```
m = [0,1;-omega^2,0];
```

```
res = zeros(2,steps);
```

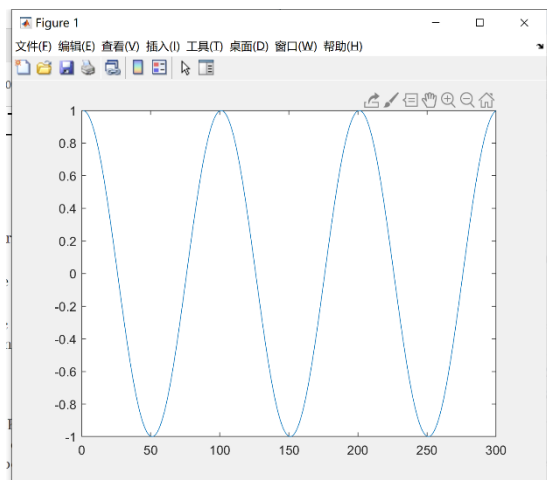
```
res(:,1) = [1;0];
```

```

for i=1:(steps-1)
    k1 = m*res(:,i);
    k2 = m*(res(:,i)+k1*interval/2);
    k3 = m*(res(:,i)+k2*interval/2);
    k4 = m*(res(:,i)+k3*interval);
    K = (k1+2*k2+2*k3+k4)/6;
    res(:,i+1) = res(:,i)+K * interval;
end
plot(res(1,:))

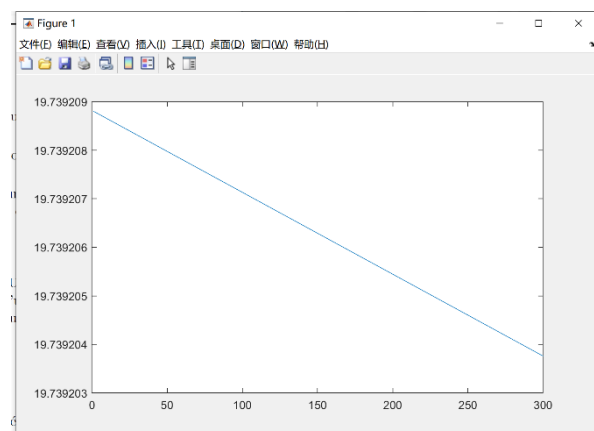
```

4.3



La figure de $q(t)$ avec la méthode de RUNGE KUTTA avec un pas de temps de 0.01s
 On peut voir que la performance est beaucoup meilleure, il n'y a pas d'atténuation ou d'amplification.

4.4 La figure de E^* , c'est presque une constante.



Exo5

5.1

5.1.1 Le code de Matlab :


```

interval = 0.01;
steps = 3/interval;
omega = 2*pi;
beta = 0.25;
gamma = 0.5;
B = [1+beta*interval^2*omega^2 0;
     gamma*interval*omega^2 1];
C = [1-(0.5-beta)*interval^2*omega^2 interval;
     -(1-gamma)*interval*omega^2 1];
A = inv(B)* C;
res = zeros(2,steps);
res(:,1) = [1;0];
for i=1:(steps-1)
    res(:,i+1) = A*res(:,i);
end
plot(res(1,:))

```

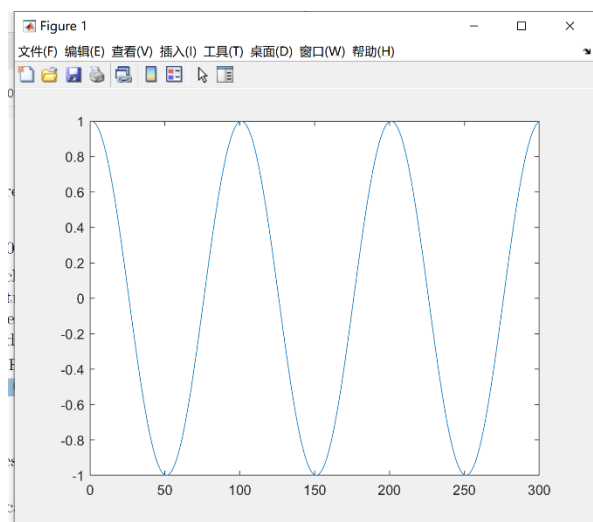
La matrice d'amplification :

```

A =
    0.9980    0.0100
   -0.3944    0.9980

```

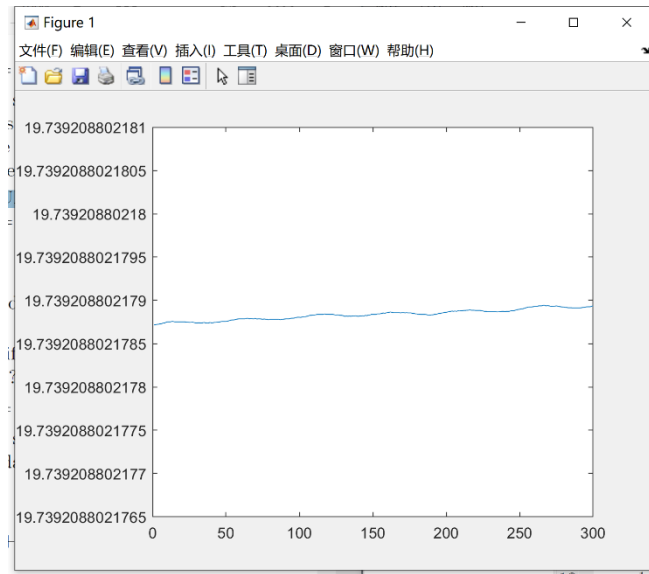
5.1.2 : La solution de NewMark (beta =0.25, gamma=0.5) :



La solution de NewMark est aussi performante que celle de RUNGE KUTTA, elles sont toutes assez stables. Elles sont très proche au résultat exact.

Mais les solutions de EULER explicite et EULER implicite sont pas assez agréables, avec une forte atténuation ou amplification.

5.1.3



La figure de E^* de NewMark est encore plus constante que celle de RUNGE KUTTA. C'est très proche du résultat exact.

```
5.1.4 >> eig(A)

ans =

    0.9980 + 0.0628i
    0.9980 - 0.0628i
```

Les parties réelles des matrices d'amplification sont toutes proches de 1.

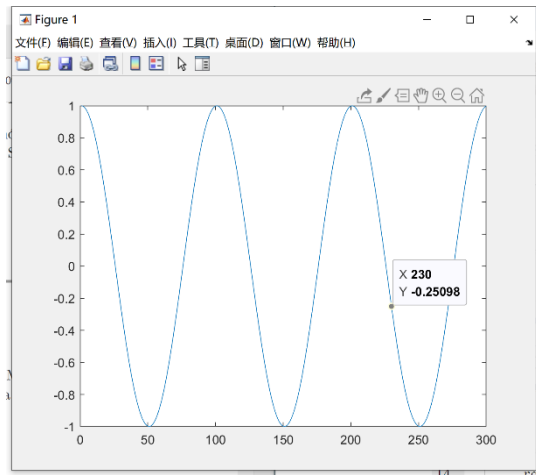
5.2

5.2.1 Le code de Matlab

```
interval = 0.01;
steps = 3/interval;
omega = 2*pi;
beta = 0;
gamma = 0.5;
B = [1+beta*interval^2*omega^2 0;
     gamma*interval*omega^2 1];
C = [1-(0.5-beta)*interval^2*omega^2 interval;
     -(1-gamma)*interval*omega^2 1];
A = inv(B)*C;
res = zeros(2,steps);
res(:,1) = [1;0];
for i=1:(steps-1)
    res(:,i+1) = A*res(:,i);
```

```
end  
plot(res(1,:))
```

5.2.2



La solution du schéma de NEWMARK avec $\gamma=0.5$, $\beta=0$

Cette solution est aussi performante que celle de RUNGE KUTTA, elles sont toutes assez stables. Elles sont très proche au résultat exact.

Mais les solutions de EULER explicite et EULER implicite sont pas assez agréables, avec une forte atténuation ou amplification.