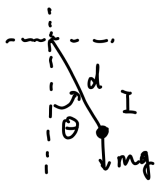
Oscillateur conservatif linéaire à un degré de liberté

Table of Contents

Sébastien SY1924130

Retrouver l'équation du mouvement du pendule simple avec les équations de Lagrange

Prenons le pendule simple pour étudier.
θ comme la coordonnée généralisée, I comme le moment d'inertie.



Donc $E_c = \frac{I}{2}\dot{\theta}^2$, $E_p = -mgd\cos\theta$ (choisissons $E_p(\theta = 90°) = 0$)

$\delta W = 0$ à cause de la conservation de l'énergie mécanique

Lagrangien $\mathcal{L} = E_c - E_p = \frac{I}{2}\dot{\theta}^2 + mgd\cos\theta$

Équation de Lagrange :

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{\theta}}\right] - \frac{\partial L}{\partial \theta} = 0$$

$$\Rightarrow I\ddot{\theta} + mgd\sin\theta = 0$$

linéaire

$$\Rightarrow I\ddot{\theta} + mgd\theta = 0$$

Montrer que le schéma explicite avec matrice d'amplification est obtenu à partir du système du premier ordre que l'on discrétise en temps de manière explicite

On applique le développement limité

$$y(t_0+h) = y(t_0) + hy'(t_0) + \frac{1}{2}h^2 y''(t_0) + O(h^3)$$

① isolation :

$$y'(t_0) \approx \frac{y(t_0+h) - y(t_0)}{h}$$

En intégrant de t_0 à t_0+h

$$y(t_0+h) - y(t_0) = \int_{t_0}^{t_0+h} y'(t, y(t))\, dt$$

Approximation par la somme de Riemann

$$\int_{t_0}^{t_0+h} y'(t, y(t))\, dt \approx h y'(t_0, y(t_0))$$

En combinant les deux équations, on retrouve la méthode d'Euler

Paramètres

```
clear global;
t_start = 0;
T0 = 3;
delta_t = 0.01;
t = t_start:delta_t:T0;
```

1.1

```
omega_0 = 2*pi;
q_exact = dsolve('D2q+omega_0^2*q=0','q(0)=1,Dq(0)=0')
```

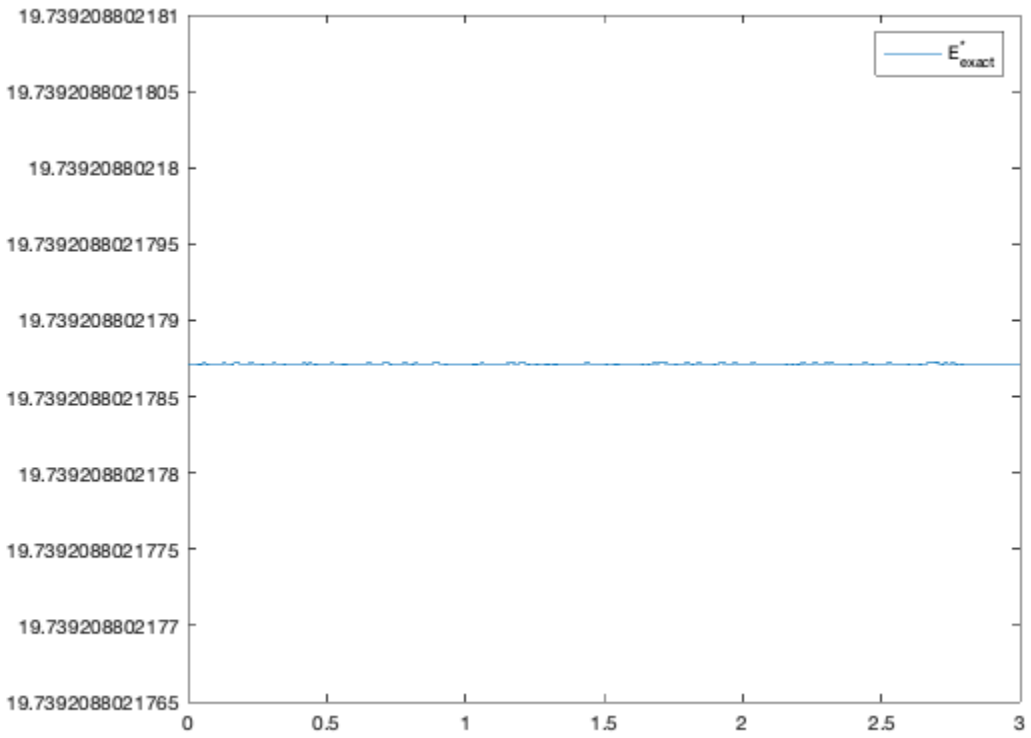
```
q_exact =

exp(-omega_0*t*1i)/2 + exp(omega_0*t*1i)/2
```

1.2

On peut voir que E_star est constant.

```
E_star_exact = 1/2 * ( diff(q_exact)^2 + omega_0^2  * q_exact^2)
plot(t, eval(E_star_exact));
legend('E^*_{exact}')
hold off
```

```
E_star_exact =

((omega_0*exp(-omega_0*t*1i)*1i)/2 -
 (omega_0*exp(omega_0*t*1i)*1i)/2)^2/2 + (2778046668940015*(exp(-
omega_0*t*1i)/2 + exp(omega_0*t*1i)/2)^2)/140737488355328
```



2.1

$$\begin{vmatrix} q_{j+1} \\ \dot{q}_{j+1} \end{vmatrix} = \begin{vmatrix} q_j \\ \dot{q}_j \end{vmatrix} + \Delta t \times \begin{vmatrix} \dot{q}_j \\ \ddot{q}_j \end{vmatrix} \quad (5)$$

$$\Rightarrow q_{j+1} = q_j + \Delta t \times \dot{q}_j$$

$$\text{et } \dot{q}_{j+1} = \Delta t \times \ddot{q}_j + \dot{q}_j$$

$$= -\omega_0^2 \Delta t\, q_j + \dot{q}_j \text{ , selon (1)}$$

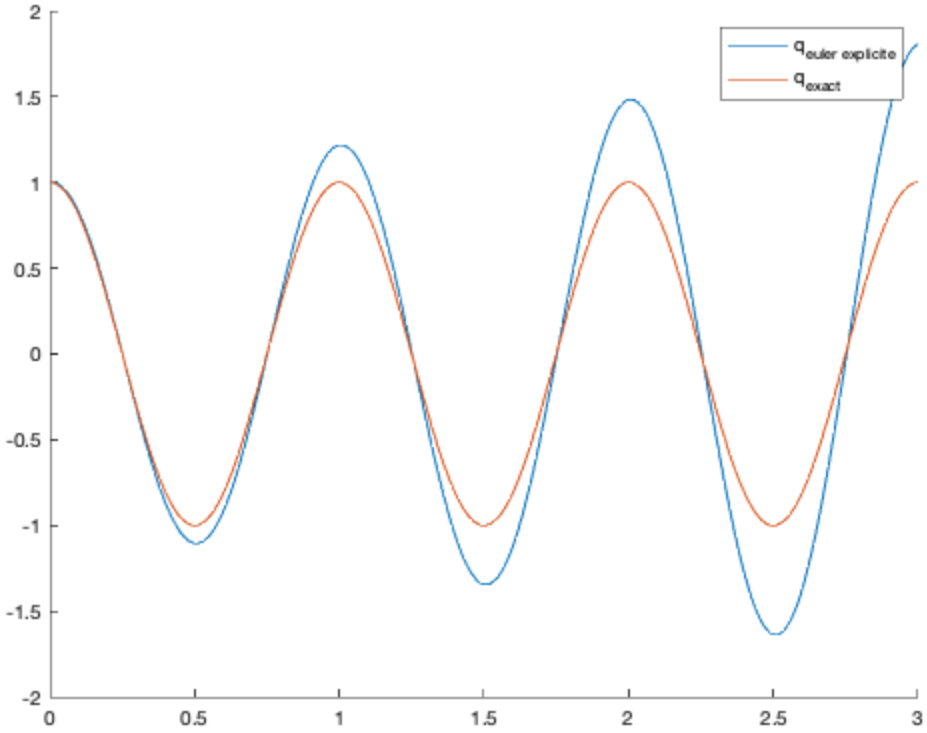
Donc (6) $\begin{vmatrix} q_{j+1} \\ \dot{q}_{j+1} \end{vmatrix} = \begin{bmatrix} 1 & \Delta t \\ -\omega_0^2 \Delta t & 1 \end{bmatrix} \begin{vmatrix} q_j \\ \dot{q}_j \end{vmatrix}$ est vrai

2.2

On utilise la Méthode 2.

```
A = [1 delta_t; -omega_0^2*delta_t 1]
U = iterate(A, t);
clf;
hold on;
q_num = U(1,:);
dq_num = U(2,:);
plot(t, q_num);
plot(t, eval(q_exact));
legend('q_{euler explicite}', 'q_{exact}')
```

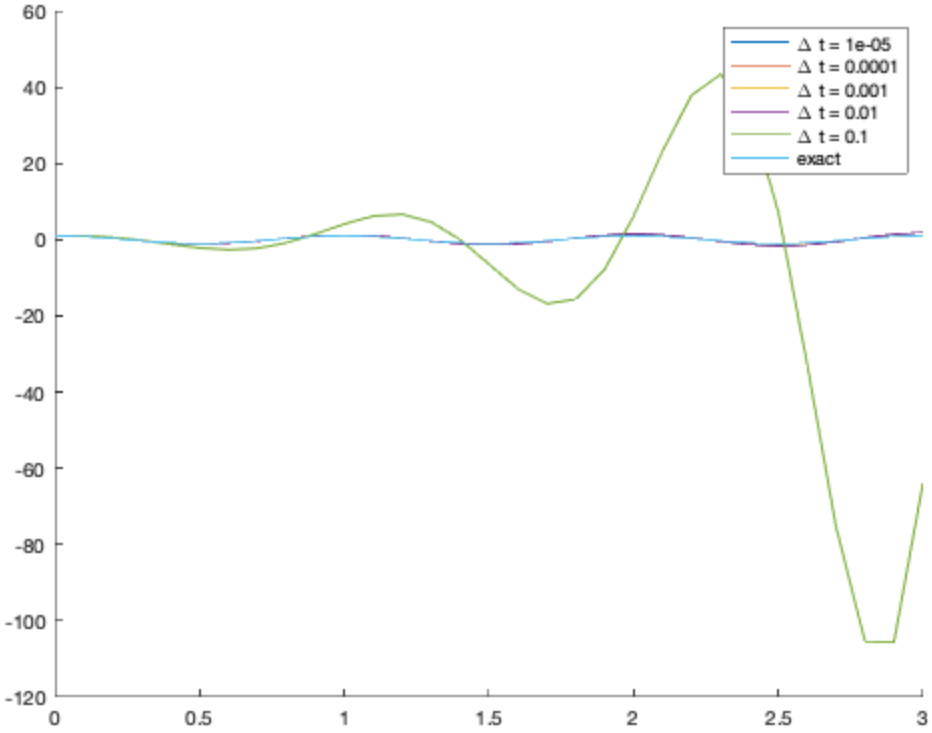
```
A =

    1.0000    0.0100
   -0.3948    1.0000
```



2.3

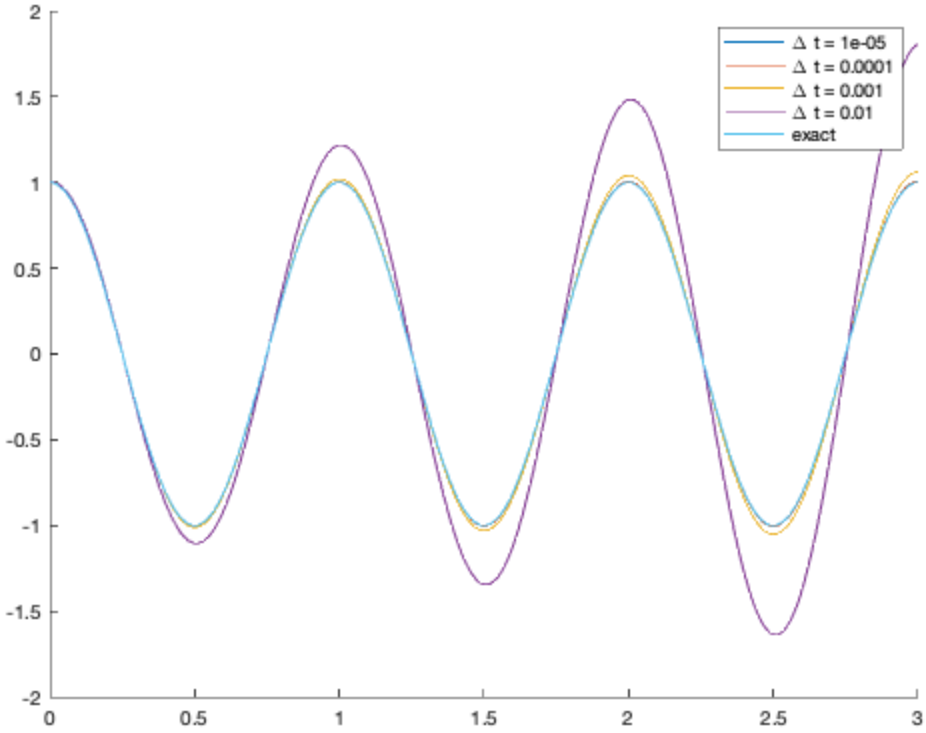
Ce schéma d'intégration est divergente, vue de la figure. Mais plus le pas de temps est petit, plus la divergence est lente.

```
delta_ts = [0.00001 0.0001 0.001 0.01 0.1];
clf;
hold on;
l = numel(delta_ts);
for i = 1:l
    delta_t = delta_ts(i);
    t = t_start:delta_t:T0;
    A = [1 delta_t; -omega_0^2*delta_t 1];
    U = iterate(A, t);
    q_num_temp = U(1,:);
    dq_num_temp = U(2,:);
    plot(t, q_num_temp,'DisplayName',['\Delta t = '
 num2str(delta_t)]);
end
delta_t = 0.01;
t = t_start:delta_t:T0;
plot(t, eval(q_exact),'DisplayName','exact');
legend('show')
```



On supprime la solution de delta_t=0,1 pour voir plus clairement.

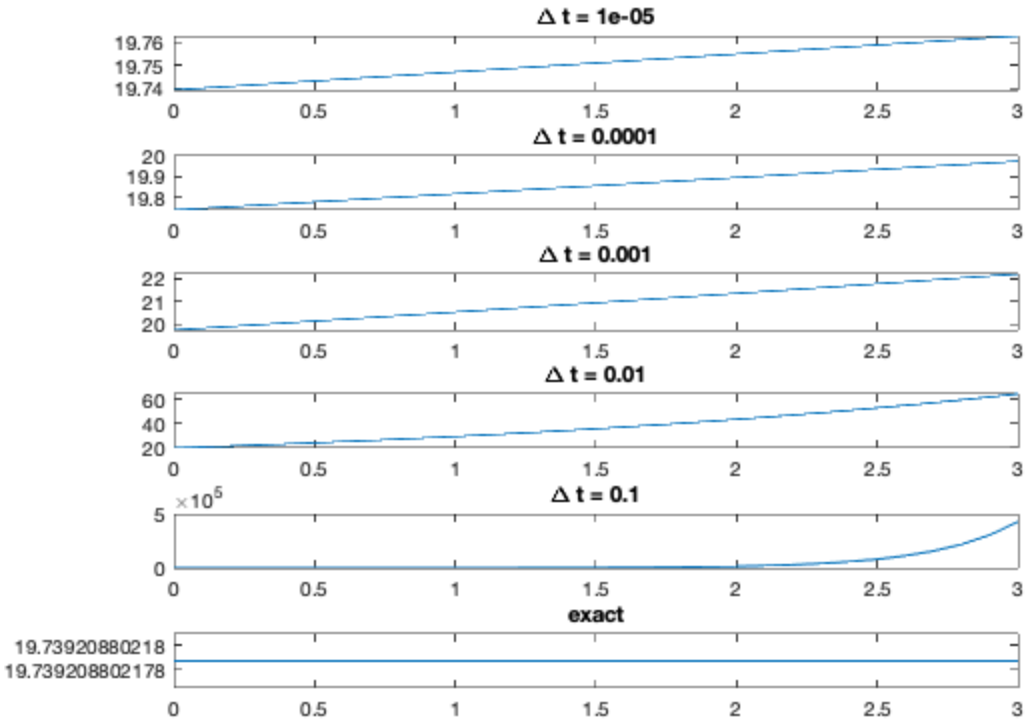
```
children = get(gca, 'children');
delete(children(2));
```



2.4

On peut voir que E_star est divergent. Plus le pas de temps est petit, plus la divergence est lente.

```
clf;
hold off;
for i = 1:numel(delta_ts)
    delta_t = delta_ts(i);
    t = t_start:delta_t:T0;
    A = [1 delta_t; -omega_0^2*delta_t 1];
    U = iterate(A, t);
    q_num_temp = U(1,:);
    dq_num_temp = U(2,:);
    E_star_num_temp = 1/2 * ( dq_num_temp.^2 + omega_0.^2  .*
 q_num_temp.^2);
    subplot(l+1,1,i);
    plot(t, E_star_num_temp);
    title(['\Delta t = ' num2str(delta_t)])
end
subplot(l+1,1,i+1);
delta_t = 0.01;
t = t_start:delta_t:T0;
plot(t, eval(E_star_exact));
title('exact')
```



2.5

Les valeurs propres sont des complexes conjugués. Et plus le pas de temps est grand, plus les modules des valeurs propres sont grandes, mais aussi la norme de ces complexes sont toujours supérieures à 1, ce qui explique l'instabilité inconditionnelle.

```
syms o0 dt
eig([1 dt; -o0^2*dt 1])
```

```
ans =

 1 - dt*o0*1i
 1 + dt*o0*1i
```

3.1

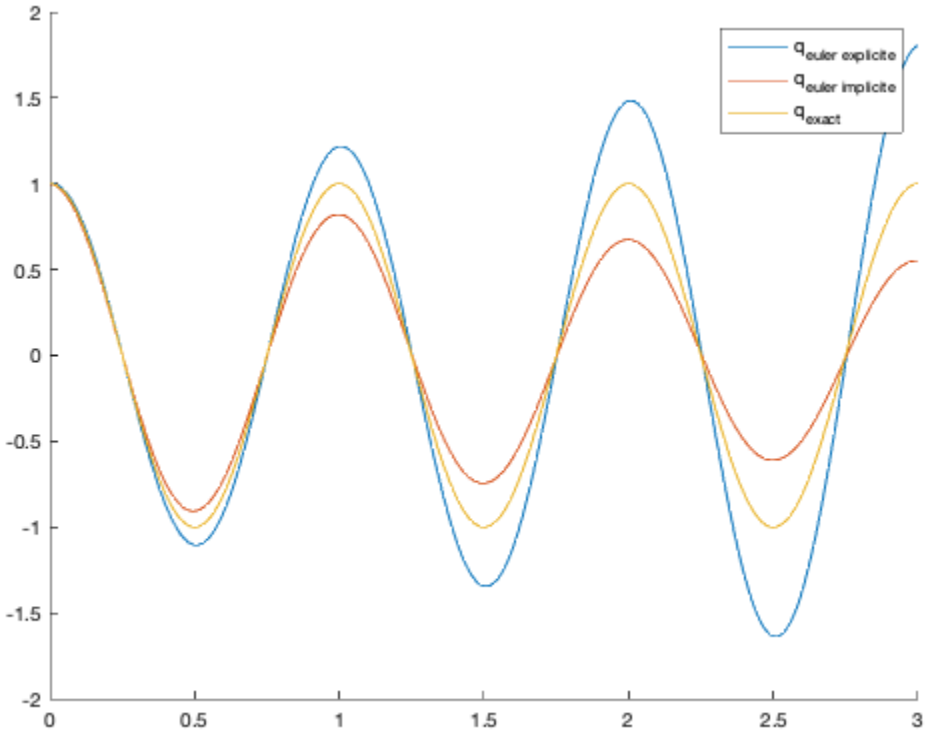
Selon les trois relations données entre les valeurs aux instants t_j et t_j+1, on obtenir la matrice d'amplification A:

```
delta_t = 0.01;
t = t_start:delta_t:T0;
A_2 = 1/(1+omega_0^2*delta_t^2) * [1 delta_t; -omega_0^2*delta_t 1]
U_2 = iterate(A_2, t);
```

```
A_2 =

    0.9961    0.0100
   -0.3932    0.9961
```

3.2

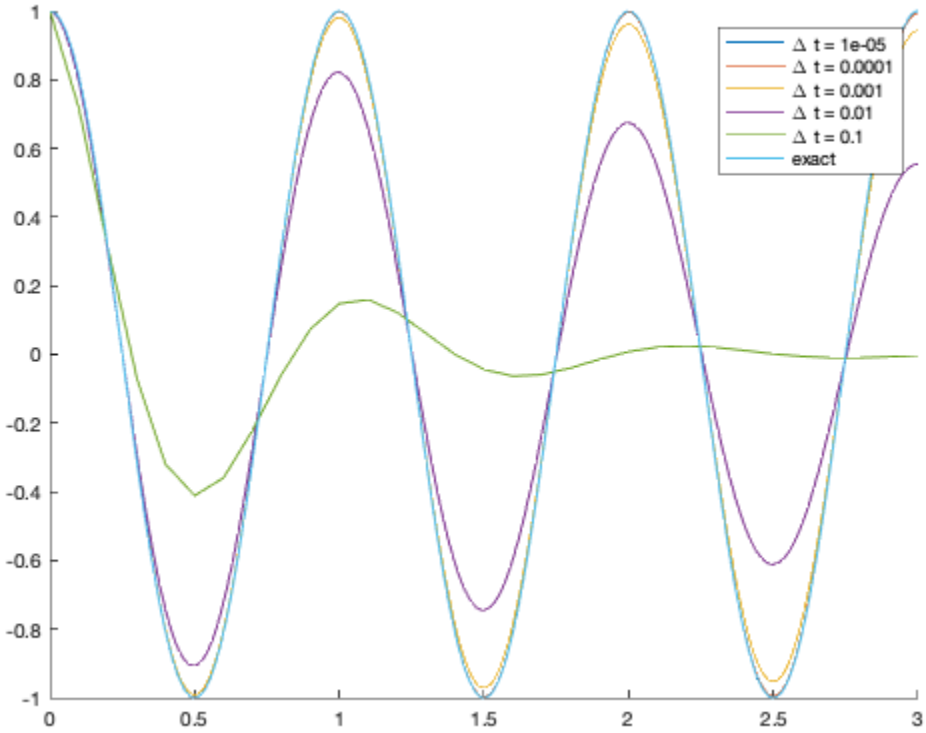
```
clf;
hold on;
q_num_2 = U_2(1,:);
dq_num_2 = U_2(2,:);
plot(t, q_num);
plot(t, q_num_2);
plot(t, eval(q_exact));
legend('q_{euler explicite}','q_{euler implicite}', 'q_{exact}')
```



3.3

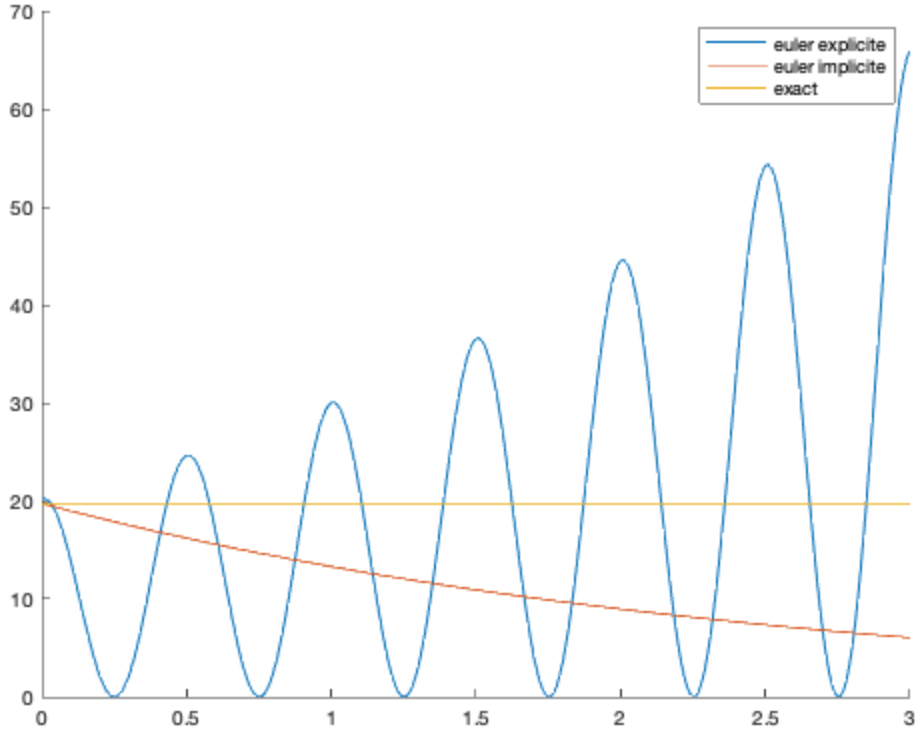
```
clf;
hold on;
l = numel(delta_ts);
for i = 1:l
    delta_t = delta_ts(i);
```

```
    t = t_start:delta_t:T0;
    A_2 = 1/(1+omega_0^2*delta_t^2) * [1 delta_t; -omega_0^2*delta_t
 1];
    U_2_temp = iterate(A_2, t);
    q_num_2_temp = U_2_temp(1,:);
    dq_num_2_temp = U_2_temp(2,:);
    plot(t, q_num_2_temp,'DisplayName',['\Delta t = '
 num2str(delta_t)]);
end
delta_t = 0.01;
t = t_start:delta_t:T0;
plot(t, eval(q_exact),'DisplayName','exact');
legend('show')
```



3.4

```
delta_t = 0.01;
t = t_start:delta_t:T0;
E_star_num = 1/2 * ( q_num.^2 + omega_0.^2  .* q_num.^2);
E_star_num_2 = 1/2 * ( dq_num_2.^2 + omega_0.^2  .* q_num_2.^2);
clf;
hold on;
plot(t, E_star_num);
plot(t, E_star_num_2);
plot(t, eval(E_star_exact));
legend('euler explicite', 'euler implicite', 'exact');
```



3.5

Les valeurs propres sont des complexes conjugués. Et plus le pas de temps est grand, plus la partie imaginaire des valeurs propres sont petites, mais aussi les modules sont toujours inférieures à 1, ce qui explique la stabilité inconditionnelle.

```
eig(1/(1+o0^2*dt^2) * [1 dt; -o0^2*dt 1])
```

```
ans =

  1i/(dt*o0 + 1i)
 1/(1 + dt*o0*1i)
```

4.1

$$\ddot{q} + \omega_0^2 q = 0$$

Soit $U = \begin{pmatrix} q \\ \dot{q} \end{pmatrix}$

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

l'equation est transformée en ordre 1.

4.2

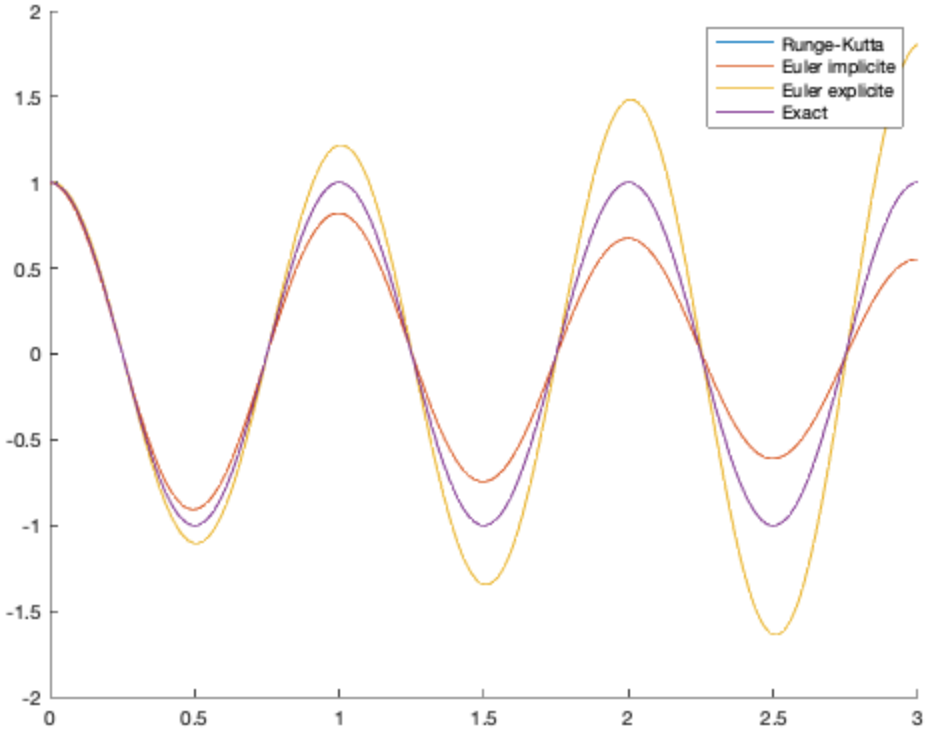
```
delta_t = 0.01;
t = t_start:delta_t:T0;
i = 0;
U_3 = [];
for t_i=t
    if i==0
        U_3(:,1) = [1;0];
    else
        k1 = f(U_3(:,i), t_i, omega_0);
        k2 = f(U_3(:,i) + k1 * delta_t/2, t_i+delta_t/2, omega_0);
        k3 = f(U_3(:,i) + k2 * delta_t/2, t_i+delta_t/2, omega_0);
        k4 = f(U_3(:,i) + k3 * delta_t, t_i+delta_t, omega_0);
        K = (k1 + 2*k2 + 2*k3 + k4) / 6;
        U_3(:,i+1)=U_3(:,i) + K * delta_t;
    end
    i=i+1;
end
```

4.3

On peut voir que RK4 est plus précis et plus stable que les méthodes précédentes. (Dans la figure, la ligne de Runge-Kutta coïncide presque complètement avec la ligne de la solution exacte)

```
clf;
hold on;
q_num_3 = U_3(1,:);
dq_num_3 = U_3(2,:);
plot(t, q_num_3);
plot(t, q_num_2);
```

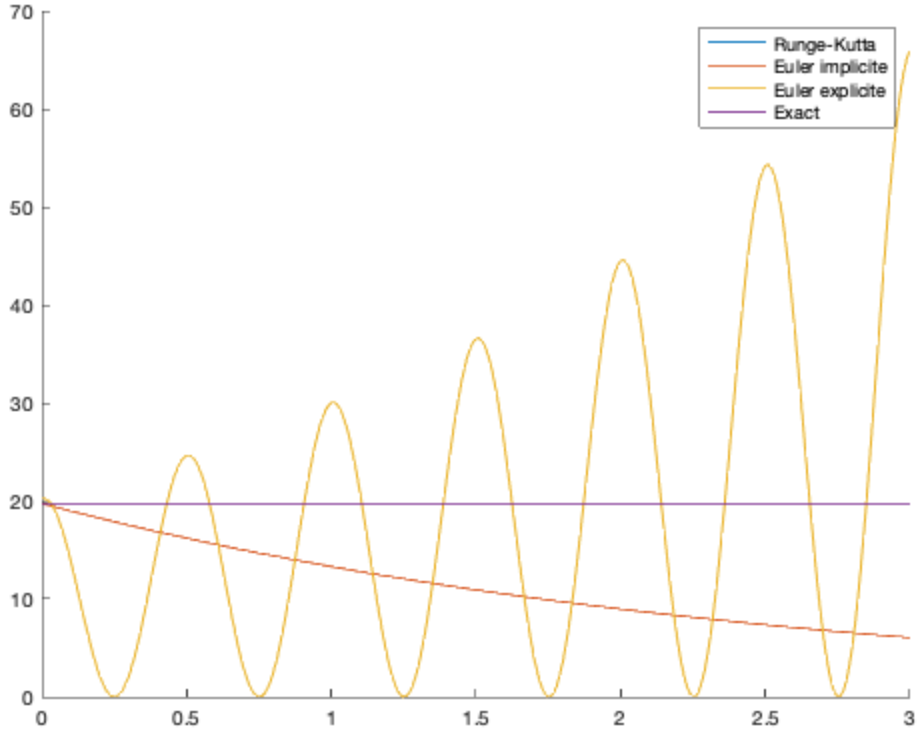
```
plot(t, q_num);
plot(t, eval(q_exact));
legend('Runge-Kutta','Euler implicite','Euler explicite','Exact')
```



4.4

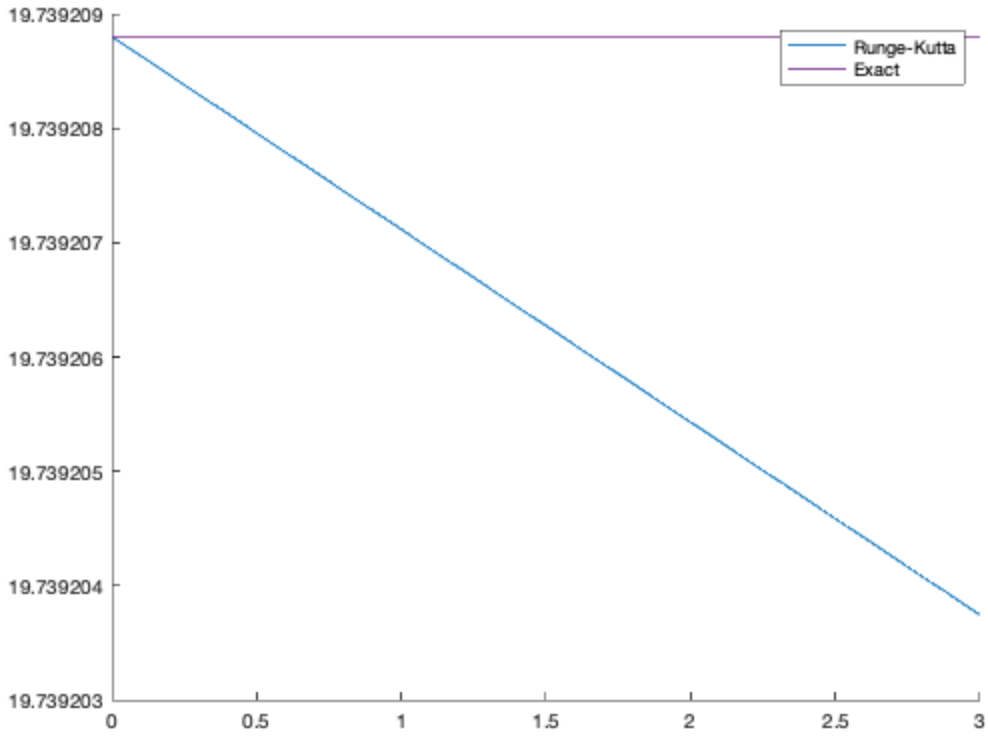
On peut voir que RK4 est plus précis et plus stable que les méthodes précédentes. (Dans la figure, la ligne de Runge-Kutta coïncide presque complètement avec la ligne de la solution exacte)

```
E_star_num_3 = 1/2 * ( dq_num_3.^2 + omega_0.^2  .* q_num_3.^2);
clf;
hold on;
plot(t, E_star_num_3);
plot(t, E_star_num_2);
plot(t, E_star_num);
plot(t, eval(E_star_exact));
legend('Runge-Kutta','Euler implicite','Euler explicite','Exact')
```



On supprime les autres lignes pour mieux comparer la ligne de Runge-Kutta avec la ligne de E_star exacte.

```
children = get(gca, 'children');
delete(children(2));
delete(children(3));
```



5.1.1

```
syms gamma beta B C dt
B = [1+beta*dt^2*omega_0^2 0; gamma*dt*omega_0^2 1];
C = [1-(0.5-beta)*dt^2*omega_0^2 dt; -(1-gamma)*dt*omega_0^2 1];
A = B\C;
A_4 = subs(A,[gamma beta dt],[0.5 0.25 0.01])
```

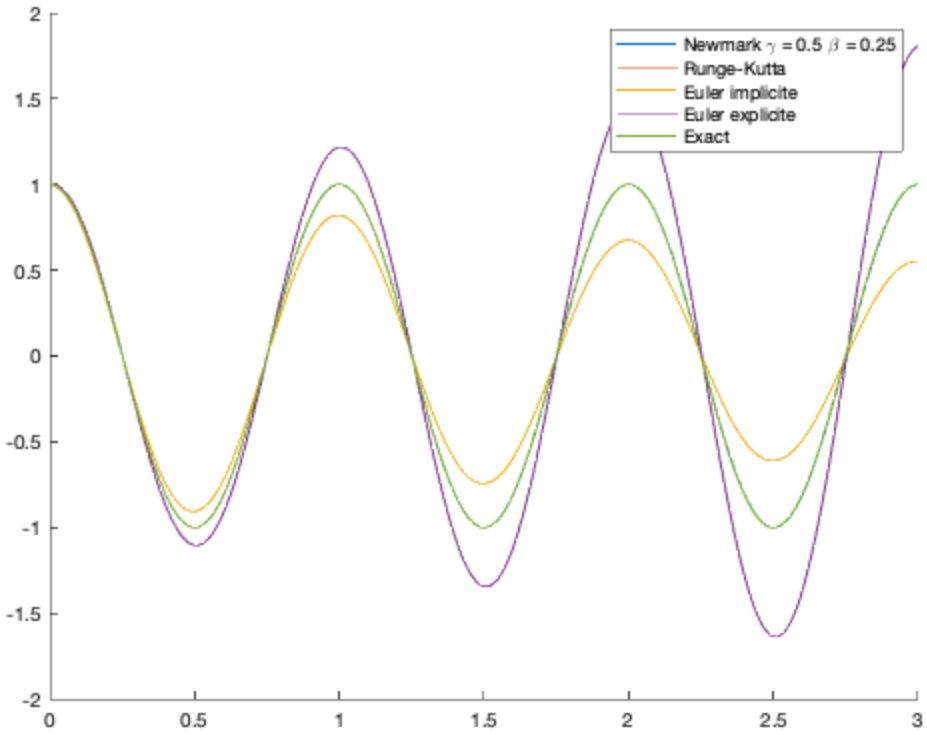
```
A_4 =

[  562394344087523997/563505562755100003,
 562949534213120/563505562755100003]
[ -222243733515201200/563505562755100003,
 562394344087523997/563505562755100003]
```

5.1.2

On peut voir que le schéma de Newmark gamma = 0,5 beta = 0,25 est aussi précis et aussi stable que Runge-Kutta. (Dans la figure, la ligne de Newmark et la ligne de Runge-Kutta coïncident presque complètement avec la ligne de la solution exacte, donc c'est difficile de les distinguer)

```
delta_t = 0.01;
t = t_start:delta_t:T0;
U_4 = iterate(A_4, t);
```

```
q_num_4 = U_4(1,:);
dq_num_4 = U_4(2,:);
clf;
hold on;
plot(t, q_num_4);
plot(t, q_num_3);
plot(t, q_num_2);
plot(t, q_num);
plot(t, eval(q_exact));
legend('Newmark \gamma = 0.5 \beta = 0.25','Runge-Kutta','Euler
 implicite','Euler explicite','Exact')
```

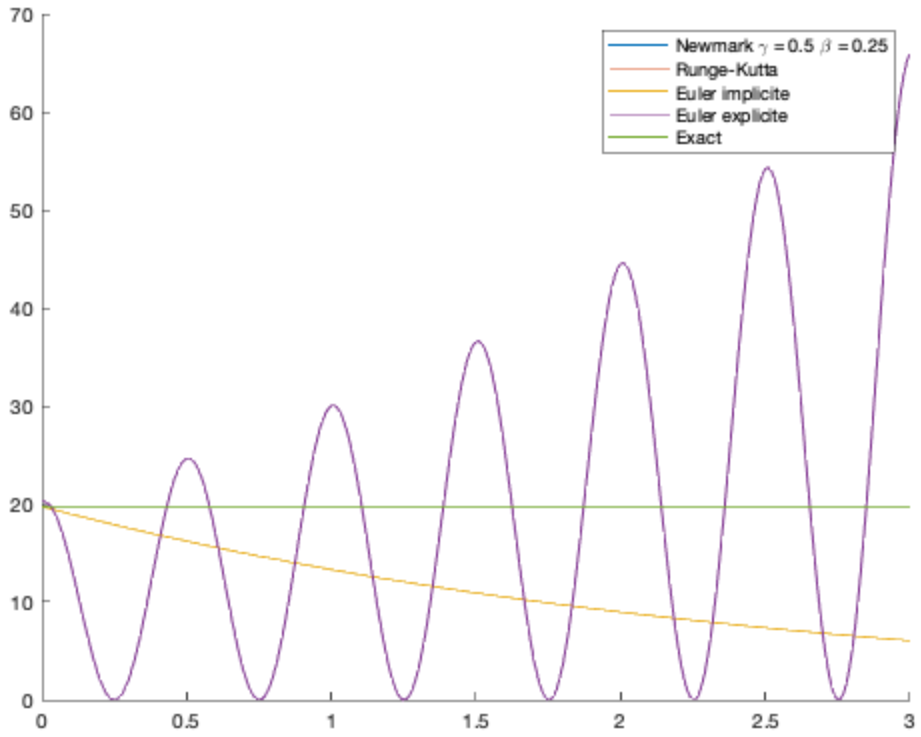


5.1.3

On peut voir que le schéma de Newmark gamma = 0,5 beta = 0,25 est plus précis et plus stable que les autres schémas.

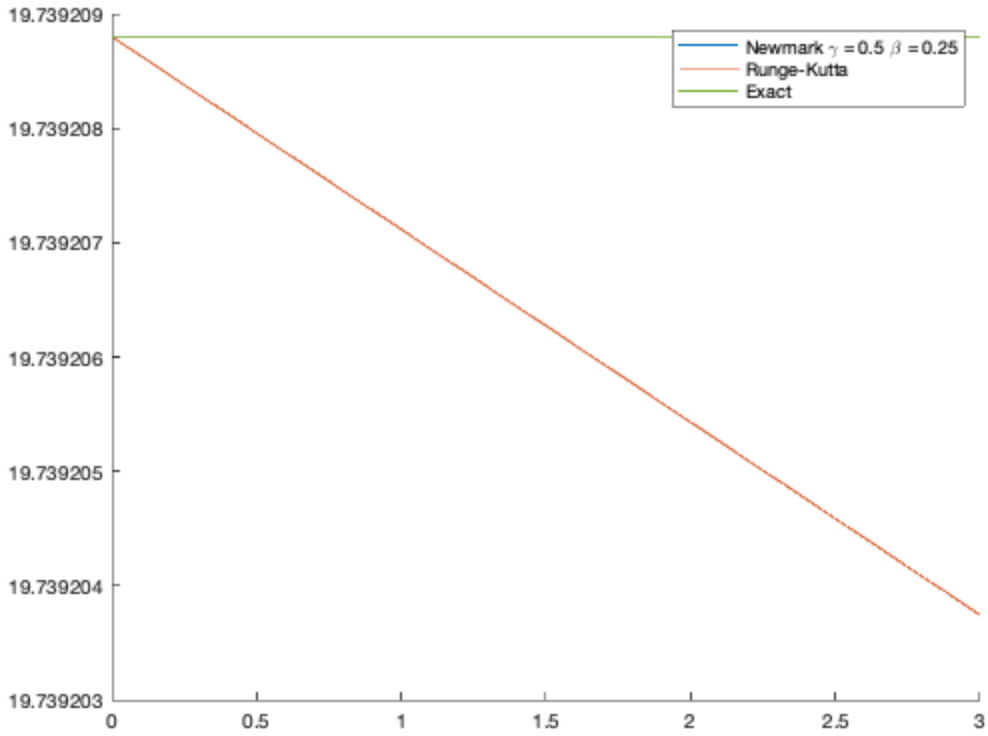
```
E_star_num_4 = 1/2 * ( dq_num_4.^2 + omega_0.^2  .* q_num_4.^2);
clf;
hold on;
plot(t, E_star_num_4);
plot(t, E_star_num_3);
plot(t, E_star_num_2);
plot(t, E_star_num);
plot(t, eval(E_star_exact));
```

```
legend('Newmark \gamma = 0.5 \beta = 0.25','Runge-Kutta','Euler
 implicite','Euler explicite','Exact')
```



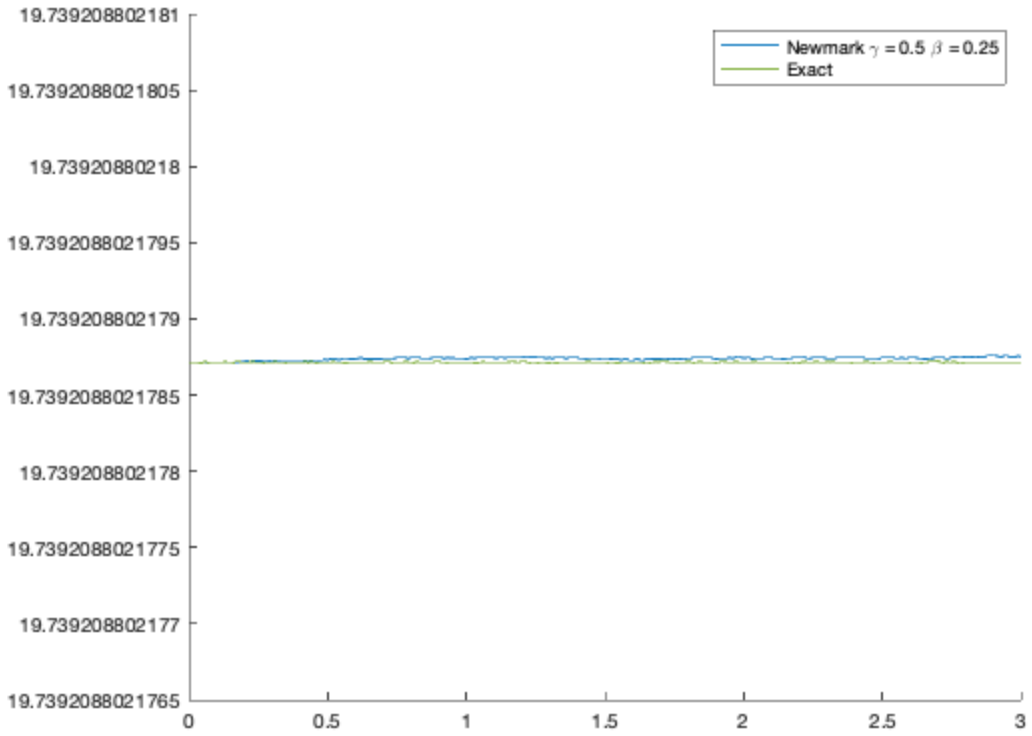
On supprime les autres lignes pour mieux comparer la ligne de Newmark avec la ligne de Runge-Kutta et la ligne de E_star exacte.

```
children = get(gca, 'children');
delete(children(2));
delete(children(3));
```



On supprime la ligne de Runge-Kutta pour mieux comparer la ligne de Newmark avec la ligne de E_star exacte.

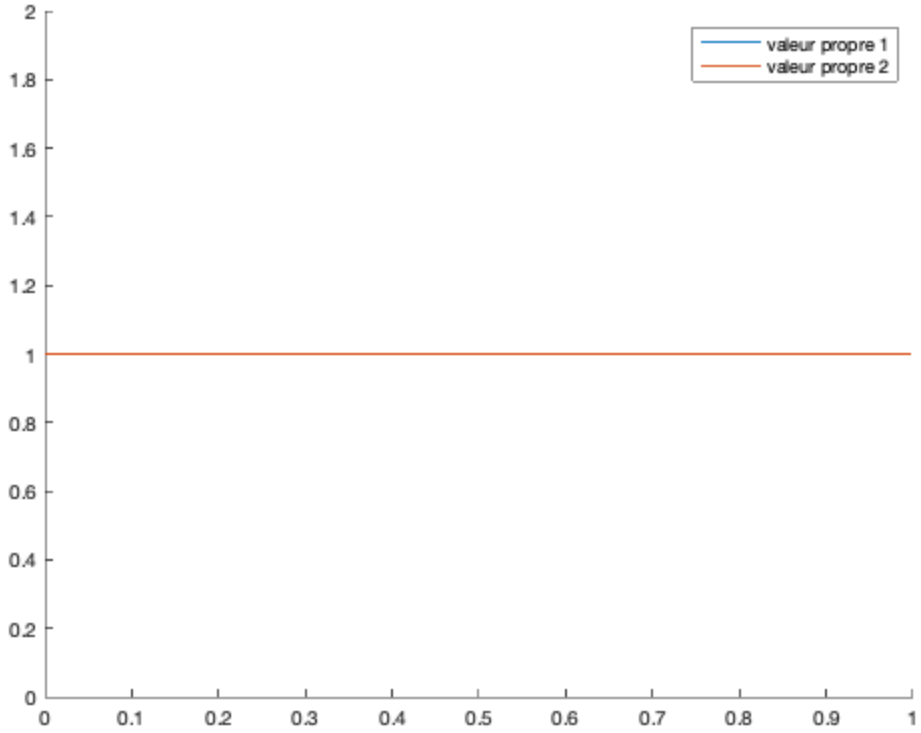
```
delete(children(4));
```



5.1.4

On peut voir que la module des valeurs propres est toujours 1, c'est-à- dire que ce schéma est inconditionnellement stable.

```
delta_ts = 0:0.01:1;
vp = [];
vp2 = [];
i=1;
for dt_i=delta_ts
    A_temp = subs(A,[gamma beta dt], [0.5 0.25 dt_i]);
    [~,d] = eig(A_temp);
    vp(i)=abs(d(1,1));
    vp2(i)=abs(d(2,2));
    i=i+1;
end
clf
hold on
plot(delta_ts,vp);
plot(delta_ts,vp2);
legend('valeur propre 1', 'valeur propre 2')
```



5.2.1

```
A_5 = subs(A,[gamma beta dt],[0.5 0 0.01])
```

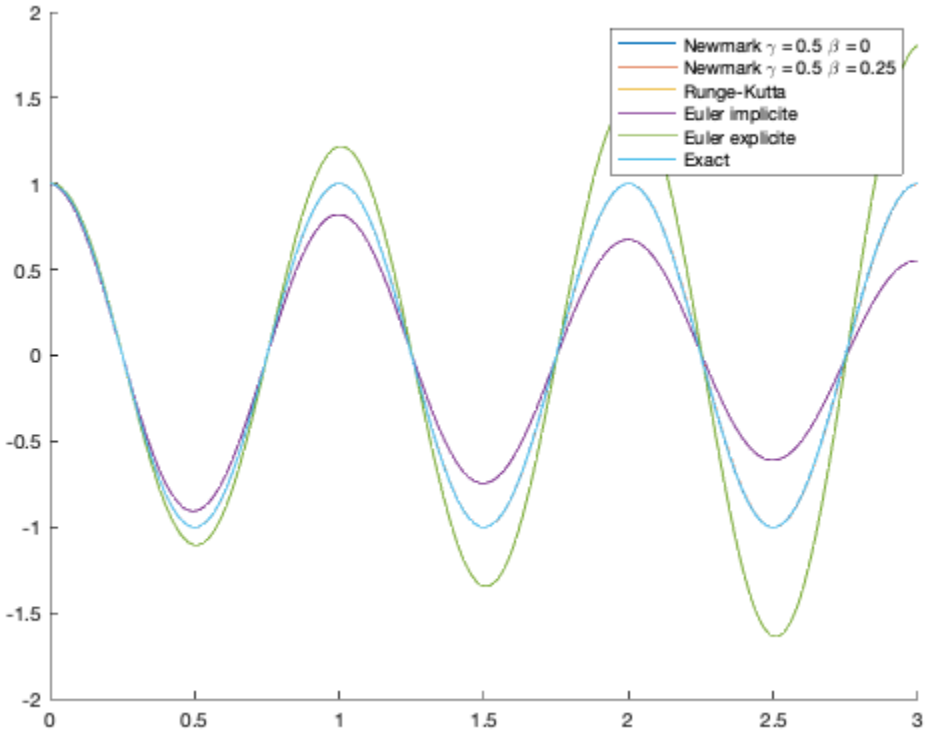
```
A_5 =

[
  280919367376867997/281474976710656000,
   1/100]
[ -312471546844610131918665173207991/792281625142643375935439503360000,
 280919367376867997/281474976710656000]
```

5.2.2

On peut voir que le schéma de Newmark gamma = 0,5 beta = 0 est aussi précis et aussi stable que Runge-Kutta et que Newmark gamme = 0,5 beta = 0,25. (Dans la figure, les lignes de Newmark et la ligne de Runge-Kutta coïncident presque complètement avec la ligne de la solution exacte, donc c'est difficile de les distinguer)

```
delta_t = 0.01;
t = t_start:delta_t:T0;
U_5 = iterate(A_5, t);
q_num_5 = U_5(1,:);
```

```
dq_num_5 = U_5(2,:);
clf;
hold on;
plot(t, q_num_5);
plot(t, q_num_4);
plot(t, q_num_3);
plot(t, q_num_2);
plot(t, q_num);
plot(t, eval(q_exact));
legend('Newmark \gamma = 0.5 \beta = 0', 'Newmark \gamma = 0.5 \beta =
 0.25','Runge-Kutta','Euler implicite','Euler explicite','Exact')
```

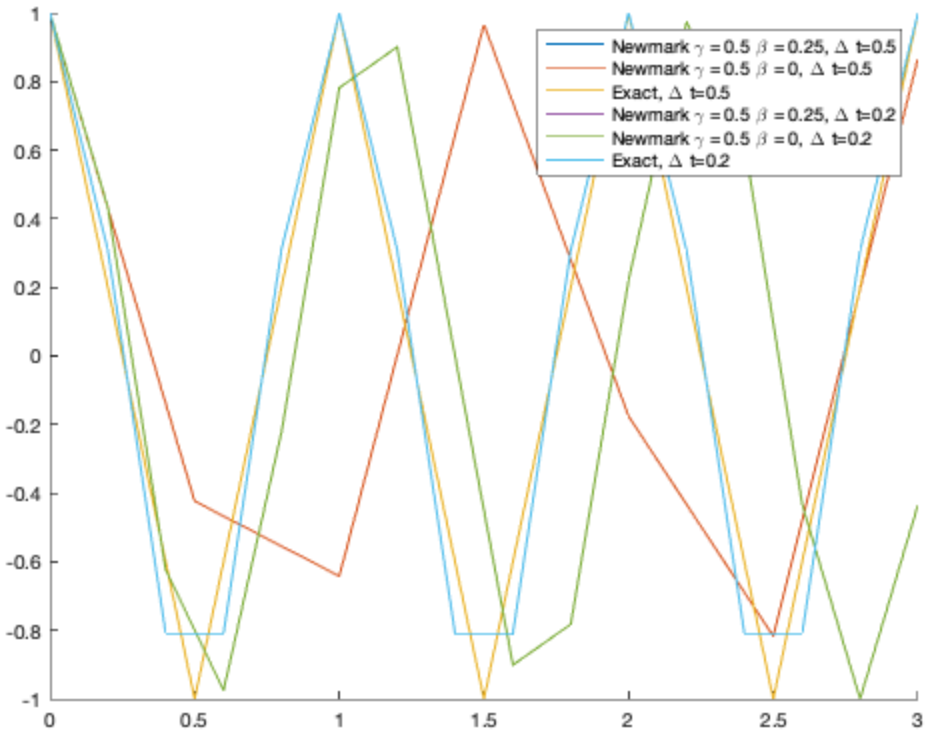


5.2.3

On peut voir que les solutions numériques de Newmark ne sont pas précis ni en amplitude ni en période. Cela veut dire que la précision d'une solution d'une fonction trigonométrique ne peut pas être garantie si le pas de temps n'est pas beaucoup plus petit que la moitié de la période de cette fonction.

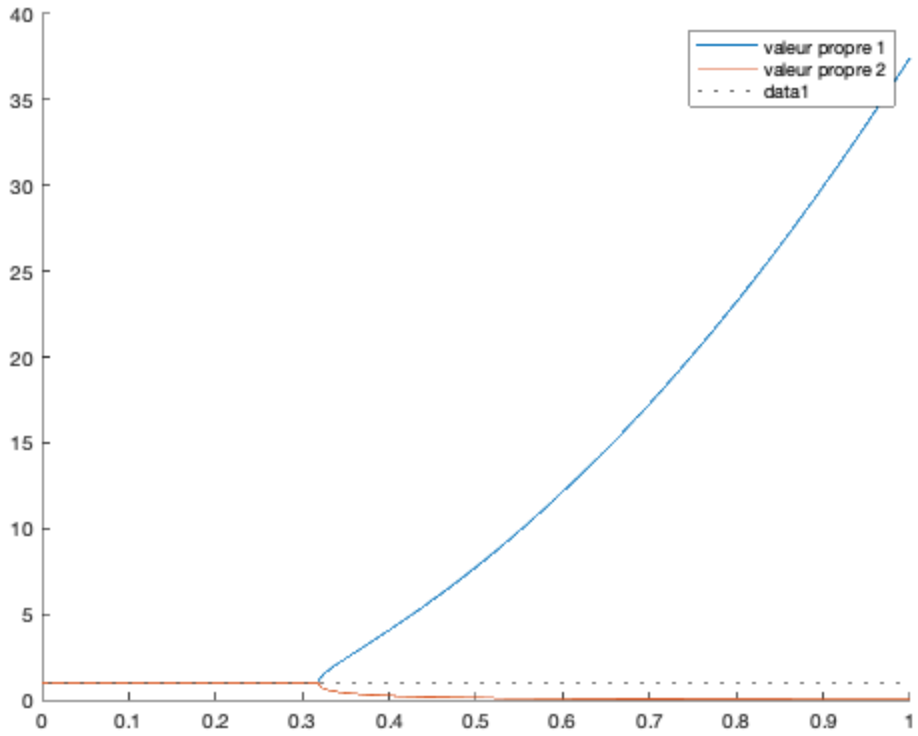
```
clf
hold on
for dt_i=[0.5,0.2]
    t = t_start:dt_i:T0;
    U_exp = iterate(subs(A,[gamma beta dt], [0.5 0.25 dt_i]), t);
    q_num_exp = U_exp(1,:);
    dq_num_exp = U_exp(2,:);
    U_imp = iterate(subs(A,[gamma beta dt], [0.5 0 dt_i]), t);
```

```
    q_num_imp = U_exp(1,:);
    dq_num_imp = U_exp(2,:);
    plot(t, q_num_exp, 'DisplayName', ['Newmark \gamma = 0.5 \beta =
 0.25, \Delta t=' num2str(dt_i)]);
    plot(t, q_num_imp, 'DisplayName', ['Newmark \gamma = 0.5 \beta =
 0, \Delta t=' num2str(dt_i)]);
    plot(t, eval(q_exact),'DisplayName',['Exact, \Delta t='
 num2str(dt_i)]);
end
legend('show')
```



5.2.4

```
[~,d] = eig(subs(A, [gamma beta], [0.5 0]));
delta_ts = 0:0.001:1;
vp1 = abs(d(1,1));
vp2 = abs(d(2,2));
vp1_num = subs(vp1,dt,delta_ts);
vp2_num = subs(vp2,dt,delta_ts);
clf
hold on
plot(delta_ts,vp1_num,delta_ts,vp2_num);
legend('valeur propre 1', 'valeur propre 2')
yline(1,':');
```



On peut voir de la figure le point du pas de temps critique

```
for i=length(vp1_num):-1:1
    vp = vp1_num(i);
    if vp <= 1
        temps_critique = delta_ts(i)
        break
    end
end
alpha = temps_critique*omega_0/2
```

```
temps_critique =

    0.3180


alpha =

    0.9990
```

Fonctions

```
function [U] = iterate(A,t)
%ITERATE U_j+1 = A*U_j, for j in t
```

```
i = 0;
U = [];
for t_i=t
    if i==0
        U(:,1) = [1;0];
    else
        U(:,i+1)=A*U(:,i);
    end
    i=i+1;
end

end

function [dU] = f(U,t,omega_0)
dU=zeros(2,1);
dU(1)=U(2);
dU(2)=-omega_0^2*U(1);
end
```

Published with MATLAB® R2019b