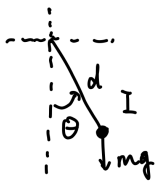
Table of Contents

TD Mécanique numérique

```
% 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 :

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

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

linéariser

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

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

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

Discrétisation :

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

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

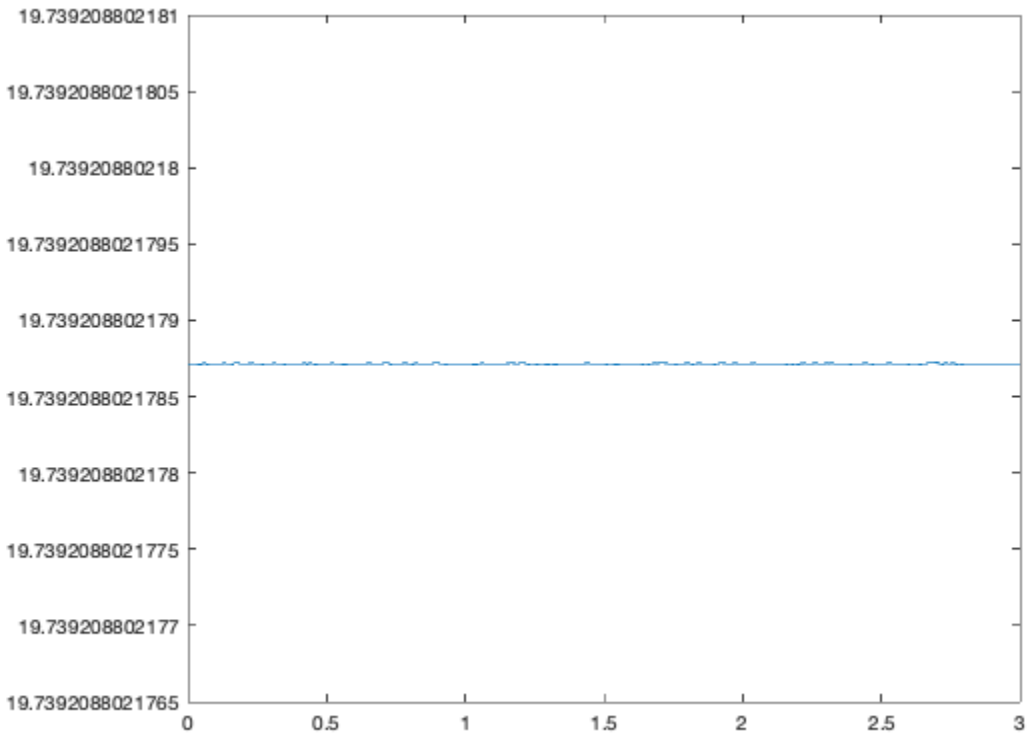
```
omega_0 = 2*pi;
q_exact = dsolve('D2q+omega_0^2*q=0','q(0)=1,Dq(0)=0')
```

```
Warning: Support of character vectors and strings will be removed in a
 future
release. Use sym objects to define differential equations instead.

q_exact =

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

1.2

```
E_star_exact = 1/2 * ( diff(q_exact)^2 + omega_0^2  * q_exact^2)
plot(t, eval(E_star_exact));
hold off

% On peut voir que E_star est constant.
```

```
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

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

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

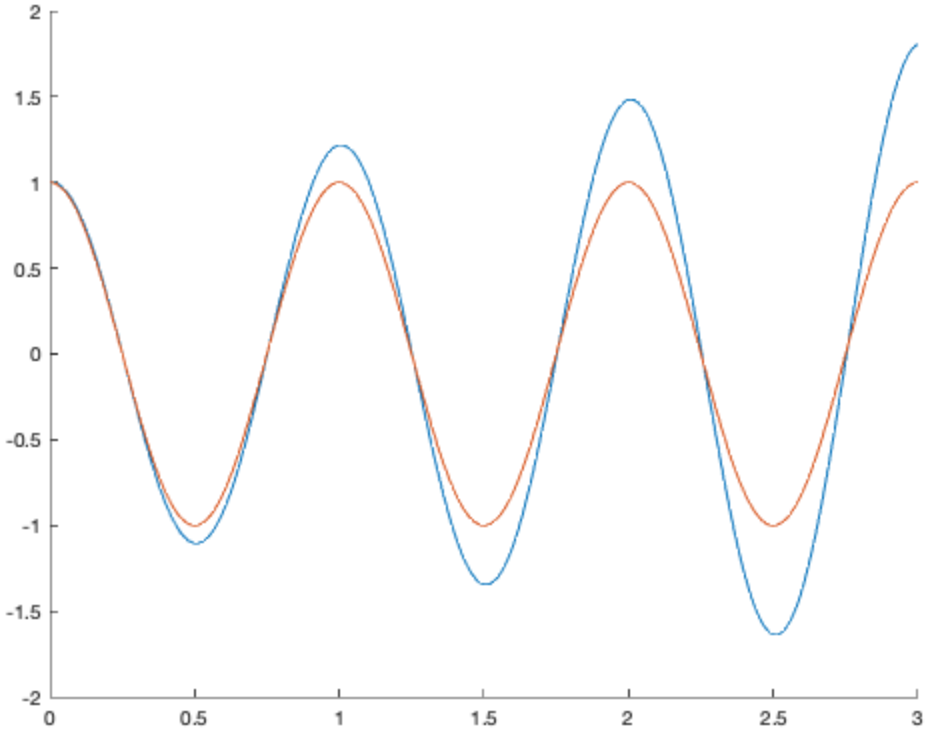
$$\text{et } \dot{q}_{j+1} = \Delta t \times \ddot{q}_j + \dot{q}_j$$

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

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

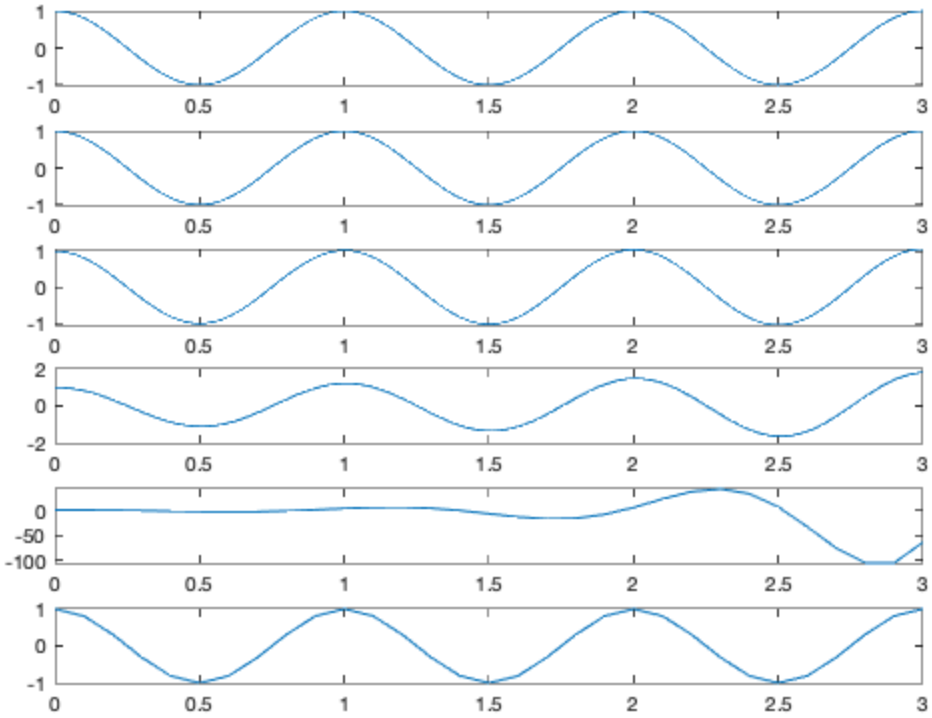
2.2

```
% 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));


A =

    1.0000    0.0100
   -0.3948    1.0000
```



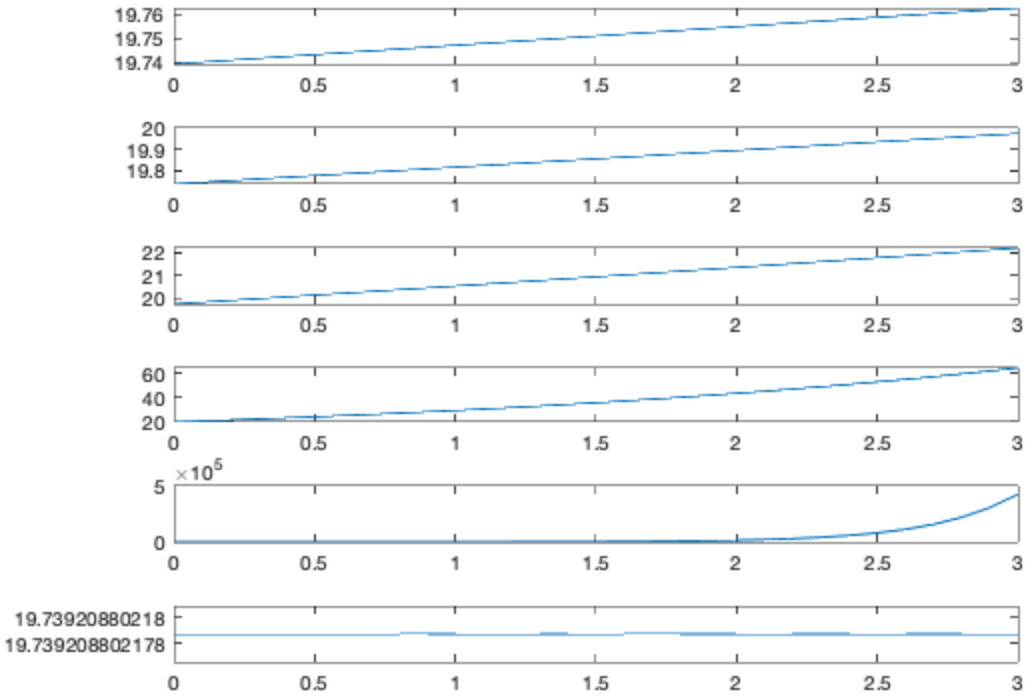
2.3

```
% Ce schéma d'intégration est divergente, voir 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 off;
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 = U(1,:);
    dq_num = U(2,:);
    subplot(l+1,1,i);
    plot(t, q_num);
end
subplot(l+1,1,i+1);
plot(t, eval(q_exact));
```



2.4

```
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 = U(1,:);
    dq_num = U(2,:);
    E_star_num = 1/2 * ( dq_num.^2 + omega_0.^2  .* q_num.^2);
    subplot(l+1,1,i);
    plot(t, E_star_num);
end
subplot(l+1,1,i+1);
plot(t, eval(E_star_exact));

% E_star est divergent.
% Plus le pas de temps est petit, plus la divergence est lente.
```



2.5

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

% 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.


ans =

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

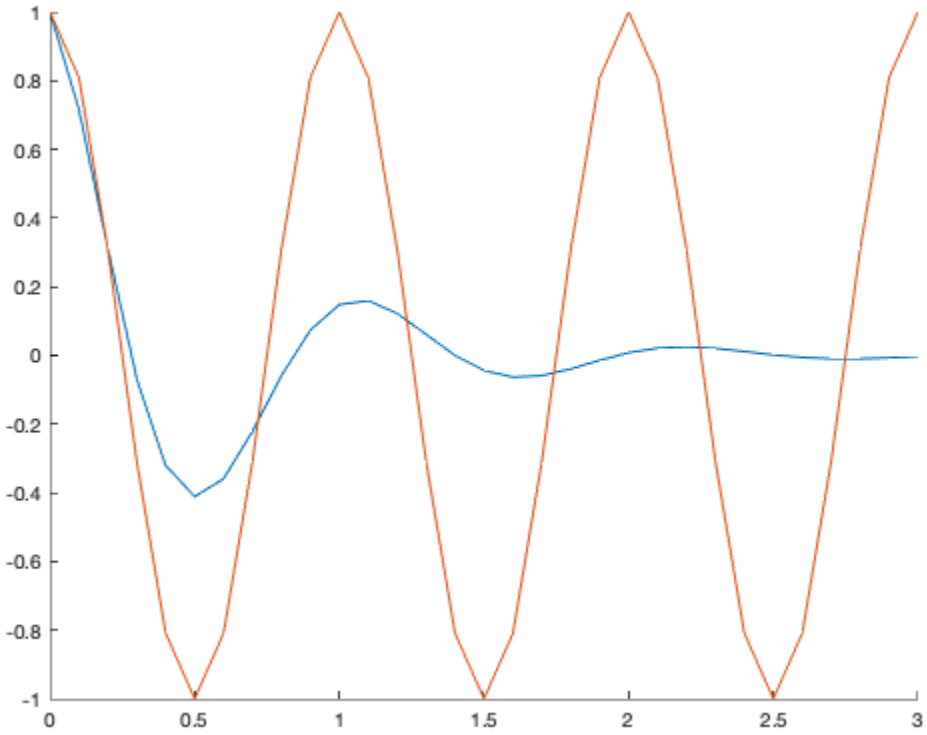
3.1

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

    0.7170    0.0717
   -2.8304    0.7170
```

3.2

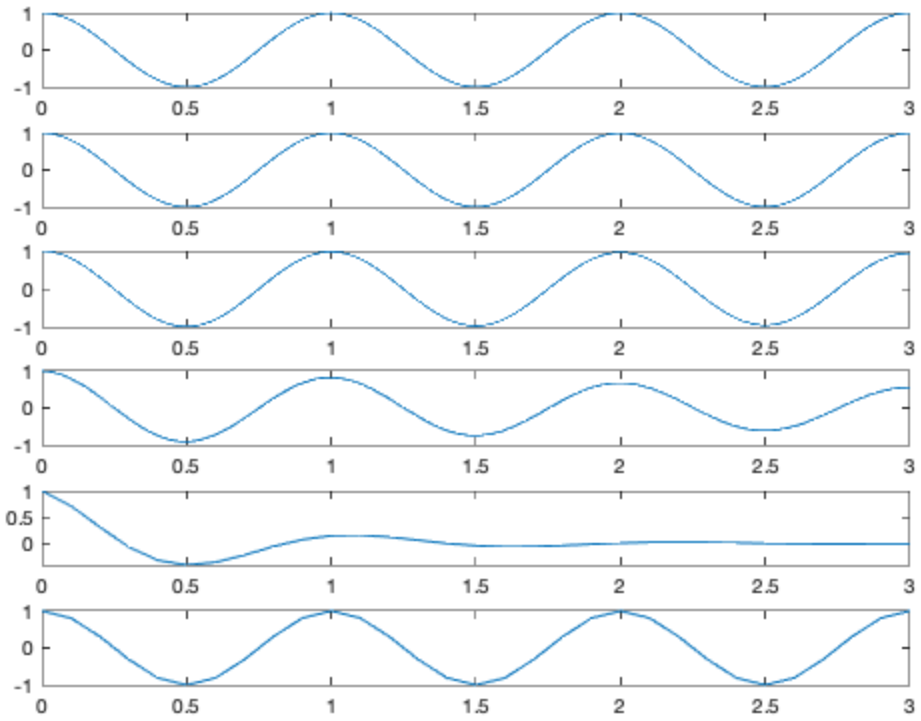
```
clf;
hold on;
q_num_2 = U_2(1,:);
dq_num_2 = U_2(2,:);
plot(t, q_num_2);
plot(t, eval(q_exact));
```



3.3

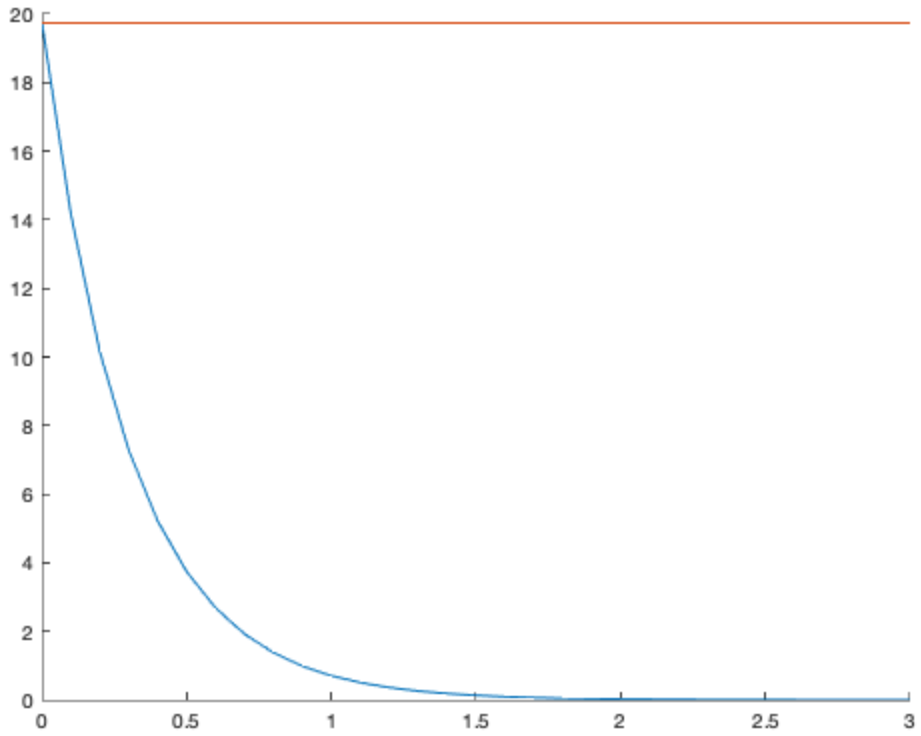
```
clf;
hold off;
l = numel(delta_ts);
for i = 1:l
    delta_t = delta_ts(i);
    t = t_start:delta_t:T0;
```

```
    A_2 = 1/(1+omega_0^2*delta_t^2) * [1 delta_t; -omega_0^2*delta_t
 1];
    U_2 = iterate(A_2, t);
    q_num_2 = U_2(1,:);
    dq_num_2 = U_2(2,:);
    subplot(l+1,1,i);
    plot(t, q_num_2);
end
subplot(l+1,1,i+1);
plot(t, eval(q_exact));
```



3.4

```
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_2);
plot(t, eval(E_star_exact));
```



3.5

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

% 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 TODO.


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.

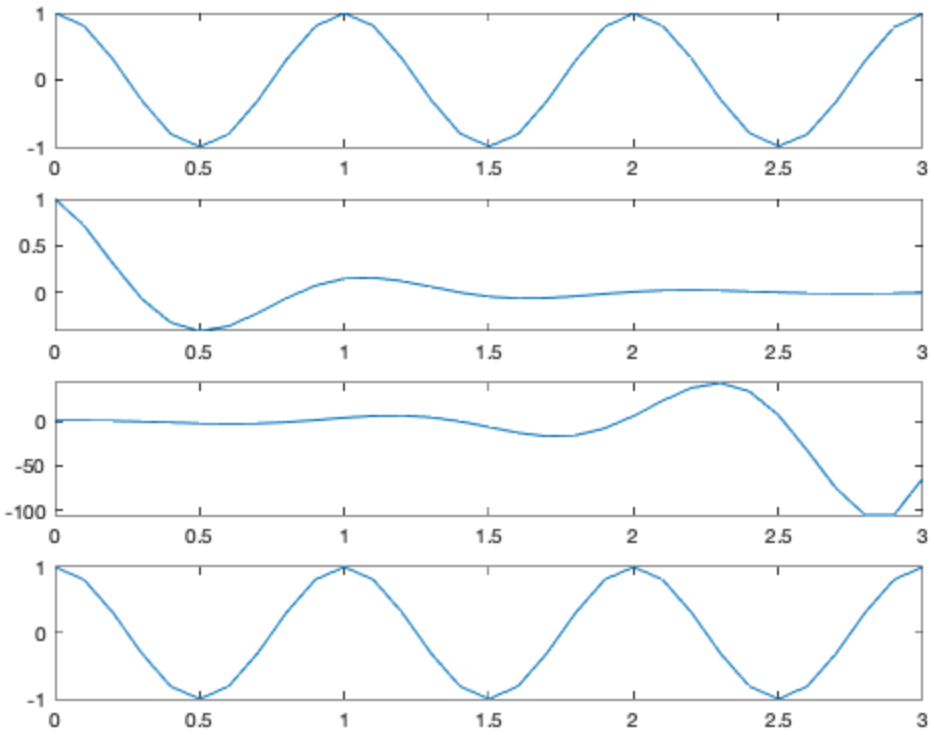
4.2

```
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

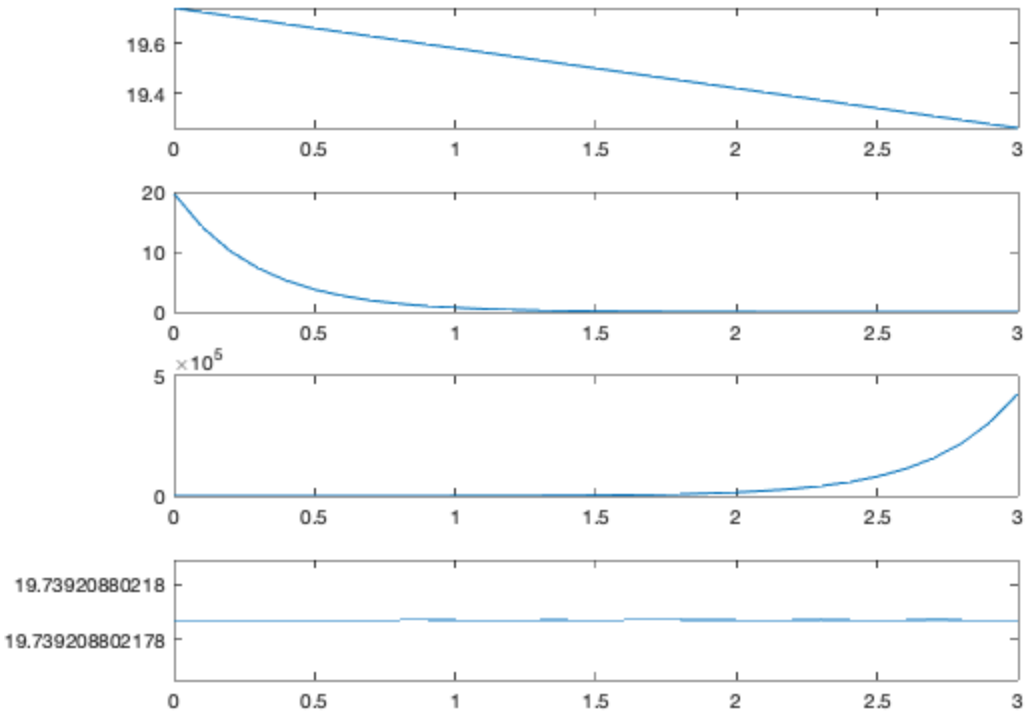
```
clf;
hold off;
q_num_3 = U_3(1,:);
dq_num_3 = U_3(2,:);
subplot(4,1,1);
plot(t, q_num_3);
subplot(4,1,2);
plot(t, q_num_2);
subplot(4,1,3);
plot(t, q_num);
subplot(4,1,4);
```

```
plot(t, eval(q_exact));

% On peut voir que RK4 est plus précis et plus stable que les méthodes
% précédentes.
```



4.4

```
E_star_num_3 = 1/2 * ( dq_num_3.^2 + omega_0.^2  .* q_num_3.^2);
clf;
hold off;
subplot(4,1,1);
plot(t, E_star_num_3);
subplot(4,1,2);
plot(t, E_star_num_2);
subplot(4,1,3);
plot(t, E_star_num);
subplot(4,1,4);
plot(t, eval(E_star_exact));

% On peut voir que RK4 est plus précis et plus stable que les méthodes
% précédentes.
```



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