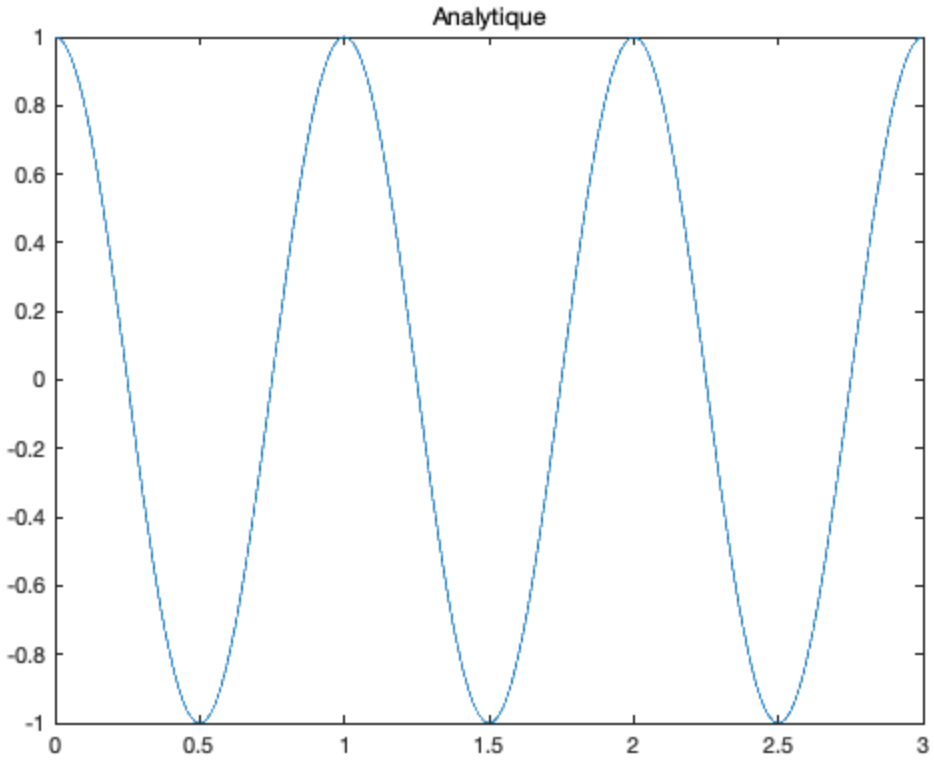
Table of Contents

Q1.1 1
Q1.2 2
Q2.1 2
Q2.2 2
Q2.3 3
Q2.4 4
Q2.5 5
Q3.1 6
Q3.2 6
Q3.3 7
Q3.4 8
Q3.5 9
Q4.1 10
Q4.2 10
Q4.3 11
Q4.4 12
Q5.1.1 13
Q5.1.2 14
Q5.1.3 15
Q5.1.4 16
Q5.2.1 17
Q5.2.2 18
Q5.2.3 18
Q5.2.4 20

Q1.1

```
q0 = 1;
dq0 = 0;
w0 = 2 * pi;
T0 = 3;

% On peut trouver que q = A * sin(w0 * t) + B * cos(w0 * t).
% D'après les conditions initiales, on peut trouver que A = 0 et B =
 1.
% Alors q = cos(2 * pi * t).

x1 = linspace(0, T0, 3000);
q1 = cos(2 * pi * x1);
plot(x1, q1);
title('Analytique');
```



Q1.2

```
% On trouve que E* = 2 * pi * pi.
% C'est égale à 0.5 * w0 * w0, une constante.
% E* ne dépand pas de t.
```

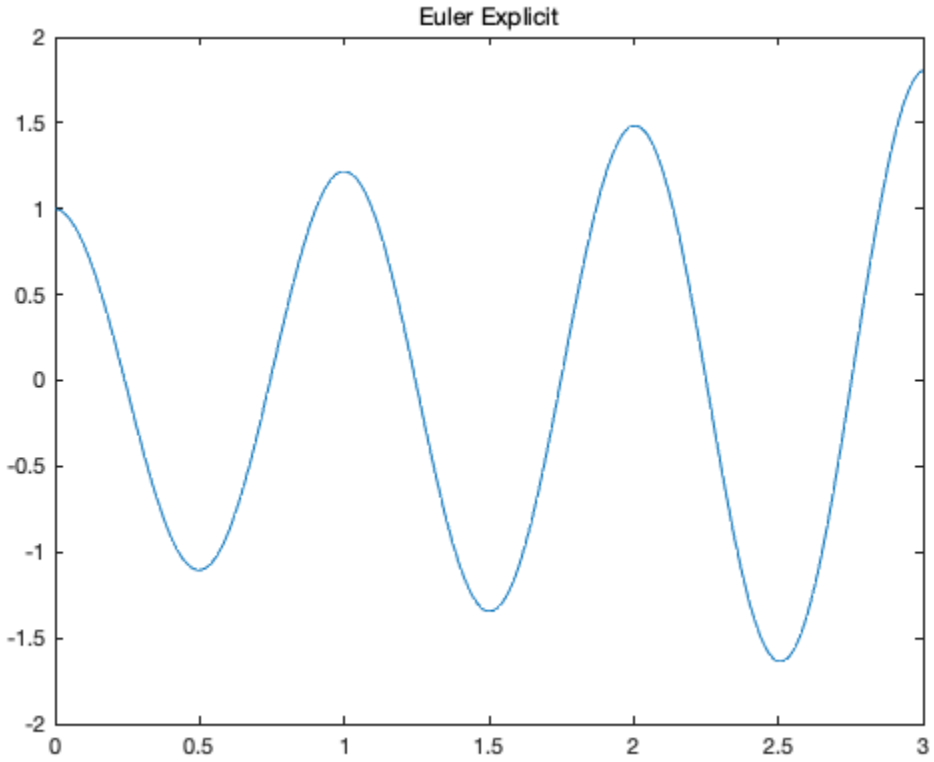
Q2.1

```
% On peut utiliser la relation (1) et remplace d2q par - w0 * w0 * q.
% Alors, on trouve le résultat.
```

Q2.2

```
dt = 0.01;
U = [q0; dq0];
q2 = [];
E2 = [];
A = [1, dt; - w0 * w0 * dt, 1];
for t = linspace(0, T0, 300)
U = A* U;
q2 = [q2, U(1)];
E2 = [E2, 0.5 * (U(2) * U(2) + w0 * w0 * U(1) * U(1))];
end
```

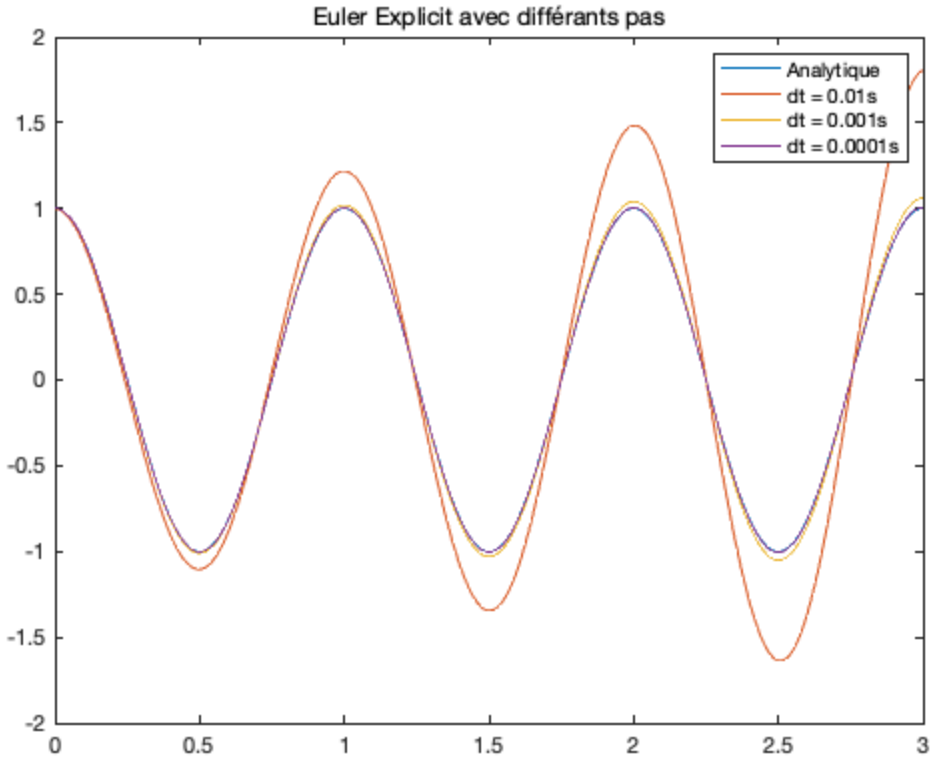
```
x2 = linspace(0, T0, 300);
plot(x2, q2);
title('Euler Explicit');
```



Q2.3

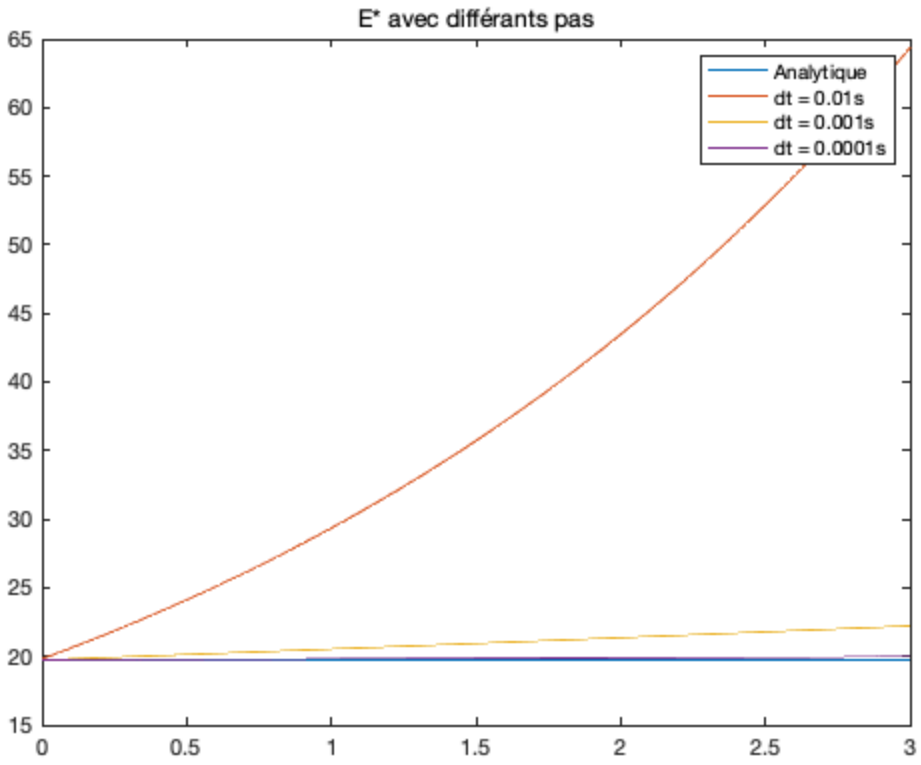
```
% On change le pas dt, et on re?ois plusieurs images.
% Quand le pas est plus petit, l'image est plus precise.

% Le pas est 0.001s
dt = 0.001;
U = [q0; dq0];
q3 = [];
E3 = [];
A = [1, dt; - w0 * w0 * dt, 1];
for t = linspace(0, T0, 3000)
U = A* U;
q3 = [q3, U(1)];
E3 = [E3, 0.5 * (U(2) * U(2) + w0 * w0 * U(1) * U(1))];
end

% Le pas est 0.0001s
dt = 0.0001;
U = [q0; dq0];
q4 = [];
E4 = [];
```

```
A = [1, dt; - w0 * w0 * dt, 1];
for t = linspace(0, T0, 30000)
U = A* U;
q4 = [q4, U(1)];
E4 = [E4, 0.5 * (U(2) * U(2) + w0 * w0 * U(1) * U(1))];
end

x3 = linspace(0, T0, 3000);
x4 = linspace(0, T0, 30000);
plot(x1, q1, x2, q2, x3, q3, x4, q4);
title('Euler Explicit avec différants pas');
legend('Analytique', 'dt = 0.01s', 'dt = 0.001s', 'dt = 0.0001s');
```



Q2.4

```
% On ajoute la calculation de E* dans Q2.3.
% Et on plot les E* de différant pas.

E1 = ones(1, 3000) * 2 * pi * pi;
plot(x1, E1, x2, E2, x3, E3, x4, E4);
title('E* avec différants pas');
legend('Analytique', 'dt = 0.01s', 'dt = 0.001s', 'dt = 0.0001s');

% On trouve que le pas varie très grave E*.
% Et quand le pas est plus petit, la précision est plus bonne.
```



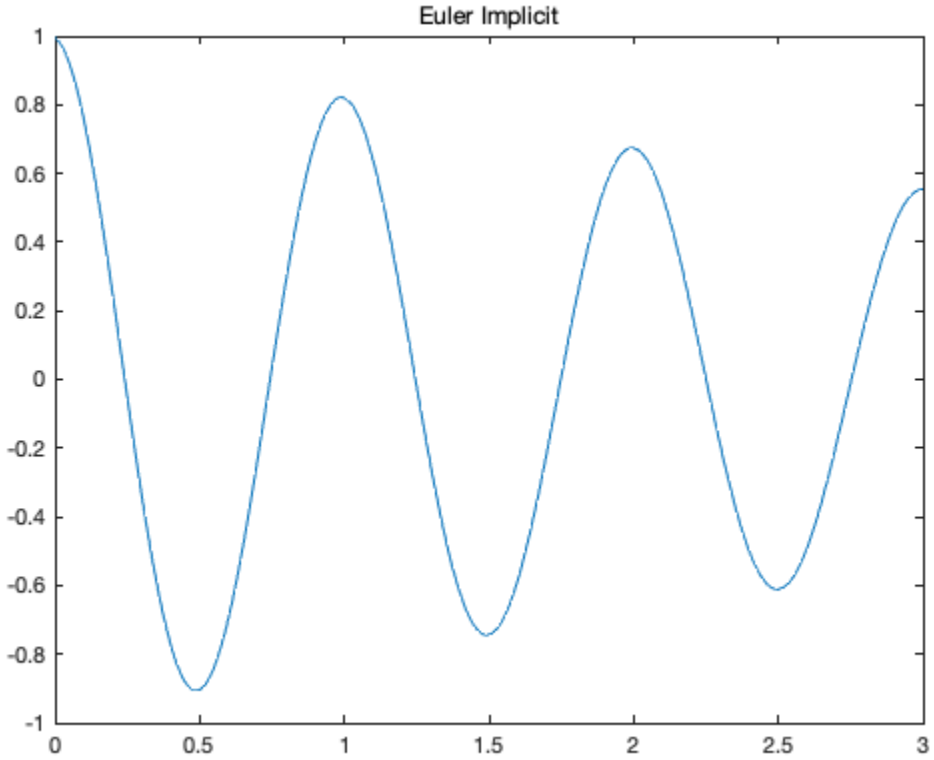
Q2.5

```
dt = 0.001; % ou 0.001 ou 0.0001
A = [1, dt; - w0 * w0 * dt, 1];
vp = eig(A);
abs(vp);

% Quand le pas est 0.01s,
% vp =
%    1.0000 + 0.0628i
%    1.0000 - 0.0628i
% abs(vp) = 1.0020
% Quand les pas est 0.001s,
% vp =
%    1.0000 + 0.0063i
%    1.0000 - 0.0063i
% abs(vp) = 1.0000
% Quand le pas est 0.0001s,
% vp =
%    1.0000 + 0.0006i
%    1.0000 - 0.0006i
% abs(vp) = 1.0000
% Matlab fait une simulation, alors le module de valeur propre est
 égale à 1.
```

```
% En fait, le module de valeur propre est supérieure à 1, c'est
 instable.
% Et quand le pas est plus petit, le module de valeur propre est plus
 petit et la divergence est plus lentement.
```

Q3.1

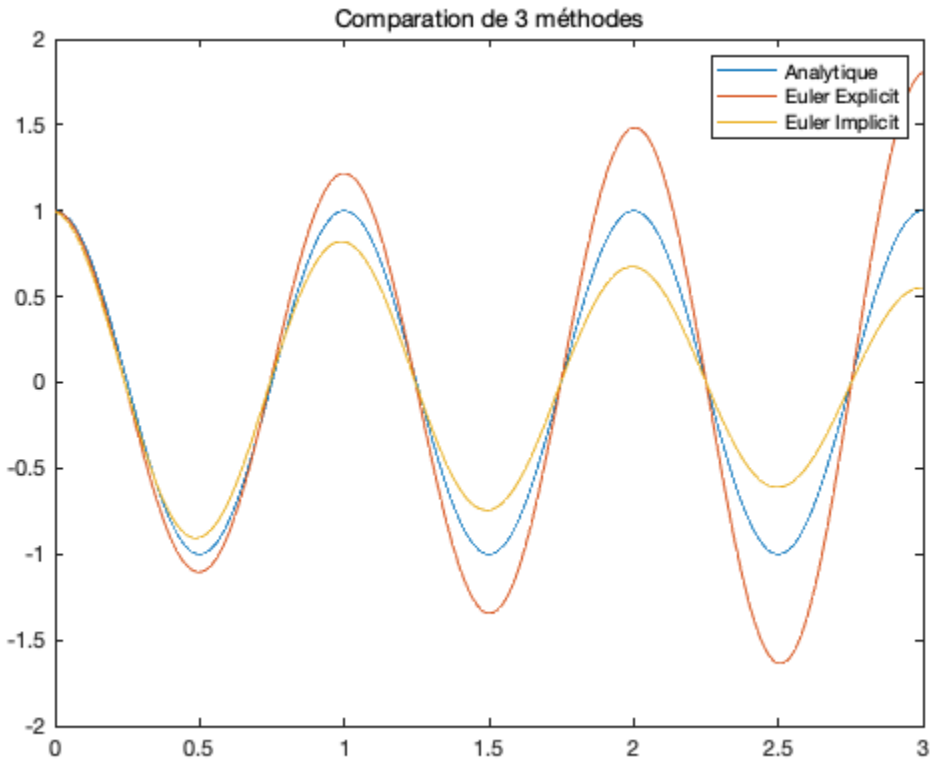
```
dt = 0.01;
U = [q0; dq0];
q5 = [];
E5 = [];
A = [1, dt; - w0 * w0 * dt, 1] / (1 + w0 * w0 * dt * dt);
for t = linspace(0, T0, 300)
U = A* U;
q5 = [q5, U(1)];
E5 = [E5, 0.5 * (U(2) * U(2) + w0 * w0 * U(1) * U(1))];
end

x5 = linspace(0, T0, 300);
plot(x5, q5);
title('Euler Implicit');
```



Q3.2

```
plot(x1, q1, x2, q2, x5, q5);
```

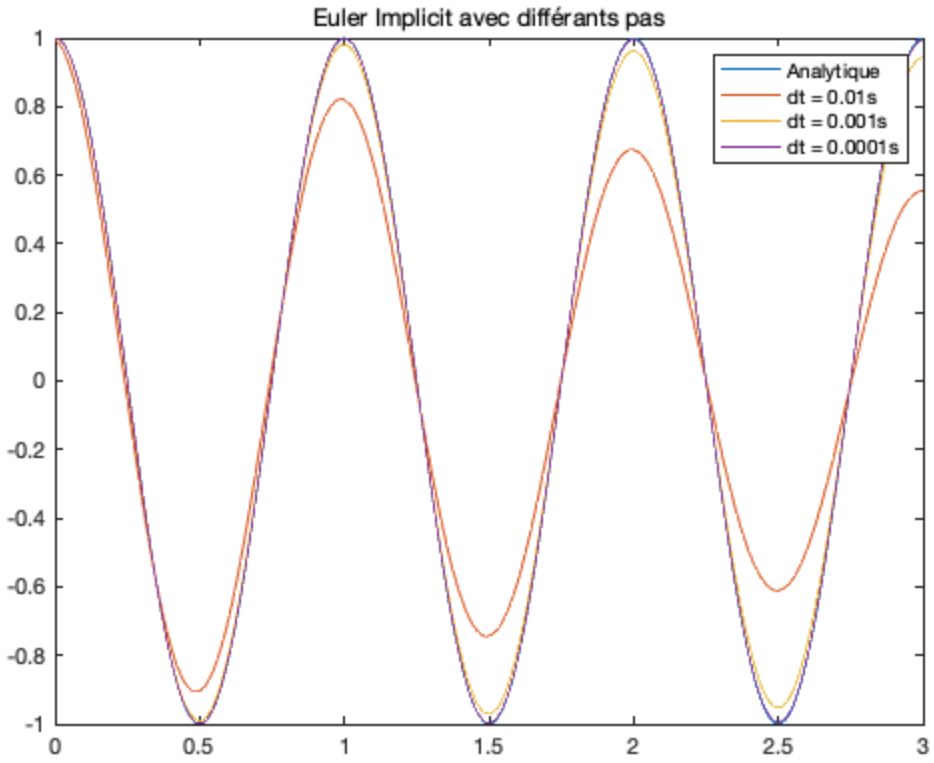
```
title('Comparation de 3 méthodes');
legend('Analytique', 'Euler Explicit', 'Euler Implicit');
```



Q3.3

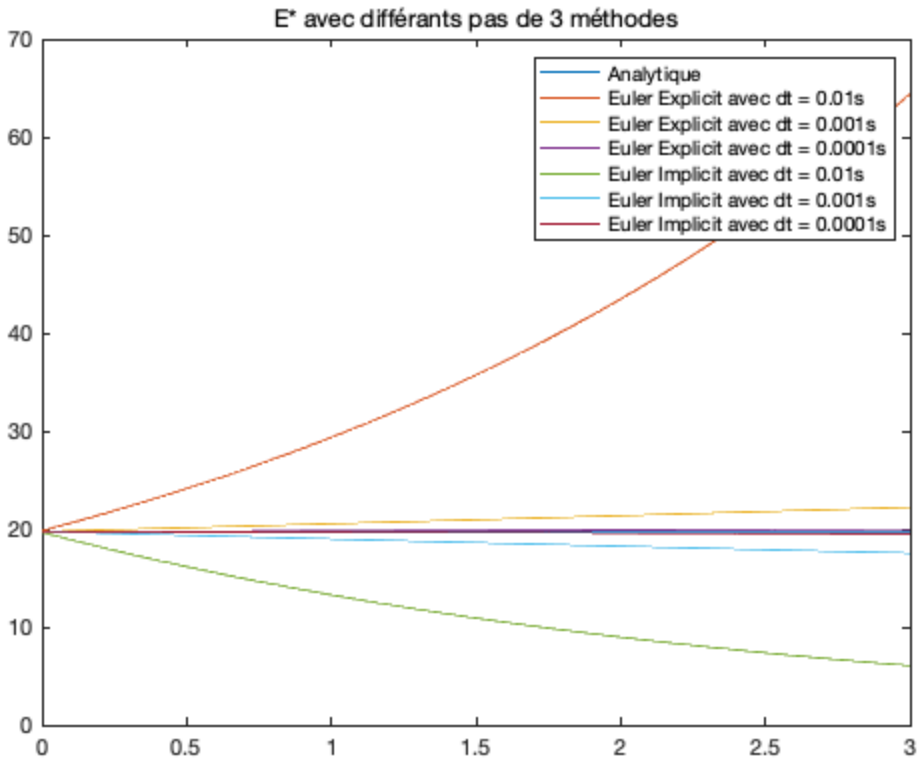
```
% Quand le pas est 0.001s
dt = 0.001;
U = [q0; dq0];
q6 = [];
E6 = [];
A = [1, dt; - w0 * w0 * dt, 1] / (1 + w0 * w0 * dt * dt);
for t = linspace(0, T0, 3000)
U = A* U;
q6 = [q6, U(1)];
E6 = [E6, 0.5 * (U(2) * U(2) + w0 * w0 * U(1) * U(1))];
end

% Quand le pas est 0.0001s
dt = 0.0001;

U = [q0; dq0];
q7 = [];
E7 = [];
A = [1, dt; - w0 * w0 * dt, 1] / (1 + w0 * w0 * dt * dt);
for t = linspace(0, T0, 30000)
U = A* U;
```

```
q7 = [q7, U(1)];
E7 = [E7, 0.5 * (U(2) * U(2) + w0 * w0 * U(1) * U(1))];
end

x6 = linspace(0, T0, 3000);
x7 = linspace(0, T0, 30000);
plot(x1, q1, x5, q5, x6, q6, x7, q7);
title('Euler Implicit avec différants pas');
legend('Analytique', 'dt = 0.01s', 'dt = 0.001s', 'dt = 0.0001s');
```



Q3.4

```
plot(x1, E1, x2, E2, x3, E3, x4, E4, x5, E5, x6, E6, x7, E7);
title('E* avec différants pas de 3 méthodes');
legend('Analytique', 'Euler Explicit avec dt = 0.01s', 'Euler Explicit
 avec dt = 0.001s', ...
    'Euler Explicit avec dt = 0.0001s', 'Euler Implicit avec dt =
 0.01s', ...
    'Euler Implicit avec dt = 0.001s', 'Euler Implicit avec dt =
 0.0001s');
```



Q3.5

```
dt = 0.01; % ou 0.001 ou 0.0001
A = [1, dt; - w0 * w0 * dt, 1] / (1 + w0 * w0 * dt * dt);
vp = eig(A);
abs(vp);

% Quand le pas est 0.01s,
% vp =
%    0.9961 + 0.0626i
%    0.9961 - 0.0626i
% abs(vp) = 0.9980
% Quand les pas est 0.001s,
% vp =
%    1.0000 + 0.0063i
%    1.0000 - 0.0063i
% abs(vp) = 1.0000
% Quand le pas est 0.0001s,
% vp =
%    1.0000 + 0.0006i
%    1.0000 - 0.0006i
% abs(vp) = 1.0000
% Matlab fait une simulation, alors le module de valeur propre est
 égale à 1.
```

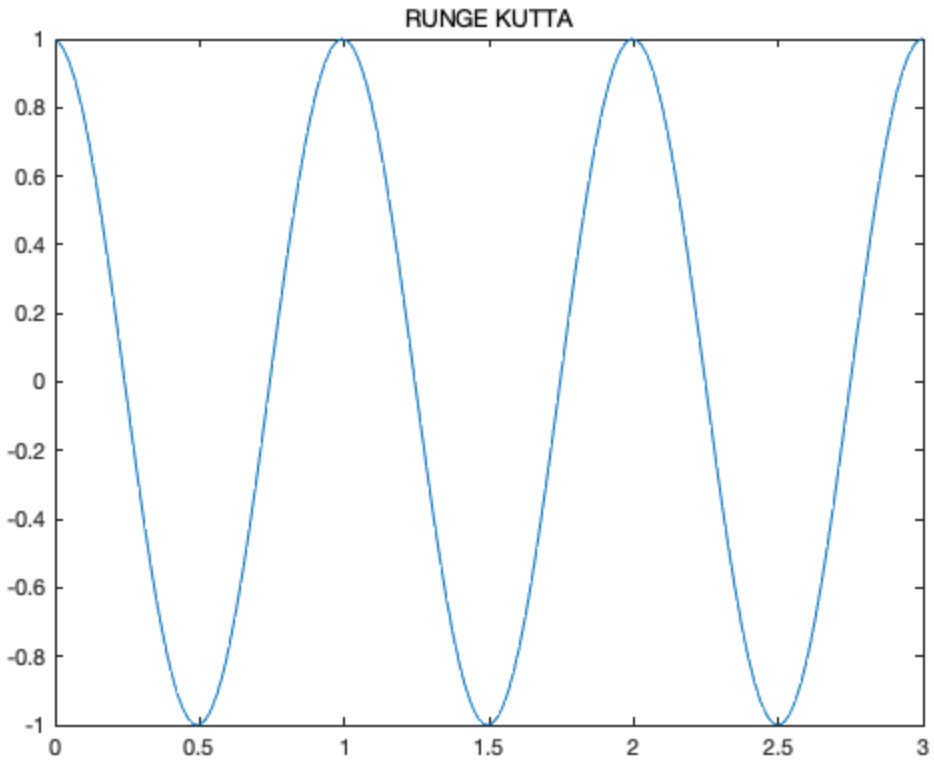
```
% En fait, le module de valeur propre est inférieure à 1, alors c'est
 stable.
% Et quand le pas est plus petit, le module de valeur propre est plus
 grand et la convergence est plus lentement.
```

Q4.1

```
% En utilisant Q = [q, dq], on a dQ = [0, 1; - w0 * w0, 0] * Q
```

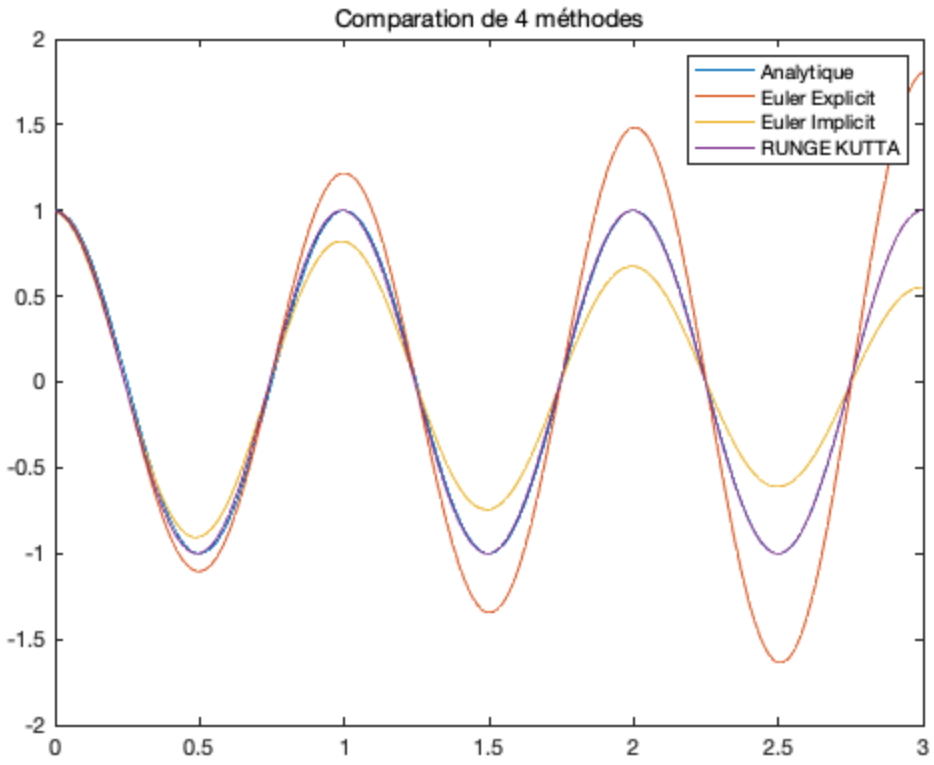
Q4.2

```
dt = 0.01;
Q = [q0; dq0];
q8 = [];
E8 = [];
A = [0, 1; - w0 * w0, 0];
for t = linspace(0, T0, 300)
K1 = A * Q;
K2 = A * (Q + K1 * dt / 2);
K3 = A * (Q + K2 * dt / 2);
K4 = A * (Q + K3 * dt);
K = (K1 + 2 * K2 + 2 * K3 + K4) / 6;
Q = Q + K * dt;
q8 = [q8, Q(1)];
E8 = [E8, 0.5 * (Q(2) * Q(2) + w0 * w0 * Q(1) * Q(1))];
end

x8 = linspace(0, T0, 300);
plot(x8, q8);
title('RUNGE KUTTA');
```



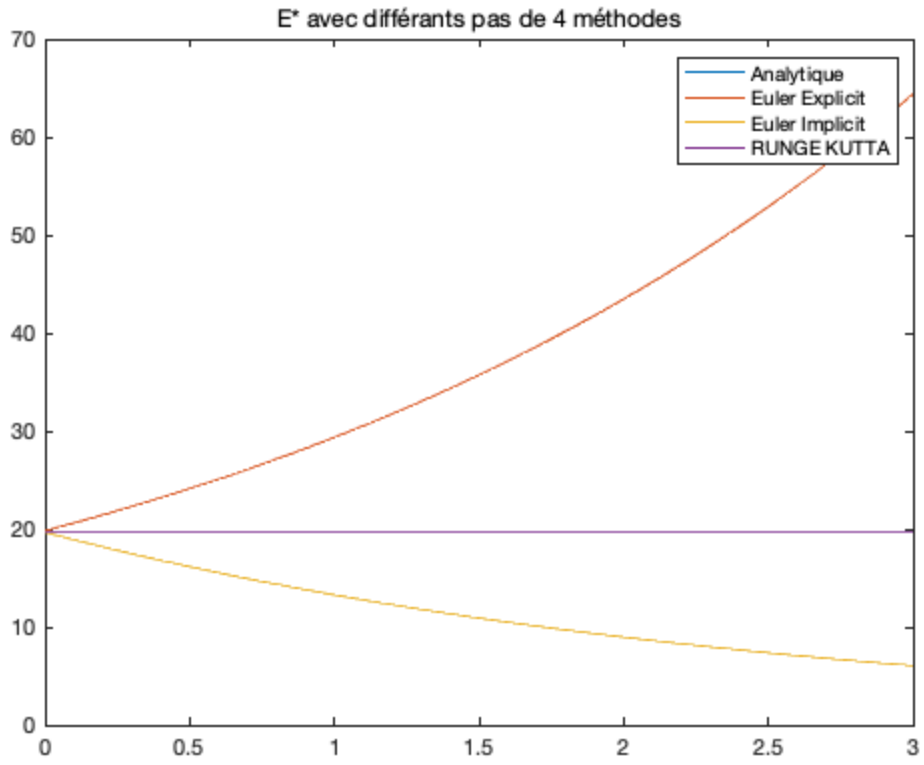
Q4.3

```
plot(x1, q1, x2, q2, x5, q5, x8, q8);
title('Comparation de 4 méthodes');
legend('Analytique', 'Euler Explicit', 'Euler Implicit', 'RUNGE
 KUTTA');

% La méthode de RUNGE KUTTA est plus précise que les autres.
```



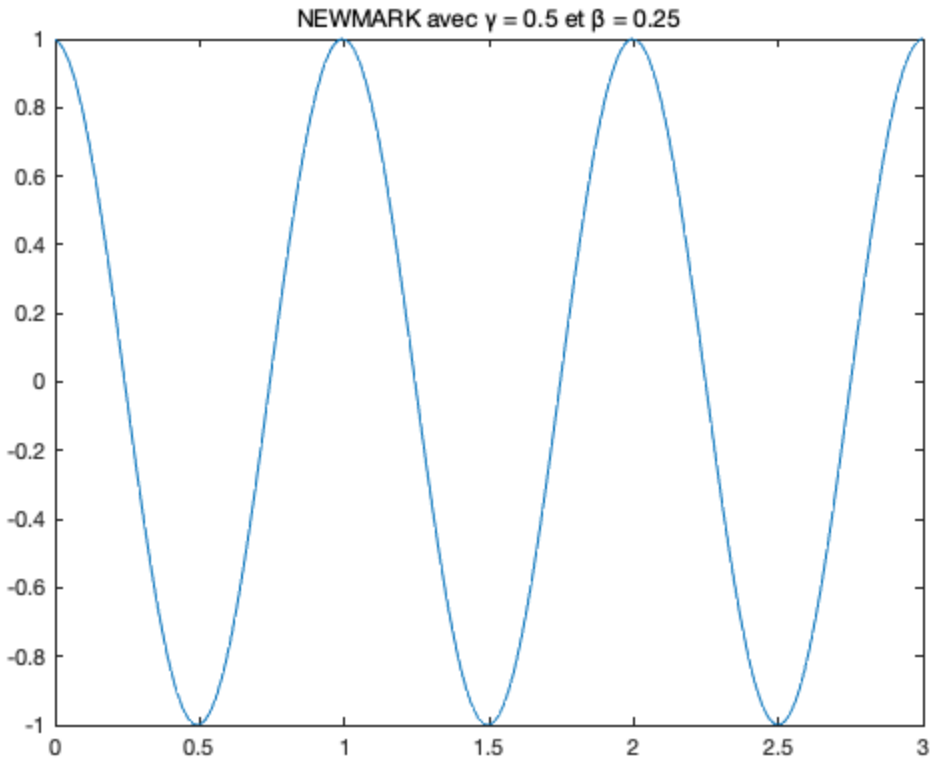
Q4.4

```
plot(x1, E1, x2, E2, x5, E5, x8, E8);
title('E* avec différants pas de 4 méthodes');
legend('Analytique', 'Euler Explicit', 'Euler Implicit', 'RUNGE
 KUTTA');

% C'est plus correcte en comparant avec les deux autres méthodes.
```



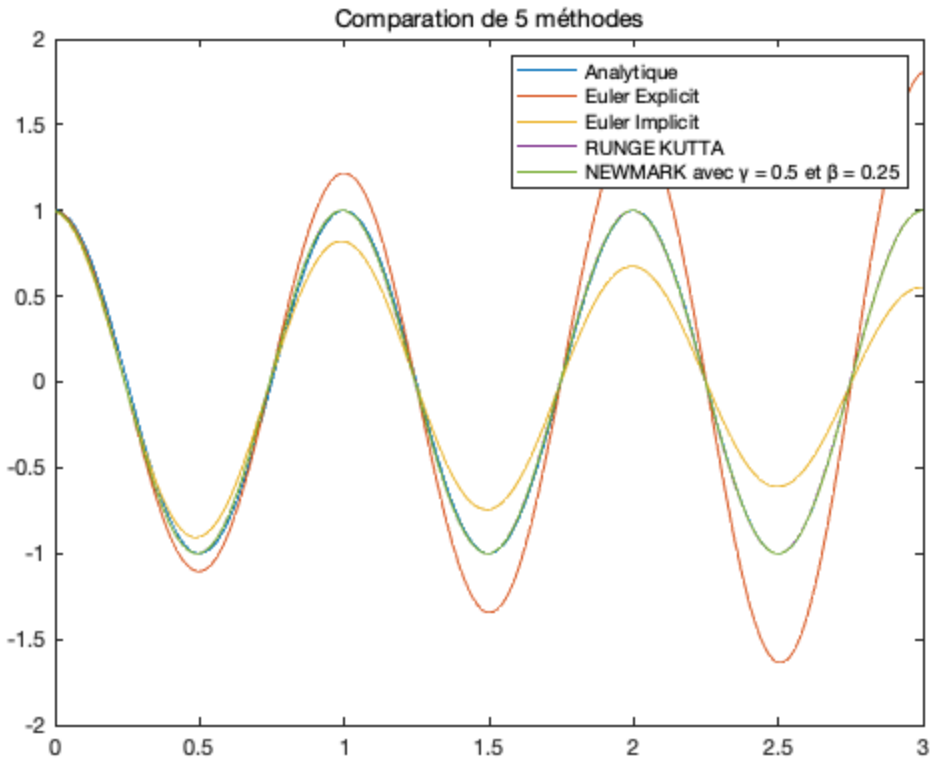
Q5.1.1

```
dt = 0.01;
gamma = 0.5;
beta = 0.25;
B = [1 + beta * w0 * w0 * dt * dt, 0; gamma * w0 * w0 * dt, 1];
C = [1 + (beta - 0.5) * w0 * w0 * dt * dt, dt; (gamma - 1) * w0 * w0 *
 dt, 1];
A = inv(B) * C;
U = [q0; dq0];
q9 = [];
E9 = [];
for t = linspace(0, T0, 300)
U = A* U;
q9 = [q9, U(1)];
E9 = [E9, 0.5 * (U(2) * U(2) + w0 * w0 * U(1) * U(1))];
end

x9 = linspace(0, T0, 300);
plot(x9, q9);
title('NEWMARK avec # = 0.5 et # = 0.25');
```



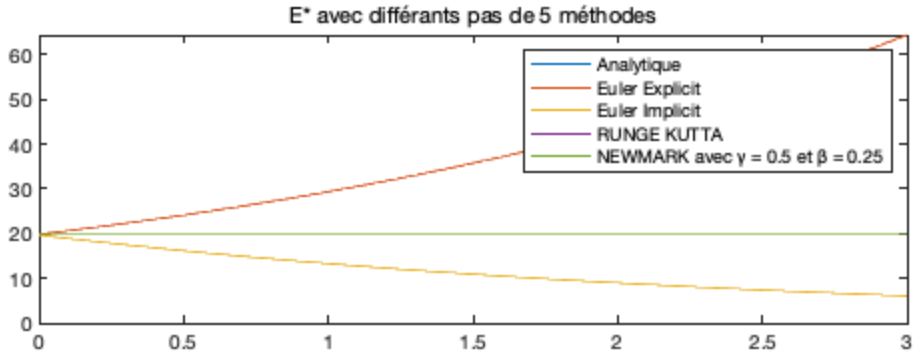
Q5.1.2

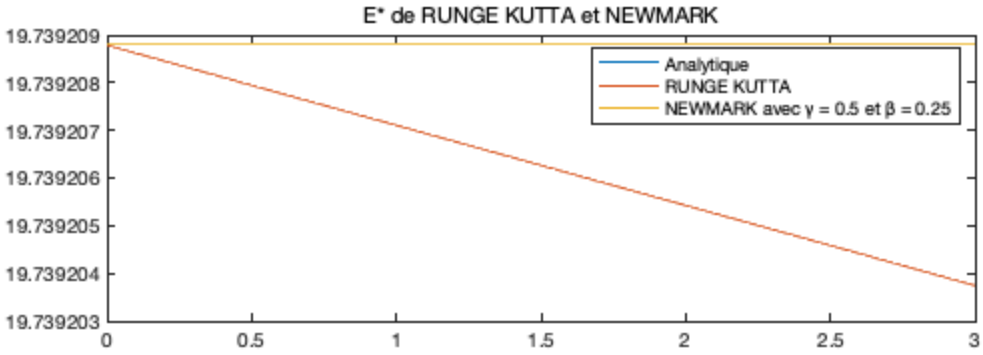
```
plot(x1, q1, x2, q2, x5, q5, x8, q8, x9, q9);
title('Comparation de 5 méthodes');
legend('Analytique', 'Euler Explicit', 'Euler Implicit', 'RUNGE
 KUTTA', ...
    'NEWMARK avec # = 0.5 et # = 0.25');
```



Q5.1.3

```
subplot(2, 1, 1);
plot(x1, E1, x2, E2, x5, E5, x8, E8, x9, E9);
title('E* avec différants pas de 5 méthodes');
legend('Analytique', 'Euler Explicit', 'Euler Implicit', 'RUNGE
 KUTTA', ...
    'NEWMARK avec # = 0.5 et # = 0.25');
subplot(2, 1, 2);
plot(x1, E1, x8, E8, x9, E9);
title('E* de RUNGE KUTTA et NEWMARK');
legend('Analytique', 'RUNGE KUTTA', 'NEWMARK avec # = 0.5 et # =
 0.25');

% La méthode de NEWMARK est plus bonne que celle de RUNGE KUTTA.
```





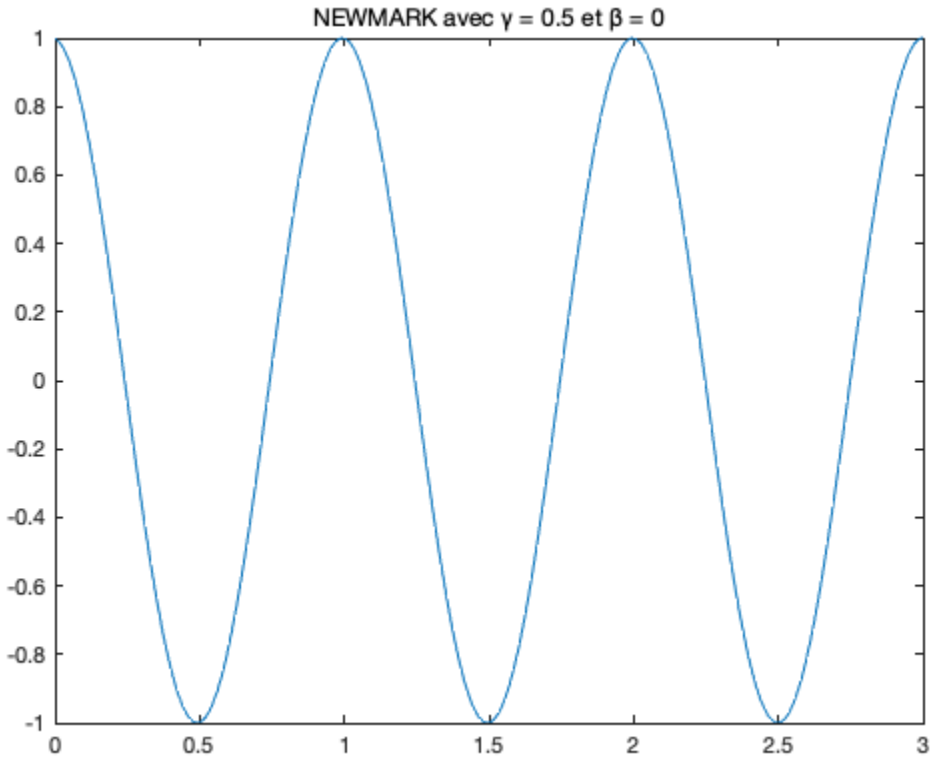
Q5.1.4

```
dt = 0.01; % ou 0.001 ou 0.1
B = [1 + beta * w0 * w0 * dt * dt, 0; gamma * w0 * w0 * dt, 1];
C = [1 + (beta - 0.5) * w0 * w0 * dt * dt, dt; (gamma - 1) * w0 * w0 *
 dt, 1];
A = inv(B) * C;
vp = eig(A);
abs(vp);

% Quand le pas est 0.1s,
% vp =
%    0.8203 + 0.5719i
%    0.8203 - 0.5719i
% abs(vp) = 1
% Quand les pas est 0.01s,
% vp =
%    0.9980 + 0.0628i
%    0.9980 - 0.0628i
% abs(vp) = 1
% Quand le pas est 0.001s,
% vp =
%    1.0000 + 0.0063i
%    1.0000 - 0.0063i
% abs(vp) = 1
```

```
% Le module de valeur propre est égale à 1 en variant le pas dt.
```

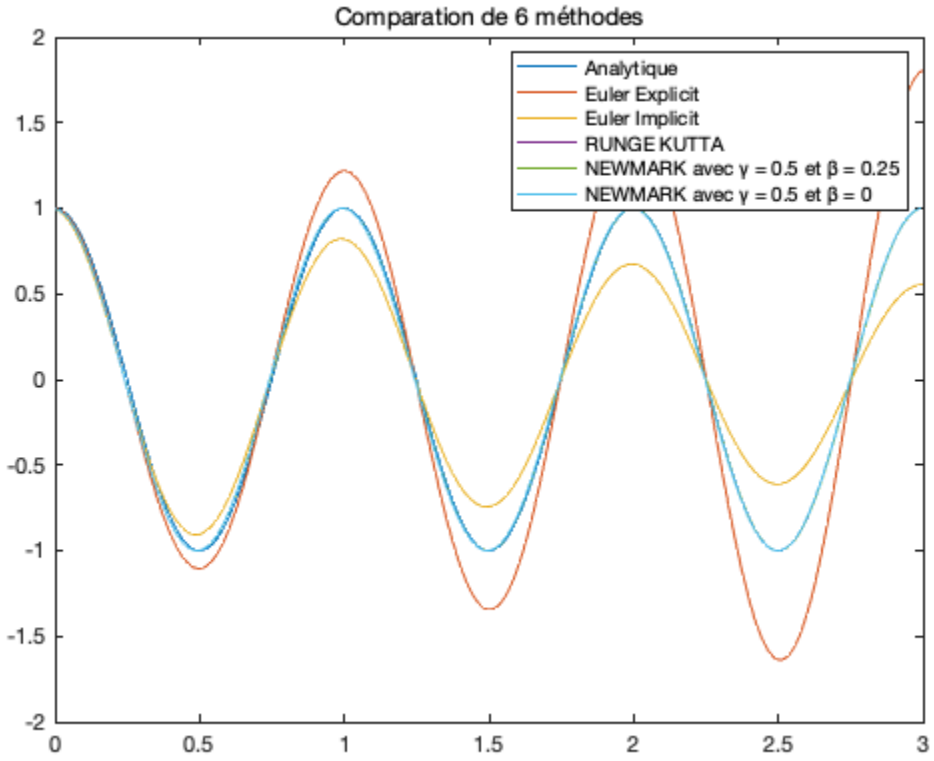
Q5.2.1

```
% On utilise la même méthode que Q5.1.1, mais # = 0.

dt = 0.01;
gamma = 0.5;
beta = 0;
B = [1 + beta * w0 * w0 * dt * dt, 0; gamma * w0 * w0 * dt, 1];
C = [1 + (beta - 0.5) * w0 * w0 * dt * dt, dt; (gamma - 1) * w0 * w0 *
 dt, 1];
A = inv(B) * C;
U = [q0; dq0];
q10 = [];
E10 = [];
for t = linspace(0, T0, 300)
U = A* U;
q10 = [q10, U(1)];
E10 = [E10, 0.5 * (U(2) * U(2) + w0 * w0 * U(1) * U(1))];
end

x10 = linspace(0, T0, 300);
subplot(1, 1, 1);
plot(x10, q10);
title('NEWMARK avec # = 0.5 et # = 0');
```



Q5.2.2

```
plot(x1, q1, x2, q2, x5, q5, x8, q8, x9, q9, x10, q10);
title('Comparation de 6 méthodes');
legend('Analytique', 'Euler Explicit', 'Euler Implicit', 'RUNGE
 KUTTA', ...
    'NEWMARK avec # = 0.5 et # = 0.25', 'NEWMARK avec # = 0.5 et # =
 0');
```

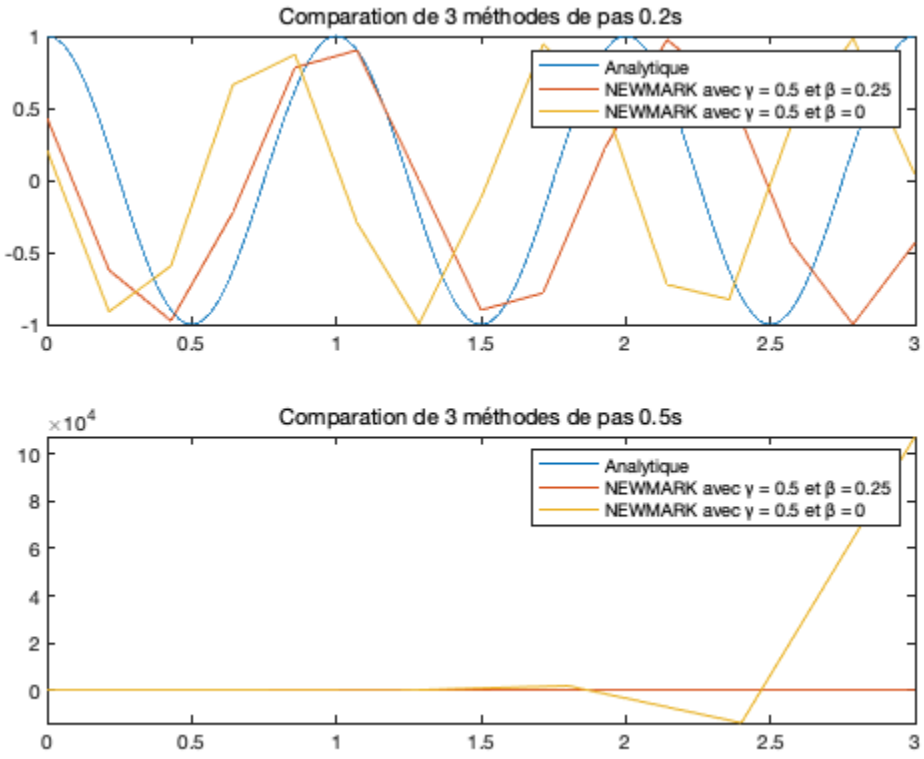


Q5.2.3

```
% On change le pas dt.

% Quand le pas est 0.2s.
dt = 0.2;
gamma = 0.5;
beta = 0.25;
B = [1 + beta * w0 * w0 * dt * dt, 0; gamma * w0 * w0 * dt, 1];
C = [1 + (beta - 0.5) * w0 * w0 * dt * dt, dt; (gamma - 1) * w0 * w0 *
 dt, 1];
A = inv(B) * C;
U = [q0; dq0];
q11 = [];
E11 = [];
for t = linspace(0, T0, 15)
```

```
U = A* U;
q11 = [q11, U(1)];
E11 = [E11, 0.5 * (U(2) * U(2) + w0 * w0 * U(1) * U(1))];
end
x11 = linspace(0, T0, 15);

gamma = 0.5;
beta = 0;
B = [1 + beta * w0 * w0 * dt * dt, 0; gamma * w0 * w0 * dt, 1];
C = [1 + (beta - 0.5) * w0 * w0 * dt * dt, dt; (gamma - 1) * w0 * w0 *
 dt, 1];
A = inv(B) * C;
U = [q0; dq0];
q12 = [];
E12 = [];
for t = linspace(0, T0, 15)
U = A* U;
q12 = [q12, U(1)];
E12 = [E12, 0.5 * (U(2) * U(2) + w0 * w0 * U(1) * U(1))];
end
x12 = linspace(0, T0, 15);

subplot(2, 1, 1);
plot(x1, q1, x11, q11, x12, q12);
title('Comparation de 3 méthodes de pas 0.2s');
legend('Analytique', 'NEWMARK avec # = 0.5 et # = 0.25', 'NEWMARK avec
 # = 0.5 et # = 0');

% Quand le pas est 0.5s.
dt = 0.5;
gamma = 0.5;
beta = 0.25;
B = [1 + beta * w0 * w0 * dt * dt, 0; gamma * w0 * w0 * dt, 1];
C = [1 + (beta - 0.5) * w0 * w0 * dt * dt, dt; (gamma - 1) * w0 * w0 *
 dt, 1];
A = inv(B) * C;
U = [q0; dq0];
q13 = [];
E13 = [];
for t = linspace(0, T0, 6)
U = A* U;
q13 = [q13, U(1)];
E13 = [E13, 0.5 * (U(2) * U(2) + w0 * w0 * U(1) * U(1))];
end
x13 = linspace(0, T0, 6);

gamma = 0.5;
beta = 0;
B = [1 + beta * w0 * w0 * dt * dt, 0; gamma * w0 * w0 * dt, 1];
C = [1 + (beta - 0.5) * w0 * w0 * dt * dt, dt; (gamma - 1) * w0 * w0 *
 dt, 1];
A = inv(B) * C;
U = [q0; dq0];
q14 = [];
```

```
E14 = [];
for t = linspace(0, T0, 6)
U = A* U;
q14 = [q14, U(1)];
E14 = [E14, 0.5 * (U(2) * U(2) + w0 * w0 * U(1) * U(1))];
end
x14 = linspace(0, T0, 6);

subplot(2, 1, 2);
plot(x1, q1, x13, q13, x14, q14);
title('Comparation de 3 méthodes de pas 0.5s');
legend('Analytique', 'NEWMARK avec # = 0.5 et # = 0.25', 'NEWMARK avec
 # = 0.5 et # = 0');

% La méthode avec # = 0 fonctionne mal que celle avec # = 0.25.
```



Q5.2.4

Published with MATLAB® R2018a