DM1 de mécanique numérique

Nom : Tom ZHU

Numéro d'étudiant : SY1924142

1. **Solution analytique de l'équation (1) :**

1 .1 : la solution générale de l'équation $\ddot{q} + w_0^2 * q = 0$ est

$$q = A * \left(e^{-iw_0 t}\right) + B * (e^{iw_0 t})$$

Selon les conditions initiaux, on peut obtenir A=B=1/2 ;
Alors, q=$\cos(2 * \pi * (t - 3))$, et donc q=$\cos(2 * \pi * t)$.

1.2 : D'après q(t)=$\cos(2 * \pi * (t - 3))$=$\cos(2 * \pi * t)$

On peut obtenir :

$E^* = \frac{1}{2} * (\dot{q}^2 + w_0^2 * q^2)$= $2 * \pi^2$, c'est un constant.

2. **Résolution de l'équation (1) avec un schéma d'EULER explicite**

2.1 : Selon l'équation(5), on a $q_{j+1} = q_j + \Delta t * \dot{q}_j$ et $\dot{q}_{j+1} = \dot{q}_j + \Delta t * \ddot{q}_j$.

En même temps: on a $\ddot{q}_j = -w_0^2 * q_j$

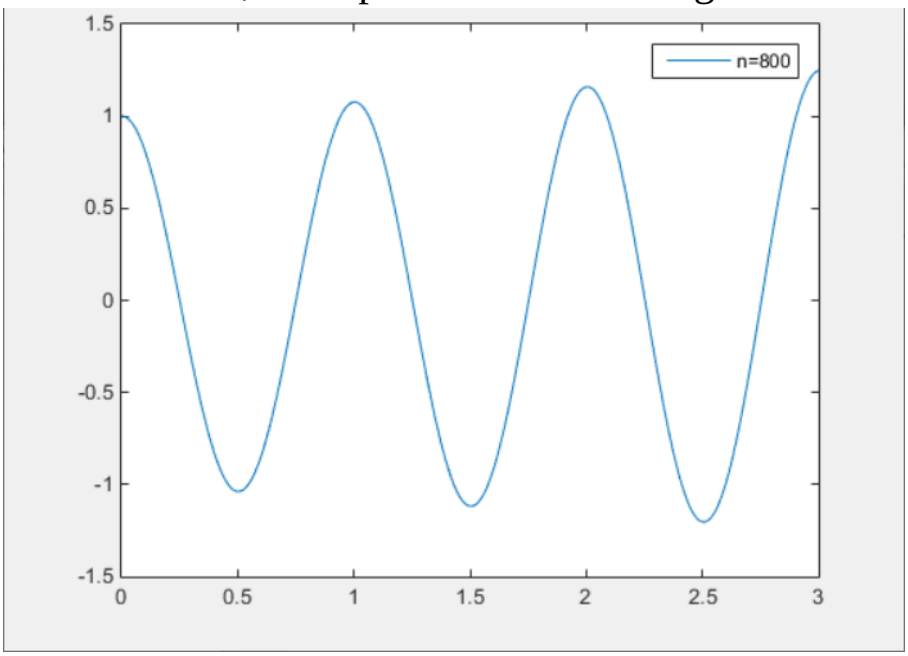
Donc on peut obtenir l'équation(6).

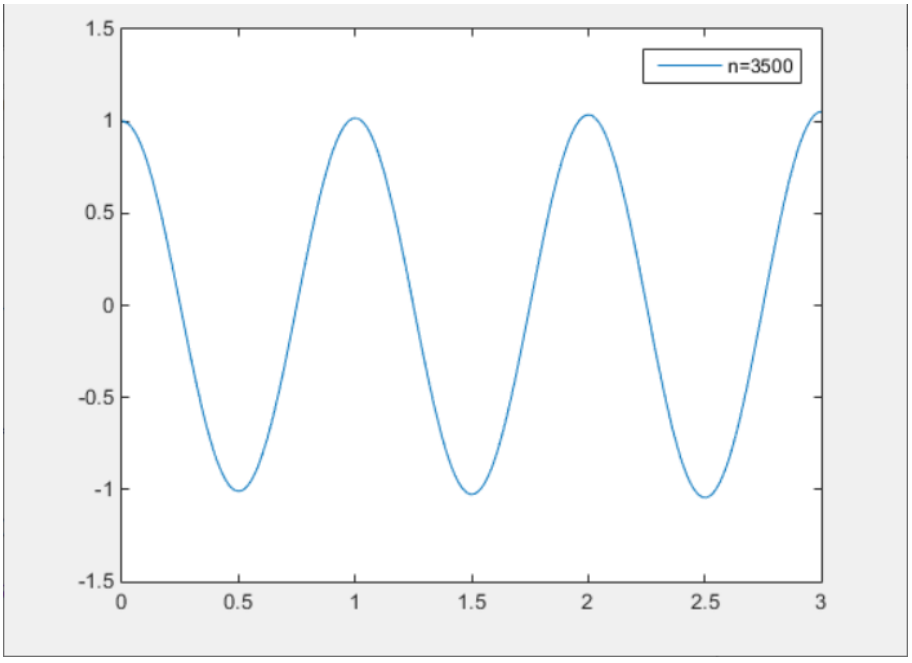
2.2 (a) Le code est ici:

main.m

```
clear all;
T=3;
w=2*pi;
n=3500;
delta_t=T/n;
qj=zeros(n,1);
qj(1,1)=1;
qj1=zeros(n,1);
qj1(1,1)=0;
E=zeros(n,1);
E(1,1)=0.5*(qj1(1,1)^2+(w^2)*qj(1,1)^2);
for i=1:n-1
    qj(i+1)=qj(i)+delta_t*qj1(i);
    qj1(i+1)=qj1(i)-delta_t*qj(i)*w^2;
    E(i+1)=0.5*(qj1(i+1)^2+(w^2)*qj(i+1)^2);
end
ti=linspace(0,3,n);
plot(ti,qj);
%plot(ti,E);

```

2.3 On prend deux différente pas de temps n=800, n=3500 ; et on peut obtenir deux figures :





Les résultats sont divergents. Et plus le pas de temps Δt est petit, plus la divergence est lente.

2.4 Selon la figure de changement de la quantité de E^* On peut obtenir que plus le pas de temps Δt est petit, plus la vitesse d'augmentation est petite.

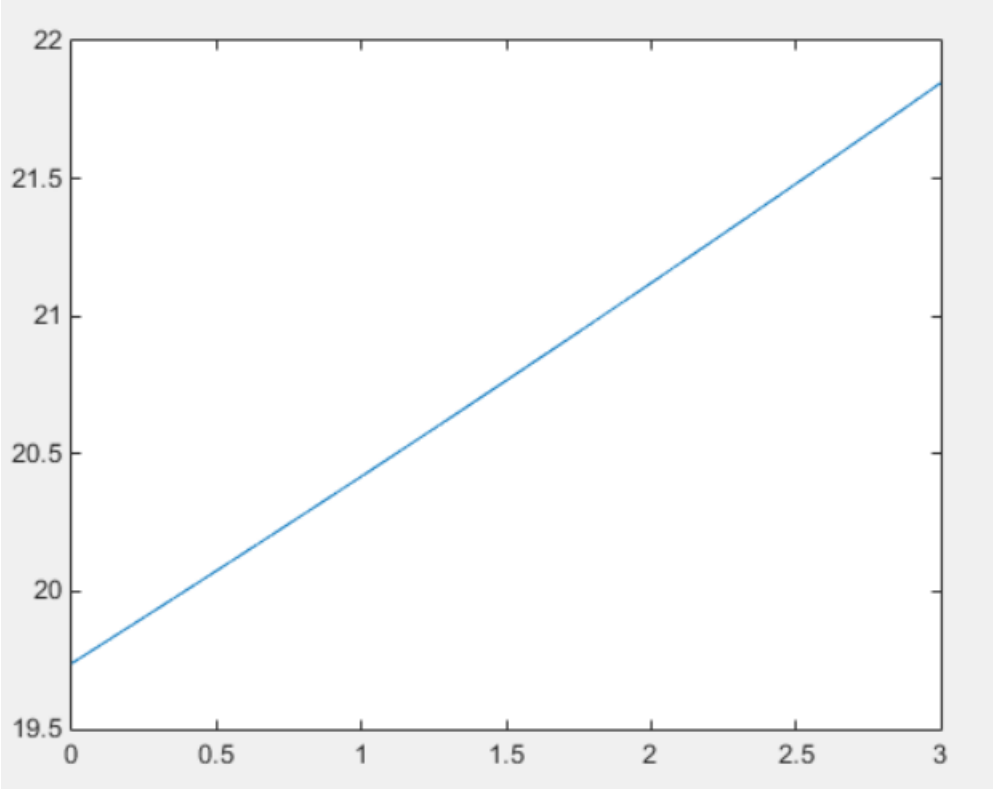


Figure 1 n=3500

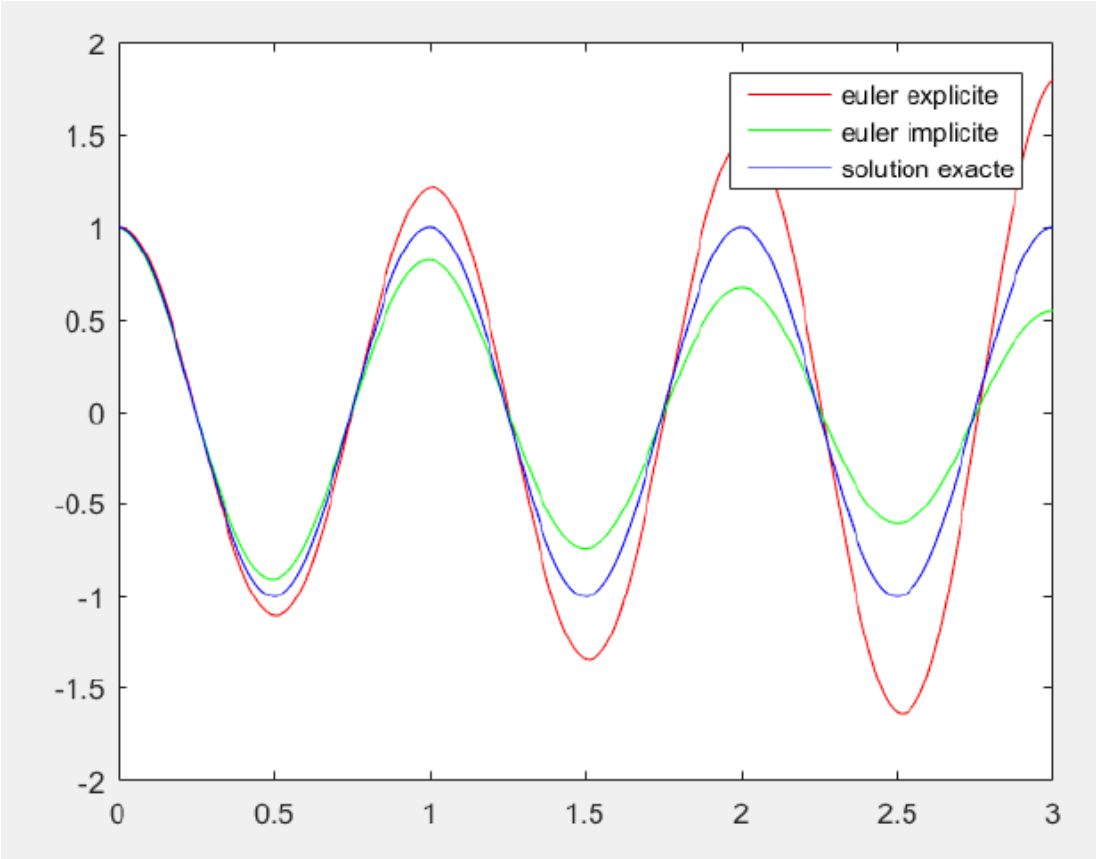
2.5 On prend delta_t=0.01s, et n=300. On obtient les valeurs propres : $1 \pm 0.0628318530717959i$; Le caractère inconditionnellement instable est la valeur absolue de minimum de valeur propre est supérieur à 1.

3. Résolution de l'´equation (1) avec un schéma d'EULER implicite

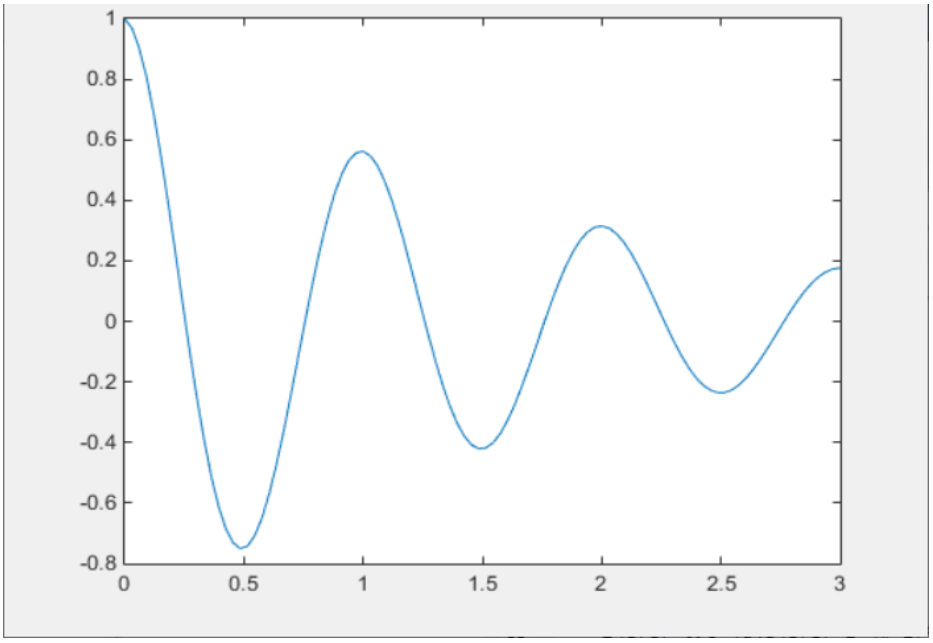
3.1 : Le code est ici :

```
clear all;
T=3;
w=2*pi;
n=1000;
delta_t=T/n;
qj=ones(n,1);
qj(1,1)=1;
qj1=ones(n,1);
qj1(1,1)=0.1;
E=zeros(n,1);
E(1,1)=0.5*(qj1(1,1)^2+(w^2)*qj(1,1)^2);
for i=1:n-1
    qj(i+1)=delta_t*qj1(i)+qj(i)/(1+(delta_t^2)*(w^2));
    qj1(i+1)=qj1(i)-delta_t*(w^2)*qj(i+1);
    E(i+1)=0.5*(qj1(i+1,1)^2+(w^2)*qj(i+1,1)^2);
end
ti=linspace(0,3,n);
plot(ti,qj);
%plot(ti,E);

```

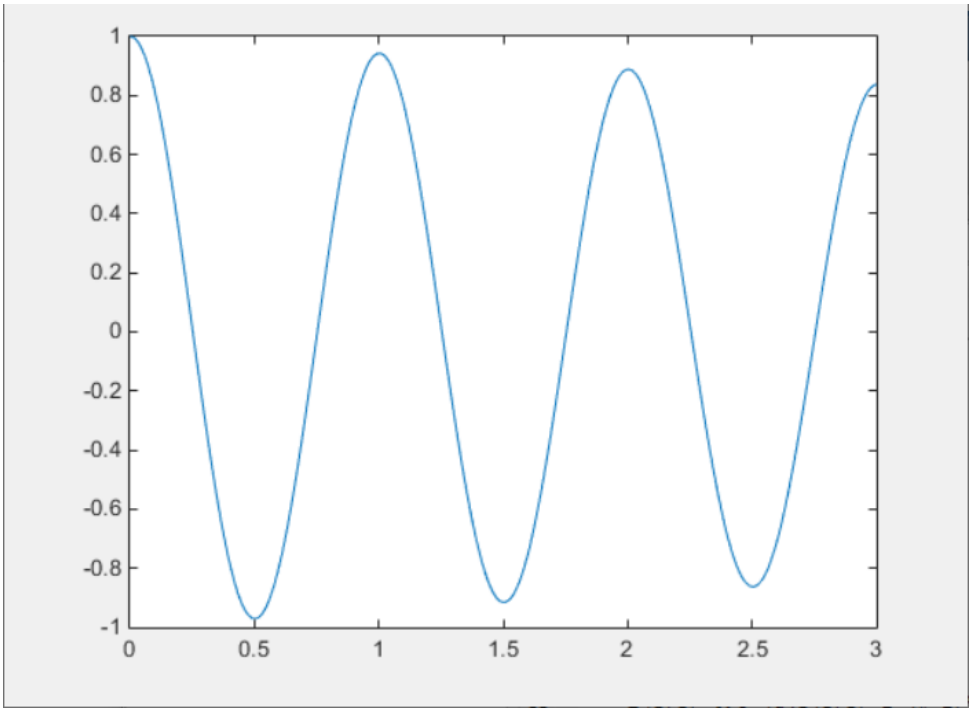
3.2 **:** On prend le pas de temps comme 0.01s, et on a la figure ici :



3.3 **:**pour n=100 et n=1000, on a deux figures :



La figure : n=100



La figure : n=1000

On peut obtenir le résultat que plus le pas de delta_t est petit, plus l'atténuation des oscillations est faible.

3.4 **:** Selon la figure de changement de la quantité de E^* , on peut obtenir que plus le pas de temps est petit, plus la vitesse d'augmentation est petite.

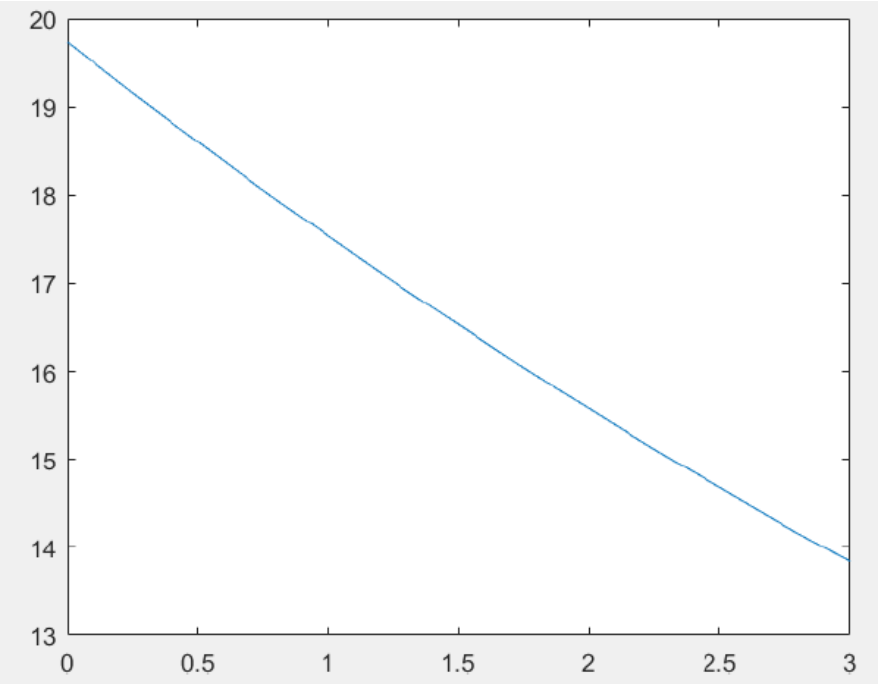


Figure de changement de la quantité de E^* (n=1000)

3.5 **:** On calcule les valeurs propres : 0.999644820438904 $\pm$0.018843i; Le caractère inconditionnellement stable : la valeur absolue de minimum de valeur propre est supérieure ou égal à 1.

4. Résolution de l'équation (1) avec un schéma de RUNGE KUTTA

4.1 On a $y_1=\dot{q}$, y_2=q. Donc l'équation est $y_1=-w_0^2 y_2$, $y_2=y_1$.

4.2 Le code est ici :

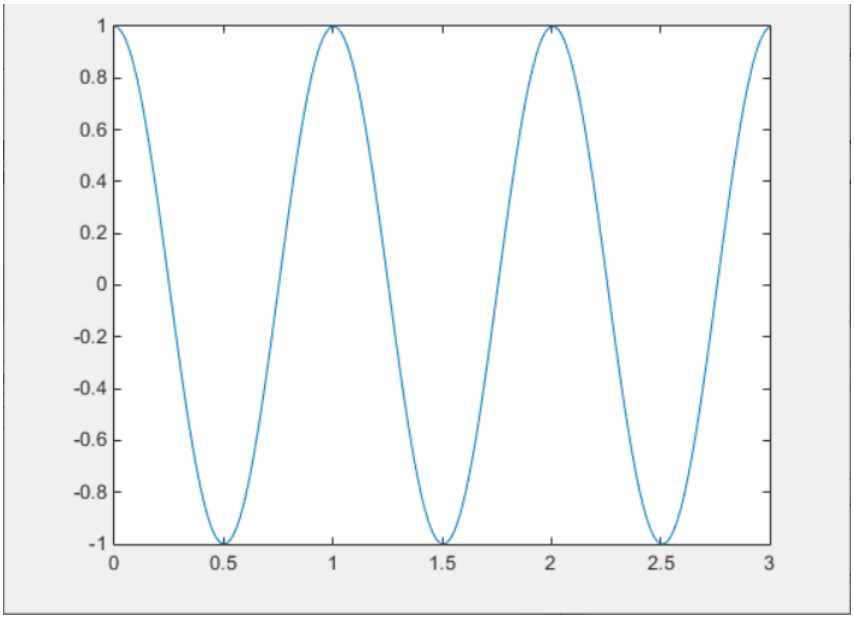
```
function [fp,E]=runge_function(x0,x01,t,T,wo,E)
n=T/t;
x=x0;
x1=x01;
h=t;
fp=zeros(n,1);
fp(1,1)=1;
for i=1:n
k1=func_x1(x,x1,wo);
l1=func_x(x,x1);
```

```
k2=func_x1(x+l1*h/2, x1+k1*h/2,wo);
l2=func_x(x+l1*h/2, x1+k1*h/2);

k3=func_x1(x+l2*h/2, x1+k2*h/2,wo);
l3=func_x(x+l2*h/2, x1+k2*h/2);

k4=func_x1(x+l3*h, x1+k3*h,wo);
l4=func_x(x+l3*h, x1+k3*h);

x1=x1+(k1+2*k2+2*k3+k4)*h/6;
x=x+(l1+2*l2+2*l3+l4)*h/6;
if(i<n)
fp(i+1)=x;
E(i+1)=0.5*(x1^2+(wo^2)*x^2);
end
end
end
function x=func_x(x,x1)
x=x1;
end
function x1=func_x1(x,x1,wo)
x1=-wo^2*x;
end
```

Et la main fonction est ici :

```
T=3;
n=300;
t=T/n;
x0=1;
wo=2*pi;
x01=0;
ti=linspace(0,3,n);
E=zeros(n,1);
E(1,1)=0.5*(x01^2+(wo^2)*x0^2);
[fp,E]=runge_function(x0,x01,t,T,wo,E);
%plot(ti,E);
plot(ti,fp)
```

4.3 Le résultat converge et il égale à la solution analytique.



4.4 Le résultat est similaire. De plus, il est plus stable que la method d'Euler.

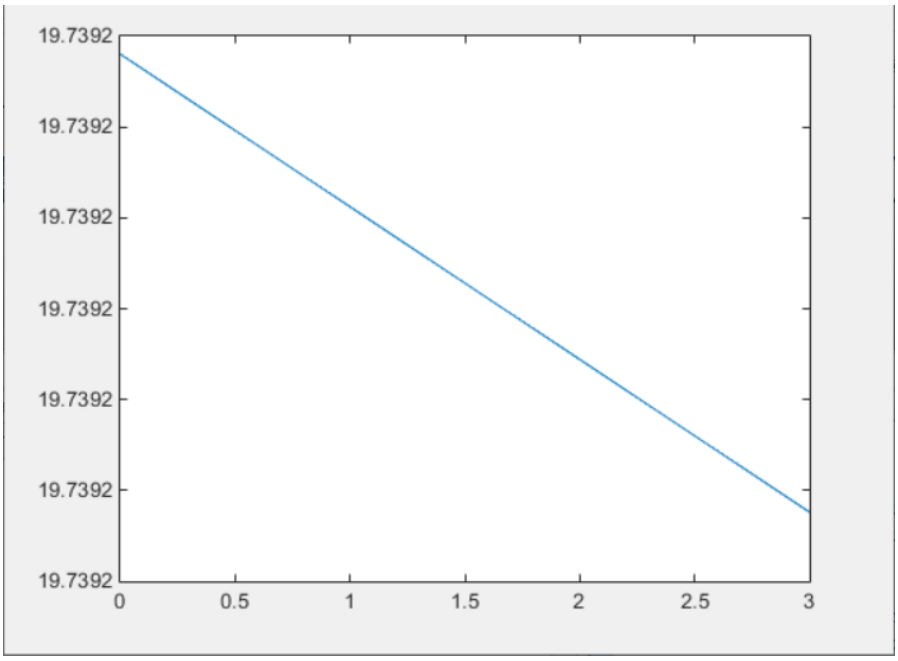


Figure changement de la quantité E*(n=300)