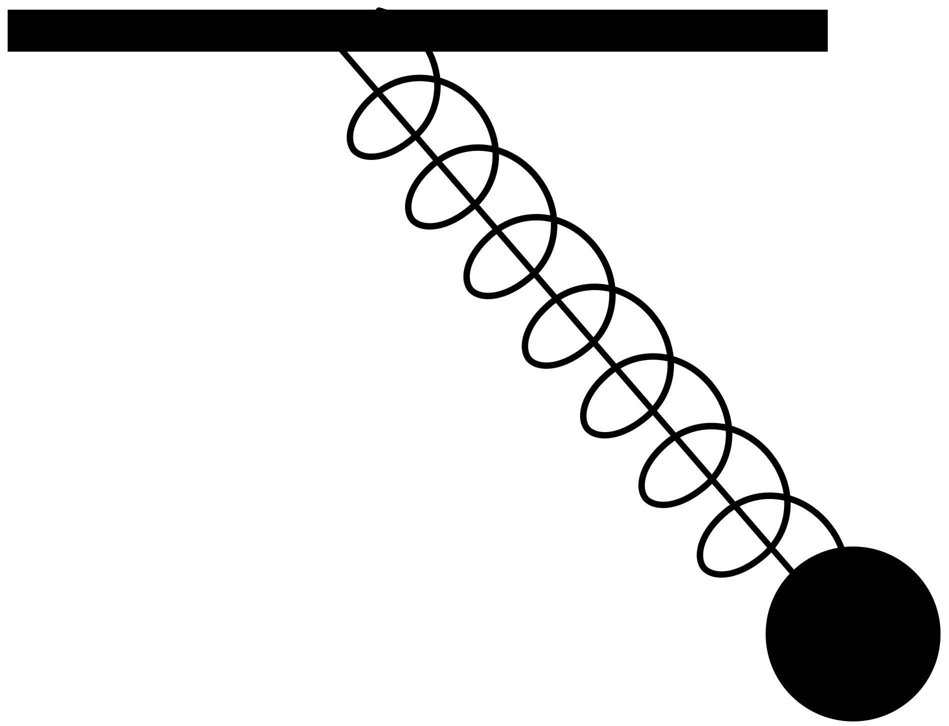
PENDULE ÉLASTIQUE

Alexandre HERGAUX
Nikola MEDVED



SOMMAIRE

ETAPES DU PROJET

Etape 1

Trouver les équations définissant le mouvement du pendule

Etape 2

Choisir par quelle méthode résoudre le problème

Etape 3

Déterminer des moyens de prouver la validité de nos résultats

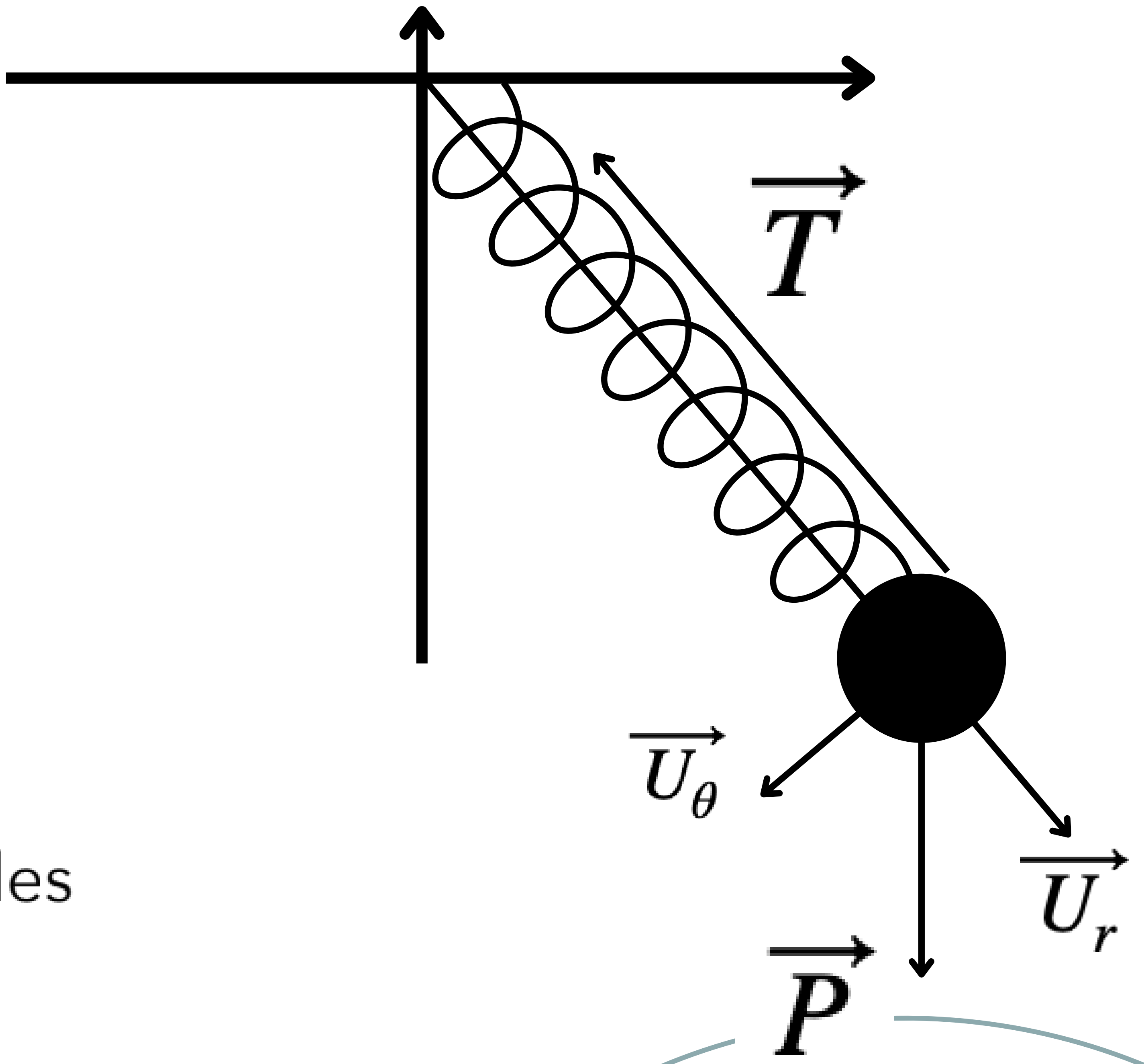
Etape 4

Analyser les résultats

ANALYSE

Bilan des des forces :

- Tension du ressort : $\vec{T} = -k \cdot (l - l_0)$
- Poids $\vec{P} = m \cdot g$
- Eventuellement des frottements $\vec{F_f} = -\alpha \cdot \vec{v}$



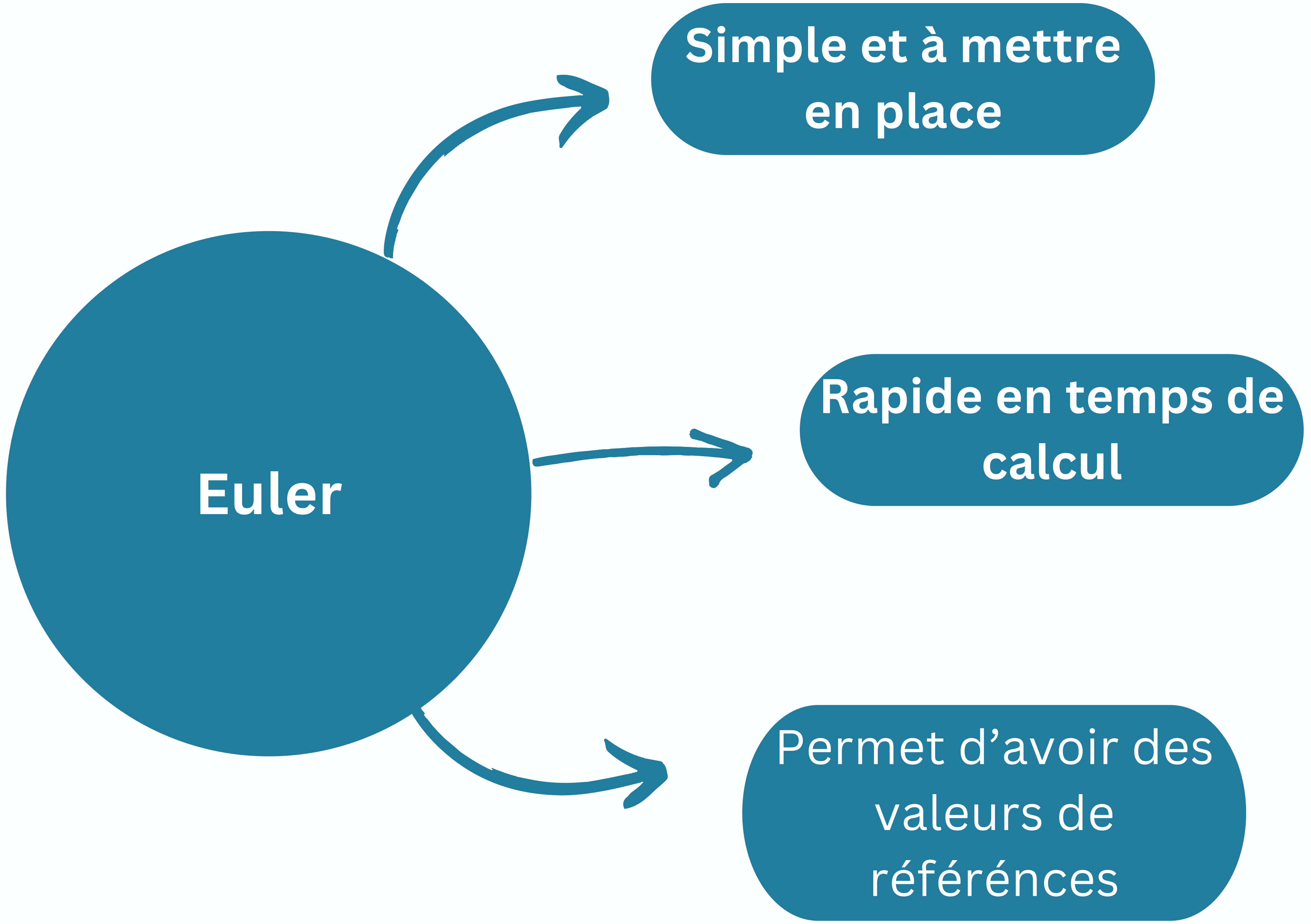
Voici les équations obtenu en réalisant en bilan des forces et en projetant sur $\vec{U_r}$ et $\vec{U_\theta}$:

$$\ddot{\theta} - \theta^2 - g \cdot \cos(\theta) - \frac{k}{m} \cdot l = \frac{k}{m} \cdot l_0$$

$$\ddot{\theta} + \ddot{l} \cdot \dot{\theta} + g \cdot \sin(\theta) = 0$$

Méthode EULER

Méthode d’approximation des équations différentiel

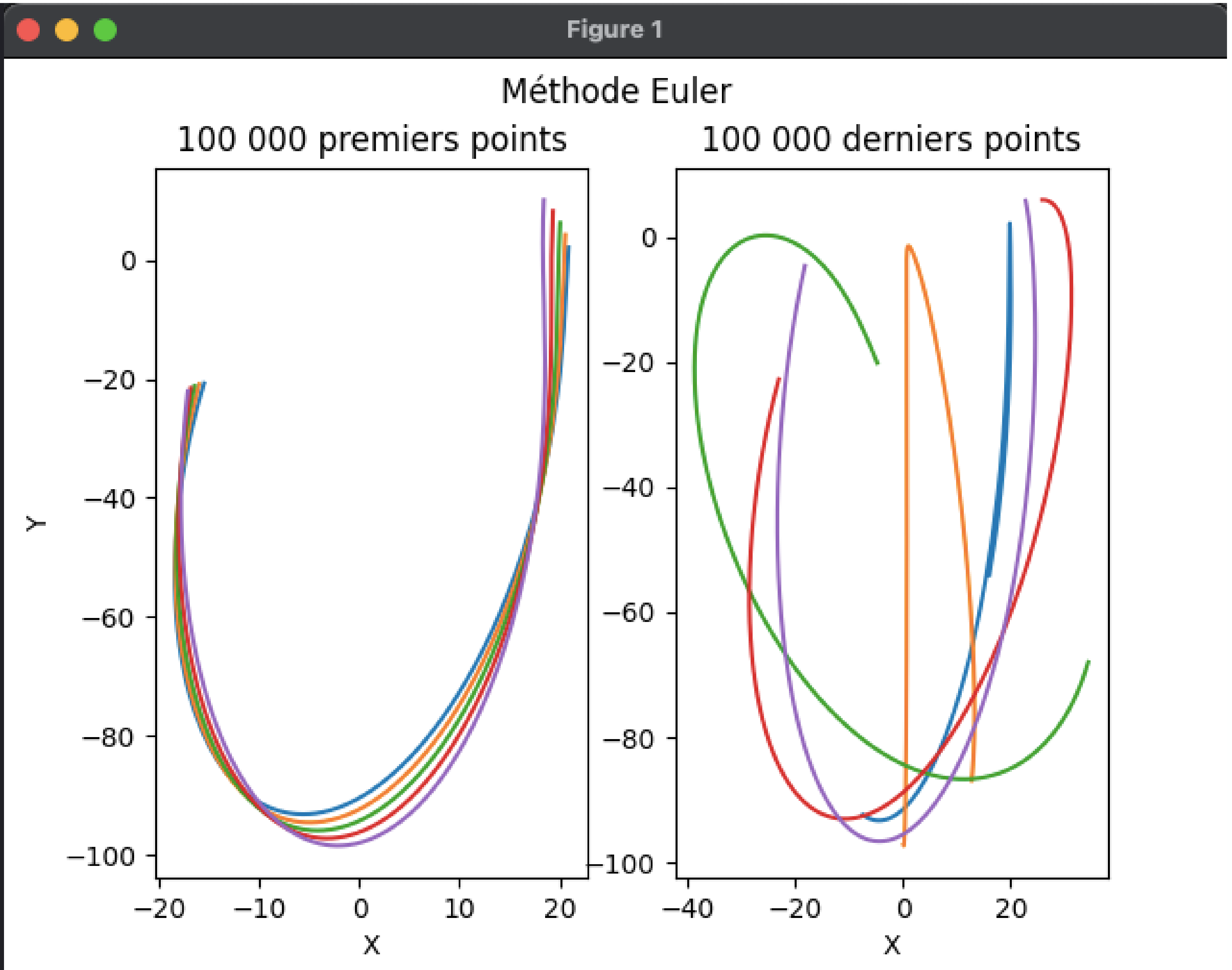


Posons :

$$Y = \begin{pmatrix} \dot{l} \\ l \\ \dot{\theta} \\ \theta \end{pmatrix} \qquad Y' = \begin{pmatrix} \ddot{l} \\ \dot{l} \\ \ddot{\theta} \\ \dot{\theta} \end{pmatrix}$$

Méthode EULER : Méthode d’approximation des équations différentiel

```c
//Euler adapté pour une matrice de 4 valeurs
void euler(int nt, double t[nt],double y0[4], double** y){
    y[0][0]=y0[0];
    y[0][1]=y0[1];
    y[0][2]=y0[2];
    y[0][3]=y0[3];
    double dt;
    double yp[4];
    double ytemp[4];
    for(int i=1;i<nt;i++){
        dt=t[i]-t[i-1];
        ytemp[0]=y[i-1][0];
        ytemp[1]=y[i-1][1];
        ytemp[2]=y[i-1][2];
        ytemp[3]=y[i-1][3];
        f(ytemp,yp);
        y[i][0]=y[i-1][0]+dt*yp[0];
        y[i][1]=y[i-1][1]+dt*yp[1];
        y[i][2]=y[i-1][2]+dt*yp[2];
        y[i][3]=y[i-1][3]+dt*yp[3];
    }
}
```



PROBLEMES RENCONTRÉS

01 - MANQUE DE MÉMOIRE

Avec l'augmentation du nombre de points, la pile était saturée

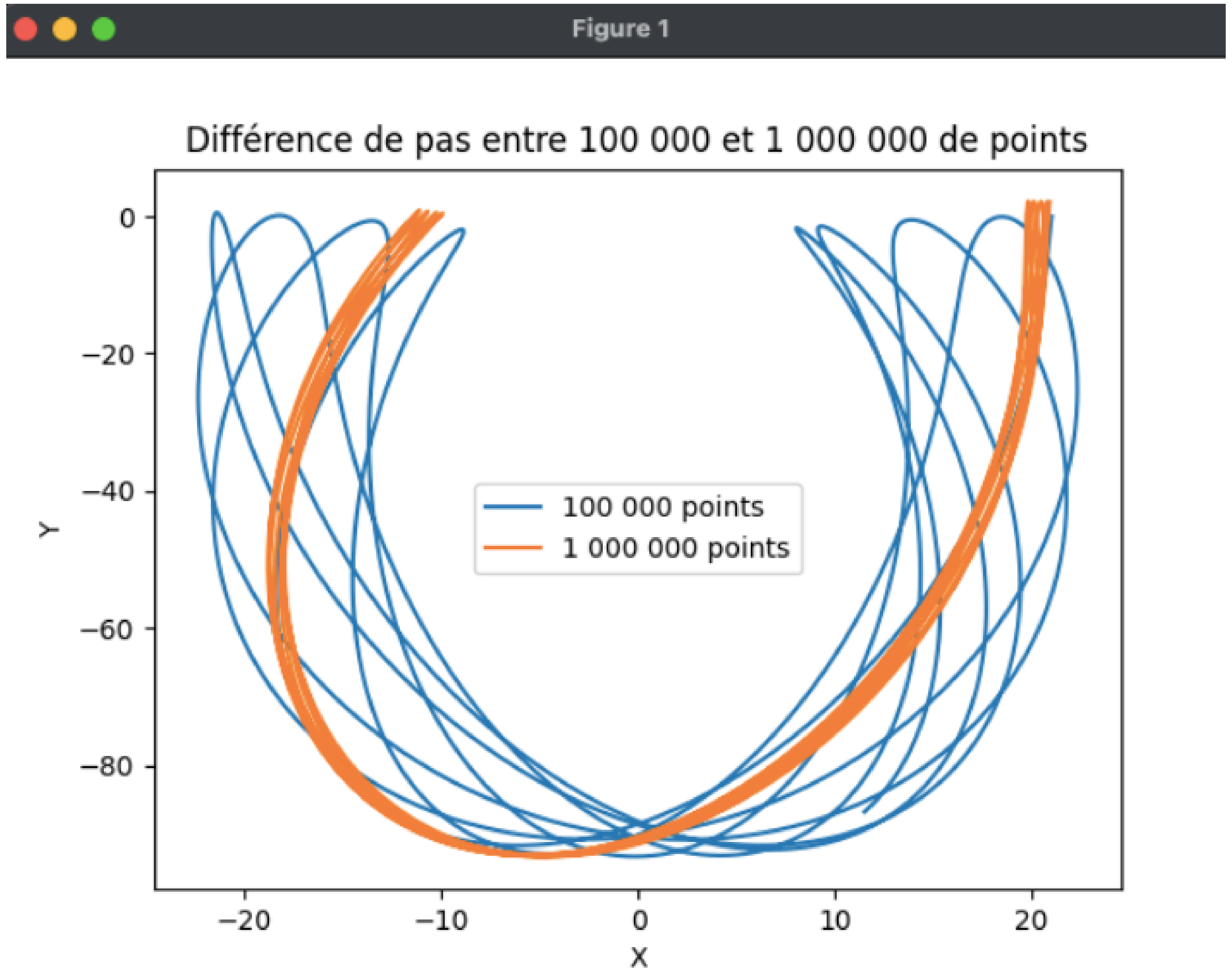
02 - LA PRÉCISION

Besoin de beaucoup de point pour augmenter la précision

01 - MANQUE DE MÉMOIRE

```c
int main(){
    char c[nEch][21];                    //Tableaux de mots
    nameCre(c,nEch);                     //Création des noms de fichier

    for(int i=0;i<nEch;i++){             //Boucle pour créer le nombre d'echantillon suffisant

        l+=lDelta;teta+=tetaDelta;lptn+=lptnDelta;tetaptn+=tetaptnDelta;k+=kDelta;

        FILE* ff=fopen(c[i],"w");        //Creation d'un fichier ou on ecrira les valeurs
        if (ff == NULL) {
            printf("Erreur d'ouverture du fichier\n");
            return 1;
        }

        int a=n/pas;                     //Répartition du nombre de points
        int r=n%pas;                     //printf("a=%i et r=%i\n",a,r);//DEBUG


        double y0[4]={lptn,l,tetaptn,teta};
        for(int i=0;i<a;i++){

            double* t=malloc(pas*sizeof(double));//Déclaration d'un tableau dynamique de temps
            timeTab(pas,t,i);

            double** y=(double**)malloc(pas*sizeof(double*));// Déclaration d'un tableau dynamique pour les valeurs
            for(int j=0;j<pas;j++){
                y[j]=(double*)malloc(4*sizeof(double));
            }


            rk4(pas,t,y0,y);

            for(int i=0;i<pas;i++){
                fprintf(ff,"%e\t\t%e\t\t%e\t\t%e\t\t%e\n",t[i],y[i][1],y[i][3],-m*g*y[i][1]*cos(y[i][3])+0.5*k*pow(y[i][1]-l0,2),0.5*m*(pow(y[i][0],2)+pow(y[i][1]*y[i][2],2)));
                /*if(y[i][1]>10e4 || y[i][3]>10e4){
                    printf("Erreur calcul");
                    return 0;
                }*/
            }
            y0[0]=y[pas-1][0];y0[1]=y[pas-1][1];y0[2]=y[pas-1][2];y0[3]=y[pas-1][3];

            for(int i=0;i<pas;i++){  //
                free(y[i]);          //
            }                        //Libération de la mémoire
            free(y);                 //
            free(t);                 //
        }
```

01 - AUGMENTATION DU NOMBRE DE POINT

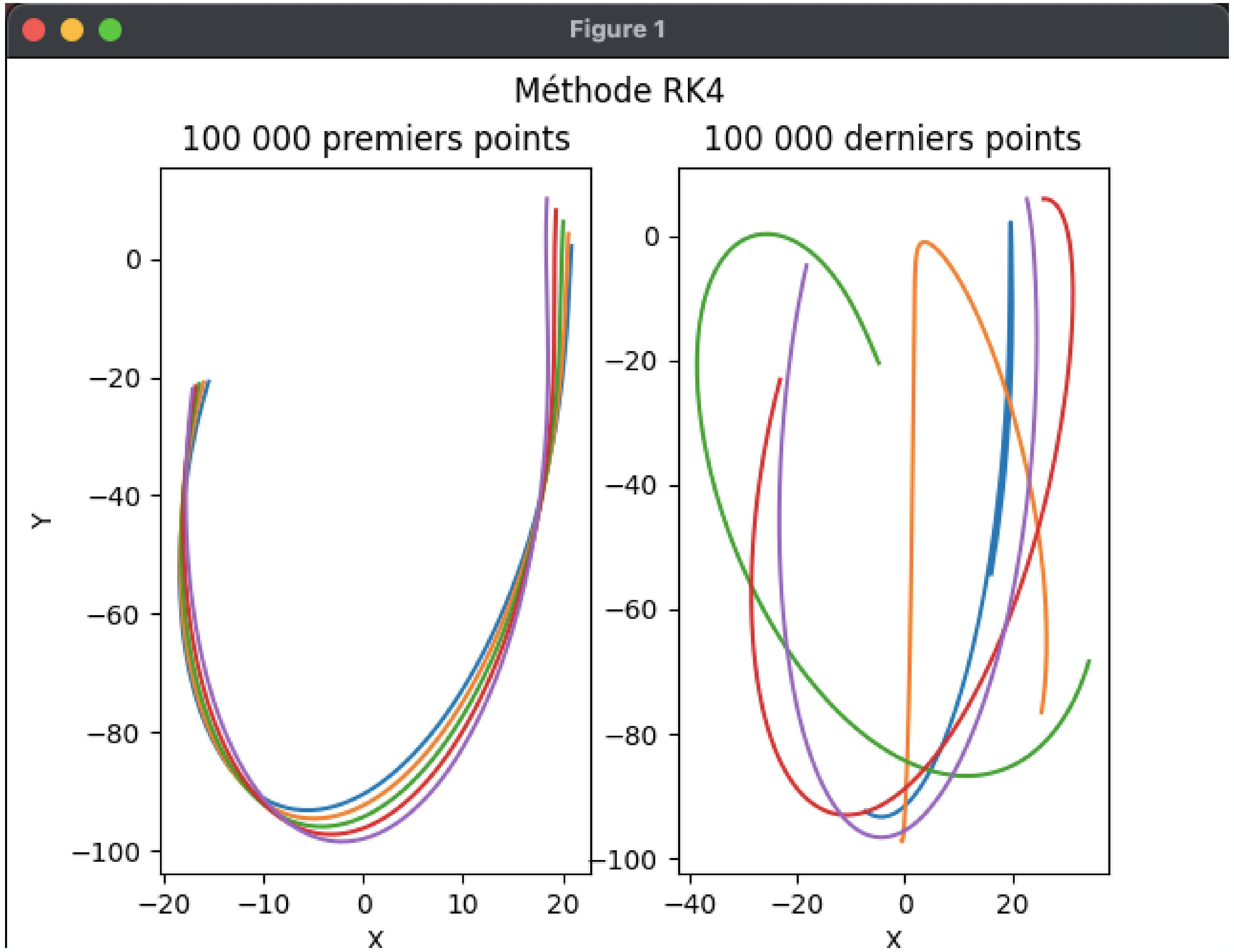


Voici ce que l’on obtient pour un temps de 100s, on voit bien ici que le pas n’était pas assez grand avec seulement 100 000 points pour la méthode Euler.

Méthode RK4 :

On utilisera les mêmes conditions initiales que pour les résolutions avec Euler.

```
//RK4 adapté pour une matrice de 4 valeurs
void rk4(int nt,double t[nt],double y0[4],double** y) {
    y[0][0]=y0[0];
    y[0][1]=y0[1];
    y[0][2]=y0[2];
    y[0][3]=y0[3];

    double dt;
    double yp[4];
    double ytemp[4];

    for(int i=1;i<nt;i++){
        dt=t[i]-t[i-1];

        for(int j=0;j<4;j++){
            ytemp[j]=y[i-1][j];
            double k1,k2,k3,k4;

            f(ytemp,yp);
            k1=dt*yp[j];

            ytemp[j]=y[i-1][j]+k1/2.0;
            f(ytemp,yp);
            k2=dt*yp[j];

            ytemp[j]=y[i-1][j]+k2/2.0;
            f(ytemp,yp);
            k3=dt*yp[j];

            ytemp[j]=y[i-1][j]+k3;
            f(ytemp,yp);
            k4=dt*yp[j];

            y[i][j]=y[i-1][j]+(k1+2*k2+2*k3+k4)/6.0;
        }
    }
}
```



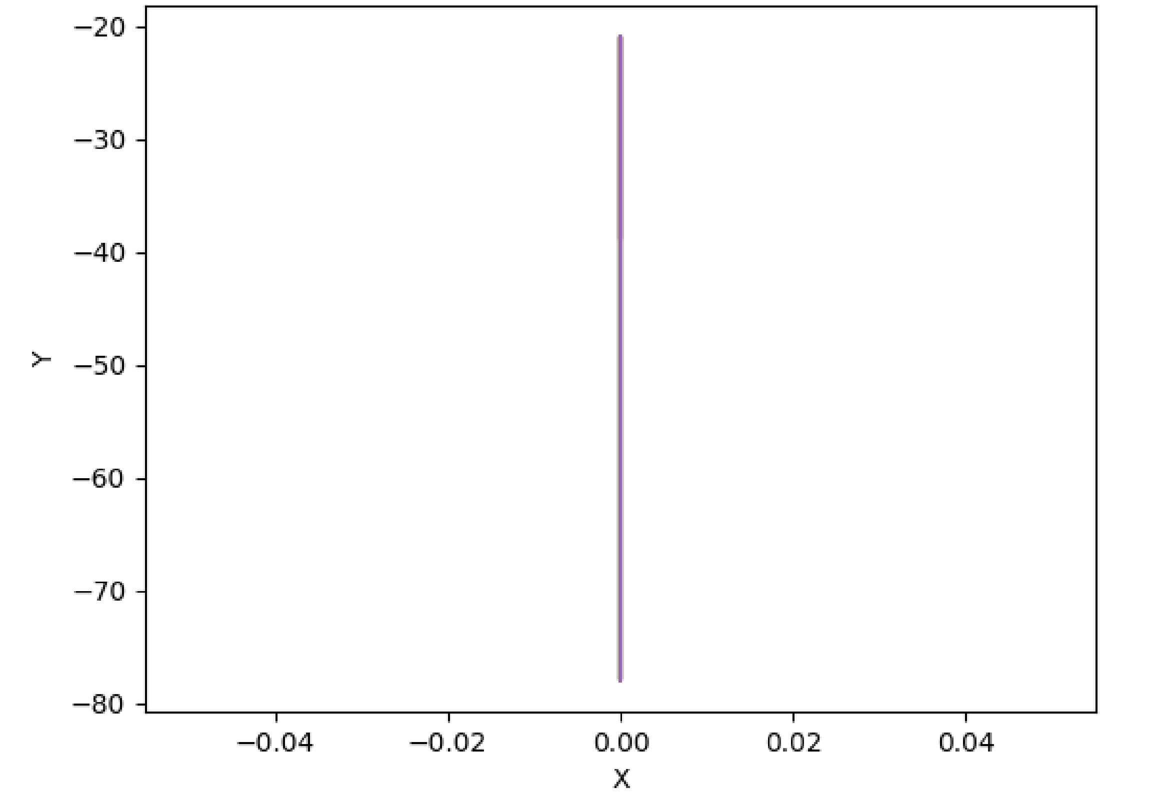
Stabilité

Conservation de l’énergie

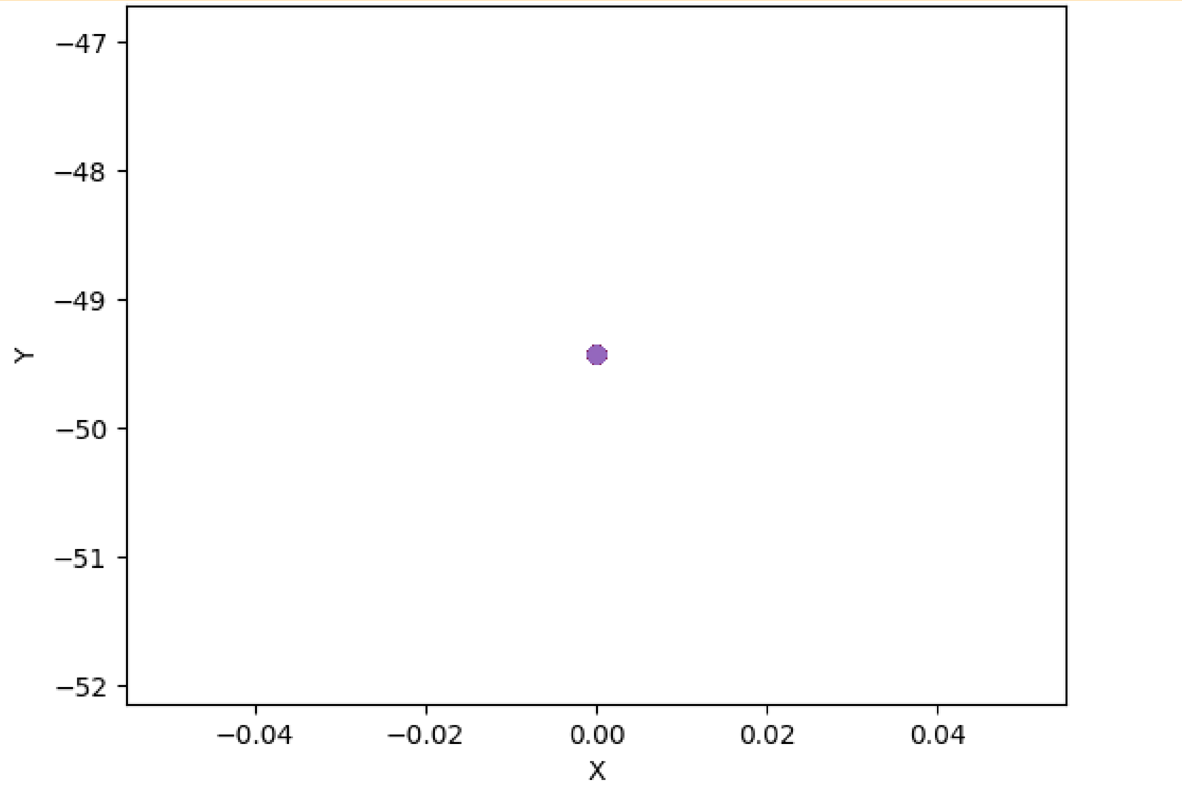
Différence entre les méthodes

Validité du programme

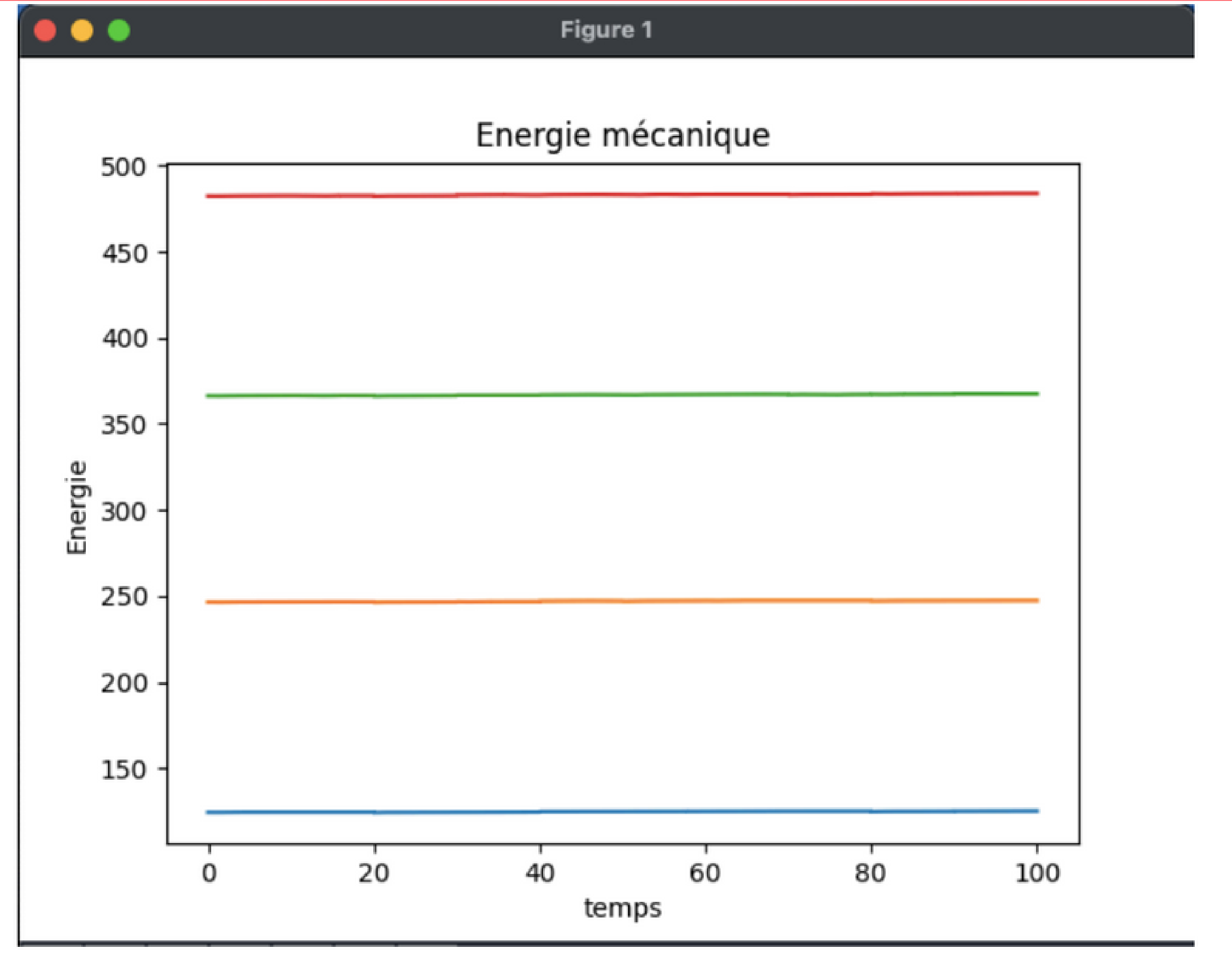
Propriété du ressort



Point d’équilibre

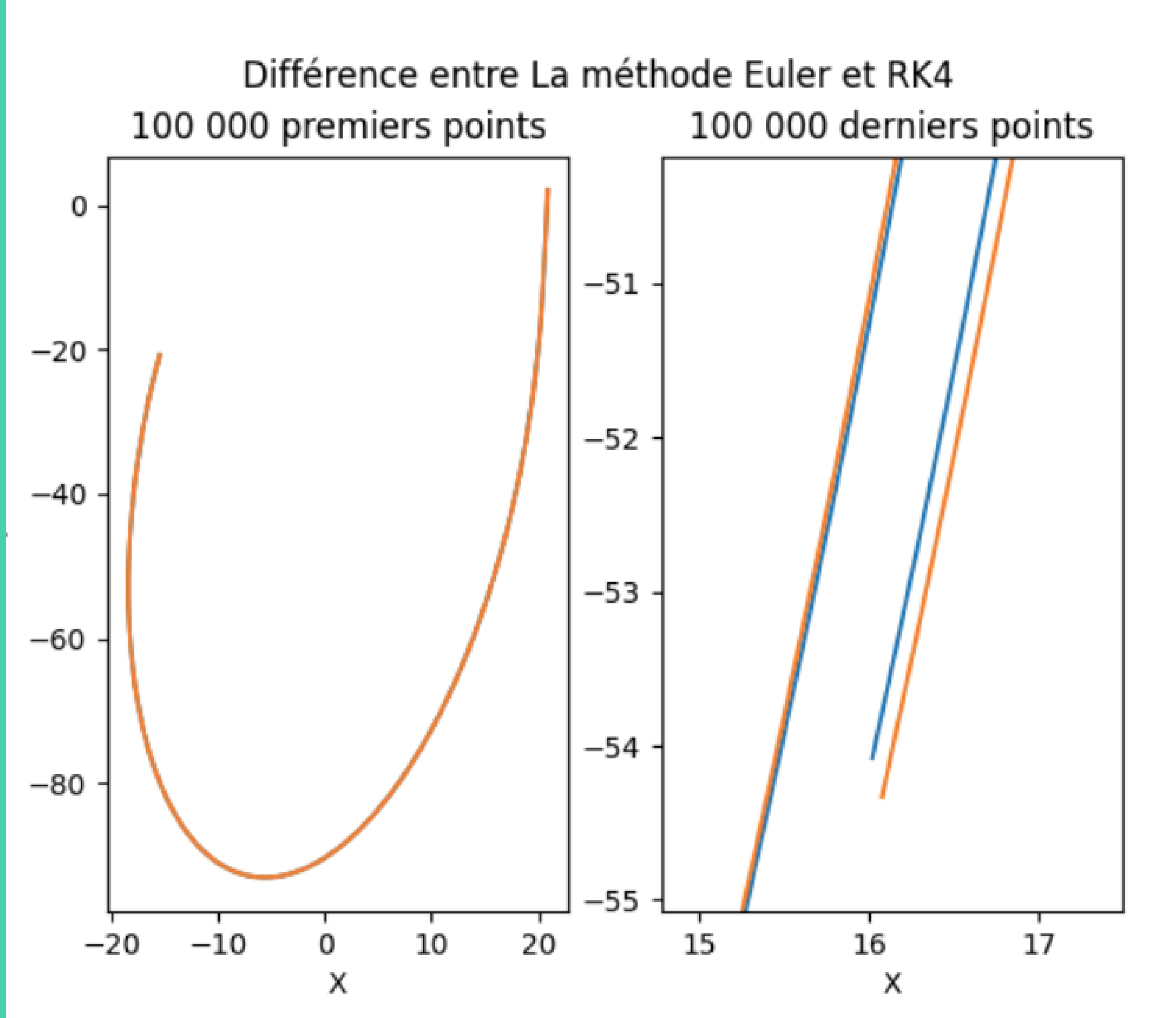


Conservation de l'énergie



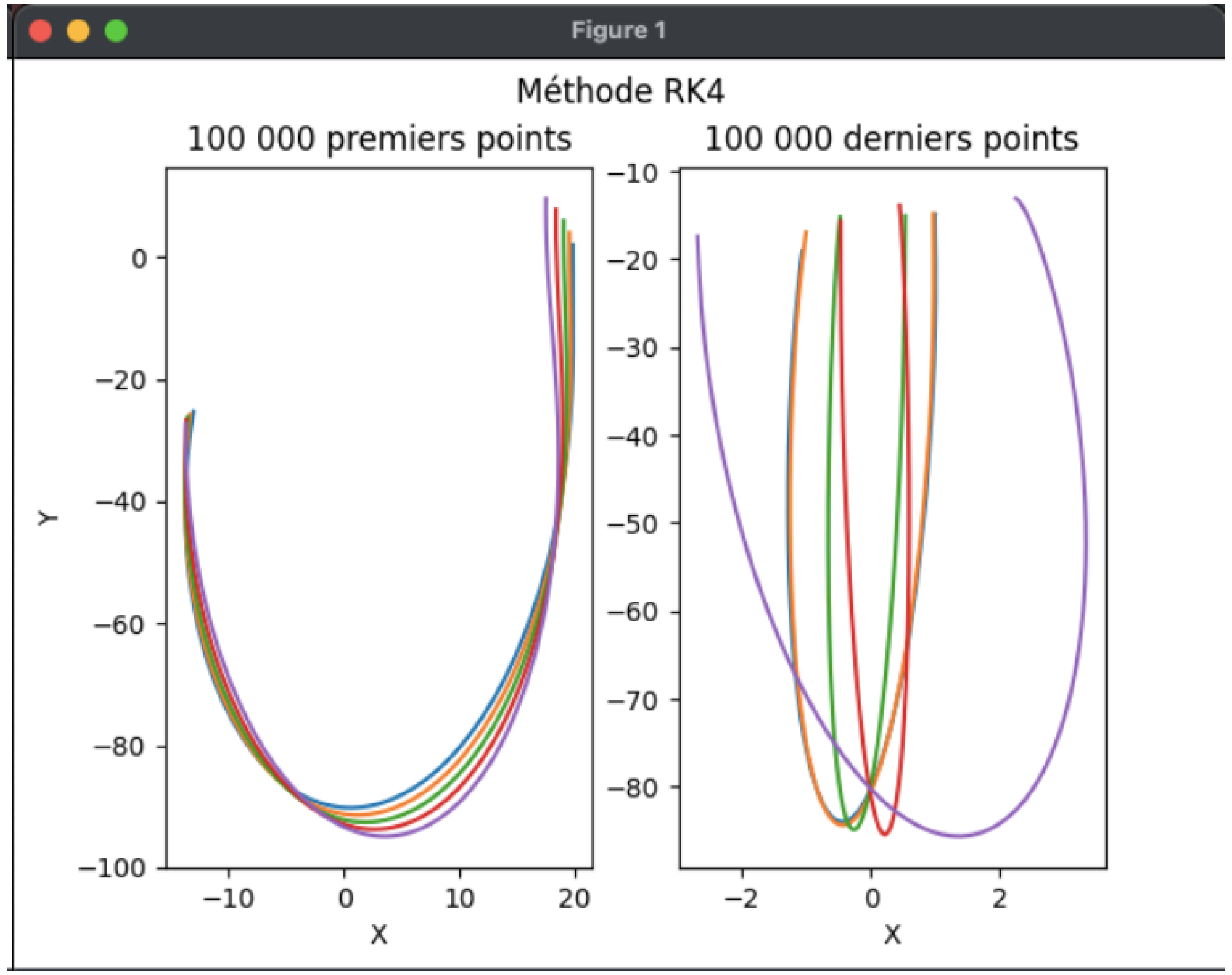
$$E_p = -m \cdot g \cdot l \cdot \cos(\theta) + 0.5 \cdot k \cdot (l - l_0)^2 \qquad E_c = \frac{1}{2} \cdot m \cdot \left(\dot{l}^2 + \left(l \cdot \dot{\theta}\right)^2\right)$$

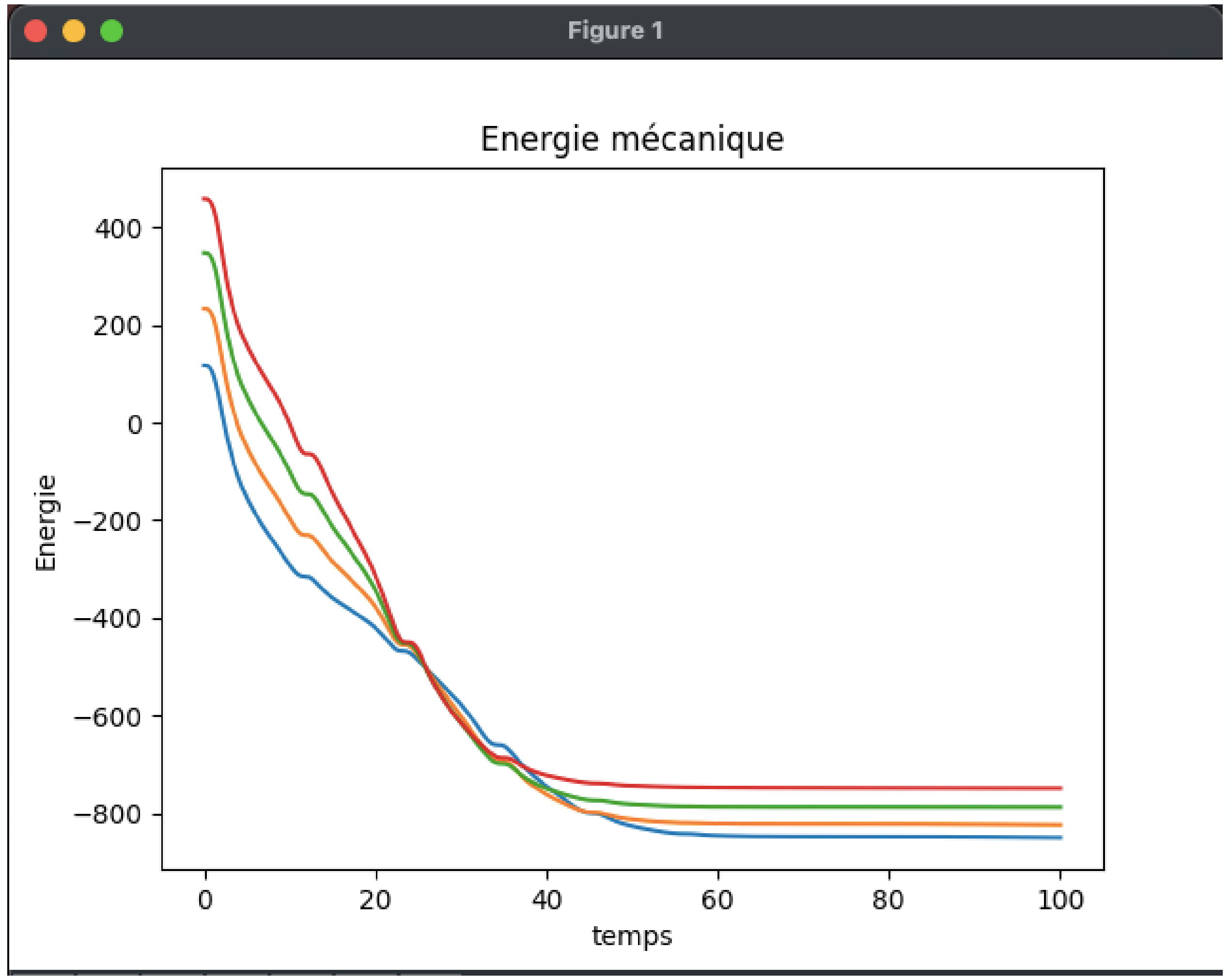
Différences ≠



Ajout de frottements

```
// fonction Y' qui donne la dérivé de Y
void f(double y[4], double dy[4]){
    dy[0]=y[1]*pow(y[2],2)+g*cos(y[3])-k/m*(y[1]-l0);
    dy[1]=y[0];
    dy[2]=-g*sin(y[3])/y[1]-2*y[0]*y[2]/y[1]-a/m*y[2];
    dy[3]=y[2];
}
```

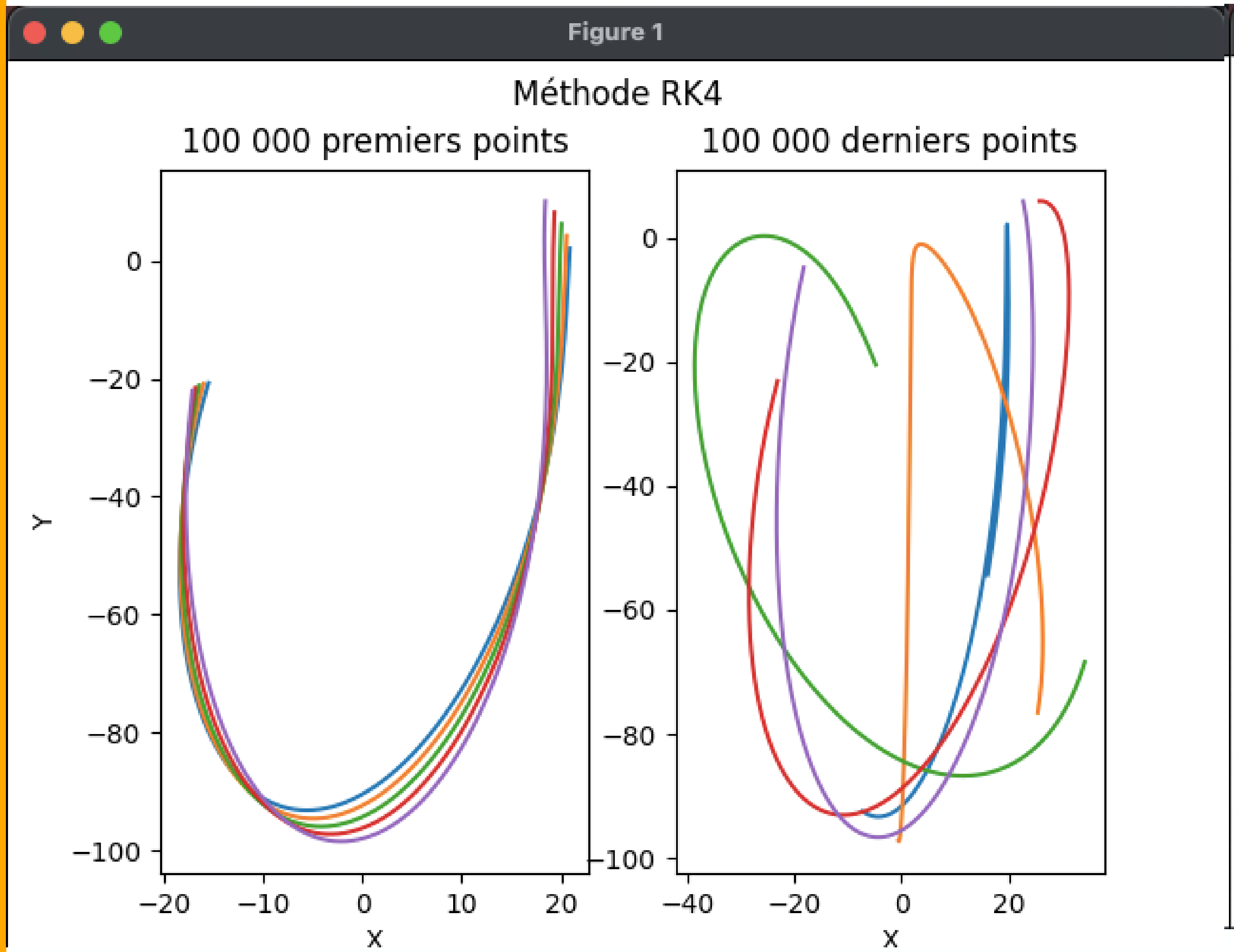


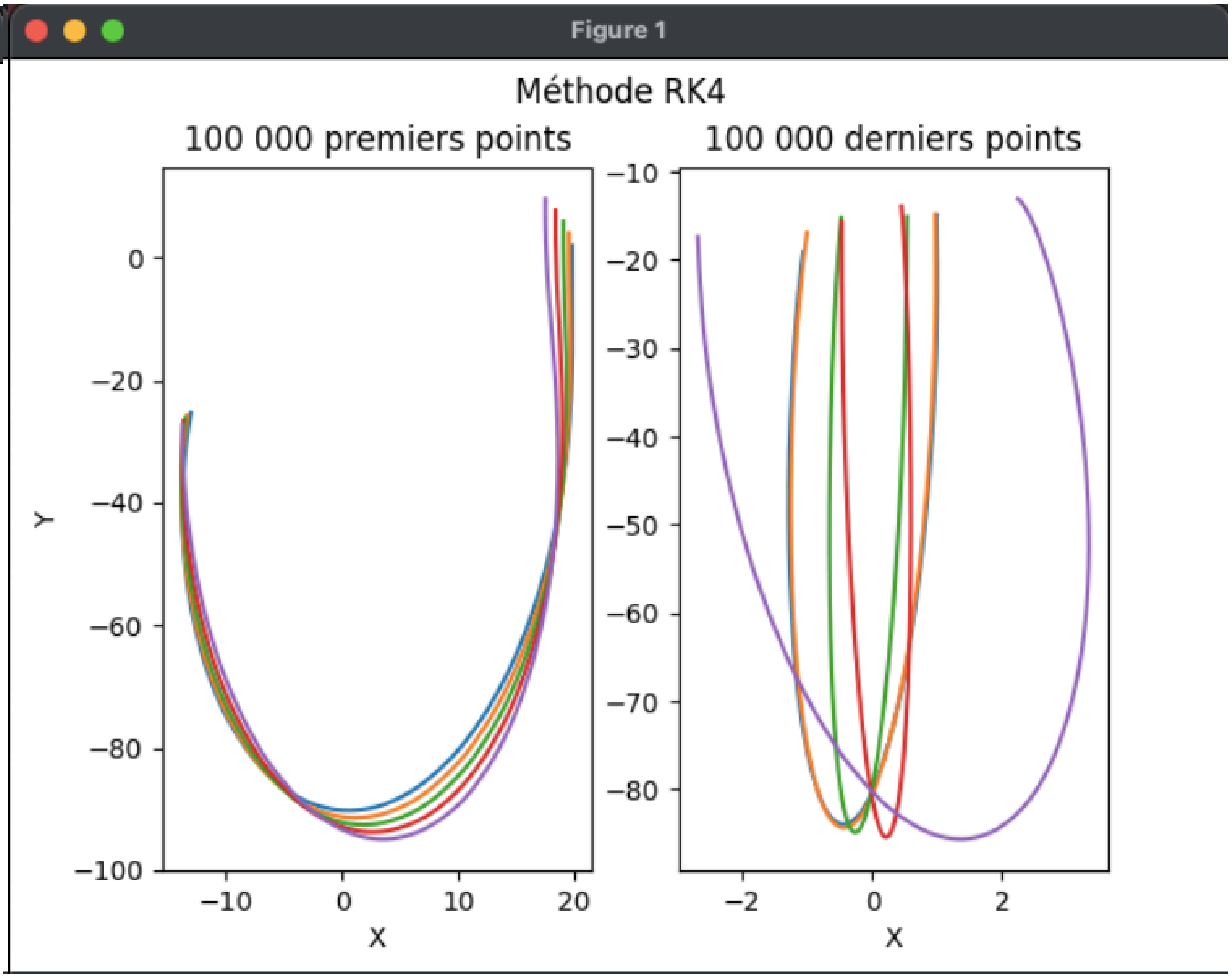


Caractère chaotique

En ce qui concerne la dépendance aux conditions initiales, on peut analyser la différence de comportement entre les courbes.
Ceci est la variation des conditions initiales entre chaque courbe :

$$\Delta l = 0.1 \quad \Delta\theta = 0.0001 \quad \Delta\dot{l} = 0.01 \quad \Delta\dot{\theta} = 0.0001$$





MERCI POUR VOTRE ECOUTE !