LA RESOLUTION NUMERIQUE DES EQUATIONS DIFFERENTIELLES

La résolution numérique des équations différentielles est une approche utilisée pour obtenir des solutions approchées à des équations différentielles. Cette méthode est couramment utilisée en physique, en ingénierie, en biologie et dans d'autres domaines où les équations différentielles sont omniprésentes.

Il existe plusieurs méthodes numériques pour résoudre les équations différentielles, parmi lesquelles les plus courantes sont les méthodes d'Euler, les méthodes de Runge-Kutta et les méthodes de différences finies. Ces méthodes consistent généralement à discrétiser le domaine de temps ou d'espace, puis à approximer les dérivées temporelles ou spatiales à l'aide de différences finies ou d'autres schémas numériques.

Sommaire

EL KHAWLANI AHMED

I- Résolution numérique des équations différentielle du premier ordre

1- Problème

Nous devons résoudre numériquement une équation différentielle qui s'écrit sous la forme :

$$\boldsymbol{y'(t) = F\big(t, y(t)\big)}$$

sur un intervalle [a, b]. La valeur initiale $y(a)$, appelée y_0, est connue et donnée.

Etape 1 : Nous procédons à la discrétisation de l'intervalle [a, b], le divisant en n petits intervalles. Cela nous donne une liste d'abscisses [t_0, t_1, t_2, ..., t_n] où t_i = a + i * h et h = (b - a) / n ;
Etape 2 : Nous devons trouver la liste des solutions [y_0, y_1, y_2, ..., y_n] où y_i est l'approximation de $y(t_i)$.

2- Comment trouver les y_i ? - Méthode d'Euler

La méthode d'Euler est l'une des méthodes les plus simples pour résoudre numériquement des équations différentielles. Elle est basée sur l'équation de réccurence suivante :

$$\boldsymbol{y_{i+1} = y_i + h\, F(t_i, y_i)}$$

Connaissant y_0, nous calculons y_1, à l'aide de la formule ci dessus, puis nous pouvons calculer y_2, et ainsi de suite.

3- Méthode d'Euler – Preuve

Nous avons :

$$y_{i+1} - y_i \approx y(t_{i+1}) - y(t_i)$$

$$= \int_{t_i}^{t_{i+1}} y'(t)dt$$

$$= \int_{t_i}^{t_{i+1}} F\big(t, y(t)\big)dt$$

Étant donné que l'intervalle [a, b] est subdivisé en n petits intervalles, nous pouvons supposer que la fonction F reste constante sur chaque sous-intervalle [t_i, t_{i+1}]. Par conséquent, la valeur de F dans l'intervalle [t_i, t_{i+1}] est représentée par F(t_i, y(t_i)).

EL KHAWLANI AHMED

Nous avons donc :

$$y_{i+1} - y_i \approx \int_{t_i}^{t_{i+1}} F\big(t_i, y(t_i)\big)dt$$

$$\approx \int_{t_i}^{t_{i+1}} F(t_i, y_i)dt$$

$$= F(t_i, y_i) \text{ x } \int_{t_i}^{t_{i+1}} dt$$

$$= (t_{i+1} - t_i)\, F(t_i, y_i)$$

$$= h\, F(t_i, y_i)$$

D'où la relation d'Euler :

$$\boldsymbol{y_{i+1} = y_i + h\, F(t_i, y_i)}$$

Pour obtenir une solution très proche de la solution mathématique exacte, nous devons augmenter le nombre de subdivisions n. En augmentant n, les intervalles $[t_i, t_{i+1}]$ deviennent plus petits, ce qui rend l'hypothèse que nous avons faite dans la preuve plus valide.

4- Fonction Euler sur Python

Nous définissons une fonction prenant en paramètres la fonction F, les bornes de l'intervalle de résolution a et b, la condition initiale y_0, et le nombre de subdivisions n. Cette fonction retourne un tuple contenant deux listes : la première liste représente les abscisses obtenues après la division de l'intervalle [a, b] en n intervalles, tandis que la deuxième liste représente les solutions (les images des abscisses de la première liste par la solution de l'équation différentielle :

$$\boldsymbol{y'(t) = F\big(t, y(t)\big)}$$

```
def Euler(F, a, b, y0, n):
  # Pas de discrétisation
  h = (b - a)/n
  # La liste des abscisses initialisé par a
  T = [a]
  # La liste des solutions initialisé par y0
  Y = [y0]
  for i in range(1, n + 1):
    # Nous ajoutons ti à la liste des abscisses T
    T.append(a + i * h)
    # Nous calculons yi à l'aide de la formule d'Euler
    y = Y[i - 1] + h * F(T[i - 1], Y[i - 1])
    # Nous ajoutons yi à la liste des solutions Y
    Y.append(y)
  # Nous retournons le tuple qui contient les deux listes T et Y
  return T, Y
```

EL KHAWLANI AHMED

5- Exemple 1 de résolution

Nous voulons résoudre l'équation différentielle suivante :

$$\boldsymbol{2y'(t) - y(t) = 0}$$

sur l'intervalle [0, 10] avec $y(0) = 1$.

Tout d'abord on écrit l'équation différentielle sous la forme $y'(t) = F\big(t, y(t)\big)$.

Nous avons $2y'(t) - y(t) = 0$ donc $y'(t) = \frac{y(t)}{2} = F\big(t, y(t)\big)$.

Avec F une fonction tel que $F(m, n) = \frac{n}{2}$

On définit F sur python :

```
def F(m, n):
  return n / 2
```

On appelle la fonction Euler avec les différentes valeurs de n (10, 100, 1000)

```
E10 = Euler(F, 0, 10, 1, 10)
E100 = Euler(F, 0, 10, 1, 100)
E1000 = Euler(F, 0, 10, 1, 1000)
```

Puisque la fonction Euler retourne un tuple donc les variables E10, E100 et E1000 sont des tuples.

Mathématiquement la solution de notre équation différentielle est y(t) = $e^{t/2}$

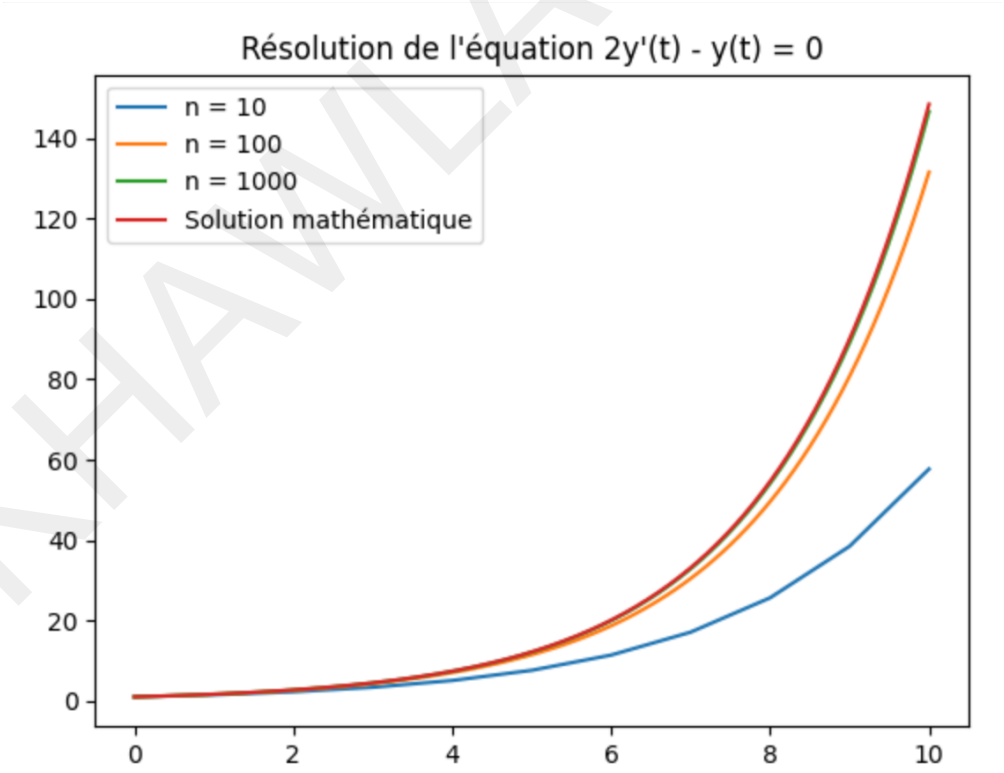
Nous traçons les courbes des solutions obtenues avec les différentes valeurs de n ainsi que la solution mathématique pour pouvoir faire une comparaison entre les différentes courbes.

Le code suivant permet ces courbes dans une même figure :

EL KHAWLANI AHMED

```
X10, Y10 = E10
X100, Y100 = E100
X1000, Y1000 = E1000

import numpy as np
import matplotlib.pyplot as plt

# On définit la solution mathématique
def solution_math(t):
  return np.exp(t / 2)

# On trace la courbe de la solution pour n = 10
plt.plot(X10, Y10, label = 'n = 10')
# On trace la courbe de la solution pour n = 100
plt.plot(X100, Y100, label = 'n = 100')
# On trace la courbe de la solution pour n = 1000
plt.plot(X1000, Y1000, label = 'n = 1000')
# On trace la solution mathématique
plt.plot(X1000, solution_math(np.array(X1000)), label = 'Solution mathématique')

# Afficher un titre pour notre figure
plt.title("Résolution de l'équation 2y'(t) - y(t) = 0")
# Afficher la légende
plt.legend()

plt.show()
```

L'éxécution du code ci dessus affiche la figure suivante :



On observe que lorsque le nombre de subdivisions, n, augmente, on se rapproche de plus en plus de la solution mathématique.

EL KHAWLANI AHMED

6- Exemple 2 de résolution

Nous voulons résoudre l'équation différentielle suivante :

$$\mathbf{7y'(t) + 2y(t) = 2t^3 - 5t^2 + 4t - 1}$$

sur l'intervalle [0, 1] avec $y(0) = 0$

Tout d'abord on écrit l'équation différentielle sous la forme $y'(t) = F\big(t, y(t)\big)$.

Nous avons $y'(t) = \frac{2t^3-5t^2+4t-1-2y(t)}{7}$

Donc $y'(t) = \frac{2t^3-5t^2+4t-1-2y(t)}{7} = F\big(t, y(t)\big)$

Avec F une fonction tel que $F(m, n) = \frac{2m^3-5m^2+4m-1-2n}{7}$

On définit F sur python :

```
def F(m,n):
  return (2 * m ** 3 - 5 * m ** 2 + 4 * m - 1 - 2 * n) / 7
```

On appelle la fonction Euler avec les différentes valeurs de n (10, 50, 100)

```
E10 = Euler(F, 0, 1, 0, 10)
E50 = Euler(F, 0, 1, 0, 50)
E100 = Euler(F, 0, 1, 0, 100)
```

Puisque la fonction Euler retourne un tuple donc les variables E10, E50 et E100 sont des tuples.

Mathématiquement la solution de notre équation différentielle est :

$$y(t) = t^3 - 13\,t^2 + 93t\ - 326\ + 326\,e^{-2t/7}$$

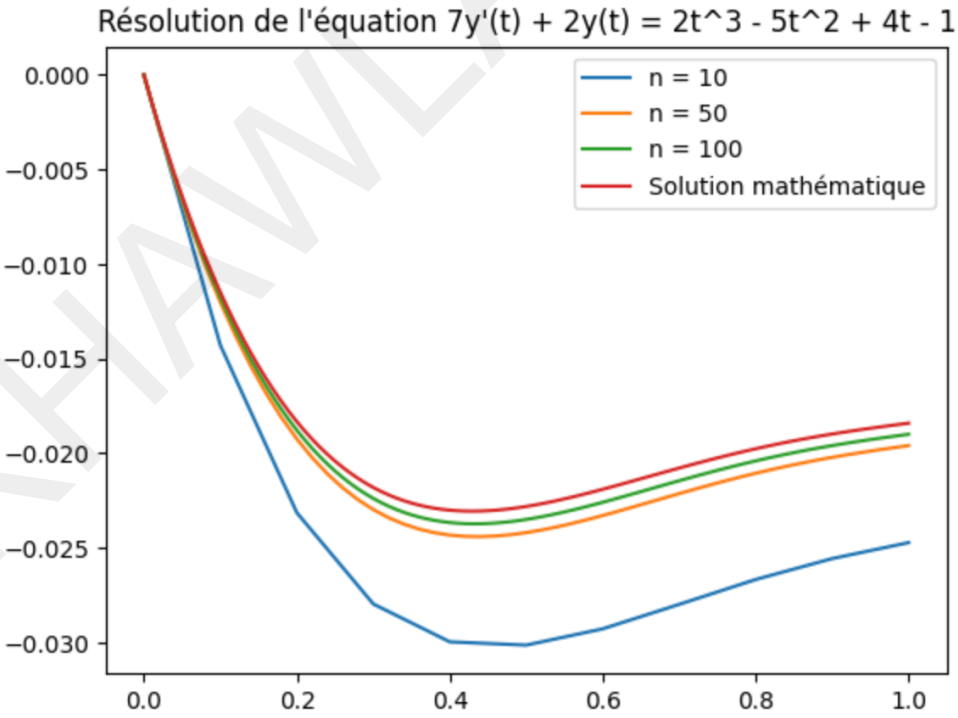
Nous traçons les courbes des solutions obtenues avec les différentes valeurs de n ainsi que la solution mathématique pour pouvoir faire une comparaison entre les différentes courbes.

Le code suivant permet ces courbes dans une même figure :

EL KHAWLANI AHMED

```
X10, Y10 = E10
X50, Y50 = E50
X100, Y100 = E100

import numpy as np
import matplotlib.pyplot as plt

# On définit la solution mathématique
def solution_math(t):
  return t ** 3 - 13 * t ** 2 + 93 * t - 326 + 326 * np.exp(-2 * t / 7)

# On trace la courbe de la solution pour n = 10
plt.plot(X10, Y10, label = 'n = 10')
# On trace la courbe de la solution pour n = 50
plt.plot(X50, Y50, label = 'n = 50')
# On trace la courbe de la solution pour n = 100
plt.plot(X100, Y100, label = 'n = 100')
# On trace la solution mathématique
plt.plot(X100, solution_math(np.array(X100)), label = 'Solution mathématique')

# Afficher un titre pour notre figure
plt.title("Résolution de l'équation 7y'(t) + 2y(t) = 2t^3 - 5t^2 + 4t - 1")
# Afficher la légende
plt.legend()

plt.show()
```

L'éxécution du code ci dessus affiche la figure suivante :



On observe une autre fois que lorsque le nombre de subdivisions, n, augmente, on se rapproche de plus en plus de la solution mathématique.

EL KHAWLANI AHMED

II- Résolution numérique d'un système d'équations différentielles du premier ordre

1- Exemple 1 de résolution

Nous voulons résoudre le système d'équations différentielles suivant à l'aide de la méthode d'Euler :

$$(\boldsymbol{S}): \begin{cases} \boldsymbol{y'(t) = 2y(t) - z(t) + 4t} \\ \boldsymbol{z'(t) = 3y(t) + z(t) - 2} \end{cases}$$

La résolution doit être dans l'intervalle [0, 1] avec $y(0) = z(0) = 1$ avec un nombre de subdivisions n = 100.

Nous devons tout d'abord transformer le système $(\boldsymbol{S})$ en une équation différentielle qui s'écrit sous la forme :

$$\boldsymbol{y'(t) = F(t, y(t))}$$

Nous posons $Y(t) = \begin{pmatrix} y(t) \\ z(t) \end{pmatrix}$. Nous aurons donc $Y'(t) = \begin{pmatrix} y'(t) \\ z'(t) \end{pmatrix}$.

Ce qui veut dire : $Y'(t) = \begin{pmatrix} 2y(t) - z(t) + 4t \\ 3y(t) + z(t) - 2 \end{pmatrix} = \begin{pmatrix} 2 & -1 \\ 3 & 1 \end{pmatrix} . \begin{pmatrix} y(t) \\ z(t) \end{pmatrix} + \begin{pmatrix} 4t \\ -2 \end{pmatrix}$

D'où $Y'(t) = \begin{pmatrix} 2 & -1 \\ 3 & 1 \end{pmatrix} . Y(t) + \begin{pmatrix} 4t \\ -2 \end{pmatrix}$

Il est maintenant possible d'écrire $Y'(t) = F(t, Y(t))$ avec $F(m, n) = \begin{pmatrix} 2 & -1 \\ 3 & 1 \end{pmatrix} . n + \begin{pmatrix} 4m \\ -2 \end{pmatrix}$

Nous procédons maintenant à la résolution de l'équation différentielle du premier ordre :

$$\boldsymbol{Y'(t) = F(t, Y(t))}$$

Sur l'intervalle [0, 1] avec $Y(0) = \begin{pmatrix} y(0) \\ z(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

On définit F sur python :

```
import numpy as np
def F(m, n):
  A = np.array([[2, -1], [3, 1]])
  return np.dot(A, n) + np.array([4 * m, -2])
```

On appelle la fonction Euler avec la fonction F, les bornes d'intervalles 0 et 1, la condition initiale [1, 1] et le nombre de subdivision n = 100.

```
R = Euler(F, 0, 1, np.array([1, 1]), 100)
```


EL KHAWLANI AHMED

Puisque la fonction Euler retourne un tuple, la variable R est un tuple. Son premier élément est une liste qui contient les abscisses obtenus après la discrétisation de l'intervalle [0, 1] et son deuxième éléments est la liste des Y (Attention on parle toujours du grand Y).

```
T = R[0]      # Liste des abscisses
Y = R[1]      # Liste des Yi
```

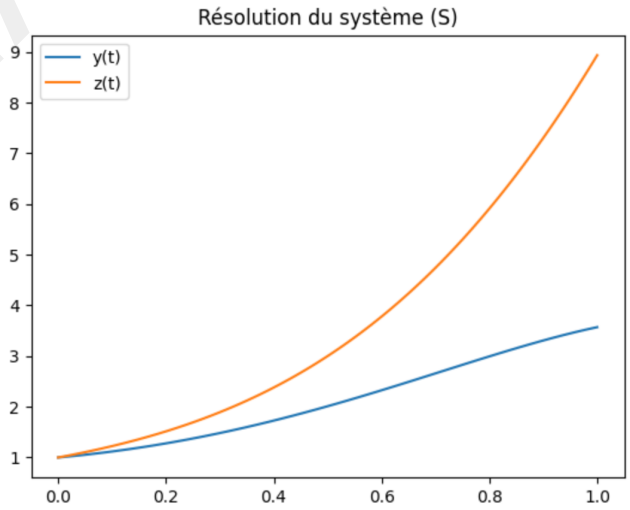
Les éléments de Y sont aussi des listes puisque chaque Y_i est représenté par $[y_i , z_i]$.
Les y_i seront donc les premiers éléments de chaque Y[i] et les z_i sont les deuxièmes éléments de chaque Y[i].
Puisque Y est une liste de listes de 2 éléments, nous pouvons dire que Y est une matrice de deux colonnes, la première colonne contient les y_i et la deuxième colonne contient les z_i . On transforme Y en une matrice array (du module numpy) pour pouvoir extraire facilement les colonnes.

```
Y = np.array(Y)
y = Y[:, 0]
z = Y[:, 1]
```

Maintenant nous pouvons dessiner les courbes des deux fonctions y et z.

```
import matplotlib.pyplot as plt

# Dessiner la courbe de y(t)
plt.plot(T, y, label = 'y(t)')
# Dessiner la courbe de z(t)
plt.plot(T, z, label = 'z(t)')

plt.legend()
plt.title("Résolution du système (S)")

plt.show()
```

L'éxécution du code ci dessus affiche la figure suivante :



EL KHAWLANI AHMED

2- Exemple 2 de résolution

Nous voulons résoudre le système d'équations différentielles suivant à l'aide de la méthode d'Euler :

$$(E): \begin{cases} x'(t) - 2x(t) + z(t) = 4 \\ y(t) - x(t) + 3y'(t) - z(t) = 2t \\ y(t) - z'(t) - 3z(t) = 0 \end{cases}$$

La résolution doit être dans l'intervalle [2, 5] avec $x(2) = 0.5,\ y(2) = 1\ et\ z(2) = 2$ avec un nombre de subdivisions n = 1000.

Pour simplifier nous devons d'abord réecrir le système (E) en posant les dérivés dans le côté gauche des équations :

$$(E) \quad est\ équivalent\ à \begin{cases} x'(t) = 2x(t) - z(t) + 4 \\ y'(t) = \frac{1}{3}x(t) - \frac{1}{3}y(t) + \frac{1}{3}z(t) + \frac{2}{3}t \\ z'(t) = y(t) - 3z(t) \end{cases}$$

Nous devons tout d'abord transformer le système $(\boldsymbol{E})$ en une équation différentielle qui s'écrit sous la forme :

$$\boldsymbol{y'(t) = F(t, y(t))}$$

Nous posons $Y(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}$ Nous aurons donc $Y'(t) = \begin{pmatrix} x'(t) \\ y'(t) \\ z'(t) \end{pmatrix}$

Ce qui veut dire : $Y'(t) = \begin{pmatrix} 2x(t) - z(t) + 4 \\ \frac{1}{3}x(t) - \frac{1}{3}y(t) + \frac{1}{3}z(t) + \frac{2}{3}t \\ y(t) - 3z(t) \end{pmatrix}$

D'où $Y'(t) = \begin{pmatrix} 2 & 0 & -1 \\ \frac{1}{3} & \frac{-1}{3} & \frac{1}{3} \\ 0 & 1 & -3 \end{pmatrix} . \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} + \begin{pmatrix} 4 \\ \frac{2t}{3} \\ 0 \end{pmatrix} = \begin{pmatrix} 2 & 0 & -1 \\ \frac{1}{3} & \frac{-1}{3} & \frac{1}{3} \\ 0 & 1 & -3 \end{pmatrix} . Y(t) + \begin{pmatrix} 4 \\ \frac{2t}{3} \\ 0 \end{pmatrix}$

Il est maintenant possible d'écrire $Y'(t) = F(t, Y(t))$ avec :

$$F(m, n) = \begin{pmatrix} 2 & 0 & -1 \\ \frac{1}{3} & \frac{-1}{3} & \frac{1}{3} \\ 0 & 1 & -3 \end{pmatrix} . n + \begin{pmatrix} 4 \\ \frac{2m}{3} \\ 0 \end{pmatrix}$$

Nous procédons maintenant à la résolution de l'équation différentielle du premier ordre :

$$\boldsymbol{Y'(t) = F(t, Y(t))}$$

EL KHAWLANI AHMED

Sur l'intervalle [2, 5] avec $Y(2) = \begin{pmatrix} x(2) \\ y(2) \\ z(2) \end{pmatrix} = \begin{pmatrix} 0.5 \\ 1 \\ 2 \end{pmatrix}$

On définit F sur python :

```
import numpy as np
def F(t, Y):
  A = np.array([[2, 0, -1],[1/3, -1/3, 1/3],[0, 1, -3]])
  return np.dot(A, Y) + np.array([4, 2 * t / 3, 0])
```

On appelle la fonction Euler avec la fonction F, les bornes d'intervalles 2 et 5, la condition initiale [0.5, 1, 2] et le nombre de subdivision n = 1000.

```
R = Euler(F, 2, 5, np.array([0.5, 1, 2]), 1000)
```

Puisque la fonction Euler retourne un tuple, la variable R est un tuple. Son premier élément est une liste qui contient les abscisses obtenus après la discrétisation de l'intervalle [2, 5] et son deuxième éléments est la liste des Y (Attention on parle toujours du grand Y).

```
T = R[0]     # Liste des abscisses
Y = R[1]     # Liste des Yi
```

Les éléments de Y sont aussi des listes puisque chaque Y_i est représenté par $[x_i , y_i , z_i]$.
Les x_i seront donc les premiers éléments de chaque Y[i], les y_i seront les deuxièmes éléments de chaque Y[i] et les z_i seront les troisièmes éléments de chaque Y[i].
Puisque Y est une liste de listes de 3 éléments, nous pouvons dire que Y est une matrice de trois colonnes, la première colonne contient les x_i , la deuxième colonne contient les y_i et la troisième contient les z_i . On transforme Y en une matrice array (du module numpy) pour pouvoir extraire facilement les colonnes.

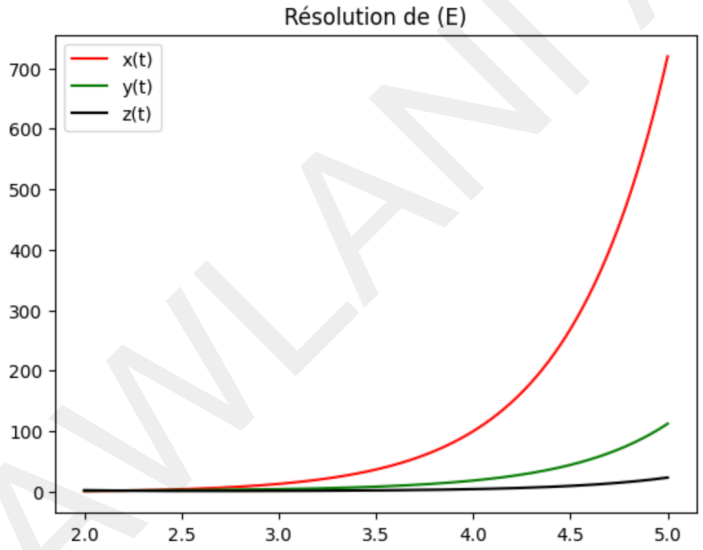
```
Y = np.array(Y)
x = Y[:, 0]
y = Y[:, 1]
z = Y[:, 2]
```

Maintenant nous pouvons dessiner les courbes des trois fonctions x, y et z.

EL KHAWLANI AHMED

```
import matplotlib.pyplot as plt

# Traçage la courbe de x(t) avec la couleur rouge
plt.plot(T, x, label = 'x(t)', color = 'red')
# Traçage la courbe de x(t) avec la couleur verte
plt.plot(T, y, label = 'y(t)', color = 'green')
# Traçage la courbe de x(t) avec la couleur noire
plt.plot(T, z, label = 'z(t)', color = 'black')

# Afficher la légenre
plt.legend()
# Afficher le titre de la figure
plt.title('Résolution de (E)')

plt.show()
```

L'éxécution du code ci dessus affiche la figure suivante :



EL KHAWLANI AHMED

III- Résolution numérique des équations différentielle du deuxième ordre

Nous voulons résoudre l'équation différentielle suivante :

$$(\boldsymbol{D})\colon \boldsymbol{y''(t) - 2ty'(t) + 6y(t) = t^2}$$

sur l'intervalle [0, 2] avec $y(0) = 5$ et $y'(0) = 2.5$ avec un nombre de subdivisions n = 500.

Pour résoudre une équation différentielle du deuxième ordre on la transforme d'abord en un système de deux équations différentielles.

Posons $z = y'$ on aura donc $z' = y''$, d'où :

$$(\boldsymbol{D})\ \boldsymbol{est\ equivalent}\ \begin{cases} \boldsymbol{y'(t) = z(t)} \\ \boldsymbol{z'(t) = -6y(t) + 2tz(t) + t^2} \end{cases}$$

Nous posons $Y(t) = \begin{pmatrix} y(t) \\ z(t) \end{pmatrix}$. Nous aurons donc $Y'(t) = \begin{pmatrix} y'(t) \\ z'(t) \end{pmatrix}$.

Ce qui veut dire : $Y'(t) = \begin{pmatrix} z(t) \\ -6y(t)+2tz(t)+t^2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -6 & 2t \end{pmatrix} . \begin{pmatrix} y(t) \\ z(t) \end{pmatrix} + \begin{pmatrix} 0 \\ t^2 \end{pmatrix}$

D'où $Y'(t) = \begin{pmatrix} 0 & 1 \\ -6 & 2t \end{pmatrix} . Y(t) + \begin{pmatrix} 0 \\ t^2 \end{pmatrix}$

Il est maintenant possible d'écrire $Y'(t) = F(t, Y(t))$ avec $F(m, n) = \begin{pmatrix} 0 & 1 \\ -6 & 2m \end{pmatrix} . n + \begin{pmatrix} 0 \\ m^2 \end{pmatrix}$

Nous procédons maintenant à la résolution de l'équation différentielle du premier ordre :

$$\boldsymbol{Y'(t) = F(t, Y(t))}$$

Sur l'intervalle [0, 2] avec $Y(0) = \begin{pmatrix} y(0) \\ z(0) \end{pmatrix} = \begin{pmatrix} 5 \\ 2.5 \end{pmatrix}$

On définit F sur python :

```
import numpy as np
def F(m, n):
  A = np.array([[0, 1], [-6, 2 * m]])
  return np.dot(A, n) + np.array([0, m ** 2])
```

On appelle la fonction Euler avec la fonction F, les bornes d'intervalles 0 et 2, la condition initiale [5, 2.5] et le nombre de subdivision n = 500.

```
R = Euler(F, 0, 2, np.array([5, 2.5]), 500)
```

EL KHAWLANI AHMED

Puisque la fonction Euler retourne un tuple, la variable R est un tuple. Son premier élément est une liste qui contient les abscisses obtenus après la discrétisation de l'intervalle [0, 2] et son deuxième éléments est la liste des Y (Attention on parle toujours du grand Y).

```
T = R[0]     # Liste des abscisses
Y = R[1]     # Liste des Yi
```

Les éléments de Y sont aussi des listes puisque chaque Y_i est représenté par $[y_i , z_i]$.
Les y_i seront donc les premiers éléments de chaque Y[i] et les z_i sont les deuxièmes éléments de chaque Y[i].
Puisque Y est une liste de listes de 2 éléments, nous pouvons dire que Y est une matrice de deux colonnes, la première colonne contient les y_i et la deuxième colonne contient les z_i . On transforme Y en une matrice array (du module numpy) pour pouvoir extraire facilement les colonnes.
Nous nous intéressons qu'à la première colonne qui contient les y (Le but était dès le début de résoudre l'équation (D)).

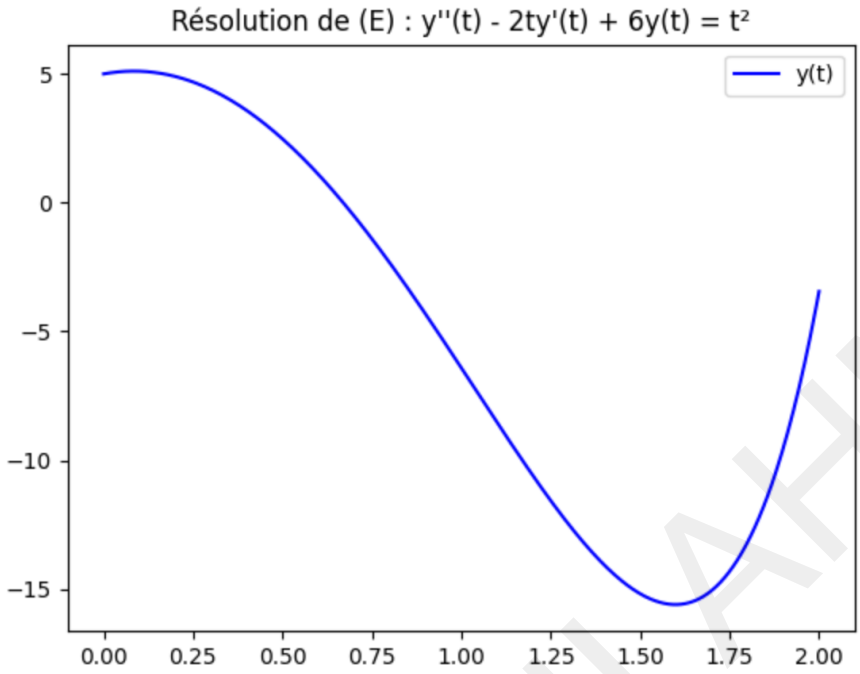
```
Y = np.array(Y)
y = Y[:, 0]
```

Maintenant nous pouvons dessiner la courbe de la fonction y.

```
import matplotlib.pyplot as plt

# Traçage de la courbe de y avec la couleur bleue
plt.plot(T, y, label = 'y(t)', color = 'blue')

# Ajouter un titre à la figure
plt.title("Résolution de (E) : y''(t) - 2ty'(t) + 6y(t) = t²")

# Ajouter une légende
plt.legend()

plt.show()
```

L'éxécution du code ci dessus affiche la figure suivante :

EL KHAWLANI AHMED



EL KHAWLANI AHMED