TP N° 1

Exercice 1

1 Import

Import

- `numpy` : Les fonctions mathématiques prédéfinies, et `linspace` pour créer les vecteur des coordonnées x de chaque fonctions.
- `matplotlib` : Dessine les graphes.
- `collections` : Crée une nouvelle structure avec `namedtuple`.

```
import numpy as np
import matplotlib.pyplot as plt
from collections import namedtuple
```

2 Définition Des Fonctions Mathématiques

```
def function_1(x):
    return x**6 - x -1;

def function_2(x):
    return 1 - 1/4 * np.cos(x)

def function_3(x):
    return np.cos(x) - np.exp(-x)

def function_4(x):
    return x**4 - 56.101*x**3 + 785.6561*x**2 - 72.7856*x + 0.0078
```

3 Génération des Vecteurs

```
x1 = np.linspace(-2, 2, 400)
x2 = np.linspace(-2*np.pi, 2*np.pi, 400)
x3 = np.linspace(-1, 2*np.pi, 400)
x4 = np.linspace(-0.1, 0.5, 400)

y1 = function_1(x1)
y2 = function_2(x2)
y3 = function_3(x3)
y4 = function_4(x4)
```

4 Nouvelle Structure root

root

On a créé une nouvelle structure en utilisant `namedtuple` root. Elle représente les racines pour $f(x) = 0$ et possède trois attributs :

- **position** : coordonnée x de la racine
- **a** : extrémité gauche de l'intervalle auquel la racine appartient
- **b** : extrémité droite de l'intervalle auquel la racine appartient

```
root = namedtuple("root", ["position","a", "b"])
```

5 Scatter

Scatter

- `scatter_single(rootElement, colorValue, marker)` : Dessine un point qui représente une racine pour $f(x)$. Elle vérifie si les extrémités de l'intervalle de la racine sont égales. Si oui, le label est : $\alpha = a$, sinon $\alpha \in [a, b]$. On utilise cette fonction si f admet une seule racine.
- `scatter_many(rootElement, colorValue, index, marker)` : Dessine un point qui représente une racine pour $f(x)$. Elle vérifie si les extrémités de l'intervalle de la racine sont égales. Si oui, le label est : $\alpha_{\text{index}} = a$, sinon $\alpha_{\text{index}} \in [a, b]$. On utilise cette fonction si f admet plusieurs racines.

Paramètres :

- **rootElement** : de type `root`
- **colorValue** : couleur du marqueur
- **index** : indice de la racine
- **marker** : style du marqueur

```
def scatter_single(rootElement,colorValue,marker):
    if rootElement.a == rootElement.b :
      plt.scatter(rootElement.position, 0, color = colorValue , marker = marker,label = fr"$\alpha = {rootElement.a}$")
    else:
      plt.scatter(rootElement.position, 0, color = colorValue , marker = marker,label = fr"$\alpha \in [{rootElement.a},{rootElement.b}]$")

def scatter_many(rootElement,colorValue,index,marker):
    if rootElement.a == rootElement.b :
      plt.scatter(rootElement.position, 0, color = colorValue , marker = marker,label = fr"$\alpha_{index} = {rootElement.a}$")
    else:
      plt.scatter(rootElement.position, 0, color = colorValue , marker = marker,
        label = fr"$\alpha_{index} \in [{rootElement.a},{rootElement.b}]$")
```

6 Draw

Draw

La fonction `draw()` dessine le graphe et les points des racines grâce à la liste des racines `rootList` et aux fonctions `scatter` que nous avons précédemment définies, Et retourne la valeur de l'indice incrémenté de 1 pour la mettre à jour.

Paramètres :

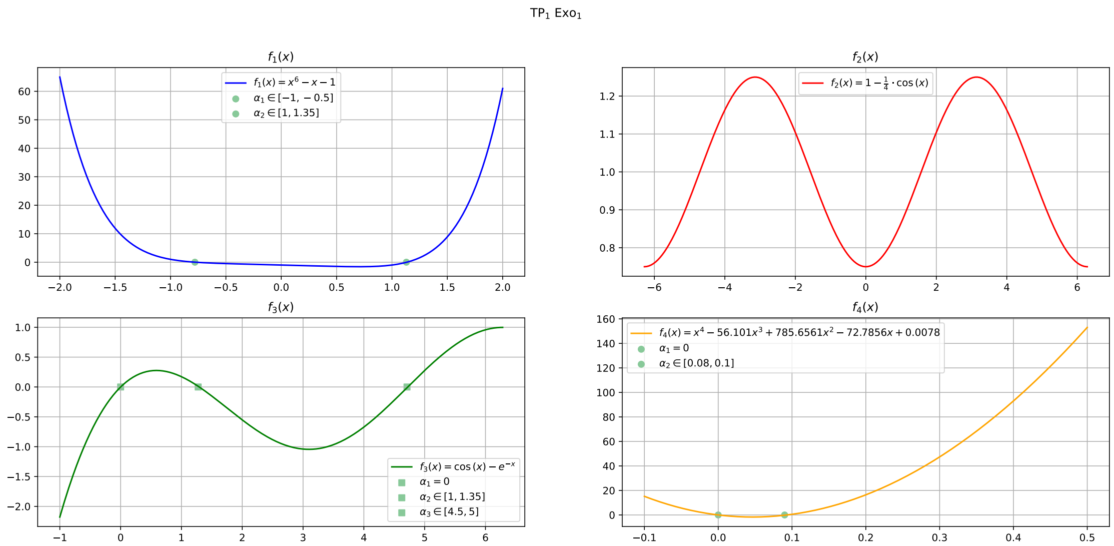
- **x** : Vecteur numpy des coordonnées x
- **y** : Vecteur numpy des coordonnées y
- **functionLabel** : Label de la fonction
- **colorValue** : Couleur du graphe
- **rootList** : Liste de racines de type `root`
- **index** : Indice du subplot
- **title** : Titre du graph
- **marker** (optionnel, valeur par défaut = `'o'`) : Style du marqueur pour les racines

```python
def draw(x,y,functionLabel,colorValue,rootList,index,title,marker='o'):
    plt.subplot(2,2,index)
    plt.plot(x, y, label=fr"{functionLabel}", color=colorValue)
    sizeList = len(rootList)

    if sizeList == 1:
        scatter_single(rootList[0],'#88c999',marker)
    elif sizeList > 1 :
      for i in range(sizeList):
        scatter_many(rootList[i],'#88c999',i+1,marker)
    plt.legend()
    plt.grid()
    plt.title(title)
    return index+1
```

7 Reste Du Code

```python

index = 1
index = draw(x1,y1,fr"$f_{index}(x) = x^6 - x - 1$","blue",[root(-0.78,-1,-0.5),root(1.13,1,1.35)],index,r"$f_1(x)$")
index = draw(x2, y2,fr"$f_{index}(x) = 1 - \frac{{1}}{{4}} \cdot \cos{{(x)}}$","red",[],index,r"$f_2(x)$",'.')
index = draw(x3, y3,fr"$f_3(x) = \cos{{(x)}} - e^{{-x}}$","green",[root(0,0,0),root(1.275,1,1.35),root(4.71,4.5,5)],index,r"$f_3(x)$",'s')
index = draw(x4, y4,fr"$f_4(x) = x^4 - 56.101x^{{3}} + 785.6561x^{{2}} - 72.7856x + 0.0078$","orange",[root(0,0,0),root(0.09,0.08,0.1)], index,r"$f_4(x)$",'o')

plt.suptitle(r"TP$_1$ Exo$_1$")
plt.savefig("fig.pdf")
plt.show()
```

8 Figure



TP$_1$ Exo$_1$
$f_1(x)$
$f_1(x) = x^6 - x - 1$
$\alpha_1 \in [-1, -0.5]$
$\alpha_2 \in [1, 1.35]$
$f_2(x)$
$f_2(x) = 1 - \frac{1}{4} \cdot \cos(x)$
$f_3(x)$
$f_3(x) = \cos(x) - e^{-x}$
$\alpha_1 = 0$
$\alpha_2 \in [1, 1.35]$
$\alpha_3 \in [4.5, 5]$
$f_4(x)$
$f_4(x) = x^4 - 56.101x^3 + 785.6561x^2 - 72.7856x + 0.0078$
$\alpha_1 = 0$
$\alpha_2 \in [0.08, 0.1]$

Exercice 2

1 Import

Import

- `numpy` : Les fonctions mathématiques prédéfinies, et `linspace` pour créer les vecteur des coordonnées x de chaque fonctions.
- `matplotlib` : Dessine les graphes.
- `collections` : Crée une nouvelle structure avec `namedtuple`.

```python
import numpy as np
import matplotlib.pyplot as plt
from collections import namedtuple
```

2 Définition Des Fonctions Mathématiques

```python
def function_1(x):
    return np.log(x) - x + 2

def function_2(x):
    return np.exp(x) + x**2/2 + x - 1
```

3 ϵ

```python
eps = 10 ** (-5)
```

4 Génération des Vecteurs

```python
x_1 = np.linspace(0.1,5,400)
x_2 = np.linspace(-2,2,400)

y_1 = function_1(x_1)
y_2 = function_2(x_2)
```

5 Nouvelle Structure root

root

On a créé une nouvelle structure en utilisant `namedtuple` root. Elle représente les racines pour $f(x) = 0$ et possède trois attributs :

- **position** : coordonnée x de la racine
- **a** : extrémité gauche de l'intervalle auquel la racine appartient
- **b** : extrémité droite de l'intervalle auquel la racine appartient
- **error** : Estimation de l'erreur de la method dichotomie.
- **iteration**: Nombre d'iteration de la method dichotomie.

```python
root = namedtuple("root", ["position","a", "b","error","iteration"])
```

6 Implementation De La Method Dichotomie

6.1 Estimation De L'erreur

```python
def ErrorEstimation(a,b):
    return (b-a)/2
```

6.2 Algorithm De Dichotomie

```python
def dichotomy(eps,a,b,function,max_iter=100):

    rootList = []
    n = 0
    a_0 = a
    b_0 = b

    rootList.append(root("None",a,b,"None","Before Algo Start"))

    while (error := ErrorEstimation(a,b)) > eps and n <=max_iter :

        x = (a+b)/2

        if function(x) * function(a) < 0:
            b = x
        elif function(x) * function(b) < 0:
            a = x
        else:
            rootList.append(root(x,a,b,"No Error",n))
            return rootList

        rootList.append(root(x,a,b,f"{error:.2e}",n))
        n = n+1

    rootList.append(root((a+b)/2,a,b,f"{error:.2e}",n))
    return rootList
```

7 Dessiner

7.1 Scatter

```
def scatter_many(a,b,position,error,index,marker="*"):
    if error == "No Error":
        plt.scatter(position,0,marker=marker,color="#88c999",label=fr"$\alpha_{index} \in [{a},{b}] = {root.position}$")
    else:
        plt.scatter(position,0,marker=marker,color="#88c999",label=fr"$\alpha_{index} \in [{a},{b}] \approx {position}$")


def scatter_single(a,b,position,error,marker="*"):
    if error=="No Error":
        plt.scatter(position,0,marker=marker,color="#88c999",label=fr"$\alpha \in [{a},{b}] = {position}$")
    else:
        plt.scatter(position,0,marker=marker,color="#88c999",label=fr"$\alpha \in [{a},{b}] \approx {position}$")

def scatter(rootMatrix,marker="*"):
    size = len(rootMatrix)

    if size == 1:
        firstElement = rootMatrix[0][0]
        lastElement = rootMatrix[0][-1]

        scatter_single(firstElement.a,firstElement.b,lastElement.position,lastElement.error)
    else:
        for index in range(size):
            firstElement = rootMatrix[index][0]
            lastElement = rootMatrix[index][-1]
            scatter_many(firstElement.a,firstElement.b,lastElement.position,lastElement.error,index)
```

7.2 Graphes

```
def draw_graph(x,y,index,color,label,rootMatrix,marker="*"):
    plt.subplot(1,2,index)
    plt.plot(x,y,color=color,label=fr"$f(x)_{index} = {label} $")
    plt.title(fr"$f_{index}(x)$")
    scatter(rootMatrix)
    plt.legend()
    plt.grid()
    return index+1
```

7.3 Tables

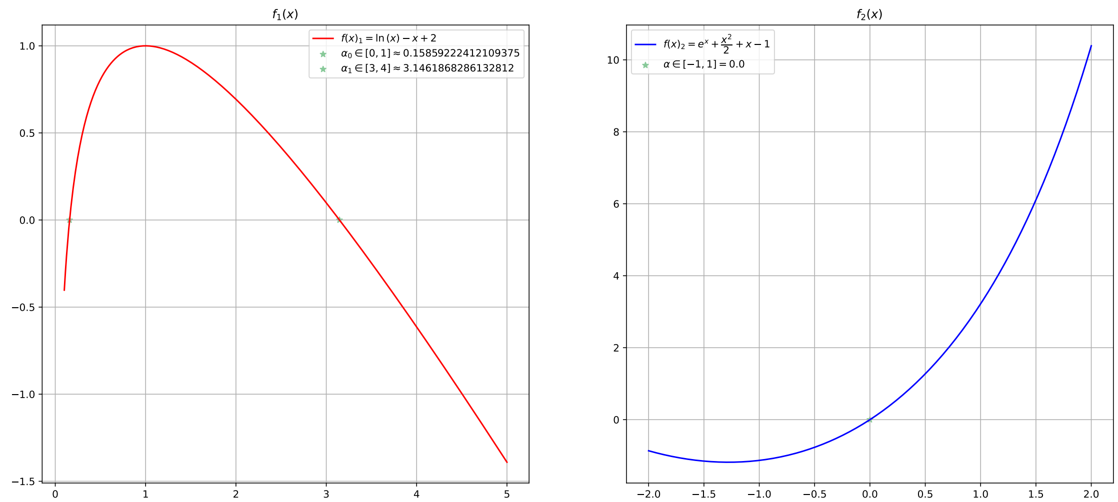
```
def draw_table(rootMatrix,index):

    for rootList in rootMatrix:
        plt.subplot(2,2,index)
        plt.subplots_adjust(top=0.85)
        plt.axis("off")
        data = [[r.position, r.a, r.b, r.error, r.iteration] for r in rootList]
        headers = ["Position", "a", "b", "Error", "Iteration"]
        the_table = plt.table(cellText=data, colLabels=headers,fontsize=12,loc="center", cellLoc="left")
        the_table.auto_set_column_width(col=list(range(len(headers))))
        index = index+1

    return index
```

8 Rest Du Code

```
label_1 = r"\ln{(x)} - x + 2"
label_2 = r"e^{x} + \dfrac{x^2}{2} + x - 1"

color_1 = "red"
color_2 = "blue"

index = 1

rootList_1 = dichotomy(eps,0,1,function_1)
rootList_2 = dichotomy(eps,3,4,function_1)

rootMatrix_1 = [rootList_1 , rootList_2]

rootMatrix_2 = [dichotomy(eps,-1,1,function_2)]

index = draw_graph(x_1,y_1,index,color_1,label_1,rootMatrix_1)
index = draw_graph(x_2,y_2,index,color_2,label_2,rootMatrix_2)

plt.suptitle(r"Tp$_{1}$ Exo$_{2}$ Functions")

plt.figure(figsize=(10, 7))
index = 1
index = draw_table(rootMatrix_1,index)
index = draw_table(rootMatrix_2,index)

plt.suptitle(r"Tp$_{1}$ Exo$_{2}$ Tables")

plt.show()
```

9 Figures

$\text{Tp}_1\ \text{Exo}_2$ Functions



$\text{Tp}_1\ \text{Exo}_2$ Tables

Position	a	b	Error	Iteration
None	0	1	None	Before Algo Start
0.5	0	0.5	5.00e-01	0
0.25	0	0.25	2.50e-01	1
0.125	0.125	0.25	1.25e-01	2
0.1875	0.125	0.1875	6.25e-02	3
0.15625	0.15625	0.1875	3.12e-02	4
0.171875	0.15625	0.171875	1.56e-02	5
0.1640625	0.15625	0.1640625	7.81e-03	6
0.16015625	0.15625	0.16015625	3.91e-03	7
0.158203125	0.158203125	0.16015625	1.95e-03	8
0.1591796875	0.158203125	0.1591796875	9.77e-04	9
0.15869140625	0.158203125	0.15869140625	4.88e-04	10
0.158447265625	0.158447265625	0.15869140625	2.44e-04	11
0.1585693359375	0.1585693359375	0.15869140625	1.22e-04	12
0.15863037109375	0.1585693359375	0.15863037109375	6.10e-05	13
0.158599853515625	0.1585693359375	0.158599853515625	3.05e-05	14
0.1585845947265625	0.1585845947265625	0.158599853515625	1.53e-05	15
0.15859222412109375	0.1585845947265625	0.158599853515625	7.63e-06	16

Position	a	b	Error	Iteration
None	3	4	None	Before Algo Start
3.5	3	3.5	5.00e-01	0
3.25	3	3.25	2.50e-01	1
3.125	3.125	3.25	1.25e-01	2
3.1875	3.125	3.1875	6.25e-02	3
3.15625	3.125	3.15625	3.12e-02	4
3.140625	3.140625	3.15625	1.56e-02	5
3.1484375	3.140625	3.1484375	7.81e-03	6
3.14453125	3.14453125	3.1484375	3.91e-03	7
3.146484375	3.14453125	3.146484375	1.95e-03	8
3.1455078125	3.1455078125	3.146484375	9.77e-04	9
3.14599609375	3.14599609375	3.146484375	4.88e-04	10
3.146240234375	3.14599609375	3.146240234375	2.44e-04	11
3.1461181640625	3.1461181640625	3.146240234375	1.22e-04	12
3.14617919921875	3.14617919921875	3.146240234375	6.10e-05	13
3.146209716796875	3.14617919921875	3.146209716796875	3.05e-05	14
3.1461944580078125	3.14617919921875	3.1461944580078125	1.53e-05	15
3.1461868286132812	3.14617919921875	3.1461944580078125	7.63e-06	16

Position	a	b	Error	Iteration
None	-1	1	None	Before Algo Start
0.0	-1	1	No Error	0

Exercice 3

1 Import

Import

- `numpy` : Les fonctions mathématiques prédéfinies, et `linspace` pour créer les vecteur des coordonnées x de chaque fonctions.
- `matplotlib` : Dessine les graphes.
- `collections` : Crée une nouvelle structure avec `namedtuple`.

```
import numpy as np
import matplotlib.pyplot as plt
from collections import namedtuple
```

2 Définition De La Fonction f

```
def function(x):
    return x - 1 - np.exp(-x)
```

3 ϵ

```
```

4 Génération Des Vecteurs

```
    plt.xlabel("x")
    plt.ylabel("y")
```

5 Nouvelle Structure root

root

On a créé une nouvelle structure en utilisant `namedtuple` root. Elle représente les racines pour $f(x) = 0$ et possède trois attributs :

- **position** : coordonnée x de la racine
- **a** : extrémité gauche de l'intervalle auquel la racine appartient
- **b** : extrémité droite de l'intervalle auquel la racine appartient
- **phi** : le label de la function point fixe φ.
- **x_0** : la valeur de depart $x_0 \in [a, b]$.
- **eps** : la valeur de la toleranceϵ.
- **error** : Estimation de l'erreur de la method dichotomie.
- **iteration**: Nombre d'iteration de la method dichotomie.

```
root = namedtuple("root", ["position","a", "b","phi","x_0","eps","error","iteration"])
```

6 Trouver φ Pour f Sur [1,2]

On a $f(x) = x - 1 - e^{-x}$ et $[a, b] = [1, 2]$

$$x - 1 - e^{-x} = 0$$
$$\boxed{x = e^{-x} + 1 = \varphi}$$

$\underline{\varphi'}$

$$\boxed{\varphi'(x) = -e^{-x}}$$

$$\boxed{\forall x \quad \varphi' < 0}$$

$\underline{\varphi''}$

$$\boxed{\varphi''(x) = e^{-x}}$$

$$\boxed{\forall x \quad \varphi'' > 0}$$

<u>Variation Table</u>

x	1 2
$\varphi''(x)$	$+$
$\varphi'(x)$	$-0.36 \nearrow -0.13$
$\varphi'(x)$	$-$
$\varphi(x)$	$1.36 \searrow 1.13$

$\varphi([1,2]) \subseteq [1,2] \Longrightarrow \varphi$ est stable en $[1,2]$

$\varphi \in C^1$ et $\sup\limits_{x\in[1,2]} |\varphi'(x)| = 0.36 < 1 \Longrightarrow \varphi$ est contractante en $[1,2]$

Conclusion :

$$\boxed{\varphi = e^{-x} + 1 \quad , \quad k = 0.36}$$

7 Implementation De La Method Point-Fixe

7.1 Estimation De L'erreur

```
def ErrorEstimation(x_0,x_1,k,n):
    return k**(n)/(1-k) * abs(x_0-x_1)
```

7.2 Definition De La Fonction φ

```
def phi_function(x):
    return np.exp(-x) + 1
```

7.3 Initialisation Des Variables

```
    plt.subplots_adjust(top=0.95)
    plt.plot(x,y,label=label,color=color)
    size = len(rootList)
    scatter(rootList[size-1],marker)
```

7.4 Algorithm De Point-Fixe

```
def Fixed_Point(eps,a,b,k,x_0,phi_function,label,max_iter=100):

    rootList=[]

    x_1 = x_n = x = phi_function(x_0)

    n = 0

    while (error:= ErrorEstimation(x_0,x_1,k,n)) > eps and n<= max_iter:
        x = phi_function(x_n)
        if x==x_n:
            rootList = [root(x_n,a,b,label,x_0,eps,"No Error",n)]
            return rootList
        x_n = x
        rootList.append(root(x_n,a,b,label,x_0,eps,error,n))
        n = n+1

    if n==0:
        rootList = [root(x_n,a,b,label,x_0,eps,"No Error",n)]
        return rootList
    elif n>max_iter:
        rootList = [root(x_n,a,b,label,x_0,eps,"Doesn't Converge Exceeded Max Number Of Iteration",n)]
    else:
        return rootList
```

8 Dessiner

8.1 Scatter

```
def scatter(rootElement,marker):

    if rootElement.error == "No Error":
        plt.scatter(rootElement.position,0,color="#88c999",marker=marker,label=fr"$\alpha \in [{rootElement.a},{rootElement.b}]  = {
            rootElement.position}$")

    else:
        plt.scatter(rootElement.position,0,color="#88c999",marker=marker,label=fr"$\alpha \in [{rootElement.a},{rootElement.b}]  \approx {
            rootElement.position}$")
```

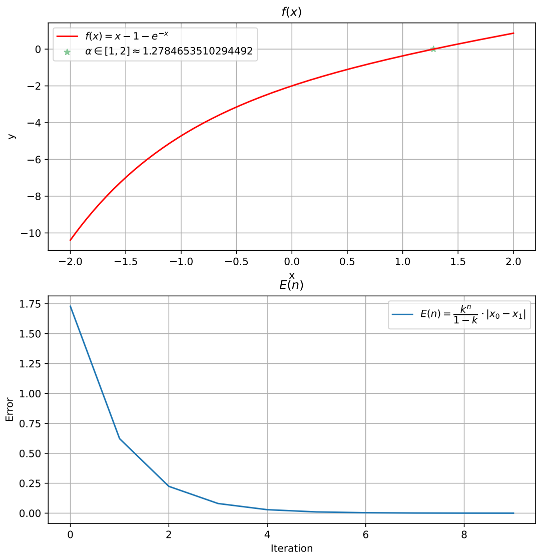
8.2 Graphe & Table

```
    x = []
    y = []
    for value in rootList:
        x.append(value.iteration)
        y.append(value.error)

    plt.subplot(2,2,3)
    plt.plot(x,y,label=r"$E(n) = \dfrac{k^n}{1-k} \cdot |x_0 - x_1|$")
    plt.title(r"$E(n)$")
    plt.xlabel("Iteration")
    plt.ylabel("Error")
    plt.legend()
    plt.grid()
```

9 Rest Du Code

```
    plt.grid()

    plt.subplot(2,2,2)
    plt.axis('off')
    data = [[r.position, r.a, r.b, r.phi , r.x_0 , r.eps , r.error, r.iteration] for r in rootList]
    headers =  ["Position","$a$","$b$",r"$\varphi$","$x_0$",r"$\epsilon$","Error","Iteration"]
    the_table = plt.table(fontsize=12,cellText=data,colLabels =headers ,cellLoc = 'center',loc='center')
    the_table.auto_set_column_width(col=list(range(len(headers))))

plt.figure(figsize=(15, 8))

eps = 10**(-4)
k = 0.36
a = 1
b = 2
x_0 = 0.5

x = np.linspace(-2,2,400)
y = function(x)

label_f = r"$ f(x) = x - 1 - e^{-x}$"
label_phi =r"$e^{-x} + 1$"

rootList = Fixed_Point(eps,a,b,k,x_0,phi_function,label_phi)

draw(x,y,label_f,rootList)

plt.suptitle(r"Tp$_1$ Exo$_3$")

drawGraphError(rootList)

plt.show()
```

10 Figure



Tp$_1$ Exo$_3$

Position	a	b	φ	x_0	ε	Error	Iteration
1.200582296575809	1	2	$e^{-x}+1$	0.5	0.0001	1.7289541558009898	0
1.301018878606898	1	2	$e^{-x}+1$	0.5	0.0001	0.6224234960883563	1
1.272254257631842	1	2	$e^{-x}+1$	0.5	0.0001	0.22407245859180824	2
1.2801992679637144	1	2	$e^{-x}+1$	0.5	0.0001	0.08066608509305097	3
1.2779819020462433	1	2	$e^{-x}+1$	0.5	0.0001	0.029039790633498347	4
1.2785989735253174	1	2	$e^{-x}+1$	0.5	0.0001	0.010454324628059406	5
1.2784271110758172	1	2	$e^{-x}+1$	0.5	0.0001	0.0037635568661013854	6
1.2784749663532786	1	2	$e^{-x}+1$	0.5	0.0001	0.0013548804717964985	7
1.2784616401753643	1	2	$e^{-x}+1$	0.5	0.0001	0.0004877569698467395	8
1.2784653510294492	1	2	$e^{-x}+1$	0.5	0.0001	0.00017559250914482623	9

Remarque

On remarque que $E(x)$ est decroissante et converge vers 0