

## 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`.

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

### 2 Définition Des Fonctions Mathématiques

```
1 def function_1(x):
2     return x**6 - x -1;
3
4 def function_2(x):
5     return 1 - 1/4 * np.cos(x)
6
7 def function_3(x):
8     return np.cos(x) - np.exp(-x)
9
10 def function_4(x):
11     return x**4 - 56.101*x**3 + 785.6561*x**2 - 72.7856*x + 0.0078
```

### 3 Génération des Vecteurs

```
1 x1 = np.linspace(-2, 2, 400)
2 x2 = np.linspace(-2*np.pi, 2*np.pi, 400)
3 x3 = np.linspace(-1, 2*np.pi, 400)
4 x4 = np.linspace(-0.1, 0.5, 400)
5
6 y1 = function_1(x1)
7 y2 = function_2(x2)
8 y3 = function_3(x3)
9 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

```
1 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

```
1 def scatter_single(rootElement, colorValue, marker):
2     if rootElement.a == rootElement.b :
3         plt.scatter(rootElement.position, 0, color = colorValue , marker = marker, label = fr"$\alpha = \{rootElement.a\}$")
4     else:
5         plt.scatter(rootElement.position, 0, color = colorValue , marker = marker, label = fr"$\alpha \in [\{rootElement.a\},\{rootElement.b\}]$")
6
7 def scatter_many(rootElement, colorValue, index, marker):
8     if rootElement.a == rootElement.b :
9         plt.scatter(rootElement.position, 0, color = colorValue , marker = marker, label = fr"$\alpha_{\{index\}} = \{rootElement.a\}$")
10    else:
11        plt.scatter(rootElement.position, 0, color = colorValue , marker = marker,
12                    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

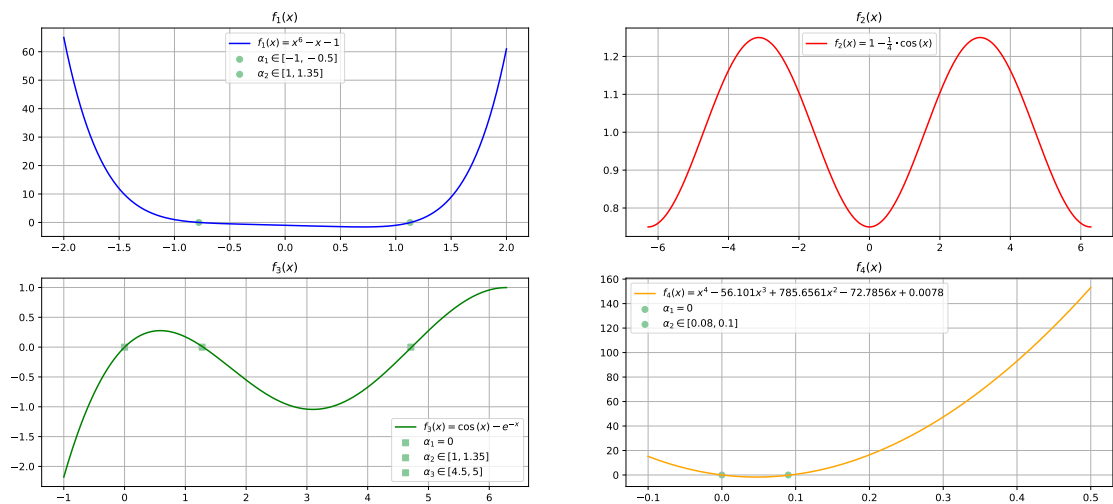
```
1 def draw(x,y,functionLabel,colorValue,rootList,index,title,marker='o'):  
2     plt.subplot(2,2,index)  
3     plt.plot(x, y, label=fr"{functionLabel}", color=colorValue)  
4     sizeList = len(rootList)  
5  
6     if sizeList == 1:  
7         scatter_single(rootList[0], '#88c999', marker)  
8     elif sizeList > 1 :  
9         for i in range(sizeList):  
10            scatter_many(rootList[i], '#88c999', i+1, marker)  
11     plt.legend()  
12     plt.grid()  
13     plt.title(title)  
14     return index+1
```

## 7 Reste Du Code

```
1  
2 index = 1  
3 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"$f_1(x)$")  
4 index = draw(x2, y2, fr"$f_{index}(x) = 1 - \frac{1}{4} \cdot \cos{(x)}$", "red", [], index, r"$f_2(x)$", '.')  
5 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')  
6 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)],  
7         index, r"$f_4(x)$", 'o')  
8  
9 plt.suptitle(r"TP$1$ Exo$1$")  
10 plt.savefig("fig.pdf")  
11 plt.show()
```

8 Figure

TP<sub>1</sub> Exo<sub>1</sub>



## 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`.

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

### 2 Définition Des Fonctions Mathématiques

```
1 def function_1(x):
2     return np.log(x) - x + 2
3
4 def function_2(x):
5     return np.exp(x) + x**2/2 + x - 1
```

### 3 $\epsilon$

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

### 4 Génération des Vecteurs

```
1 x_1 = np.linspace(0.1,5,400)
2 x_2 = np.linspace(-2,2,400)
3
4 y_1 = function_1(x_1)
5 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.

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

## 6 Implementation De La Method Dichotomie

### 6.1 Estimation De L'erreur

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

### 6.2 Algorithm De Dichotomie

```
1 def dichotomy(eps,a,b,function,max_iter=100):  
2  
3     rootList = []  
4     n = 0  
5     a_0 = a  
6     b_0 = b  
7  
8     rootList.append(root("None",a,b,"None","Before Algo Start"))  
9  
10    while (error := ErrorEstimation(a,b)) > eps and n <= max_iter :  
11  
12        x = (a+b)/2  
13  
14        if function(x) * function(a) < 0:  
15            b = x  
16        elif function(x) * function(b) < 0:  
17            a = x  
18        else:  
19            rootList.append(root(x,a,b,"No Error",n))  
20            return rootList  
21  
22        rootList.append(root(x,a,b,f"{error:.2e}",n))  
23        n = n+1  
24  
25    rootList.append(root((a+b)/2,a,b,f"{error:.2e}",n))  
26    return rootList
```

## 7 Dessiner

### 7.1 Scatter

```
1 def scatter_many(a,b,position,error,index,marker="*"):
2     if error == "No Error":
3         plt.scatter(position,0,marker=marker,color="#88c999",label=fr"${\alpha}_{index} \text{ in } [{a},{b}] = {root.position}$")
4     else:
5         plt.scatter(position,0,marker=marker,color="#88c999",label=fr"${\alpha}_{index} \text{ in } [{a},{b}] \approx {position}$")
6
7
8 def scatter_single(a,b,position,error,marker="*"):
9     if error=="No Error":
10         plt.scatter(position,0,marker=marker,color="#88c999",label=fr"${\alpha} \text{ in } [{a},{b}] = {position}$")
11     else:
12         plt.scatter(position,0,marker=marker,color="#88c999",label=fr"${\alpha} \text{ in } [{a},{b}] \approx {position}$")
13
14 def scatter(rootMatrix,marker="*"):
15     size = len(rootMatrix)
16
17     if size == 1:
18         firstElement = rootMatrix[0][0]
19         lastElement = rootMatrix[0][-1]
20
21         scatter_single(firstElement.a,firstElement.b,lastElement.position,lastElement.error)
22     else:
23         for index in range(size):
24             firstElement = rootMatrix[index][0]
25             lastElement = rootMatrix[index][-1]
26             scatter_many(firstElement.a,firstElement.b,lastElement.position,lastElement.error,index)
```

### 7.2 Graphes

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

### 7.3 Tables

```
1 def draw_table(rootMatrix,index):
2
3     for rootList in rootMatrix:
4         plt.subplot(2,2,index)
5         plt.subplots_adjust(top=0.85)
6         plt.axis("off")
7         data = [[r.position, r.a, r.b, r.error, r.iteration] for r in rootList]
8         headers = ["Position", "a", "b", "Error", "Iteration"]
9         the_table = plt.table(cellText=data, colLabels=headers, fontsize=12, loc="center", cellLoc="left")
10        the_table.auto_set_column_width(col=list(range(len(headers))))
11        index = index+1
12
13    return index
```

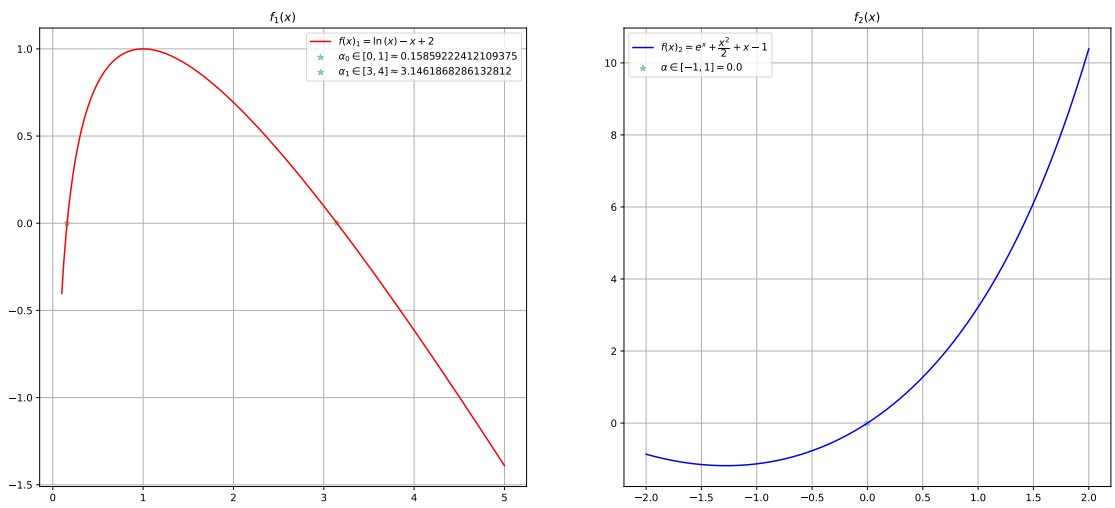
## 8 Rest Du Code

```
1 label_1 = r"\ln{(x)} - x + 2"
2 label_2 = r"e^{x} + \dfrac{x^2}{2} + x - 1"
3
4 color_1 = "red"
5 color_2 = "blue"
6
7 index = 1
8
9 rootList_1 = dichotomy(eps,0,1,function_1)
10 rootList_2 = dichotomy(eps,3,4,function_1)
11
12 rootMatrix_1 = [rootList_1 , rootList_2]
13
14 rootMatrix_2 = [dichotomy(eps,-1,1,function_2)]
15
16 index = draw_graph(x_1,y_1,index,color_1,label_1,rootMatrix_1)
17 index = draw_graph(x_2,y_2,index,color_2,label_2,rootMatrix_2)
18
19 plt.suptitle(r"Tp$_{1}$ Exo$_{2}$ Functions")
20
21 plt.figure(figsize=(10, 7))
22 index = 1
23 index = draw_table(rootMatrix_1,index)
24 index = draw_table(rootMatrix_2,index)
25
26 plt.suptitle(r"Tp$_{1}$ Exo$_{2}$ Tables")
27
28 plt.show()
```



9 Figures

Tp<sub>1</sub> Exo<sub>2</sub> Functions



Tp<sub>1</sub> Exo<sub>2</sub> 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`.

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

### 2 Définition De La Fonction $f$

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

### 3 $\epsilon$

```
1 eps = 10**(-4)
```

### 4 Génération Des Vecteurs

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

## 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 fonction point fixe  $\varphi$ .
- **x\_0** : la valeur de depart  $x_0 \in [a, b]$ .
- **eps** : la valeur de la tolerance.
- **error** : Estimation de l'erreur de la method dichotomie.
- **iteration**: Nombre d'iteration de la method dichotomie.

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

## 6 Trouver $\varphi$ Pour $f$ Sur $[-1,1]$

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

$$e^x + \frac{x^2}{2} + x - 1 = 0$$

$$x = 1 - e^x - \frac{x^2}{2} = \varphi$$

$\varphi'$

$$\varphi'(x) = -e^x - x$$

$$\varphi'(x) = -(e^x + x)$$

$\varphi''$

$$\varphi''(x) = -e^x - 1$$

$\varphi''(x) = -(e^x + 1)$

$\forall x \quad \varphi'' \leq 0$

**Variation Table**

$x$	$-1$	$\alpha$	$-0.25$	$1$
$\varphi''(x)$	—			
$\varphi'(x)$	0.63	0	-0.52	-3.71

$$\inf_{x \in [-1,1]} |\varphi'(x)| < 1 \implies \varphi \text{ ne diverge pas dans } [-1,1]$$

$$\varphi \in C^1 \text{ et } \sup_{x \in [-1,1]} |\varphi'(x)| = 3.71 \geq 1 \implies \varphi \text{ n'est pas contractante en } [-1,1]$$

Le problem est dans l'extremite 1 , donc on va prendre un nouvelle intervalle [-1,-0.25] :

$$\varphi \in C^1 \text{ et } \sup_{x \in [-1,-0.25]} |\varphi(x)| = 0.63 < 1 \implies \varphi \text{ est contractante en } [-1,-0.25]$$

**Conclusion :**

$\varphi = 1 - e^x - \frac{x^2}{2} \quad , \quad k = 0.63$

## 7 Implementation De La Method Point-Fixe

### 7.1 Estimation De L'erreur

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

### 7.2 Definition De La Fonction $\varphi$

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

### 7.3 Initialisation Des Variables

```
1 k = 0.63
2 a = -1
3 b = -0.25
4 x_0 = -0.5
```

### 7.4 Algorithm De Point-Fixe

```
1 def Fixed_Point(eps,a,b,k,x_0,phi_function,label,max_iter=100):
2
3     rootList=[]
4
5     x_1 = x_n = x = phi_function(x_0)
6
7     n = 0
8
9     while (error:= ErrorEstimation(x_0,x_1,k,n)) > eps and n<= max_iter:
10         x = phi_function(x_n)
11         if x==x_n:
12             rootList = [root(x_n,a,b,label,x_0,eps,"No Error",n)]
13             return rootList
14         x_n = x
15         rootList.append(root(x_n,a,b,label,x_0,eps,f"{error:.2e}",n))
16         n = n+1
17
18     if n==0:
19         rootList = [root(x_n,a,b,label,x_0,eps,"No Error",n)]
20         return rootList
21     elif n>max_iter:
22         rootList = [root(x_n,a,b,label,x_0,eps,"Doesn't Converge Exceeded Max Number Of Iteration",n)]
23     else:
24         return rootList
```

## 8 Dessiner

### 8.1 Scatter

```
1 def scatter(rootElement,marker):
2
3     if rootElement.error == "No Error":
4         plt.scatter(rootElement.position,0,color="#88c999",marker=marker,label=fr"$\alpha$ \in [{rootElement.a},{rootElement.b}] = {
5             rootElement.position}$")
6
7     else:
8         plt.scatter(rootElement.position,0,color="#88c999",marker=marker,label=fr"$\alpha$ \in [{rootElement.a},{rootElement.b}] \approx {
9             rootElement.position}$")
```

### 8.2 Graphe & Table

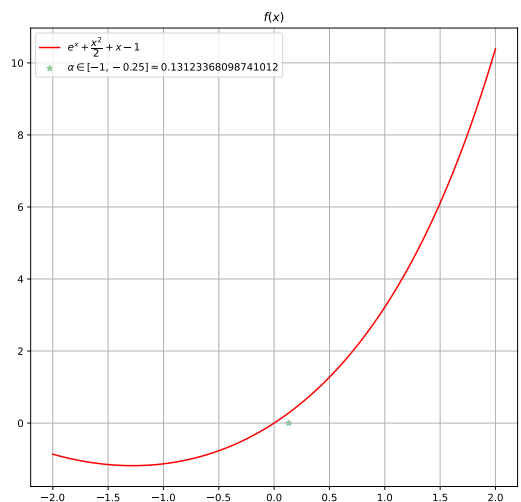
```
1 def draw(x,y,label,rootList,color='red',marker='*'):
2     plt.subplot(1,2,1)
3     plt.plot(x,y,label=label,color=color)
4     size = len(rootList)
5     scatter(rootList[size-1],marker)
6     plt.title(fr"$f(x)$")
7     plt.legend()
8     plt.grid()
9
10    plt.subplot(1,2,2)
11    plt.axis('off')
12    data = [[r.position, r.a, r.b, r.phi, r.x_0, r.eps, r.error, r.iteration] for r in rootList]
13    headers = ["Position", "$a$", "$b$", r"$\varphi$", "$x_0$", r"$\epsilon$", "Error", "Iteration"]
14    the_table = plt.table(fontsize=12,cellText=data,colLabels = headers, cellLoc = 'center',loc='center')
15    the_table.auto_set_column_width(col=list(range(len(headers))))
```

## 9 Rest Du Code

```
1 label_f = r"$e^{\{x\}} + \dfrac{\{x^2\}^2}{2} + x - 1$"
2 label_phi = r"$1 - e^{\{x\}} - \dfrac{\{x^2\}^2}{2}$"
3
4 rootList = Fixed_Point(eps,a,b,k,x_0,phi_function,label_phi)
5
6 draw(x,y,label_f,rootList)
7
8 plt.suptitle(fr"Tp$_1$ Exo$_3$")
9 plt.show()
```

10 Figure

Tp1 Exo3



Position	a	b	$\varphi$	$x_0$	$\varepsilon$	Error	Iteration
-0.34399876804842855	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	2.08e+00	0
0.23190262237442605	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	1.31e+00	1
-0.28788634309466743	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	8.24e-01	2
0.2087139140655841	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	5.19e-01	3
-0.25387321274354313	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	3.27e-01	4
0.1919840398088326	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	2.06e-01	5
-0.23008011439679985	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	1.30e-01	6
0.17906161900682951	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	8.18e-02	7
-0.21212597574959355	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	5.15e-02	8
0.16863849115499163	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	3.25e-02	9
-0.19791161673591706	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	2.05e-02	10
0.15997313267298566	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	1.29e-02	11
-0.18627504390363636	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	8.12e-03	12
0.15260553931620008	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	5.12e-03	13
-0.17650962001960474	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	3.22e-03	14
0.1462314613149858	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	2.03e-03	15
-0.16815588534174014	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	1.28e-03	16
0.14063973242860342	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	8.06e-04	17
-0.16089966890626056	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	5.08e-04	18
0.13567816181635478	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	3.20e-04	19
-0.15451751054051968	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	2.01e-04	20
0.13123368098741012	-1	-0.25	$1 - e^x - \frac{x^2}{2}$	-0.5	0.0001	1.27e-04	21