# Méthodes de résolution
# $f(x) = 0$

Objectif :

Résoudre l'équation : $g(x) = x.e^{-x^2} - 1/4 = 0$
Avec une précision $\varepsilon$ sur $g(x)$ de $10^{-16}$

Une première étape sera d'étudier la résolution de manière numérique l'équation (évidente) :

$$f(x) = x.e^{-x^2} = 0$$

## Partie 1 : Prise en main :

Créer 2 fonctions (au sens algorithmique du terme) :

- La première renvoyant à l'argument $x$ : $x.e^{-x^2}$ (valeur de $f$ en $x$)
- La seconde renvoyant à l'argument $x$ : la dérivée de $f$ en $x$

(choisir des noms de fonctions explicites)

Tracer les deux courbes associées à ces deux fonctions dans l'intervalle $[-2; 2]$

## Partie 2 : Dichotomie

**Q1 :** A quelle condition la résolution par dichotomie est-elle opérationnelle ?
**Q2 :** Rappeler les arguments nécessaires à la méthode de dichotomie
**Q3 :** A partir du tracé de la courbe de $f$, préciser l'intervalle $[a, b]$ où elle est monotone.

En utilisant l'algorithme du cours, programmer une fonction qui aux arguments : $fonction$, $[a, b]$, $\varepsilon$, renvoie la valeur trouvée.

L'utiliser pour résoudre $f(x) = 0$

**Q4 :** Quelle valeur trouvez-vous (avec 4 chiffres significatifs) ?

## Partie 3 : Méthode de Newton

**Q5 :** Rappeler les arguments nécessaires à la méthode de Newton

En utilisant l'algorithme du cours, programmer une fonction qui aux arguments : $fonction$, $derivee\_fonction$, $x_0$, $\varepsilon$, $max\_it$ renvoie la valeur recherchée.

L'utiliser pour résoudre $f(x) = 0$
Tester le programme pour la valeur initiale $x_0 = 0{,}1$

**Q6 :** Quelle valeur trouvez-vous (avec 4 chiffres significatifs) ?

Tester le programme pour différentes valeurs initiales.

**Q7 :** Que constate-t-on ? Comment l'expliquer ? Quelle précaution faut-il prendre alors lors de l'étude de cette équation ?

## Partie 4 : Module scipy

Le module scipy propose ces méthodes en utilisant des méthodes numériques optimisées. Voici le nom et les arguments des méthodes associées (voir l'aide pour plus de détail)

| ***Nom de la méthode*** | ***Utilisation avec scipy*** |
|---|---|
| Dichotomie | optimize.bisect(fonction, borne inférieure, borne supérieure) |
| Newton | optimize.newton(fonction, valeur initiale) |

**Q8 :** Comparer la précision des résultats obtenus par vos algorithmes et ceux fournis par scipy

## Partie 5 : Performance temporelle

Le module `time` peut aussi tester le temps nécessaire à la mise en œuvre de chacune des méthodes. Pour cela, il faut :

- enregistrer l'instant de l'horloge au début de l'algorithme (en seconde)

```
init=time.clock()
```

- enregistrer l'instant de l'horloge à la fin de l'algorithme

```
fin=time.clock()
```

- afficher la durée écoulée

```
print(fin-init)
```

## Partie 6 : Tableau comparatif

**Q9 :** Dresser le tableau comparatif des 4 méthodes, en indiquant pour chacune la valeur trouvée (n'indiquer que les chiffres significatifs) et le temps d'exécution nécessaire.

**Q10 :** Rappeler les contraintes nécessaires à la mise en œuvre de chacune.

## Partie 7 : Retour à l'objectif

**Q11 :** A partir du tableau précédent, choisir la méthode permettant de résoudre « au mieux » l'équation : $g(x) = x.e^{-x^2} - 1/4$

**Q12 :** Quelle valeur r trouvez-vous ? Vérifier en calculant $g(r)$. Conclure.