Travaux dirigés
Recherche de zéro et équations non-linéaires n°7
Mathématiques pour l'informatique
—IMAC 2—

▸ **Exercice 1.** ***Dichotomie***

1. *Nous voulons résoudre l'équation* $\cos x = 2x$. *Montrer que cette équation a une solution unique* x_0 *sur* $[-\pi, \pi]$.

2. *Récupérez le code en C++ sur le site de l'enseignant. Compilez, lisez le code et testez.*

3. *Ce code contient la fonction* $f(x) = \cos x - 2x$, *pour laquelle on voudrait trouver la valeur* x_0 *telle que* $f(x_0) = 0$. *Codez la méthode de dichotomie. Le prototype de votre fonction sera :*

   ```
   double dichotomy(std::function<double(const double&)> &f,
                    double lowerBound, // copy
                    double upperBound, // copy
                    const unsigned int nbIterations)
   ```

   et cette fonction renverra le milieu de l'intervalle trouvé. Testez avec la fonction f *sur l'intervalle* $[-\pi, \pi]$.

4. *Nous considérons que nous coupons à chaque itération l'intervalle considéré en son milieu. Trouver une expression de l'erreur absolue du calcul en fonction de la taille de l'intervalle de départ et du nombre d'itérations. Codez une fonction qui calcule le nombre d'itérations nécessaires pour obtenir une précision donnée et pour l'intervalle considéré. Demandez un résultat à* 10^{-7} *près et vérifier que votre racine vérifie bien cette contrainte.*

▸ **Exercice 2.** ***Méthode de Newton***

La méthode de Newton permet de trouver le zéro d'une fonction $f(x)$ *à partir d'une estimation initiale* x_0 *de la solution. Cette méthode itère selon le schéma suivant :*

Algorithm 1: Newton

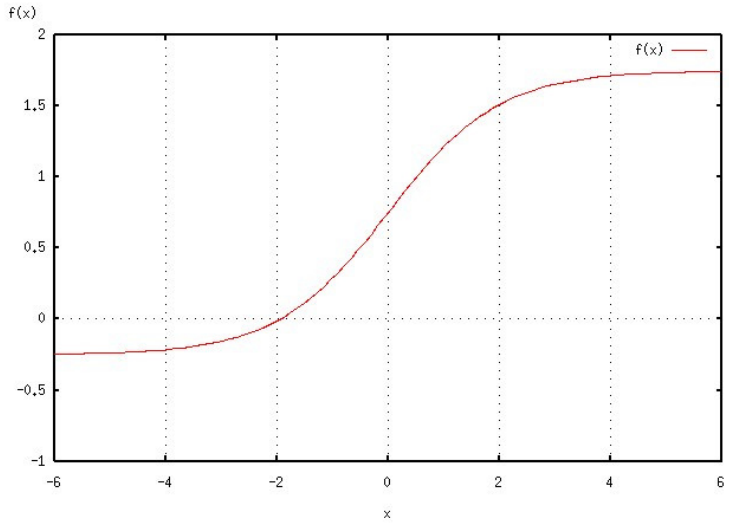
input: fonction f, solution initiale x_0 et un seuil ε

$x = x_0$
repeat
  $x = x - \dfrac{f(x)}{f'(x)}$
until $|f(x)| < \varepsilon$
return x

On s'arrête quand $|f(x)|$ est suffisamment petit.

1. *Nous voulons résoudre l'équation $f(x) = 0$ pour :*

$$f(x) = \frac{e^x - 1}{e^x + 1} + \frac{3}{4} \text{ et } f'(x) = \frac{2e^x}{(e^x+1)^2}$$



Codez la méthode de Newton. Le prototype de votre fonction sera :

```
double Newton(std::function<double(const double&)> &f,
              std::function<double(const double&)> &derivative,
              const double &input,
              const double &threshold,
              const unsigned int maxIterations = 1000)
```

Testez votre programme à partir du point $x_0 = 0$ puis $x_0 = 2$. Que se passe-t-il?

2. *Testez la méthode de Newton sur la fonction $f(x) = \cos x - 2x$ de l'exercice précédent. Sa dérivée est $f'(x) = -\sin x - 2$. Comparez le nombre d'itérations nécessaires à chacune des méthodes pour converger vers un résultat de précision équivalente.*

▸ **Exercice 3.** ***Méthode de Newton duale***

La méthode de Newton implique de calculer la dérivée de la fonction traitée. Une façon de contourner cette contrainte consiste à utiliser le nombres duaux utilisés dans un tp précédent.

1. *Récupérez le code des nombres duaux et mettez le dans le répertoire* `src`.
2. *Mettre à jour le fichier* `CmakeList.txt` *en lui spécifiant de compiler également les fichiers relatifs aux nombres duaux.*

3. *Faites une nouvelle version de votre méthode de Newton, pensez à :*
    - *mettre les bons includes*
    - *reformuler la fonction à traiter avec les opérateurs des nombres duaux*
    - *reformuler votre méthode de Newton*
4. *vérifiez si vous trouvez les mêmes résultats qu'avec la formulation initiale.*