L3 - Méthodes numériques
TP 2 - Calcul de fonctions et résolution d'équations

Support de TP

Les TPs visent à appliquer de manière concrète les notions vues en cours. Il est donc conseillé de venir avec vos supports de cours ou de les télécharger sur le web.

Les TPs notés doivent être soumis sur la page Moodle du cours dans les deux semaines suivant le TP. Vous devez soumettre un fichier zip contenant le code **sans bug** avec un scripte python **TP2_1.py,TP2_2.py,..** par partie du TP ainsi qu'un court rapport de 2 pages maximum en **PDF** (sans page de titre). Le rapport ne doit contenir aucune sortie terminal ou ligne de code. Vous devez par contre pour chaque section du TP rédiger une réflexion sur les notions utilisées et ce que vous avez appris.

1 Affichage de courbes sous matplotlib

Dans ce TP, vous serez amené à tracer des courbes. Vous utiliserez pour cela la bibliothèque matplotlib/pylab qui s'importe avec `import pylab as pl`. On supposera par la suite que pylab et numpy ont été importés.

1. Initialiser un vecteur `x` contenant 500 valeurs entre -10 et 10 (`np.linspace`).
2. Calculer $y1 = |sin(x) - x|$ et $y2 = 2x - 1$ (`np.sin,np.abs`).
3. Les tracer dans une figure avec une légende (`pl.figure,pl.plot`)
4. Initialiser un vecteur `t` de 500 points entre 0 et 5π. Utiliser ce vecteur pour tracer une spirale paramétrée par $t\sin(t)$ et $t\cos(t)$ and une nouvelle figure.

2 Complexité algorithmique et temps de calcul

- Coder les fonctions `puissance` et `func` définies dans l'Exercice 5 du cours 1.
- Coder la version avec récurrence de la fonction `func` discutée dans le cours 2 (l'appeler `func2`).
- Afficher le temps de calcul nécessaire à chacune des fonctions à l'aide des fonctions `tic` et `toc` définies dans le fichier `utils.py`. Tester des valeurs de `n` allant de 10 à 1000.
- Créer une liste de 10 valeurs n (entières) allant de 10 à 1000 par pas logarithmique (`np.logspace`) et la convertir en entiers (méthode `.astype(np.int)`).
- Pour chacune des valeurs de la liste appeler `func` et `func2` et stocker leur temps de calcul respectifs dans un vecteur (`np.zeros,utils.toq`).
- Tracer les temps de calcul en fonction de n pour les deux méthodes une première fois avec des axes linaires (`pl.plot`) puis dans une autre figure avec des axes logarithmique (`pl.loglog`)

3 Résolution d'équations

Dans cette section vous allez calculer la valeur de $\sqrt{2}$. Ceci se fera par la résolution de l'équation suivante

$$f(x) = x^2 - 2 = 0$$

1. Déclarer dans votre code la valeur :
   `sqrt2=1.4142135623730504880`
   Imprimer cette valeur avec 16 décimales.

2. Calculer l'erreur entre `np.sqrt(2)`,`math.sqrt(2)` et la valeur définie ci-dessus. Conclusions concernant la précision de la fonction `sqrt`.
3. Coder la méthode de la dichotomie en prenant $a = 1$ et $b = 2$ pour l'estimation de $\sqrt{2}$. Remplir le long des itérations un vecteur contenant l'erreur $|x_k - \sqrt{2}|$. On prendra un nombre d'itérations max `n=60`.
4. Visualiser l'évolution de l'erreur en fonction des itérations avec `gnuplot`.
5. Coder la méthode de la fausse position en prenant $a = 1$ et $b = 2$ pour l'estimation de $\sqrt{2}$. Remplir le long des itérations un vecteur contenant l'erreur $|x_k - \sqrt{2}|$.
6. Coder la méthode de Newton vue en cours pour l'estimation de $\sqrt{2}$. Remplir le long des itérations un vecteur contenant l'erreur $|x_k - sqrt2|$. On initialisera avec $x_0 = 1$.
7. Visualiser l'évolution de l'erreur en fonction des itérations pour la dichotomie, la fausse position et la méthode de Newton. Comparer leur vitesse de convergence. Combien d'itérations sont nécessaires pour que chaque méthode converge ?
8. Étudier l'effet de l'initialisation x_0 sur la convergence de la méthode de Newton.

4 Calcul de fonction

Dans cette section vous allez coder l'évaluation de la fonction $\exp(x)$. On supposera ici que la fonction $exp(x)$ définie de numpy retourne le double le plus proche de la valeur exacte.

1. Coder une fonction `exptaylor(x,n)` qui calcule la valeur de $exp(x)$ en utilisant la série de Taylor suivante jusqu'à l'ordre n :
$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$
2. Tracer l'erreur relative $|exp(x) - exptaylor(x)|/|exp(x)|$ de cette approximation pour $n = 10$ et $n = 100$ sur l'intervalle $[-10, 10]$ en échelle linéaire et log. Que remarquez vous pour les grandes valeurs de x ?
3. Implémenter la réduction de l'intervalle par rapport à $ln(2)$ vue en cours. Utiliser pour cela l'approximation de Taylor à l'ordre 100 autour de 0. Vous pourrez utiliser la fonction puissance vue en cours mais attention à la modifier pour gérer les puissances négatives. La valeur de $ln(2)$ peut être définie par :

   ```
   ln2=0.69314718055994530942
   ```
4. Tracer l'erreur relative lorsque l'on utilise la réduction de l'intervalle.Votre implémentation est elle précise par rapport à la précision machine ?