Mathématiques supérieures — le 16 Mars 2018

TP n°9 - Algorithmes numériques

Dans ce T.P., nous allons apprendre à résoudre numériquement des équations et à calculer numériquement des intégrales. Pour cela, il est nécessaire de travailler avec les nombres flottant, les nombres de type `float` en Python.

1. Premier exemple : résolution d'une équation du second degré

On cherche dans cette partie, à écrire une fonction permettant de déterminer numériquement les solutions réelles d'une équation polynomiale à coefficients réels du second degré de la forme $ax^2+bx+c=0$:

> On rappelle le fonctionnement par défaut de la fonction `print` qui permet d'afficher dans la console ses arguments :
> l'instruction `print('texte',3.12,'texte')` affichera : `texte 3.12 texte`. Par défaut, la fonction `print` affiche dans la console les arguments qui lui sont donnés les uns à la suites des autres en les séparant par **un espace**.

Exercice 1.

Écrire une fonction `solutions_polynome(a,b,c)` qui **affiche** :

- ***Pas de racine réelle*** si l'équation $ax^2+bx+c=0$ n'a pas de solution réelle ;
- ***Deux racines simples*** `r1` ***et*** `r2` si l'équation $ax^2+bx+c=0$ possède deux solutions réelles distinctes - où `r1` et `r2` sont les valeurs des solutions trouvées ;
- ***Une racine double*** `r0` si l'équation $ax^2+bx+c=0$ possède une unique solution réelle - où `r0` est la valeur de la solution trouvée ;

Remarque : N'oubliez pas d'importer le fonction `sqrt` du module `math` ou du module `numpy` !

Maintenant, testons notre fonction `solution_polynome` dans les différents cas suivants et comparons avec un calcul manuel :

i) $x^2+2x+1=0$;

ii) $2x^2-5x+5=0$;

iii) $x^2+x-1=0$;

iv) $0,0025x^2+0,1x+1=0$;

v) $0,011025x^2+0,21x+1=0$.

Question 1.

- Sous quel nom est connu la racine positive de l'équation iii) ?
- Que dire de la résolution numérique de iv) et v) par rapport au calcul manuel ?

En effet, nous avons déjà vu que Python pouvait commettre des "erreurs" de calcul avec les nombres flottants, la faute à la représentation de ces nombres en machine ! (test simple : `0.1+0.1+0.1==0.3`). Ainsi, les tests de comparaison comme par exemple `==`, `>` ou `<` sont perturbés par les approximations issues des opérations sur les nombres flottants. La parade est simple, il faut introduire une marge d'erreur lorsque l'on compare deux flottants issus de calculs entre eux.

En mathématiques, on dit que deux nombres réels x et y sont égaux à ε près (ou ε-proches) où $\varepsilon > 0$ si $|x - y| \leq \varepsilon$. Ce ε est appelée **marge d'erreur absolue** :

Exercice 2.

Écrire une fonction `estproche_abs(x,y,e=10**-9)` :

- qui prend pour arguments *obligatoires* deux nombres flottants `x` et `y` ;
- qui prend pour argument facultatif la marge d'erreur absolue `e`, de valeur par défaut 10^{-9} ;
- et qui *renvoie* `True` si $|x - y| \leq \varepsilon$ et qui `False` sinon.

Remarque : N'oubliez pas d'importer le fonction **`abs`** du module `numpy` !

Tester la fonction sur $x = 0,1 + 0,1 + 0,1$ et $y = 0,3$ puis sur $x = 10 * * - 10$ et $y = 10 * * - 15$. Que dire de ce dernier exemple ?

On se rend compte que la marge d'erreur absolue ne tient pas compte des différences d'ordre entre deux nombres : dans ce dernier cas, x et y sont d'ordre très différents, mais très petits par rapport à la marge d'erreur absolue de 10^{-9}. La fonction `estproche_abs(x,y)` ne les discerne donc pas alors que la différence est conséquente en terme d'ordre.
Pour pallier ce souci, nous allons introduire la **marge d'erreur relative** !

Soit x, y des réels et ε un réel strictement positif. On dit que x et y sont ε-proches relativement à y si $|x - y| \leq \varepsilon|y|$ et on appelle **marge d'erreur relative** le nombre ε. On définit de manière analogue la relation x et y sont ε-proches relativement à x. Grâce à cette relation, la proximité de deux nombres est relative à l'ordre de grandeur d'un de ces deux nombres, ce qui permet de contourner le problème précédent même en fixant au préalable la marge d'erreur relative ε.

L'inconvénient immédiat de cette relation est qu'elle n'est pas symétrique ! On contourne une nouvelle fois cette obstacle de la manière suivante :

On dit que x et y sont ε-proches relativement à $\max(|x|, |y|)$ si $|x - y| \leq \varepsilon \max(|x|, |y|)$.

Exercice 3.

Écrire une fonction `estproche_rel(x,y,e=10**-9)` :

- qui prend pour arguments *obligatoires* deux nombres flottants `x` et `y` ;
- qui prend pour argument facultatif la marge d'erreur relative `e`, de valeur par défaut 10^{-9} ;
- et qui *renvoie* `True` si $|x - y| \leq \varepsilon \max(|x|, |y|)$ et qui `False` sinon.

Remarque : on obtient le maximum de x et y grâce à la fonction **`max(x,y)`** déjà présente dans Python.

Tester la fonction `estproche_rel` sur l'exemple précédent $x = 10^{-10}$ et $y = 10^{-15}$.

Maintenant que l'on a résolu le problème de la proximité de deux nombres flottants dans différents contextes, il ne nous reste plus qu'à modifier notre fonction `solution_polynome` pour qu'elle tiennent compte des approximations de calculs de Python :

Exercice 4.

On considère une équation du second degré à coefficient réels. On remarque que lorsque le discriminant Δ est différent de 0 et est très petit devant $|b|$, les racines r_1 et r_2 (qu'elles soient réelles ou complexes) sont quasiment confondues. Ainsi, si Δ et très petit devant $|b|$, on considérera numériquement que $\Delta = 0$ et qu'il n'y a qu'une racines double.

Modifier la fonction `solution_polynome` pour qu'elle tienne compte des problèmes de calculs du déterminant en utilisant la remarque précédente et une marge d'erreur **absolue** $\varepsilon = 10^{-9}|b|$ (on convient ici que $10^{-9}|b|$ est très petit devant $|b|$).

2. Résolution numérique par dichotomie

Dans cette partie nous allons, étant donnée une fonction continue $f : [a, b] \to \mathbb{R}$ résoudre numériquement l'équation $f(x) = 0$.

Exercice 5.

Citer le théorème des valeurs intermédiaires.

Décrivons la méthode de recherche d'une solution à l'équation $f(x) = 0$ sachant que $f(a)f(b) \leq 0$.

On va chercher à approximer une solution x_0 de l'équation en découpant successivement l'intervalle de recherche en 2 :
On calcule $f(\frac{a+b}{2})$. Comme $f(a)f(b) \leq 0$, on a nécessairement :

$$\text{soit } f(a)f(\frac{a+b}{2}) \leq 0, \qquad \text{soit } f(\frac{a+b}{2})f(b) \leq 0\,;$$

Ainsi, en appliquant une nouvelle fois le théorème des valeurs intermédiaires, il existe une solution de $f(x) = 0$ soit dans l'intervalle $\left[\frac{a+b}{2}, b\right]$, soit dans $\left[a, \frac{a+b}{2}\right]$.
Puis on réitère le processus : on obtient alors les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ définies par la relation de récurrence suivante :

$$a_0 = a, \quad b_0 = b;$$

$$(a_{n+1}, b_{n+1}) = \begin{cases} (a_n, \frac{a_n+b_n}{2}) & \text{si } f(a_n)f(\frac{a_n+b_n}{2}) \leq 0; \\ (\frac{a_n+b_n}{2}, b_n) & \text{sinon.} \end{cases}$$

Montrons alors que ces suites tendent bien vers une solution x_0 de $f(x) = 0$:

Exercice 6.

1. Montrer que pour tout $n \in \mathbb{N}$, $a_n \leq b_n$.
2. Montrer que $(a_n)_{n\in\mathbb{N}}$ est croissante et que $(b_n)_{n\in\mathbb{N}}$ est décroissante.

3. Montrer que $\lim\limits_{n\to+\infty} b_n - a_n = 0$ (*Indice :* que dire de $b_n - a_n$ par rapport à $b - a$?).
4. Déduire des questions précédentes que $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ convergent vers une même limite $x_0 \in [a, b]$.
5. Montrer que $f(x_0) = 0$.

Ainsi, pour approximer une solution x_0 de $f(x) = 0$, on va programmer les suites suivantes et renvoyer, lorsqu'on arrive à une marge d'erreur "respectable", l'un des deux termes des suites, ou mieux, le milieu de ces deux termes.
En effet, disons que l'on désire une approximation de x_0 à ε près. Supposons que $b_n - a_n \leq 2\varepsilon$. Par définition des suites précédentes, on a :

$$\text{soit } x_0 \in \left[a_n, \frac{a_n + b_n}{2}\right]; \quad \text{soit } x_0 \in \left[\frac{a_n + b_n}{2}, b_n\right];$$

par suite, $|\frac{a_n + b_n}{2} - x_0| \leq \varepsilon$.

On arrêtera donc notre algorithme de calcul des termes des suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ lorsque n vérifie $b_n - a_n \leq 2\varepsilon$ puis on renverra $\frac{a_n + b_n}{2}$ qui est une approximation à ε près d'une solution x_0 de $f(x) = 0$ dans $[a, b]$ où ε est l'erreur maximum d'approximation que l'on commet.

Exercice 7.

1. En suivant la description précédente, écrire une fonction `dichotomie(f,a,b,e=10**-12)` :
   — qui prend pour arguments une fonction `f`, les bornes de l'intervalle de recherche $[a, b]$ et la précision `e` que l'on désire pour l'approximation (on prendra cet argument comme facultatif de valeur par défaut 10^{-12}) ;
   — qui **renvoie** l'approximation d'une solution de $f(x) = 0$ dans $[a, b]$ à `e` près.
2. Quelle est la complexité de cet algorithme en fonction du nombre n de décimales voulues pour l'approximation ?

Exercice 8.

1. Déterminer une approximation de π à 10^{-12} grâce à la fonction `dichotomie` appliquée à la fonction sin sur $[3, 4]$.
2. Déterminer une approximation à 10^{-14} près de $\sqrt{2}$.
3. Déterminer une approximation à 10^{-12} près de $\sqrt[3]{7}$.