Fiche professeur

TP : Dichotomie progressive

Motivation

L'introduction en seconde de l'algorithme de dichotomie peut paraitre délicate, car il mêle plusieurs difficultés :

- La conception d'un algorithme qui fait implicitement intervenir la notion de suite
- Une condition astucieuse, mais peu naturelle : $f(a) \times f(b) < 0$
- La prise en compte indispensable du cas particulier où $f(m) = 0$, si l'on utilise une inégalité stricte pour caractériser que $f(a)$ et $f(b)$ sont de signes contraires.
- Ce TP a été annoncé à la classe comme débouchant sur **un algorithme qui permettrait de résoudre presque toutes les équations à une inconnue réelle**, mais en fournissant une valeur approchée d'une solution. Cette performance a piqué la curiosité des élèves.

  Le théorème des valeurs intermédiaires est pressenti et nous assure l'existence d'une solution, « on peut tracer sa représentation graphique sans lever le crayon ».

Le TP

On étudie une équation issue d'un problème. La durée du TP est d'environ 1 h 30, on peut gagner du temps demandant de traiter la partie A à la maison. La partie D est plus difficile, bien qu'intéressante pour montrer l'intérêt de la vérification d'algorithme (ici, l'oubli de cas particuliers). Elle peut être réservée aux élèves rapides en approfondissement.

Un premier algorithme : cas d'une fonction croissante.

Pour donner une approche progressive et mieux comprise, on commence par étudier le cas d'une fonction croissante sur $[a\ ;\ b]$. Dans cette première partie l'équation étudiée est $f(x) = k$, ce qui est plus naturel que $f(x) - k = 0$: **les résultats calculés gardent leur sens par rapport au problème**.

Effectivement, lors de la recherche par les élèves, je n'ai pas observé de difficulté notoire de compréhension de ce premier algorithme.

Un deuxième algorithme : plus général, mais plus difficile

Le passage d'une fonction croissante à une fonction qui n'a pas un sens de variation imposé complique l'étude, la condition $f(a) \times f(b) < 0$ simplifie l'algorithme, mais est beaucoup plus abstraite., et elle pose aussi un problème, comme on peut le voir dans la partie D.

la condition $f(a) \times f(b) \leq 0$, ne poserait pas ce problème.

Ce deuxième algorithme introduit d'avantage de questions de logique, la maitrise des « et » et des « ou », les négations. La transposition de l'équation à une équation du type $f(x) = 0$ nous éloigne du contexte concret.

Selon le temps que l'on consacre à cette partie de programme, il peut être intéressant de travailler d'abord l'algorithme de la partie D, puis de faire constater que cet algorithme ne répond pas à toutes les situations (si, au

cours de l'exécution, *m* prend la valeur α, les conditions nécessaires à l'application du théorème des valeurs intermédiaires ne sont plus remplies, la réponse fournie par l'algorithme est FAUSSE !).

Un troisième algorithme, plus simple, mais incomplet[1]

L'algorithme de la partie D semble correct, cependant il ne traite pas le cas particulier où les conditions sont telles que la valeur de α est atteinte par *m* en cours d'exécution. On peut le constater facilement en prenant $f(x) = x$, a=-1 et b=1 par exemple. L'encadrement de α fournit par le programme est du type : [0 ,999 ; 1], si la précision demandée est le millième, alors que α vaut 0 !

Cela vient du fait que la condition $f(a) \times f(b) < 0$ ne permet pas de reconduire l'invariant de boucle dans ce cas particulier, autrement dit, si pour une valeur de *m*, $f(m) = 0$, alors les conditions d'application du théorème des valeurs intermédiaires ne sont plus remplies à la sortie de la boucle.

Cette situation se produit à chaque fois qu'il existe des nombres entiers naturels n et k tels que : $\alpha = \frac{k(b-a)}{2^n}$.

On peut corriger cet algorithme soit une utilisant une inégalité large : la condition devient :

$f(a) \times f(b) \leq 0$, soit en testant le cas où une racine est atteinte.

Autre approfondissement possible : lien entre la précision demandée, les valeurs de *a* et *b*, et le nombre de passages dans la boucle. En particulier, on peut faire observer le fait que 2^{10} est supérieur à 1000. Ceci peut permettre d'aborder la **notion de performance de l'algorithme**, même s'il faudra encore attendre pour le comparer à d'autres algorithmes.

Programmation de l'algorithme

Lors de la séance de TP, la programmation a été effectuée sur calculatrice, de façon à ce que le programme de la partie C puisse devenir un outil à portée de main. Pendant que les élèves manipulent leur calculatrice, le programme est saisi sur une calculatrice virtuelle vidéoprojetée.

La programmation peut être réalisée sans problème dans les langages de programmation courants.

Les tests, les erreurs constatées pendant la séance:

- Entrées ne satisfaisant pas aux conditions du théorème des valeurs intermédiaires
- Sur calculatrice : oubli de la saisie de la fonction.

[1] Merci à Gilles Charrier pour son rappel sur ce cas particulier.

Conclusion

La séance, réalisée avec deux groupes, s'est déroulée sans encombre ; au-delà de la découverte d'un nouvel algorithme, les élèves semblaient satisfaits de repartir avec un outil puissant dans leur cartable.