TP3 - DICHOTOMIE – METHODE DES TRAPEZES – DECOMPOSITION EN SERIE

Le but de ce TP est d'aborder les sous-programmes à travers trois méthodes de calcul numérique usuelles. On retrouve ici les structures de contrôle vues précédemment : l'alternative et les boucles. À ces structures, on ajoute les notions de procédure et de fonction. Le TP comprend trois parties :

- *La recherche d'un zéro d'une fonction dans un intervalle donné en appliquant le principe de recherche par dichotomie*
- *Le calcul de l'intégrale d'une fonction dans un intervalle donné à l'aide de la méthode des trapèzes*
- *Le calcul de sin(x) à partir de la décomposition en série de Taylor et d'une précision donnée*

1.2 Bibliothèques Standard & fonctions mathématiques

Les trois exercices suivants font appel aux fonctions mathématiques fournies par les bibliothèques standard des langages C et C++. Dans le contexte de ce TP, on utilise les macros et déclarations définies par le standard POSIX IEEE Std 1003.1 : http://www.unix.com/man-page/posix/7posix/math.h/

En C++ le catalogue des macros et déclarations est inclus au début du code source avec la directive : #include <cmath>

Préparation

1. *Comment le nombre π est-il défini dans la bibliothèque standard ?*
2. *Quels sont les types utilisés avec la définition de la fonction sinus ?*
3. *Quels sont les types utilisés avec la fonction qui renvoie la valeur absolue d'un nombre réel ?*
4. *Comment appelle-t-on la fonction qui permet de calculer les puissances sous la forme y^x ?*

1.3 Recherche d'un zéro d'une fonction par dichotomie

La recherche par dichotomie suppose que l'on connaisse à l'avance un intervalle [a, b] dans lequel la fonction étudiée passe par zéro. Cette fonction doit aussi être strictement croissante ou décroissante dans l'intervalle choisi.

Le principe (ou l'algorithme !) est alors le suivant :

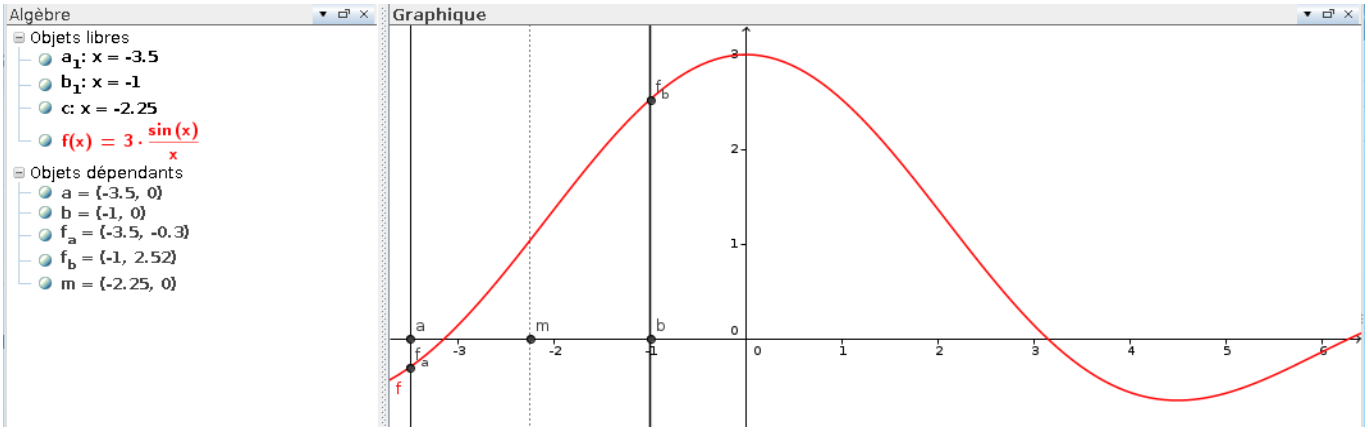
- *Le milieu m de l'intervalle [a, b] est calculé.*
- *Si f(m) et f(a) sont de même signe, l'intervalle de recherche devient [m, b], sinon cet intervalle devient [a, m].*

Le traitement précédent est réitéré jusqu'à ce que l'intervalle de recherche soit plus petit (en valeur absolue) que la précision ε (epsilon) souhaitée. Le zéro de la fonction est alors égal à l'une des deux bornes de l'intervalle.

Dans le contexte de ce TP, la fonction choisie pour la recherche de zéro est la suivante :

$$f(x) = 3 \cdot \frac{\sin(x)}{x} avec f(0) = 3$$

Voici une copie d'écran obtenue avec GeoGebra[1] illustrant un choix particulier d'intervalle [a, b] et la valeur du milieu m correspondant. Ici, f(m) et f(a) sont de signes différents et le nouvel intervalle de recherche devient [a, m].



Il est possible de choisir d'autres intervalles pour lesquels la fonction f(x) choisie passe par zéro.

Préparation

1. *Écrire l'algorithme d'un sous-programme* **SinusCardinal** *qui effectue le calcul de la fonction* $f(x) = 3 \cdot \frac{\sin(x)}{x} avec f(0) = 3$. *Choisir le type de la valeur donnée en entrée et le type de la valeur renvoyée par la fonction.*
2. *Écrire l'algorithme du sous-programme* ***dichotomie*** *qui reçoit en paramètres : les bornes de l'intervalle de recherche initial ainsi que la précision recherchée et renvoie la valeur du zéro de la fonction ainsi que le nombre d'itérations nécessaires pour trouver la solution.*

   Ce sous-programme appellera le sous-programme **SinusCardinal**.
3. *Écrire l'algorithme du programme principal qui demande à l'utilisateur de fournir les bornes de l'intervalle de recherche initial a et b puis appelle le sous-programme* ***dichotomie*** *et enfin affiche la valeur de x pour laquelle la fonction f(x) s'annule ainsi que le nombre d'itérations.*

La précision ε (epsilon) sera définie comme une constante.

4. *Préparer un jeu de test complet de calcul d'un zéro de la fonction f(x) en utilisant des bornes d'intervalle différentes de celles de la copie d'écran GeoGebra donnée ci-avant.*

   Choix des valeurs des bornes de l'intervalle [a, b] initial : a = ___ et b = ___

[1] *fichier GeoGebra a faire_dichotomie.ggb*

Bornes de l'intervalle de recherche	| a – b | < $\mathcal{E}$? VRAI / FAUX	Itération N° ?	Valeur de x pour laquelle *f(x)* ≈ 0

Réalisation

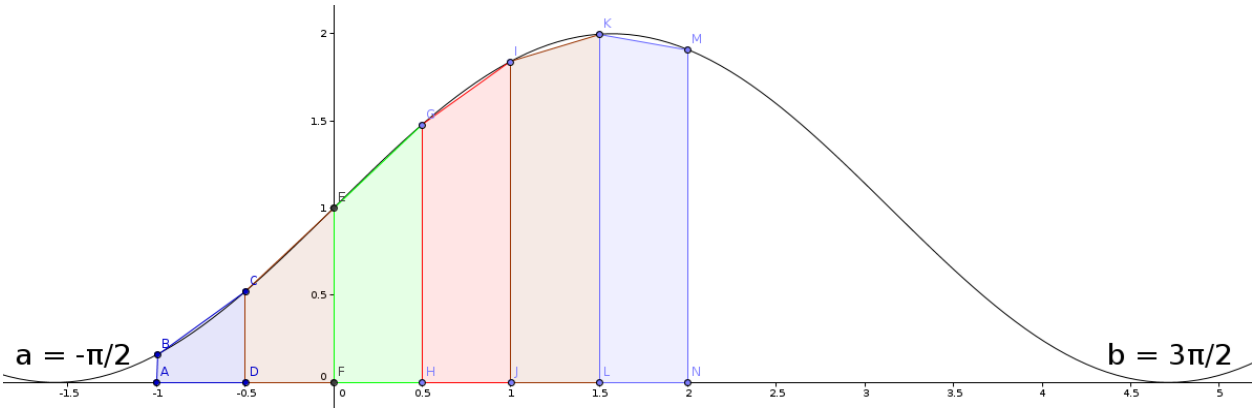
1. *Ouvrir, dans **Netbeans**, un nouveau projet ayant pour fichier source dichotomie.cpp.*
2. *Traduire l'algorithme du programme en C++ et l'éditer dans **Netbeans**.*
3. *Tester le programme en validant le fonctionnement de chaque sous-programme individuellement.*

Vous compléterez le code du sous-programme dichotomie de façon à afficher les valeurs des bornes de l'intervalle de recherche pour chaque itération.

5. *Corriger le code si nécessaire puis conclure.*

1.4 Calcul d'une intégrale par la méthode des trapèzes

L'objectif de cet exercice est de calculer l'intégrale de la fonction $f(x) = \sin(x) + 1$ sur un intervalle défini [a, b] à l'aide de la méthode des trapèzes. Cette méthode consiste à découper l'intervalle choisi en n trapèzes de même largeur dont on sait calculer l'aire.



La somme des aires des trapèzes est une approximation de l'intégrale de la fonction sur l'intervalle [a, b]. Plus le nombre de trapèzes est important, meilleure est l'approximation.

L'expression de la largeur l d'un trapèze est : $l = \frac{b-a}{n}$ où a et b sont les bornes de l'intervalle choisi pour le calcul et n le nombre de trapèzes dont on doit calculer l'aire.

L'expression de l'aire du trapèze ABCD dans la copie d'écran ci-dessus est : $\frac{AD\cdot(AB+DC)}{2} = l \cdot \frac{f(A)+f(D)}{2}$

En se basant sur les bornes de l'intervalle [a, b], l'expression de l'aire de chaque trapèze est de la forme :

Trapèze 1 : $l \cdot \frac{f(a)+f(a+l)}{2}$

Trapèze 2 : $l \cdot \frac{f(a+l)+f(a+2\cdot l)}{2}$

…

Trapèze n-1 : $l \cdot \frac{f(a+(n-2)\cdot l)+f(a+(n-1)\cdot l)}{2}$

Trapèze n : $l \cdot \frac{f(a+(n-1)\cdot l)+f(a+n\cdot l)}{2}$

L'aire totale correspond à la somme de toutes les aires que l'on exprime sous forme développée puis simplifiée :

$$AireTotale = l \cdot [\frac{f(a)}{2} + \frac{f(a+l)}{2} + \frac{f(a+l)}{2} + \cdots + \frac{f(a+(n-1\cdot l))}{2} + \frac{f(a+(n-1\cdot l))}{2} + \frac{f(b)}{2}]AireTotale = l \cdot [\frac{f(a)}{2} + \frac{f(b)}{2} + \sum_{i=1}^{n-1} f(a+i\cdot l)]$$

Préparation

1. *Donner la primitive de la fonction* $f(x) = \sin(x) + 1$*sur l'intervalle [a, b]*$\int_a^b \sin(x) + 1 =?$
2. *Choisir un jeu de bornes a et b en radians et calculer le résultat théorique à approcher par la méthode des trapèzes.*
3. *Préparer un jeu de tests à partir d'un découpage en 5 trapèzes entre les bornes a et b.*

a = ____ *b =* ____ *l =* ____ *f(a)/2 =* ____ *f(b)/2 =* ____

Itération *i* =	***f(a+i.l)* =**	**Aire =**	**Le fonctionnement est-il conforme ?**

Aire Totale = ____

4. *Écrire l'algorithme du sous-programme* ***SinusPlusUn*** *qui effectue le calcul de la fonction f(x) définie ci-avant. Choisir le type de la valeur donnée en entrée et le type de la valeur renvoyée par la fonction.*

5. *Écrire l'algorithme du sous-programme **trapeze** qui reçoit en paramètres : les bornes de l'intervalle de calcul ainsi que le nombre de trapèzes. Ce sous-programme renvoie le résultat du calcul par la méthode des trapèzes*

Ce sous-programme appellera le sous-programme **SinusPlusUn**.

6. *Écrire l'algorithme du programme principal qui demande à l'utilisateur de fournir les bornes de l'intervalle de calcul de l'intégrale a et b ainsi que le nombre de trapèzes. Ce programme principal appelle la fonction **trapeze** et affiche la résultat du calcul intégral par la méthode des trapèzes.*

Le programme principal doit aussi afficher le résultat du calcul intégral «théorique» obtenu avec la primitive de la fonction f(x) ainsi que l'erreur relative entre les deux méthodes de calcul.

Réalisation

1. *Traduire l'algorithme du programme en C++ et l'éditer dans **Netbeans**.*
2. *Tester le programme en validant le fonctionnement de chaque sous-programme individuellement.*

*Vous compléterez le code du sous-programme **trapeze** de façon à afficher la valeur de l'aire de chaque trapèze et la valeur intermédiaire de l'aire totale.*

4. *Corriger le code si nécessaire puis conclure.*

1.5 Calcul de sin(x) à l'aide d'une série de Taylor

Le but de cet exercice est de déterminer le nombre de termes de la série de Taylor à calculer pour une valeur de sin(x) avec une précision donnée. Une fois la précision souhaitée atteinte, on détermine l'erreur relative à l'aide du calcul direct de sin(x).

On utilise une valeur caractéristique de x pour faciliter la mise au point du programme : $x = \frac{\pi}{6}$

L'expression générale de la série de Taylor permettant de calculer sin(x) est la suivante :

$$\sin(x) = x + \sum_{n=1}^{\infty} (-1)^n . \frac{x^{(2n+1)}}{(2n+1)!}$$

L'expression développée de cette même série devient :

$$\sin(x) = x - \frac{x^3}{(3)!} + \frac{x^5}{(5)!} - \frac{x^7}{(7)!} + \frac{x^9}{(9)!} - \frac{x^{11}}{(11)!} + \ldots$$

Préparation

1. *Préparer un jeu de test de façon à obtenir une valeur de* $\sin(\frac{\pi}{6})$ *avec une précision de* 10^{-9}.

Compléter le tableau ci-après :

Terme n=	$(-1)^n$	$x^{(2n+1)}$	$(2n+1)!$	$(-1)^n.\frac{x^{(2n+1)}}{(2n+1)!}$	**Somme =**

Valeur approchée de *sin(x)* = _____

2. *Reprendre l'algorithme de la fonction **puissance** étudiée en TD et le modifier pour prendre en compte le type de la valeur donnée en entrée et le type de la valeur renvoyée par la fonction.*
3. *Reprendre l'algorithme de la fonction **factorielle** donnée en préambule du TP n°1 et le modifier afin de prendre en compte le type de la valeur donnée en entrée et le type de la valeur renvoyée par la fonction.*
4. *Écrire l'algorithme du sous-programme **sinusTaylor** qui recoit en paramètres la valeur de x et la précision ε (epsilon) souhaitée. Ce sous-programme calcule les termes de la série de Taylor les uns après les autres. Le calcul s'arrête dès que le terme calculé est inférieur à la précision souhaitée. La valeur approchée du sinus est alors renvoyée au programme appelant ainsi que le nombre de termes calculés pour atteindre la précision souhaitée.*

Ce sous-programme fait appel aux deux précédents pour les calculs des termes de la série.

5. *Écrire l'algorithme du programme principal qui demande à l'utilisateur de fournir la valeur de x et la précision souhaitée pour la valeur approchée de sin(x). Ce programme principal appelle le sous-programme **sinusTaylor** et affiche la résultat du calcul.*

Le programme principal doit aussi afficher le résultat du calcul direct de sin(x) ainsi que l'erreur relative entre les deux méthodes de calcul.

Réalisation en séance

5. *Traduire l'algorithme du programme en C++ et l'éditer dans **Netbeans**.*
6. *Tester le programme en validant le fonctionnement de chaque sous-programme individuellement.*

*Vous compléterez le code du sous-programme **sinusTaylor** de façon à afficher la valeur de chaque terme calculé et la valeur intermédiaire de sin(x).*

8. *Corriger le code si nécessaire puis conclure.*