MONET
Maxime
RT2-A1
20/02/21

TP n°2 Calcul d’intégrales

Methode 1 : La méthode des rectangles.

Mon approche ici va être de créer une fonction ***f1*** qui va calculer une fonction pour un x donné qui sera passé en paramètre, puis une fonction ***f2*** qui sera une fonction différente.

Je m’occupe ensuite de la fonction ***integrale*** qui va prendre en paramètre des doubles et qui va renvoyer un double.

Parmi les paramètres de ma fonction ***integrale*** on va retrouver :

- Un pointeur vers ma fonction ***f***.

- Un double ***a*** qui sera ma première borne.

- Un double ***b*** qui sera ma deuxième borne.

- Un double ***h*** qui sera le nombre de rectangle que je vais choisir entre ***a*** et ***b*** (la découpe).

Dans ma fonction je déclare une variable double aire, un double x, et un double L qui définir la largeur de mon rectangle.

Pour définir ma largeur je fais la formule suivante :

```
(b-a) / h
```

Le but va être de créer une boucle pour calculer l’air de chaque rectangle autant de fois que j’ai choisis avec h.

```
for(n = 0 ; n < h; n++)
  {
     aire += f(a+(L*n)) * L;
     //printf("aire : %f\n",aire);
     //x+= x+L;
     n++;
   }
```

Nous allons ajouter à chaque fois l'aire avec l'aire précédent et incrémenter mon pas avec n qui s'incrémente avec ma boucle.

Pour effectuer mes tests, j'ai choisis un cosinus car il possède des parties négatives et positives.

Cela me permettra de tester si l'aire considérée est signée.

Si on augmente la valeur de h, nous allons découper de plus en plus notre fonction en petit rectangle, ce qui va le rendre très précis !

Pour observer ce phénomène j'ai crée une boucle dans mon main qui fera varier la valeur de h, ainsi nous pourrons observer son impact sur la précision du résultat.

```
    int n = 5 ;
    while (n<=10000)
    {
        double methode1=integral(f,1,2,n);
        printf("intégral : %lf avec h = %d\n",methode1,n);
        n = n*5;
    }
```

```
Intégral :0.096613 avec h=5
intégral :0.045065 avec h=25
intégral :0.036088 avec h=125
intégral :0.034346 avec h=625
intégral :0.034000 avec h=3125
```

Plus on avance plus on remarque que l'on tend vers une valeur précise et stable, l'influence de h est donc très importante et se doit d'être assez élever (> 100) pour obtenir une valeur précise.

Méthode 2: La méthode des rectangles (encadrement de la solution).

Pour aborder la méthode 2 nous allons nous heurter à un problème que nous allons devoir résoudre qui est le changement entre la grande air et la petite air suivant que la courbe soit croissante ou décroissante.

Nous allons commencer par récupérer notre fonction ***f*** de la méthode numéro 1.

```
double f (double x) // Fonction pour la fonction f(x)
{
      double fonction = cos( x );
      return fonction;
}
```

Ensuite nous allons devoir calculer deux fois l'aire pour une largeur donnée et ensuite les soustraire.

Nous allons donc partir sur deux fonction : ***Integralplus*** et ***Integralmoins***.

Commencons par la fonction ***Integralplus***.

On commence par initialiser nos variables :

```
double aire = 0;
    double x = a;
    double L = (b-a)/h;
    int n;
```

Nous allons ensuite boucler autour de h comme la méthode 1.

```
for(n = 0 ; n < h; n++)
    {
        double petitaire = f(a+(L*n)) * L;
        double grandaire = f(a+(L*n)+L) * L;
         x = (grandaire>petitaire)?grandaire:petitaire;
         aire += x;
        // printf("aire plus : %f\n",aire);
    }
```

L'enjeu ici est de toujours sélectionner l'air supérieur que la courbe soit croissante ou décroissante.

Pour ce faire nous allons à chaque fois calculer f(a) et f(a+L) et déterminer quelle est l'aire la plus grand pour ensuite l'ajouter dans ma variable et passer à l'occurrence suivante.

Une petite subtilité ici, j'ai remplacer un if par une petite ligne qui est la suivante :

```
x = (grandaire>petitaire)?grandaire:petitaire;
```

si grande aire > petit aire donc x = grand aire , else x=petit aire.

Enchaînons avec la fonction ***Integralmoins :***

La seule chose qui change ici sera mon test, je vais choisir le plus petit des deux.

```
        for(n = 0 ; n < h; n++)
    {
        double petitaire = f(a+(L*n)) * L;
        double grandaire = f(a+(L*n)+L) * L;
        x = (petitaire<grandaire)?petitaire:grandaire;
        aire += x;
        //printf("aire moins : %f\n",aire);

    }
```

Dans mon main je vais ensuite afficher l'influence encore une fois de h sur les bornes S+ et S- avec l'implémentation d'une boucle qui sera la même que pour la méthode 1 :

```
    int n = 5;
    while (n<10000)
    {
    double Splus = integralplus(f,1,2,n);
    double Smoins = integralmoins(f,1,2,n);
    printf("S+ : %lf pour h = %d\n",Splus,n);
    printf("S- : %lf pour h = %d\n\n",Smoins,n);
    n = n*5;
    }
```

```
S+ : 0.163245 pour h=5
S- : -0.028045 pour h=5

S+ : 0.086946 pour h=25
S- : 0.048688 pour h=25

S+ : 0.071652 pour h=125
S- : 0.064000 pour h=125
S+ : 0.068592 pour h=625
S- : 0.067061 pour h=625

S+ : 0.067979 pour h=3125
S- : 0.067673 pour h=3125
```

Méthode précise mais on nous n'arrivons pas au résulat pour h = 3125 !

Méthode 3 : La méthode des trapèzes :

Je récupère à nouveau ma fonction f de mes précédentes méthode :

```
double f (double x) // Fonction pour la fonction f(x)
{
      double fonction = cos( x );
      return fonction;
}
```

On commence par définir une fonction trapeze qui prends les même arguments que les autres méthodes.

On initialise également les variables que nous allons utiliser.

```
double trapeze (double (*f)(double ),double a,double b,double h)
{
    double aire = 0;
    double L = (b-a)/h;
    int n;
```

Je vais appliquer la formule de calcul d'aire d'un trapèze qui est ***((a+b)*L) /L*** , nous allons ensuite boucler sur la valeur de h.

```
for(n = 0 ; n < h; n++)
    {
        aire += ( ( f(a+(L*n)) + f(a+(L*n)+L) ) * L ) / 2;
    }
```

La seule chose qui change ici est la formule pour calculer un trapèze que j'ai appliqué.

Je vais encore faire varier h pour voir l'influence de h sur la précision du résultat.

```
Aire avec methode trapèze: 0.067600 pour h=5
Aire avec methode trapèze: 0.067817 pour h=25
Aire avec methode trapèze: 0.067826 pour h=125
Aire avec methode trapèze: 0.067826 pour h=625
Aire avec methode trapèze: 0.067826 pour h=3125
```

Nous remarquons ici que malgré un h petit, nous obtenons un résultat très précis et rapidement comparer aux deux méthodes précédentes. On remarque qu'avec h = 125, nous trouvons la solution.

Méthode 4 : La méthode des polynômes :

Je récupère mon fonction qui me donne ma fonction cos(x) comme pour les autres puis je m'occupe de ma fonction qui va calculer l'intégral.

```
double polynome (double (*f)(double ),double a,double b,double h)
{
    int n;
    double aire = 0;


    for(n = 0 ; n < h; n++)
    {
        double L = (b-a)/h;
        double La = a+(L*n) ;
        double Lb = a+(L*n)+L;
        //printf("La : %lf\n",La);
        //printf("Lb : %lf\n",Lb);
        //printf("L : %lf\n",L);
        // équivalent à  (b-a)/6 * ( f(a) + 4*f( (a+b)/2 )+ f(b) ) formule du
TP
        aire+=(((Lb)-La)/6) * (f(La)+4*f((La+(Lb))/2)+f(Lb));
    }
return aire;
}
```

Nous allons encore boucler autour de h. Cette fois ci je vais mettre ma borne a et ma borne b en fonction de n dans une variable pour faciliter la lecture du calcul.

Nous allons utiliser la même boucle pour faire varier h et observer son influence.

```
int n = 1;
    while (n<26)
    {
        double meth4 = polynome(f,1,2,n);
        printf("Aire methode polynomes : %lf pour h = %d\n",meth4,n);
        n = n*5;
    }
```

```
Aire methode polynomes : 0.067851 pour h = 1
Aire methode polynomes : 0.067826 pour h = 5
Aire methode polynomes : 0.067826 pour h = 25
```

Nous remarquons l’efficacité de cette méthode, car oui avec un h = 5 nous obtenons le bon résultat !

Même avec h = 1 nous arrivons à un résultat très très précis !