TP C#10 : Mathmageddon

1 Consignes de rendu

A la fin de ce TP, vous devrez rendre une archive respectant l'architecture suivante :

```
rendu-tp-firstname.lastname.zip
|-- firstname.lastname/
    |-- AUTHORS
    |-- README
    |-- TP10/
        |-- TP10.sln
        |-- TP10/
            |-- Everything except bin/ and obj/
```

N'oubliez pas de vérifier les points suivants avant de rendre :

- Remplacez `prenom.nom` par votre propre login et n'oubliez pas le fichier `AUTHORS`.
- Les fichiers `AUTHORS` et `README` sont obligatoires.
- Pas de dossiers `bin` ou `obj` dans le projet.
- Respectez scrupuleusement les prototypes demandés.
- Retirez tous les tests de votre code.
- **Le code doit compiler !**

AUTHORS

Ce fichier doit contenir une ligne formatée comme il suit : une étoile (*), un espace, votre login et un retour à la ligne. Voici un exemple (où '$' est un retour à la ligne et ' ' un espace) :

```
* firstname.lastname$
```

Notez que le nom du fichier est `AUTHORS` sans extension. Pour créer simplement un fichier `AUTHORS` valide, vous pouvez taper la commande suivante dans un terminal :

```
echo "* firstname.lastname" > AUTHORS
```

README

Vous devez écrire dans ce fichier tout commentaire sur le TP, votre travail, ou plus généralement vos forces / faiblesses, vous devez lister et expliquer tous les bonus que vous aurez implémentés. Un `README` vide sera considéré comme une archive invalide (malus).

2 Introduction

2.1 Objectifs

La semaine dernière, vous avez fait de la physique. Cette semaine, on va lier de manière concrète une autre matière avec la programmation : les maths et plus particulièrement le domaine de l'analyse. Il est bien de savoir écrire ces formules et les résoudre sur papier mais il serait encore plus intéressant de savoir coder des notions primordiales tels que les dérivées et les intégrales de n'importe quelle fonction. Vous allez apprendre durant ce TP à approximer ces valeurs, et à réduire la marge d'erreur pour avoir la meilleur précision possible. Les notions abordées ne seront pas faciles, mais si vous réussissez ce TP, vous aurez encore un peu amélioré votre capacité à appliquer l'informatique à votre environnement.

3 Cours

Avant de commencer le cours, il faut comprendre qu'il est extrêmement nécessaire. Vous risquez de perdre beaucoup de temps si vous ne le lisez pas, car il comporte beaucoup d'informations utiles que vous allez devoir utiliser. Si vous ne comprenez pas quelque chose, n'hésitez pas à demander de l'aide aux assistants.

3.1 Les dérivées

Commençons par redéfinir les bases. Avant de parler de dérivée d'une fonction, il faut d'abord commencer par parler du nombre dérivé en un point réel.

Definition

Le nombre dérivé en un « point » réel x d'une fonction f à variable et valeurs réelles est le coefficient directeur de la tangente au graphe de f au point $(x, f(x))$. Cette tangente est l'approximation affine de f en x.

La dérivée d'une fonction f est donc une fonction qui, à tout nombre pour lequel f admet un nombre dérivé, associe ce nombre dérivé.

Lorsque vous dérivez des fonctions en analyse, vous le faites généralement avec des formules toutes faites tel que $(sin(x))' = cos(x)$. Même s'il est possible de faire cela en programmation avec des manipulations de strings, des parsers, etc, ce n'est pas du tout l'objectif de la semaine. Dans ce TP, on ne va pas essayer de trouver la formule générale de la dérivée d'une fonction, mais simplement des valeurs en certains points. Si jamais on prend beaucoup de points et qu'on trace ces points sur un graphique, on verra quelque chose qui ressemble à la dérivée sans l'être véritablement.

Vous connaissez tous la formule suivante, même si beaucoup d'entre vous on choisit de l'oublier pour diverses raisons.

Formule

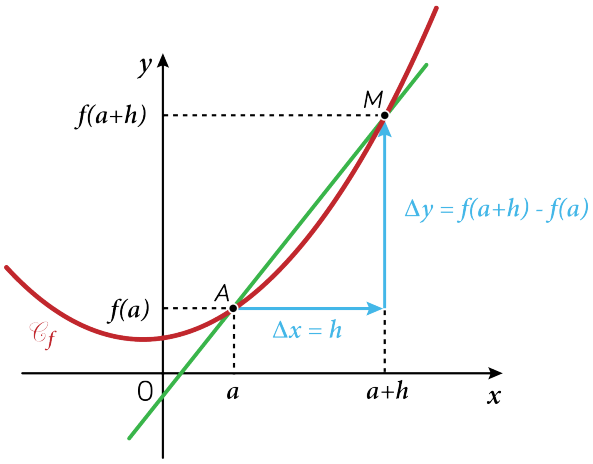
Soit f une fonction définie sur un intervalle I de $\mathbb{R}$, et $a \in I$. Soit h un nombre réel non nul tel que $a+h \in I$. On appelle **taux d'accroissement** de la fonction f entre a et $a+h$, le nombre réel

$$\frac{f(a+h) - f(a)}{h} = \frac{\Delta y}{\Delta x}$$

Par exemple, le taux d'accroissement de la fonction $f : x \to x^2$ entre 1 et $1 + h$ est

$$\frac{f(1+h) - f(1)}{h} = \frac{(1+h)^2 - 1^2}{h} = \frac{2h + h^2}{h} = 2 + h$$

Voici un petit graphique expliquant visuellement le taux d'accroissement



Note

Si vous voyer des Δ en maths, cela veut dire qu'on veut faire varier ces valeurs (généralement vers 0).

A partir du taux d'accroissement, on peut définir la formule mathématique de la dérivée d'une fonction en un point.

Formule

Soit f une fonction définie sur un intervalle I de $\mathbb{R}$, et $a \in I$. On dit que **la fonction est dérivable en a** si le taux d'accroissement, noté $f'(a)$, lorsque h tend vers 0 est

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

Le nombre $f'(a)$ – lorsqu'il existe – désigne le coefficient directeur de la droite tangente à la courbe au point d'abscisse a.

On connaît tous la dérivée de $f(x) = x^2$. C'est $f'(x) = 2x$. On peut facilement voir une relation apparaître. D'après le calcul en fin de page précédente : si jamais h est zéro, alors le taux d'accroissement de 1 à 1 (donc au point $x = 1$) est 2. Hors on sait que $f'(1) = 2$. On voit donc bien qu'en théorie aussi bien qu'en pratique, lorsque h est 0, on arrive au résultat.

Lorsqu'on fait la transition entre les maths et la programmation, il faut savoir comment traduire les formules qui possèdent une limite. Un humain arrive facilement à faire l'abstraction nécessaire pour comprendre une limite. Mais un ordinateur n'y arrivera jamais. Autrement dit, il est impossible d'écrire une telle formule en code.

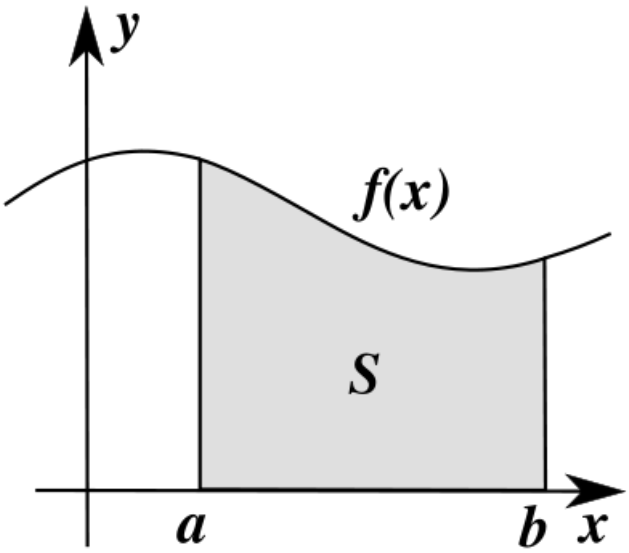
Ainsi en programmation, si on ne connaît pas la formule exacte de la dérivation on ne peut qu'approximer la dérivée, en prenant des valeurs pour h proches de zéro. Ce choix se traduit par une **marge d'erreur** que l'on peut contrôler selon nos besoins.

3.2 Les intégrales

L'intégration est comme chacun sait l'opération contraire de la dérivation. Mais une question se pose. En quoi est-ce utile de savoir faire cela ? Encore une fois, les maths ne sont qu'un outil, et le dernier TP devrait vous avoir donné une idée de leur importance pour résoudre des problèmes divers, comme de mécanique par exemple.

Pour trouver l'approximation de la dérivée, nous sommes partis de la formule. Or pour les intégrales, il n'est pas possible de faire comme cela. Il est beaucoup plus de simple de se référer au point de vue géométrique pour trouver une approximation de l'intégrale étudiée.

Comme vous le savez, l'intégrale d'une fonction entre a et b est simplement l'aire sous la courbe (ici S).



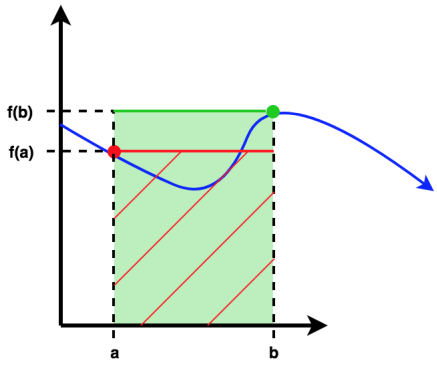
Il existe plusieurs méthodes qui nous permettent d'estimer la valeur de l'intégrale d'une fonction.

Attention

On ne parle que d'intégrale définie, donc toujours sur un intervalle entre un point a et un point b avec $a \leq b$

3.2.1 Méthode des rectangles

La méthode des rectangles est aussi appelée *méthode à un point*. Elle consiste en l'approximation de l'aire sous la représentation graphique de $f(x)$ par un rectangle. Ce rectangle s'obtient en traçant le segment horizontal partant d'un des deux points de la courbe. Le segment a donc pour couple de points $\{(a, f(a)), (b, f(a))\}$ (rectangle rouge) ou $\{(a, f(b)), (b, f(b))\}$ (rectangle vert).



Formule

L'aire du rectangle inférieur (rouge) est

$$\int_a^b f(a)d(x) = b * f(a) - a * f(a) = f(a) * (b - a)$$

L'aire du rectangle supérieur (vert) est $\int_a^b f(b)d(x) = b * f(b) - a * f(b) = f(b) * (b - a)$.

Il est difficile de ne pas remarquer que cette méthode d'approximation manque cruellement de précision, telle que présentée dans le graphe. Cependant, on peut grandement améliorer le résultat en divisant l'intervalle étudié en une multitude de segments de même taille d'intervalle. Plus deux points sont proches, plus l'approximation se rapproche de la valeur réelle.

On partage donc un intervalle $I = [a, b]$ en n intervalles de même largeur $\frac{(b-a)}{n}$. L'intégrale de $f(x)$ sur cet intervalle peut être calculée comme une somme des intégrales de chaque intervalle.

Formule

L'aire de $f(x)$ sur I est

$$\int_a^b f(x)dx = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + ... + \int_{x_{n-1}}^{x_n} f(x)dx$$

On remplace la fonction $f(x)$ par des fonctions constantes, contenant un des deux points de l'intervalle $[x_i, x_{i+1}]$, facilement intégrables (l'aire étant réduite à un rectangle) ce qui nous donne deux approximations :

Formule

rectangles inférieurs :

$$A_1 = \int_{x_0}^{x_1} f(x_0)dx + \int_{x_1}^{x_2} f(x_1)dx + ... + \int_{x_{n-1}}^{x_n} f(x_{n-1})dx$$

$$A_1 = \frac{b-a}{n} \sum_{i=0}^{n-1} f(x_i)$$

rectangles supérieurs :

$$A_2 = \int_{x_0}^{x_1} f(x_1)dx + \int_{x_1}^{x_2} f(x_2)dx + ... + \int_{x_{n-1}}^{x_n} f(x_n)dx$$

$$A_2 = \frac{b-a}{n} \sum_{i=1}^{n} f(x_i)$$

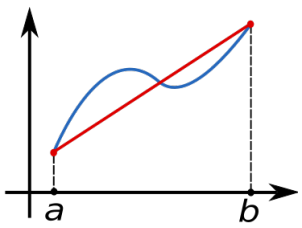
Bien évidemment, $x_0 = a$ et $x_n = b$.

Note

La moyenne de ces deux valeurs correspond à l'approximation de l'intégrale $\int_a^b f(x)dx$ par la méthode des trapèzes.

3.2.2 Méthode des trapèzes

La méthodes des trapèzes est aussi appelée *méthode à deux points*. Elle marche en approximant la région en dessous du graphe de la fonction $f(x)$ par un trapèze et en calculant son aire. Pour faire cela, la méthode va prendre les deux points d'abscisse a et b et tracer une droite affine entre $(a, f(a))$ et $(b, f(b))$ (voir figure ci-dessous). Cette droite va servir de haut au trapèze.



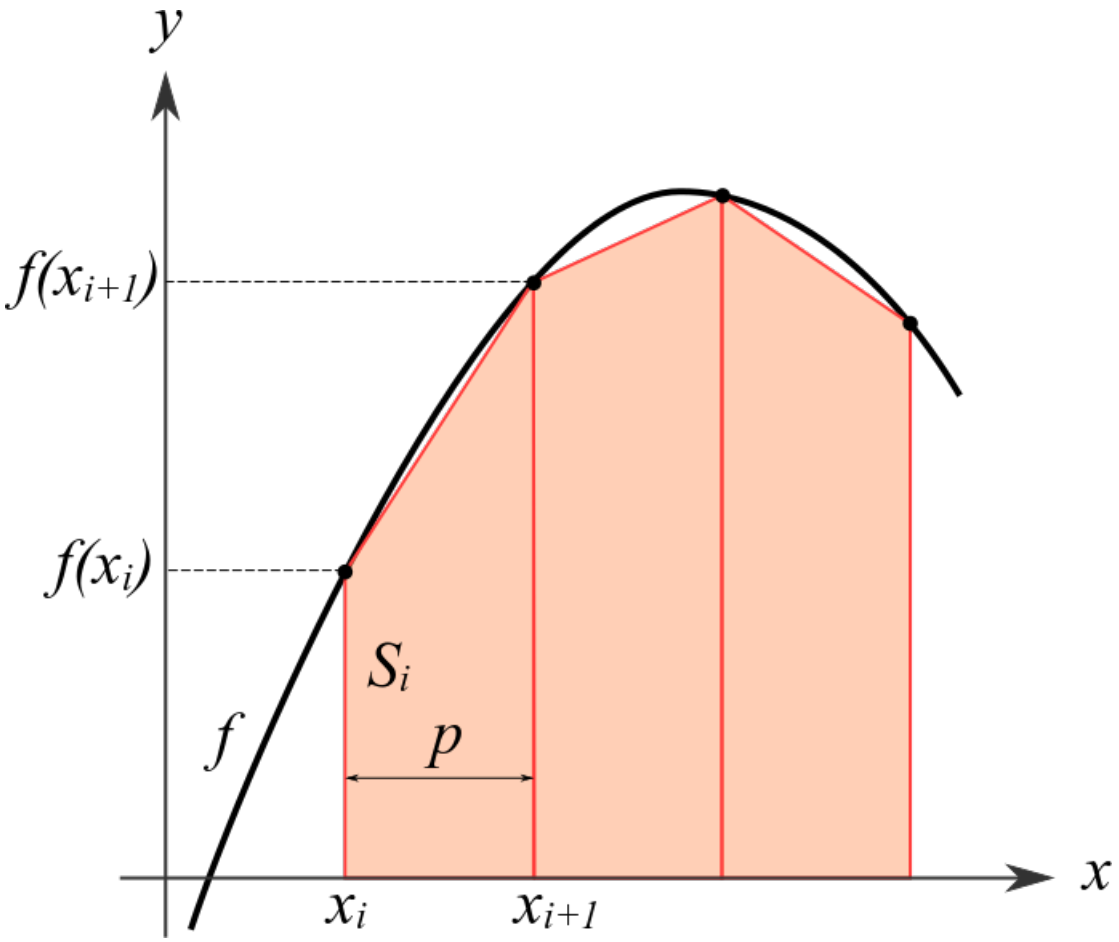
On peut facilement voir pourquoi cette méthode ne donne qu'une approximation. Elle approxime la fonction par une droite affine passant par deux points qui peuvent être très éloigné l'un de l'autre. Cependant, plus les points sont proches, plus la marge d'erreur est petite.

Formule

L'aire sous la courbe est approximativement égale à l'aire du trapèze.

$$\int_a^b f(x)dx \approx (b-a)[\tfrac{f(a)+f(b)}{2}]$$

De même que pour la méthode des rectangles, l'approximation s'améliore avec le rétrécissement de l'intervalle étudié.



Ainsi on peut définir une nouvelle formule

Formule

Soit $\{x_k\}$ un intervalle de $[a, b]$ tel que $a = x_0 < x_1 < ... < x_{N-1} < x_N = b$ alors

$$\int_a^b f(x)dx \approx \sum_{k=1}^{N} \frac{f(x_{k-1}) + f(x_k)}{2}(x_k - x_{k-1})$$

Cette formule a l'air longue et compliquée mais soyez tranquille, on ne va pas l'utiliser tel quel. Cette formule est aussi valable pour des sous-intervalles qui n'ont pas la même taille. Si on décide de n'utiliser plus que des intervalles de même taille, on peut simplifier et arriver a la formule finale.

Formule

Soit $\Delta x_k = \Delta x = \frac{b-a}{N}$ avec N le nombre d'intervalle de même taille alors

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2} \sum_{k=1}^{N} (f(x_{k-1}) + f(x_k))$$

3.2.3 Méthode de Simpson

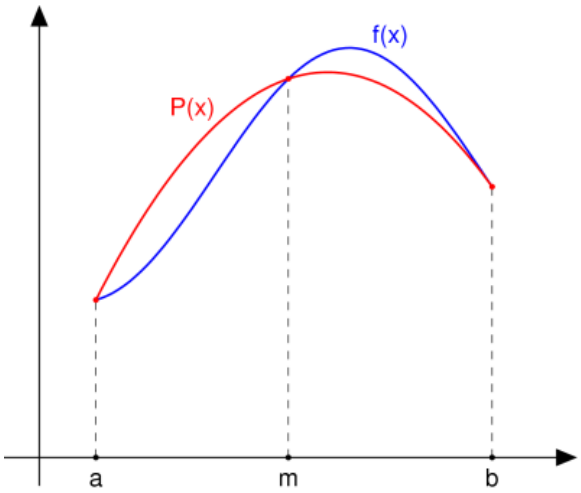
La méthode de Simpson est aussi appelée *méthode à trois points*. Le défaut principal de la méthode des rectangles ou de la méthode des trapèzes est qu'elles manquent de finesse pour

approximer correctement des courbes. La méthode de Simpson est comparable aux précédentes mais va approximer la fonction par un polynôme de degré 2. Ce polynôme est défini par trois points d'abscisses a, b, et $m = \frac{a+b}{2}$ ayant les mêmes ordonnées que sur la courbe de f.

Formule

En choisissant les points $(a, f(a))$, $(b, f(b))$, $(m, f(m))$, avec $m = \frac{a+b}{2}$, le polynôme cherché peut s'écrire par interpolation lagrangienne

$$P(x) = f(a)\frac{(x-m)(x-b)}{(a-m)(a-b)} + f(m)\frac{(x-a)(x-b)}{(m-a)(m-b)} + f(b)\frac{(x-a)(x-m)}{(b-a)(b-m)}$$



Le polynôme est une fonction très facile à intégrer, et la zone d'erreur est très amoindrie comparée à celles des deux méthodes précédentes.

Formule

L'aire sous la courbe est approximativement égale à celle du polynôme.

$$\int_a^b f(x)dx \approx \frac{b-a}{6}[f(a) + 4f(\frac{a+b}{2}) + f(b)]$$

Une fois encore, plus l'intervalle est petit, meilleure est l'approximation. Introduisons donc la formule de Simpson composite.

Formule

Soient n sous-intervalles dans $[a, b]$ avec n **pair**, $h = \frac{b-a}{n}$ la longueur des sous-intervalles, et $x_i = a + ih$ pour i entier de 0 à n,

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(x_0) + 2\sum_{j=1}^{\frac{n}{2}-1} f(x_{2j}) + 4\sum_{j=1}^{\frac{n}{2}} f(x_{2j-1}) + f(x_n)]$$

Note

La méthode de Simpson présentée est la méthode usuelle de Simpson, dite méthode 1/3. La méthode de Simpson 3/8 emploie un polynôme de degré 3 et sa formule est beaucoup plus lourde.

3.2.4 Méthode de Newton-Cotes

La méthode de Newton-Cotes est la généralisation de toutes les méthodes vues jusqu'à maintenant. On va choisir des points équidistants pour créer des sous intervalles de même taille. Autrement dit, si on prend un point avec cette méthode, on arrive à la méthode des rectangles, 2 points à la méthode des trapèzes et trois points à la méthode de Simpson.

Le but de cette formule est de créer par interpolation lagrangienne un polynôme de degré n, facilement intégrable.

Formule

Soit $x_i = a + i\Delta$, pour $i = 0, ..., n$, $\Delta = (b-a)/n$ et $y = (x-a)/\Delta$. La formule de degré n est définie par

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i)$$

avec

$$w_i = \frac{b-a}{n} \frac{(-1)^{n-i}}{i!(n-i!)} \int_0^n \prod_{k=0, k\neq i}^{n} (y-k)dy$$

Comme vous pouvez le voir, la formule pour calculer w_i contient une intégrale, nous allons devoir passer par une approximation. Voici donc un petit tableau de référence avec les w_i calculés pour vous.

Degré (n)	Nombre de Points	Nom
1	2	Méthode des trapèzes
2	3	Méthode de Simpson 1/3
3	4	Méthode de Simpson 3/8
4	5	Méthode de Boole-Villarceau
6	7	Méthode de Weddle-Hardy
7	8	Méthode à 8 points
8	9	Méthode à 9 points
9	10	Méthode à 10 points
10	11	Méthode à 11 points

Degré (n)	w (de 0 a n)	constante
1	1, 1	$\frac{b-a}{2}$
2	1, 4, 1	$\frac{b-a}{6}$
3	1, 3, 3, 1	$\frac{b-a}{8}$
4	7, 32, 12, 32, 7	$\frac{b-a}{90}$
6	41, 216, 27, 272, ...	$\frac{b-a}{840}$
7	751, 3577, 1323, 2989, ...	$\frac{b-a}{17280}$
8	989, 5888, -928, 10496, -4540 ...	$\frac{b-a}{28350}$
9	2857, 15741, 1080, 19344, 5778, ...	$\frac{b-a}{89600}$
10	16067, 106300, -48525, 272400, -260550, 427368, ...	$\frac{b-a}{598752}$

L'intégration du polynôme du théorème de Simpson est un bon exemple de l'utilisation des valeurs ci-dessus. Plus génériquement, on a :

Formule

Soient n sous-intervalles dans $[a, b]$ avec n pair, $h = \frac{b-a}{n}$ la longueur des sous-intervalles, et $x_i = a + ih$ pour i entier de 0 à n,

$$\int_a^b f(x)dx \approx < constant > [w_0 f(x_0) + w_1 f(x_1) + ... + w_n f(x_n)]$$

3.2.5 Quadrature de Gauss-Legendre (Bonus)

On va maintenant parler de la quadrature de Gauss. En règle générale, une quadrature est une approximation de l'intégrale définie qu'on remplace par une somme pondérée prise en un certain nombre de points dans le domaine d'intégration. La méthode de quadrature de Gauss est une méthode de quadrature exacte pour un polynôme de degré 2n – 1 avec n points pris sur le domaine d'intégration.

Formule

Pour un intervalle $[a, b]$,

$$I = \int_a^b f(x)\varpi(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

où $\varpi : (a, b) \to \mathtt{R}_+$ est une fonction de pondération, qui peut assurer l'intégrabilité de f.

Dans ce TP, on va parler de la variante de cette méthode, appelée la méthode de Gauss-Legendre. Elle ne peut résoudre que les problèmes d'intégration les plus classiques. Il s'agit d'intégrer la fonction f sur $[a, b]$. Les n nœuds sont les racines du n-ième polynôme de Legendre $P_n(x)$.

Formule

Les coefficients sont donnés par

$$w_i = \frac{-2}{(n+1)P_n'(x_i)P_{n+1}(x_i)} = \frac{-2}{(1-x_i^2)P_n'(x_i)^2}$$

Voici un petit tableau avec des formules

Nombre de points, n	Poids (w_i)	Points (x_i)	Polynôme de Legendre
1	2	0	x
2	1, 1	$\check{}\sqrt{1/3}$,$\sqrt{1/3}$	(3x2 – 1)/2
3	5/9, 8/9, 5/9	$\check{}\sqrt{3/5}, 0$,$\sqrt{3/5}$	(5x3 – 3x)/2

Un petit exemple pour une intégration simple

$$\int_{-1}^{1}(x+1)^2dx = 1(\frac{1}{\sqrt{3}}+1)^2 + 1(-\frac{1}{\sqrt{3}}+1)^2 = \frac{8}{3}$$

et on peut facilement vérifier car on connaît la primitive de $(x+1)^2$

$$\int_{-1}^{1}(x+1)^2 = \left[\frac{(x+1)^3}{3}\right]_{-1}^{1} = \frac{8}{3}$$

4 Marge d'erreur

Un des buts de ce TP est de vous montrer que toutes ces techniques semblent identiques mais n'ont pas les mêmes résultats. Elles calculent toutes les mêmes choses mais avec des degrés de précision bien différents les uns des autres. Nous vous avons présenté ces méthodes de la moins précise a la plus précise.

Formule

Empiriquement, la formule pour connaître la marge d'erreur (δ) est

$$\delta = \int_a^b f(x)dx - \text{'la valeur trouvée par vos calculs pour la même intégrale'}$$

Pour les exercices de précision demandés pour chaque méthode, il faudra utiliser ce calcul en fonction du nombre de sous-intervalles choisis entre a et b.

5 Enfin, du code.. ou presque

5.0.1 Les delegates et les expressions lambdas

Nous allons utiliser des notions assez sympas que vous avez utilisé durant le TP Photoshop, appelées delegates et expressions lambdas. Ce sont des notions compliquées mais qu'on va aborder facilement.

L'utilisation principale des ces lambdas est de pouvoir passer en arguments de fonctions de manière générique des fonctions mathématiques tel que $f : x \to exp(x)$ ou $f : x \to sin(x) : w$. Voici comment on déclare ce type :

```
1  Func<double, double> f = x => x * x;
```

Cette ligne crée une variable *Func* (aussi appellée *delegate*) avec le nom f. On remarque des chevrons après le mot-clé *Func* : ils indiquent que la fonction prend un double aussi bien en input qu'en output. La partie de droite est une lambda expression, la ligne est compréhensible d'elle même. À tous les x qu'on lui donne, elle va transformer le x d'entrée en $exp(x)$. Ce qui est exactement la définition d'exponentielle de x.

Ainsi on peut faire des choses comme ça :

```
1  Func<double, double> f = x => Math.Exp(x);
2  Func<double, double> g = x => Math.Log(x);
3  Func<double, double> h = x => x * x;
4  Console.WriteLine(f(g(2)) + h(4)); // 18
5
6  // avec
7  public double example(Func<double, double> f)
8  {
9      return f(2);
10 }
11
12 Console.WriteLine(example(h)); // 4
```

5.0.2 Les dictionnaires

Profitons de ce TP pour voir les dictionnaires. Ce sont des structures de données souvent utilisées quand on a besoin d'appliquer les mêmes notions à une grande collection d'objet. Le dictionnaire est composé d'une paire *key*, *value*. La *key* permet d'accéder à sa *value* en la passant au dictionnaire.

```
1  // on le déclare ainsi
2  Dictionary<string, string> profs = new Dictionary<string, string>();
3
4  // et on l'utilise ainsi
5  profs.Add("Prof1","Krisboul");
6  profs.Add("Prof2", "Junior");
7  profs.Add("Prof3", "ChoKaPeek");
8
9  Console.WriteLine(profs["Prof1"]); // "Krisboul"
10 profs.Remove("Prof3"); // remove "ChoKaPeek" from the dictionnary
```

C'est bon, la partie cours est enfin terminée... Il y a énormément d'informations, n'hésitez pas à relire, ou poser des questions aux assistants. Maintenant place au fun.

6 Exercices

6.1 Dérivée

Commençons bien, commençons simple. Vous allez commencer par coder une fonction qui prends un point a, la valeur h, et un pointeur sur fonction et qui retourne le taux d'accroissement de cette fonction. Il aura une précision de 6 (soit 6 chiffre après la virgule). N'oubliez pas de gérer d'éventuelles exceptions !

```
public static double RateOfChange(double a, double h,
    Func<double, double> f);
```

Ensuite, il nous faut une fonction qui génère une liste de points qui correspondent au résultat du taux d'accroissement entre les points d'un intervalle $[a, b]$ donné et avec un incrément t donné.

```
public static List<double> GeneratePointsForPlot(double a, double b,
    double t, Func<double, double> f);
```

On vous invite à jouer avec, pour remarquer les différences de précision en fonction de h sur différents Func<double, double>. Essayez notamment $sin(x)$, $cos(x)$, $exp(x)$, $ln(x)$, et d'autres polynômes de votre choix.

6.2 Intégrales

6.2.1 Méthode des rectangles

Commençons par faire une fonction qui calcule l'intégrale d'une fonction sur un intervalle $[a, b]$ avec la méthode des rectangles (inférieurs).

```
public static double IntegralRectangleRule(double a, double b,
    Func<double, double> f);
```

Après cet amuse-bouche, codez une fonction qui divise l'intervalle $[a, b]$ en plus petit intervalles de taille n, appliquer la méthode des rectangles sur chacun, et sommer le tout.

```
public static double CompositeIntegralRectangleRule(double a, double b,
    double n, Func<double, double> f)
```

Enfin il faut calculer la marge d'erreur de la technique. Attention il vaut mieux tester cette fonction avec des fonctions que vous savez intégrer car un des paramètres est l'intégrale mathématique de f : F. Donc par exemple, f serait $sin(x)$ et F serait $-cos(x)$

```
public static double CIRRErrorMargin(double a, double b,
    double n, Func<double, double> f, Func<double, double> F)
```

Tentez de jouer avec les fonctions, essayez surtout $sin(x)$, $cos(x)$, $exp(x)$, $ln(x)$, et quelques polynômes de votre choix.

6.2.2 Méthode des trapèzes

Écrivez une fonction qui calcule l'intégrale d'une fonction sur une intervalle $[a, b]$ avec la méthode des trapèzes.

```
public static double IntegralTrapezoidalRule(double a, double b,
    Func<double, double> f);
```

Ensuite, codez une fonction qui divise l'intervalle $[a, b]$ en plus petit intervalles de taille n, appliquer la méthode des trapèzes sur chacun, et sommer le tout.

```
public static double CompositeIntegralTrapezoidalRule(double a, double b,
    double n, Func<double, double> f)
```

Enfin il faut calculer la marge d'erreur de la technique. Attention il vaut mieux tester cette fonction avec des fonctions que vous savez intégrer car un des paramètres est l'intégrale mathématique de $f : F$. Donc par exemple, f serait $sin(x)$ et F serait $-cos(x)$.

```
public static double CITRErrorMargin(double a, double b,
    double n, Func<double, double> f, Func<double, double> F)
```

Tentez de jouer avec les fonctions, essayez surtout $sin(x)$, $cos(x)$, $exp(x)$, $ln(x)$, et quelques polynômes de votre choix. Que devient la marge d'erreur avec des droites affines ?

6.2.3 Méthode de Simpson

Écrivez une fonction qui calcule l'intégrale d'une fonction sur une intervalle $[a, b]$ avec la méthode de Simpson.

```
public static double IntegralSimpsonRule(double a, double b,
    Func<double, double> f);
```

Ensuite, codez une fonction qui divise l'intervalle $[a, b]$ en plus petit intervalles de taille n, appliquer la méthode de Simpson sur chacun, et sommer le tout.

```
public static double CompositeIntegralSimpsonRule(double a, double b,
    double n, Func<double, double> f)
```

Enfin il faut calculer la marge d'erreur de la technique. Attention il vaut mieux tester cette fonction avec des fonctions que vous savez intégrer car un des paramètres est l'intégrale mathématique de $f : F$. Donc par exemple, f serait $sin(x)$ et F serait $-cos(x)$.

```
public static double CISRErrorMargin(double a, double b,
    double n, Func<double, double> f, Func<double, double> F)
```

Tentez de jouer avec les fonctions, essayez surtout $sin(x)$, $cos(x)$, $exp(x)$, $ln(x)$, et quelques polynômes de votre choix. Que devient la marge d'erreur avec les polynômes de degrés 2 ?

6.2.4 Methode de Newton-Cotes

Écrivez une fonction qui calcule l'intégrale d'une fonction sur une intervalle $[a, b]$ avec la méthode de Newton-Cotes. Prenez 6 points. Vous devez commencer par trouver vos constantes, gardez en tête que 6 points veut dire degré 5.

```
public static double IntegralNewtonQuadratureAt6(double a, double b,
    Func<double, double> f);
```

Ensuite, appliquez la méthode de Newton-Cotes sur l'intervalle $[a, b]$. Ici n est le degré qu'on choisit, donc il peut être avisé de créer un dictionnaire qui lie le nombre de points avec leur w_i. Vous devez l'implémenter dans le scope de la classe.

```
public static void InitDictionary();
public static double IntegralNewtonQuadrature(double a, double b,
    int n, Func<double, double> f)
```

Ensuite, appliquez la méthode Composite de Newton-Cotes sur l'intervalle $[a, b]$. Ici n est la taille des intervalles, et d est le degré.

```
public static double CompositeIntegralNewtonQuadrature(double a, double b,
    double n, int d, Func<double, double> f)
```

Enfin il faut calculer la marge d'erreur de la technique. Attention il vaut mieux tester cette fonction avec des fonctions que vous savez intégrer car un des paramètres est l'intégrale mathématique de $f : F$. Donc par exemple, f serait $sin(x)$ et F serait $-cos(x)$.

```
public static double CINQErrorMargin(double a, double b,
    double n, int d, Func<double, double> f, Func<double, double> F)
```

Tentez de jouer avec les fonctions, essayez surtout $sin(x)$, $cos(x)$, $exp(x)$, $ln(x)$, et quelques polynômes de votre choix. Que devient la marge d'erreur avec les polynômes de degrés 2 et inférieurs ? Pourquoi est-il rare d'utiliser cette méthode pour des degrés supérieurs a 7 ?

6.2.5 Quadrature de Gauss-Legendre (BONUS)

Cette partie va être plus difficile. Cependant, accrochez-vous et essayez de le faire. Les exercices précédents sont courts pour que vous ayez le temps de faire le bonus. Faites attention, ici n est le nombre de points sur l'interval $[a, b]$.

La fonction principale est :

```
public static double IntegralGaussQuadrature(double a, double b,
    uint n, Func<double, double> f)
```

Pouvez vous montrez en quoi la quadrature de Gauss-Legendre est meilleure que les autres ?

These violent deadlines have violent ends !