Nom, Prenom:_____ **Groupe:**_____

TP4 : Dérivation et intégration numériques [2 heures]

Préparation

a. Calculer de façon analytique la dérivée de la fonction $f(x)=\sqrt[3]{x}$ au point $x=1$.

b. Calculer de façon analytique $\int_{2\pi}^{3\pi} \frac{\sin(x)}{x} dx$

Travail pratique

Exercice 1

De la même façon qu'il est possible de passer des variables en paramètre des fonctions, il est également possible de passer un pointeur de fonction. Par exemple, nous allons écrire une fonction Derivate qui va recevoir en paramètre le pointeur d'une autre fonction.
Lorsque l'on écrira Derivate (sin, ...), cela calculera la dérivée de la fonction sinus.
De même l'écriture Derivate (cos, ...) calculera la dérivée de la fonction cosinus.

a. Créer un projet et tester le programme Ex1.cpp[1].

b. Compléter ce programme avec une nouvelle fonction Derivee_DFR calculant la dérivée numérique (différence finie retrograde).

```
double Derivee_DFR(double(*pt_fct)(double), double x0, double h )
```

1 cf. annexes

Nom, Prenom:_____ **Groupe:**_____

c. Compléter le programme avec une nouvelle fonction Derivee_DFC calculant la dérivée numérique(différence finie centrée).

```
double Derivee_DFC(double(*pt_fct)(double), double x0, double h )
```

d. Modifier le programme de façon à ce qu'il calcule les trois dérivées numériques de la fonction $f(x)=\sqrt[3]{x}$ au point $x=1$.

e. Compléter le tableau suivant:

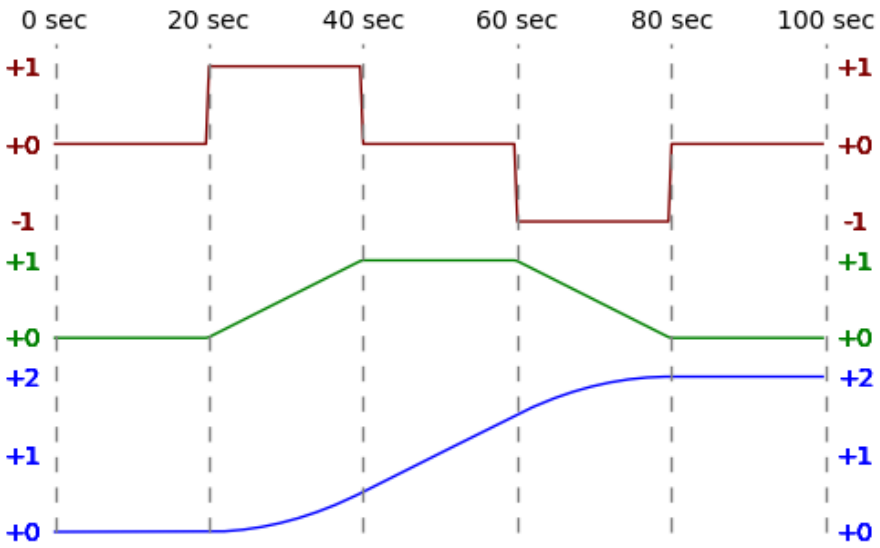
h	DFP	DFR	DFC
0.1			
0.01			
0.001			
10^{-8}			
10^{-9}			
10^{-20}			

f. Expliquer pourquoi la précision diminue à partir de 10^{-9}.

Nom, Prenom:_____ **Groupe:**_____

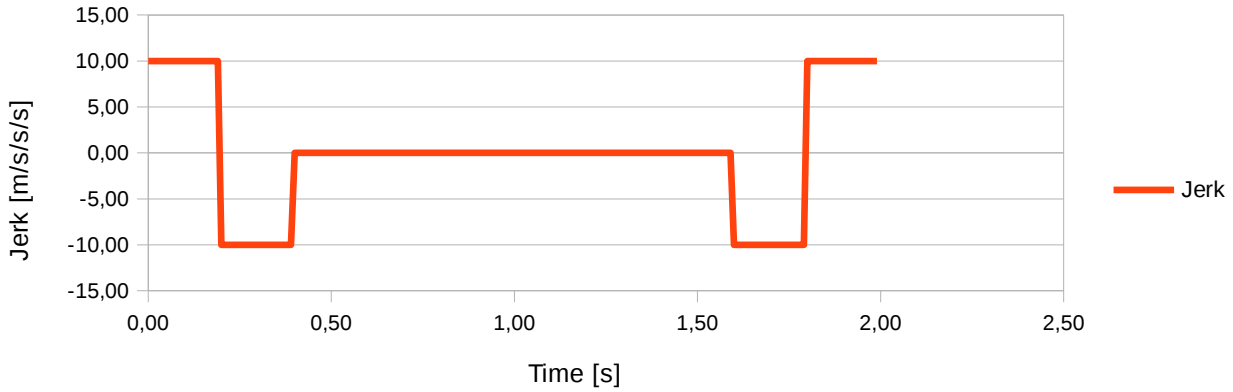
Exercice 2

Dans les systèmes robotisés, pour éviter les à-coups au démarrage et à l'arrêt des moteurs, les concepteurs utilisent des lois de commande progressives appellées rampes d'accelération illustrée ci-dessous:



Relation entre l'accélération (haut), la vitesse (milieu) et la position (en bas)

On rapelle que la vitesse est la dérivée de la position et que l'accélération est la dérivée de la vitesse. Mais il existe d'autres lois de commande, dont une qui utilise le Jerk (dérivée de l'accélération). Nous souhaitons connaître la position finale d'un robot dont le Jerk est donné ci-dessous.



Pour connaître cette position finale, il est nécessaire d'intégrer le Jerk trois fois. Nous allons utiliser une intégration numérique avec une période d'échantillonage de 10ms. La trame du programme est fournie[2] (Ex2.cpp).

2 cf. annexes

Nom, Prenom:_____ **Groupe:**_____

a. Compléter le programme afin de réaliser cette triple intégration (méthode du rectangle à droite) et déterminer la position finale.

Position finale : Unité :

b. Effectuer un collage spécial de vos données dans le fichier tableur OpenOffice (TP4-Ex2-Jerk.ods). Cochez les options [Separated by Tab] et [Detect special numbers]. Choisir [Language English (USA)] pour éviter les problèmes liés aux points (USA) et virgules (France). Imprimer les courbes sur une seule page que vous joindrez à votre compte-rendu.

Exercice 3

a. Ecrire une fonction Integrale_Simpson(double(*pt_fct)(double) , a, h) qui approxime l'intégrale $\int_a^{a+h} f(x)dx$ par la méthode de Simpson.

```
double Integrale_Simpson(double(*pt_fct)(double), double a, double h )
```

b. Utiliser la fonction pour calculer l'integrale suivante $\int_{2\pi}^{3\pi} \frac{\sin(x)}{x} dx$.

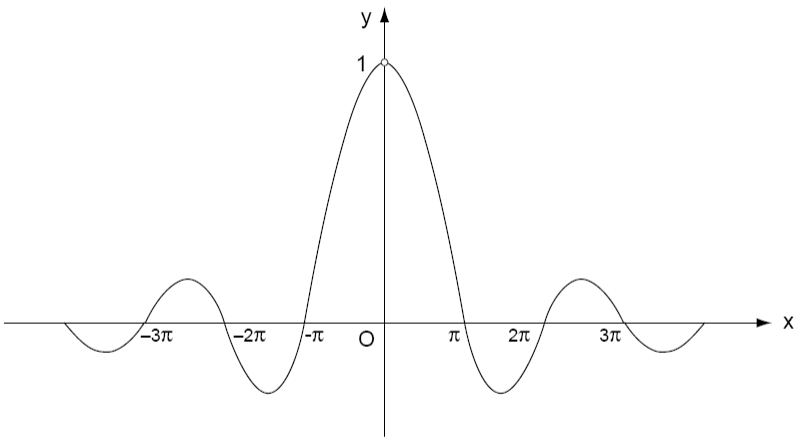


Illustration de la fonction sin(x)/x

$\int_{2\pi}^{3\pi} \frac{\sin(x)}{x} dx =$

c. Calculer de nouveau cette integrale avec la méthode du rectangle à gauche en divisant par 1000 l'intervalle d'intégration.

$$\int_{2\pi}^{3\pi} \frac{\sin(x)}{x}\, dx =$$

d. Quel est la meilleure approximation ? Pourquoi ?

Exercice 4

Vérifier le résultat de l'exercice 2 de façon analytique :

Nom, Prenom:_____ **Groupe:**_____

Annexe

Ex1.cpp

```
#include <stdio.h>
#include <math.h>


// Fonction à dériver (elle sera passée en paramètre)
double Fct(double);

// Calcule la dérivée de la fonction passée en paramère
// en x=x0 avec h=0.1
// Utilise la différence finie progressive
double Derivee_DFP(double(*pt_fct)(double), double x0, double h );


int main(int argc, char *argv[])
{
    // On calcule la dérivée de la fonction Fct en x=1 avec h=0.1
    double D=Derivee_DFP(Fct,1,0.1);
    // On affiche le résultat avec 12 chiffres après la virgule
    printf ("%0.12f",D);
    return 0;
}


// Fonction à dériver (elle sera passée en paramètre)
double Fct(double x)
{
    return x*x + 2*x + 3;       // Retourne x²+2x+3
}

// Calcule la dérivée de la fonction passée en paramère
// en x=x0 avec h=0.1
// Utilise la différence finie progressive
double Derivee_DFP(double(*pt_fct)(double), double x0, double h )
{
    return ( pt_fct(x0+h) - pt_fct(x0) ) / (h);
}
```

Nom, Prenom:_____ **Groupe:**_____

Ex2.cpp

```
#include <stdio.h>

#define     DeltaT  10e-3           // Période d'échantillonnage (10ms)

// Initiliase le Jerk selon la loi de commande désirée
void InitJerk(double *Jerk);

// Cette fonction intègre les données Data
// et place le résultat de l'intégration dans le tableau Integ
void Integrate(double *Data, double *Integ);


// Programme principal
int main(int argc, char *argv[]) {
    // Déclaration et initialisation des tableaux
    double Jerk[200]={0};
    double Acc[200]={0};
    double Speed[200]={0};
    double Pos[200]={0};
    // Initialise le Jerk
    InitJerk(Jerk);
    // Réalise les intégrations successives

    // ::: A COMPLETER :::

    // Affiche le résultat sous forme numérique dans la console
    printf ("t \t Jerk \t Acc \t Speed \t Position\n");
    for (int i=0;i<200;i++)
        printf ("%f \t %f \t %f \t %f \t %f\n" ,
                    i*DeltaT,Jerk[i],Acc[i],Speed[i],Pos[i]);
    return 0;
}


// Cette fonction intègre les données Data
// et place le résultat de l'intégration dans le
// tableau Integ
void Integrate(double *Data, double *Integ)
{
    // ::: A COMPLETER :::
}

// Initiliase le jerk
void InitJerk(double *Jerk) {
    for (int i=0;i<200;i++)
    {
        if (i>=0 && i<20) Jerk[i]=10.0;
        if (i>=20 && i<40) Jerk[i]=-10.0;
        if (i>=40 && i<160) Jerk[i]=0.0;
        if (i>=160 && i<180) Jerk[i]=-10.0;
        if (i>=180 && i<200) Jerk[i]=10.0;
    }
}
```