

***L2 X22I050***

*Algorithmique numérique*

*Projet*

*2020\_2021*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ATALLA Salim

L2 – Informatique\_Groupe\_485k

## Algorithme de Remez :

Le but de ce projet est de déterminer l’approximation du polynôme donnée par la méthode de Remez en plusieurs étapes :

1. On choisit des points de départ dans un intervalle donné.
2. Après avoir choisir ces points on considère qu’entre chaque deux points de cet intervalle on a un sous-intervalle et on détermine le maximum de ces intervalles pour la fonction f donnée.
3. Ensuite, on créer la matrice A de la forme (E1), et le vecteur de droite.
4. On aura des matrices de la forme (A z = b), où A c’est la matrice de (E1), et b le vecteur de droite, ensuite, on va déterminer le vecteur z qui représente les coefficients du polynôme !
5. En fin, pour trouver z on utilise la méthode de pivot de gauss pour résoudre le système linéaire pour obtenir les coefficients qui se déterminent ce polynôme.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Tâches 1 et 8 :**

Rôle : Déterminer l'approximation d'un polynôme par la méthode de Remez.

Fonction *remez* (pointeur vers une fonction d f,

réel d a, réel d b, entier d n) : pointeur vers une tableau de réels

// pré : n >= 3 et b > a

Variables : pointeur vers une tableau de réels : x\_i, z\_i, A, C, B, r

Début :

x\_i 🡨 allocation(tableau de réels[1:n])

z\_i 🡨 allocation(tableau de réels[1:n-1])

A 🡨 allocation(tableau de réels[1:n\*n])

C 🡨 allocation(tableau de réels[1:n])

B 🡨 allocation(tableau de réels[1:n])

x\_i 🡨 Div\_Intervalle (a, b, n)

z\_i 🡨 Z\_i\_Max (f, x\_i, n)

A 🡨 construire\_Matrice\_A (n, n, x\_i)

C 🡨 construire\_Vecteur\_C (f, z\_i, n)

B 🡨 gauss (A, C, n)

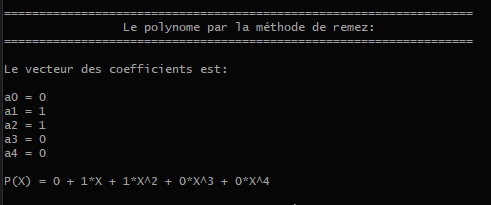
désallocation(x\_i)

désallocation(z\_i)

désallocation(A)

désallocation(C)

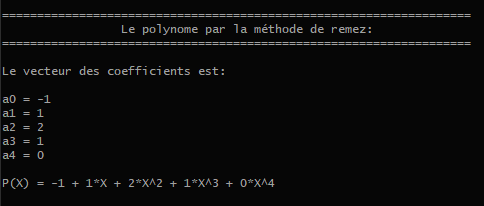
retourner B

 Fin

Jeux de tests :

Pour une fonction donnée : f1(x) = x2 + x pour un intervalle : [2, 6] et pour n = 5

On aura le résultat ci-contre :



Pour une fonction donnée :

f2(x) = x3 + 2x2 + x - 1

On aura le résultat ci-contre :

## On observe qu’on a bien le vecteur des coefficients qui se détermine le polynôme P(x) pour chaque fonction f1 et f2.

## **Tâches 2 :**

Rôle : construir un tableau de n points équidistants sur une intervalle [a, b] donnée.

Fonction *Div\_Intervalle* (réel d a, réel d b, entier d n) : pointeur vers une tableau de réels

// pré : n >= 3 et b > a

Variables : pointeur vers une tableau de réels : x\_i, entier : i

Début :

x\_i 🡨 allocation(tableau de réels[1:n])

Pour i de 1 à n faire :

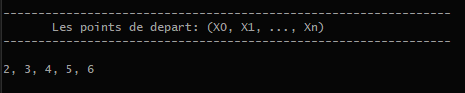
x\_i[i] 🡨 a+((b-a)div(n))\*i

finPour

retourner x\_i

Fin

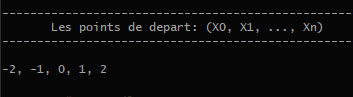
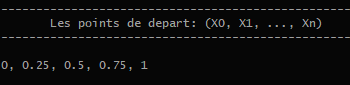
Jeux de tests :

Pour un intervalle donné : [2, 6] et pour n = 5

On aura le résultat ci-contre :

Supposons d’autres intervalles : [-2, 2]  ; [0, 1]

On aura les résultats suivants :

[-2, 2] , et pour n = 5 [0, 1] , et pour n = 5

## On a le premier point c’est bien le point a de l’intervalle, et le dernier point est bien le point b de l’intervalle, et le nombre de point est 5 puisque n = 5, et les différences entre les points sont égaux.

## **Tâches 3 :**

Rôle : Chercher le point max sur l'axe des abscisses.

Fonction *X\_i\_Max* (pointeur vers une fonction d f, réel d a, réel d b) : réel

// pré : b > a

Variables : réel max, x, eps // eps est une constante

// c’est une variable globale.

Début :

x 🡨 a

max 🡨 x

Tantque (x <= b) faire:

Si (f(x) > f(max)) alors

max <- x

finSi

x 🡨 x + eps

finTanque

retourner max

Fin

//-----------------------------------------------------------------------------------------

Rôle : Chercher le point max sur l'axe des ordonnées.

Fonction *Fx\_i\_Max* (pointeur vers une fonction d f, réel d a, réel d b) : réel

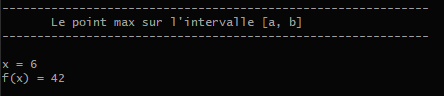
// pré : b > a

Début :

retourner f(X\_i\_Max(f, a, b))

Fin

Jeux de tests :

Pour une fonction donnée :

f(x) = x2 + x et pour un intervalle [2, 6]

On aura le résultat ci-contre :

## C’est une fonction polynomiale où le maximum sur c’est intervalle c’est lorsque

## x=6, et f(x)=42, et donc à partir de ce point-là la fonction sera strictement croissante.

## **Tâches 4 :**

Rôle : construir un tableau de n-1 points d'extremes sur des sous-intervalles [x\_i, x\_i+1] données.

Fonction *Z\_i\_Max* (pointeur vers une fonction d f,

pointeur vers une tableau de réels d x\_i,

entier d n) : pointeur vers une tableau de réels

// pré : n >= 3

Variables : pointeur vers une tableau de réels : z\_i, entier : i

Début :

z\_i 🡨 allocation(tableau de réels[1:n-1] )

Pour i de 1 à n-1 faire :

z\_i[i] 🡨 Fx\_i\_Max(f, x\_i[i], x\_i[i+1])

finPour

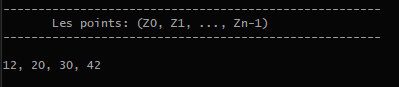
retourner z\_i

Fin

Jeux de tests :

Pour une fonction donnée :

f(x) = x2 + x , et pour un tableau x\_i [n] = {2, 3, 4, 5, 6}, où n = 5

On aura le résultat ci-contre :

Cela signifie que pour cette fonction sur les intervalles : [2, 3] ; [3, 4] ; [4, 5] ; [5, 6]

Les points maximaux sont 12, 20, 30, 42 dans l’ordre.

## **Tâches 5 :**

Rôle : créer une matrice A de n lignes et m colonnes.

Fonction *creer\_Matrice\_Vide* (entier d n, entier d m) : pointeur vers une tableau de réels

// pré : n > 0 et m > 0

Variables : pointeur vers une tableau de réels : M

Début :

M 🡨 allocation(tableau de réels[1:n\*m])

retourner M

Fin

//-----------------------------------------------------------------------------------------

Rôle : créer une matrice A de (E1), de n lignes et m colonnes.

Fonction *construire\_Matrice\_A* (entier d n, entier d m,

pointeur vers une tableau de réels d x\_i) : pointeur vers une tableau de réels

Variables : pointeur vers une tableau de réels : A, entier i, j

Début :

A 🡨 créer\_Matrice\_Vide(n, m)

Pour i de 1 à n faire :

Pour j de 1 à m faire :

Si (j mod(m-1) = 0 et j mod m != 0) alors

Si (i mod 2 = 0) alors

A [m\*i+j] 🡨 1

Sinon

A [m\*i+j] 🡨 -1

finSi

Sinon

A [m\*i+j] <- (x\_i[i]^j)

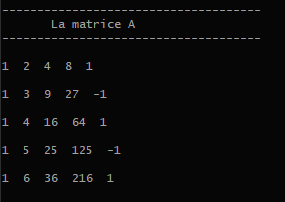
finSi

finPour

finPour

retourner A

Fin



Jeux de tests :

Pour n = m = 5, et pour un tableau x\_i [n] = {2, 3, 4, 5, 6}

On aura le résultat ci-contre qui représente la matrice A de la forme (E1) :

## **Tâches 6 :**

Rôle : créer un vecteur V vide de taille n.

Fonction *creer\_Vecteur\_Vide* (entier d n) : pointeur vers une tableau de réels

// pré : n > 0

Variables : pointeur vers une tableau de réels : V

Début :

V 🡨 allocation(tableau de réels[1:n])

retourner V

Fin

//-----------------------------------------------------------------------------------------

Rôle : créer un vecteur C à partir d'un tableau de points z\_i de taille n.

Fonction *construire\_Vecteur\_C* (pointeur vers une fonction d f,

pointeur vers un tableau de réels d x\_i,

entier d n) : pointeur vers une tableau de réels

// pré : n > 0

Variables : pointeur vers une tableau de réels : C, entier i

Début :

C 🡨 créer\_Vecteur\_Vide(n)

Pour i de 1 à n faire :

C[i] 🡨 f(x\_i[i])

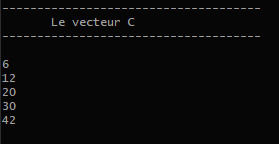
finPour

retourner C

Fin

Jeux de tests :

Pour une fonction donnée : f(x) = x2 + x , et pour un tableau x\_i [n] = {2, 3, 4, 5, 6}, où n = 5

On aura le résultat suivant qui représente le vecteur C de la forme (E1) :

Où C [i] 🡨 f (x\_i [i])

## **Tâches 7 :**

Rôle : résoudre un système linéaire par la méthode de pivot de gauss.

Fonction gauss (pointeur vers un tableau de réels d A, pointeur vers un tableau de réels d V

entier d n) : pointeur vers un tableau de réels

// pré : n > 0

Variables : pointeur vers une tableau de réels : z,

entier i, j, k, réel pivot, p, sol

Début :

// pivotation

Pour i de 1 à n-1 faire :

pivot 🡨 A[n\*i+i]

Pour j de i+1 à n-1 faire :

p 🡨 -(A[n\*j+i]) div pivot

Pour k de 1 à n faire :

A[n\*j+k] 🡨 A[n\*j+k] + p\* A[n\*i+k]

finPour

V[j] 🡨 V[j] + p\*V[i]

finPour

finPour

// Résolution

Pour i de n à 1 (i 🡨 i-1) faire :

sol 🡨 0

pour j de i+1 à n-1 faire :

sol 🡨 sol + A[n\*i+j] \* z[j]

z[i] 🡨 (V[i]-sol) / A[n\*i+i]

finPour

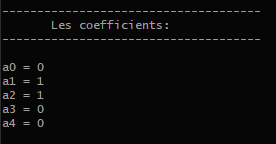
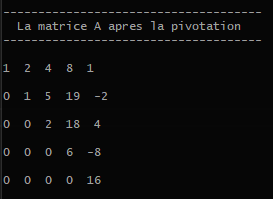
finPour

retourner z

Fin

Jeux de tests :

Pour une matrice A et un vecteur V données (dans les tâches 5 et 6), et pour n = 5

On aura le résultat suivant :

En utilisant la méthode de gauss on a pu résoudre le système linéaire.