TP3: Interpolation polynomiale : Base de Newton.

Introduction *: Le but de ce Tp est d'explorer une variante des méthodes d'interpoler une fonction par un polynôme vues au TP1 et TP2.*

Préambule *: Ne comptez pas sur une amélioration de la précision de l'interpolation. Cependant, ajouter des points d'interpolation nous obligeait à recalculer la matrice de Vandermonde et recalculer le polynôme P solution dans le TP1. Le même inconvénient est présent dans le TP2 : la base de Lagrange doit être recalculée de bout en bout donc P aussi. Or, la mise à jour de P suite à l'acquisition de nouvelles données (ce qui revient à ajouter des points) peut être menée plus rapidement en conservant l'ancien polynôme P déjà trouvé. C'est le but de ce TP3*

C'est parti :

Soit : $a=x_0<x_1<\ldots\ldots\ldots<x_n=b$ *une subdivision d' un intervalle [a,b]*

Une fonction f dont connaît les valeurs $y_i=f(x_i)$ *; pour i allant de 0 à n.*

Objectif *: Nous cherchons un polynôme P de degré au maximum égale à n tel que :*

$y_i=P(x_i)$ *; pour i allant de 0 à n.**

Travaillons dans la base de Newton $(N_0,N_1,N_2,\ldots\ldots N_{n-1})$:

0)- Rappeler la définition de la base de Newton

Cherchons P sous la forme : $P(x)= a_0+ a_1.N_1 +a_2. N_2+\ldots\ldots\ldots a_n.N_n$

1)-Après avoir traduit : $y_i=f(x_i)$ *; pour i allant de 0 à n ; écrire ces égalités sous forme d'un système triangulaire l'exemple (uniquement pour le compte rendu)*

2)- *Rappeler la résolution par différences divisées en utilisant l'exemple (uniquement pour le compte rendu)*

x	*y*		
-1	*-12*		
1	*0*		
3	*-20*		
7	*2724*		

TP3: Interpolation polynomiale : Base de Newton.

3) Ajoutons le point de coordonnées (10, 10053) . Mettre à jour la dernière ligne du tableau.

(uniquement pour le compte rendu)

4) Tracer sur un même graphique une fonction f ainsi que son polynôme d'interpolation associé.(TP)

5) Refaire la question 4) après rajout d'un point. (TP)

Indications techniques :

a. *Entrées : a=0, b=2pi , n=4 ,f=sin ; en sortie X et Y*
b. *Calcul des coefficients a_i par différences divisées. : Ecrire une procédure divis() qui renvoie les a_i sous forme d'array nommé coeff*
c. *Créer une fonction Newton(coeff,x) qui évalue le polynôme P en x réel donné .*
d. *Ecrire une fonction Ajout_un_point(xaj,yaj) : elle aura pour rôle de mettre à jour X,Y et coeff sans refaire tout le tableau des différences divisées. Pour ce TP, on ajoutera le point A (1,sin(1))*
e. *Passer à l'affichage en choisissant le nombre de points (au-dessus de 500 svp) pour tracer f (Y_exact en rouge) et P (Y_estim en bleue) et P après ajout du point A (new_Y en vert)*
f. *Besoin d'imports : math, matplotlib.pyplot, numpy*

Attendus: En binômes, le compte rendu succinct, un code qui marche, sorties sous le format qui vous convient

****

La vie est belle jeune gens.

Joyeuse Toussaint.

****