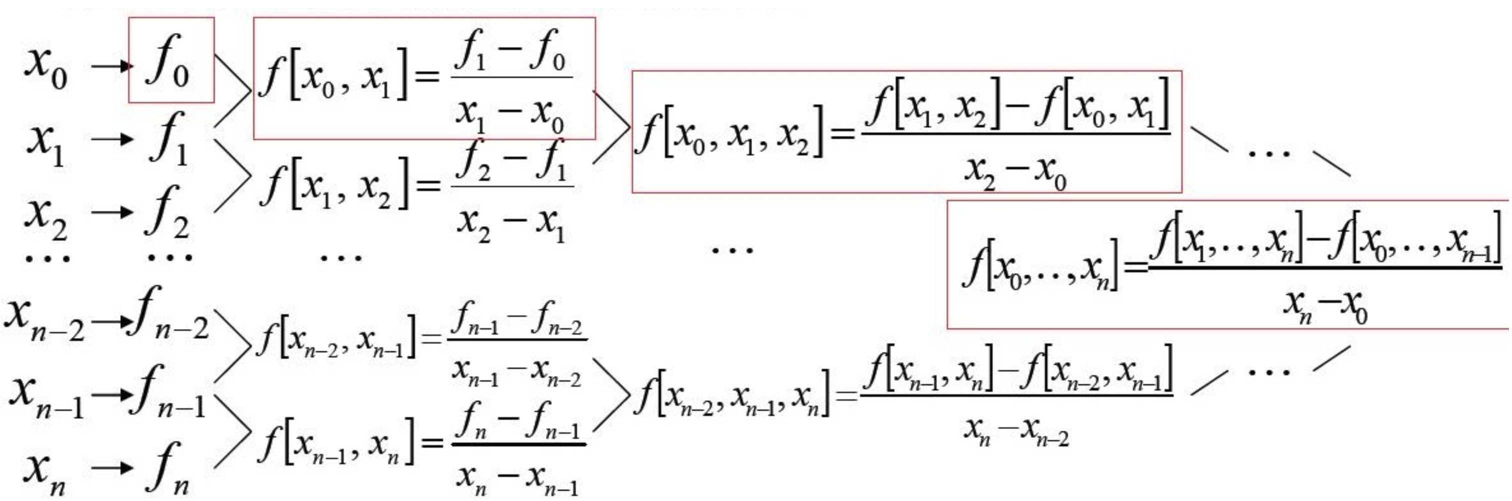
PRÀCTICA 2 D'INTRODUCCIÓ A LA COMPUTACIÓ CIENTÍFICA (ICC): Àlgebra lineal numèrica



Oscar De Caralt Roy
Enginyeria Informàtica UB:
2n de Carrera
Curs 2021-2022

Índex:

<u>Introducció i objectius:</u>

En aquesta pràctica volem implementar en C el mètode de les diferències dividides de Newton per a obtenir un polinomi interpolador i veure'n alguns exemples.

<u>Treball fet:</u>

funs_interp.c:
double horner(double z, double *x, double *c, int n):

Rebrà com a paràmetres: el valor on volem avaluar el polinomi z, el vector x= (x0, x1, . . . , xn) d'abcisses, el vector c= (c0, c1, . . . , cn) de coeficients i el grau del polinomi (n).

Dins d'aquesta funció declarem 2 variables, un int que anomenarem i per a recòrrer el bucle i un double p al que assignarem el valor de c[n].
Aquesta funció tindrà un bucle que començarà en i = n-1 i que en cada iteració anirà disminuint en 1 el valor de i, fins que i sigui menor a 0 tal i com diu aquesta part de l'enunciat: $p = c_n$, $\forall i = n-1, n-2, \dots, 1, 0 \quad p \leftarrow p * (z - x_i) + c_i$.
i en cada iteració aplicarem p = p *(z-x[i]) + c[i] tal com ens diu en aquesta altra part de l'enunciat: $p(z) = \sum_{i=0}^{n} c_i \left(\prod_{j=0}^{i-1} (z - x_j) \right)$.

int difdiv(double *x, double *f, int n):

Rebrà com a paràmetres: les abcisses x= (x0, x1, . . . , xn) , els valors de la funció a interpolar en les abcisses f= (f0, f1, . . . , fn) i el grau del polinomi (n)

declararem 2 variables (i)(k) per a recòrrer els 2 fors que ens demanen i 1 variable tol que contindrà la tolerància amb la qual compararem.

Els fors, tal i com es pot veure, seran de la següent manera: $\forall i = n, n-1, \ldots, k \;;\; \forall k = 1, 2, \ldots, n$.

Un primer for on k començarà amb valor 1 i anirà incrementant en 1 el seu valor en cada iteració fins que arribi a valer n i un segon for on i començarà amb valor n, i anirà decrementant en 1 el seu valor en cada iteració fins que arribi a valer k.

Dins d'aquest segon for comparem si el denominador és menor que la tolerància, si es dóna en alguna iteració retornem un -1 i sortim del programa. en cas contrari, apliquem: $f[i] = (f[i] - f[i-1])/(x[i] - x[i-k])$ tal i com ens diu l'enunciat.

Q: Com es relaciona la forma recursiva anterior amb l'esquema triangular del càlcul de les diferències dividides de Newton (p.11 slides de teoria)?

R: L'esquema triangular del càlcul de les diferències dividides de Newton es basa en aplicar $f[i] = (f[i] - f[i-1])/(x[i] - x[i-k])$ pels diferents intervals i anar conservant els resultats d'aquestes operacions per a futures operacions, que és exactament el mateix que emprem dins del segon bucle.

double fun_log(double z):

Aquesta funció retorna el logaritme neperià d'un valor que passem per paràmetre.

Q: És compatible l'error d'interpolació observat amb la fita de l'error en nodes equiespaiats (p.16 slides teoria)? Calculeu la fita i compareu

R: Si es compleix que xi = xo + i*h per i =0,1,...,n, amb h = (b-a)*h, i |f(n+1)(x)| <= Mn+1 per a tot x pertanyent a l'interval a i b (extrems inclosos) llavors podem calcular la fita amb la següent fòrmula: |f(x)-pn(x)| <= (Mn+1)/(4*(n+1))*(((b-a)/n)^(n+1))

double fun_runge(double z):

Aquesta funció retorna 1/(1+25*z*z) on z és un valor que passem per paràmetre.

funs.interp.h:

Aquí declarem les funcions que emprem en funs.interp.c

Q: Llegiu les pàgines 18 i 19 de les slides de teoria i comenteu els resultats que obteniu en (4.2). És bona idea considerar molts nodes d'interpolació en un interval per tenir una millor aproximació?

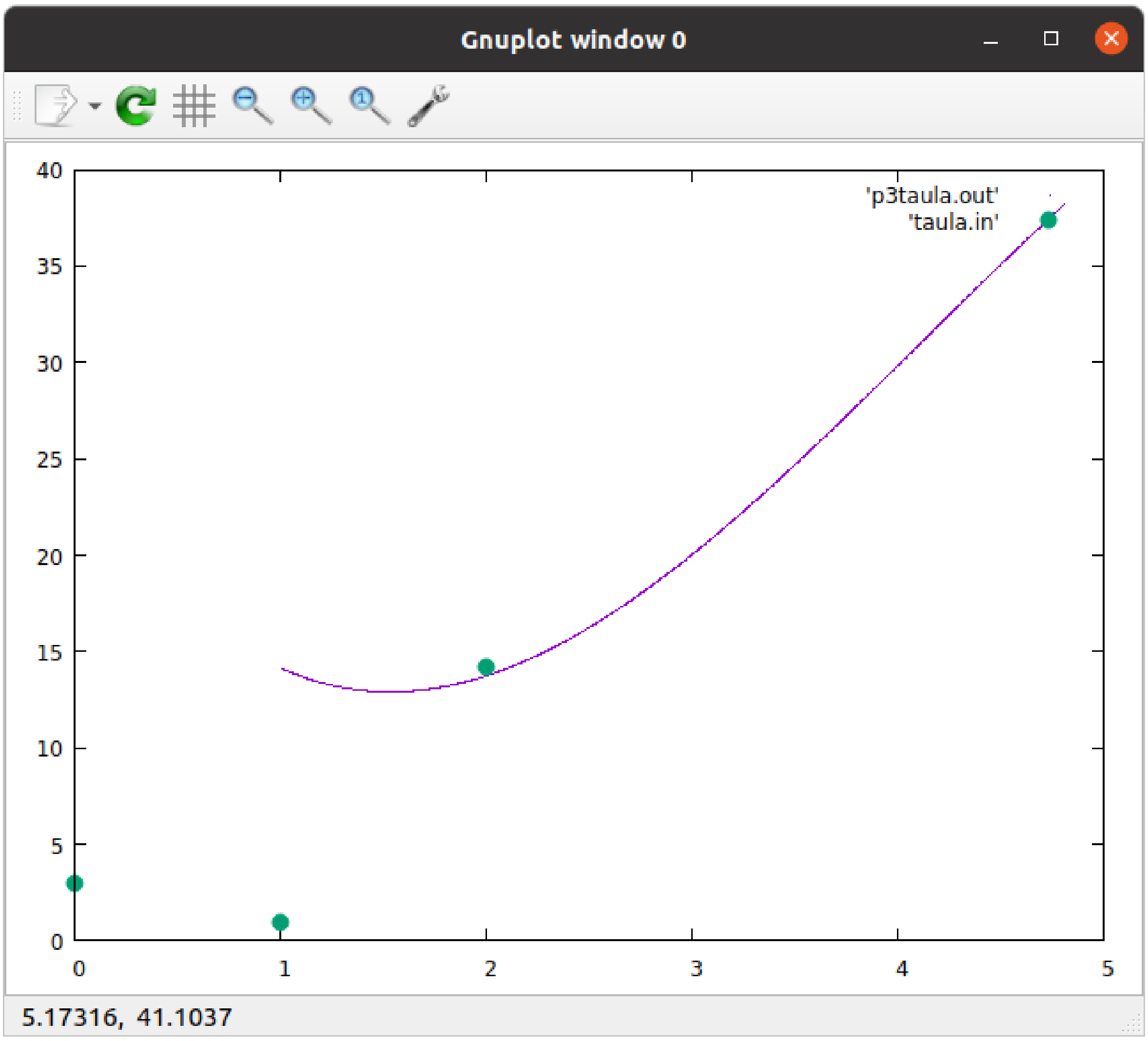
R: Observant els resultats obtinguts, puc afirmar que no és una bona idea utilitzar un grau alt per interpolar, és a dir, no és una bona idea considerar un alt nombre de nodes.

main_classe1.c :

En aquest main el que hem fet ha estat declarar 3 ints (n per a la dimensió del polinomi, i per als bucles i comptador per a comprovar que s'executés correctament), 2 doubles a i b que representaran l'interval en el qual volem treballar, dos vectors de doubles (x que contindrà les variables de control de la taula, i f que contindrà el resultat d'avaluar aquelles x en una funció), un double res que contindrà el resultat de fer difdiv(), un altre double anomenat equidist que representa la distància a la que està un punt equidistant respecte un altre contigu a aquest ((b-a)/999), i les variables u i j que ens ajudaran a escriure per fitxer. A més també tindrem 2 variables tipus FILE, un fitxer que anomenarem fitxer (taula.in) que contindrà el grau del polinomi i els valors dels vectors x i f i un fitxer que anomenarem fitxer_sortida (p3taula.out) on escriurem el punt on avaluarem i el valor que obtindrem en aplicar horner, per a cadascun dels 1000 punts de l'interval.

Primer de tot demanem a l'usuari que entri per teclat l'interval on volem avaluar els 1000 punts. Després declarem els 2 fitxers amb el mode amb el qual tractarem amb ells. Comprovem que els haguem declarat bé i després reservem memòria pels vectors x i f fent ús de: (double*)malloc((n+1)*sizeof(double)). Després demanem que llegeixi del fitxer el grau del polinomi i amb l'ajuda de dos for's fem que llegeixi les 2 files del fitxer (amb el primer for que llegeixi la primera fila i assigni el que vagi trobant al vector x i amb el segon for que llegeixi la segona fila i assigni el que vagi trobant al vector f). Després guardem en una variable res el que retorni la funció difdiv en aplicar-la i comprovem que s'hagi trobat una solució, en cas que no, acabem el programa, en cas contrari recorrem cadascun dels punts equidistants de l'interval i apliquem horner per a cadascun d'ells i printegem al fitxer de sortida.

Finalment, tanquem el fitxer de sortida amb un fclose i alliberem la memòria dinàmica emprada per als vectors x i f.



(p3taula.out representada a través de gnuplot amb les dades de la taula.in)

main_errinterp.c :

En aquest main el que hem fet ha estat declarar 4 ints (n per a la dimensió del polinomi, i per als bucles, comptador per a comprovar que s'executés correctament i eleccio per a poder escollir si mirar a través de fun_log() o fun_runge()), 2 doubles a i b que representaran l'interval en el qual volem treballar, tres vectors de doubles (x que contindrà les variables de control de la taula, f que contindrà el resultat d'avaluar aquelles x en una funció i un vector i f2 que serà una còpia del vector f però que imprimirem al fitxer), un double res que contindrà el resultat de fer difdiv(), un altre double anomenat equidist que representa la distància a la que està un punt equidistant respecte un altre contigu a aquest ((b-a)/999), i les variables u i j que ens ajudaran a escriure per fitxer. A més també tindrem una variable tipus FILE, un fitxer que anomenarem log.out o runge.out (segons el que l'usuari escolleixi) on escriurem el punt on avaluarem(zj), el resultat d'avaluar en aquell punt segons la funció (f(zj)) i el valor que obtindrem en aplicar horner pn(zj), per a cadascun dels 1000 punts de l'interval.

Primer de tot demanem a l'usuari que entri per teclat l'interval on volem avaluar els 1000 punts. Després declarem el fitxer amb el mode amb el qual tractarem. Comprovem que els haguem declarat bé i després reservem memòria pels vectors x, f i f2 fent ús de: (double*)malloc((n+1)*sizeof(double)). Després demanem que introdueixi per paràmetre el grau del polinomi i que esculli a través de quina funció vol avaluar els punts. Segons el que escolleixi amb l'ajuda del for fem que vagi omplint el vectors f i f2 amb uns valors o altres. Després guardem en una variable res el que retorni la funció difdiv en aplicar-la i comprovem que s'hagi trobat una solució, en cas que no, acabem el programa, en cas contrari recorrem cadascun dels punts equidistants de l'interval i apliquem horner per a cadascun d'ells i printegem al fitxer de sortida cadascun dels valors del vector x (zj) amb el seu valor corresponent en avaluar-los en la funció escollida, és a dir el vector f2 (f(zj)) i el corresponent resultat d'aplicar horner.
Finalment, tanquem el fitxer de sortida amb un fclose i alliberem la memòria dinàmica emprada per als vectors x, f i f2.

Problemes:

Em donava error els log.out quan tractava amb un grau parell el polinomi i en la resta em donava a la tercera columna (la de pn(zj)) sempre el mateix valor: -nan . I la part final de gnuplot no sabia com realitzar-la amb 3 variables a llegir (no sabia les comandes)

Conclusions:

He aconseguit treballar d'una millor manera que en la pràctica 1 però no estic del tot satisfet amb el treball realitzat ja que no està 100% bé. Però em veig molt millor que al començament. Aquesta pràctica m'ha ajudat a aprendre bastant millor els algorismes practicats a les classes de problemes, el llenguatge c i com programar amb el terminal de linux. A més, he pogut concloure a través de l'anàlisis i l'observació que, a menor grau del polinomi amb el que interpolem, millors resultats obtenim (menys error).