ICC Pràctica 2(I): Interpolació polinomial

Objectiu: L'objectiu d'aquesta pràctica és implementar en C el mètode de les diferències dividides de Newton per a obtenir un polinomi interpolador i veure'n alguns exemples.

Estructurarem l'enunciat de la pràctica en diferents parts per tal de facilitar-ne la comprensió i la implementació de les funcions. Les diferents parts s'aniran explicant en les successives classes de laboratori d'ordinadors i l'enunciat s'anirà actualitzant progressivament.

Important: No es corregiran les entregues que no segueixin les indicacions indicades al llarg de l'enunciat i/o en les classes de laboratori d'ordinadors.

Organització dels fitxers:

- En començar la pràctica creeu un directori `Cognom1Cognom2Nom_prac2` on anireu creant els diferents arxius en C de la pràctica (amb el nom que s'indica a l'enunciat).
- Les funcions principals estaran en fitxers `main_XXX.c`. La resta de funcions necessàries per compilar i executar els codis relatius a la resolució de sistemes lineals en el fitxer `interpolacio.c`.

Instruccions per entregar: A la corresponent tasca del Campus Virtual s'entregarà un únic fitxer `Cognom1Cognom2Nom_prac2.tgz` que contindrà *exclusivament*:

- Els arxius `.c` que s'indiquen al llarg de l'enunciat amb el codi C de les funcions programades per a la realització de la pràctica.
- El fitxer `interpolacio.h` amb les capçaleres de les funcions.
- Els fitxer de *input* per executar els codis (independentment de si les `main_*` llegeixen per pantalla o de fitxer).
- Un fitxer `.pdf` amb una petita explicació del que s'ha fet, els resultats obtinguts, comprovacions fetes, detalls d'implementacions, respostes dels enunciats, etc.

Assegureu-vos que no hi ha cap altre arxiu en el directori: elimineu objectes, executables, ocults, etc.

Per crear l'arxiu `.tgz` executeu des del directori pare la comanda

```
tar -czvf Cognom1Cognom2Nom_prac2.tgz ./Cognom1Cognom2Nom_prac2/
```

Data (límit) d'entrega: **Dimecres 20 de desembre de 2023 a les 23:55h.**
No s'acceptaran entregues més enllà del dia **22 de desembre de 2023 a les 23:55h**.

---

1 Interpolació: diferències dividides de Newton

1. Implementeu una funció amb capçalera

```
double hornerNewton(double z, double *x, double *c, int n )
```

per avaluar un polinomi donat en una base de Newton en un punt.

   - Rebrà com a paràmetres: el valor on volem avaluar el polinomi `z`, el vector `x`$= (x_0, x_1, \ldots, x_n)$ d'abcisses, i el vector `c`$= (c_0, c_1, \ldots, c_n)$ de coeficients.
   - La funció avaluarà per Horner i retornarà el valor

$$p(z) = \sum_{i=0}^{n} c_i \left(\prod_{j=0}^{i-1} (z - x_j) \right) .$$

2. Implementeu una funció amb capçalera

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

per a calcular les diferències dividides de Newton.

   - Rebrà com a paràmetres: les abcisses `x`$= (x_0, x_1, \ldots, x_n)$ i els valors de la funció a interpolar en les abcisses `f`$= (f_0, f_1, \ldots, f_n)$. A la sortida, el vector `f` contindrà les diferències dividides associades a la taula de valors (x_i, f_i) , $i = 0, 1, \ldots, n$.
   - Si el procés s'ha pogut fer sense cap entrebanc, la funció retorna el valor 0. En canvi, si algun dels denominadors que surten en el procés té valor absolut menor que 10^{-12} llavors el procés no continua i la funció retorna el valor -1.

Les diferències dividides s'implementara usant la forma recursiva

$$f[i] = (f[i] - f[i-1])/(x[i] - x[i-k]) \quad \forall i = n, n-1, \ldots, k \; ; \; \forall k = 1, 2, \ldots, n \; .$$

3. Implementeu una funció principal `main_interpfun.c` que llegeixi el grau del polinomi n i els extrems d'un interval $[a, b]$, i escrigui en un fitxer el resultat d'avaluar el polinomi interpolador en una xarxa de 1000 punts equidistants en $[a, b]$. Les dades d'interpolació correspondran a una funció coneguda $f(x)$, i.e. $f_j = f(x_j),\ x_j \in [a, b]\ \forall j$. El programa permetrà a l'usuari triar entre els següents exemples de funcions i abcisses:

   - Exemples de funcions i intervals:
     - $f(x) = \exp(x)$, $x \in [0, 1]$,
     - $f(x) = \sin(x)$, $x \in [0, \frac{\pi}{2}]$,
     - $f(x) = \frac{1}{1+25x^2}$, $x \in [-1, 1]$.
   - Exemples d'abscisses:
     - equidistants: $x_j = a + jh\ \ \forall j = 0, 1, \ldots, n$, $h = \frac{b-a}{n}$;
     - de Chebyshev: en el cas de l'interval $[-1, 1]$, $x_j = \cos(\frac{2j+1}{n+1} \frac{\pi}{2})\ \ \forall j = 0, 1, \ldots, n$.

Usant `gnuplot`, compareu les gràfiques de la funció inicial i del polinomi d'interpolació trobat per a diversos valors de n. Representeu també l'error d'interpolació.

2 Interpolació d'Hermite (simple)

Sigui $n \geq 0$ i considerem $n+1$ ternes de valors reals (x_i, f_i, g_i) , $i = 0, 1, \ldots, n$ on les abscisses x_i són totes diferents entre si. Volem trobar un polinomi $p \in P_{2n+1}(x)$ que verifiqui les condicions: $p(x_i) = f_i$, $p'(x_i) = g_i$, $\forall i = 0, 1, \ldots, n$.

4. Implementeu una funció `int difdivherm(double *x, double *f, int m)` tal que:
   - A l'entrada, `x` i `f` són vectors coneguts, de $m + 1 = 2n + 2$ components.
   - A la sortida, el vector `f` conté les diferències dividides generalitzades associades a la taula de valors (x_i, f_i, g_i) , $i = 0, 1, \ldots, n$.

     Recordem que les fórmules per a calcular les diferències en el cas del problema d'Hermite simple són com en el cas d'interpolació de Newton excepte en el primer pas, on cal tenir en compte el conveni:

     $$f[x_i, x_i] \equiv g_i \;\; \forall i = 0, 1, \ldots, n \, .$$

   - Si el procés s'ha pogut fer sense cap entrebanc, la funció retorna el valor 0. En canvi, si algun dels denominadors que surten en el procés té valor absolut menor que 10^{-12} llavors el procés no continua i la funció retorna el valor -1.

5. Feu una rutina principal `main_Hermite.c` que:
   - Llegeixi un valor natural n i reservi memòria per a 2 vectors (x i f) de $2n+2$ components reals.
   - Llegeixi $n+1$ ternes de valors ((x_i, f_i, g_i) , $i = 0, 1, \ldots, n$) i ompli els vectors x i f per tal que quedin en la forma

     $$\mathtt{x} = (x_0, x_0, x_1, x_1, \ldots, x_n, x_n) \, , \;\; \mathtt{f} = (f_0, g_0, f_1, g_1, \ldots, f_n, g_n) \, .$$

   - Cridi la funció `difdivherm` per a calcular les diferències dividides generalitzades.
   - Si s'han pogut calcular els coeficients del polinomi interpolador d'Hermite $p(x)$, s'avaluarà aquest polinomi, així com la seva derivada primera, en 501 punts equidistants de l'interval $[x_0, x_n]$, i s'escriuen en un fitxer les ternes $(z_i, p(z_i), p'(z_i))$, $i = 0, 1, \ldots, 500$.

6. Tenim les següents dades corresponents al moviment d'un cotxe:

temps (h)	0	3	5	8	13
distància (km)	0	225	383	623	993
velocitat (km/h)	75	77	80	74	72

   Estimeu la posició i la velocitat del cotxe quan $t = 10$ usant interpolació d'Hermite. Useu `gnuplot` per a pintar les gràfiques dels polinomis $p(x)$ i $p'(x)$.

Nota: Per a avaluar $p(z)$ i $p'(z)$ cal feu una adaptació del mètode de Horner,

```
void horner2(double z, double *x, double *c, double *pdz, int n)
```

Concretament, si

$$p(z) = c_0 + c_1(z - y_0) + c_2(z - y_0)(z - y_1) + \cdots c_m(z - y_0)(z - y_1)\cdots(z - y_{m-1})$$

(on hi pot haver nodes y_i repetits). Els valors de $p(z)$ i $p'(z)$ s'obtenen a partir dels valors inicials

$$p_m = c_m, \qquad p'_m = 0,$$

i calculant recurrentment

$$p_j = c_j + (z - y_j)p_{j+1} \qquad p'_j = p_{j+1} + (z - y_j)p'_{j+1} \qquad \forall j = m-1, m-2, \ldots, 1, 0 \ .$$

Es pot demostrar que $p(z) = p_0$ i $p'(z) = p'_0$. La variable `pdz` és un vector de dues components que, a la sortida de la funció `horner2`, conté els valors p_0 i p'_0.

De forma similar a com s'implementa Horner per avaluar un polinomi, no cal usar dos vectors per guardar els valors p_j i p'_j, sinó que dues variables simples són suficients, si els càlculs es fan en l'ordre adequat.

ICC Pràctica 2(II): Integració numèrica

Objectiu: Volem aproximar integrals definides usant la **regla dels trapezis** i **el mètode de Romberg**.

Usarem una mateixa funció $f(x)$ per comparar els dos mètodes, que implementareu en una funció amb capçalera `double f(double x)`. Com a exemple particular, aproximeu $I = \int_0^1 \exp(x)dx$ i reporteu els resultats per diferents precisions i nombre d'iterats.

Recordem que la **regla dels trapezis** s'expressa com

$$I \equiv \int_a^b f(x)dx \approx T_0(h) \equiv h\left(\frac{1}{2}f(a) + \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}f(b)\right) \ , \tag{1}$$

on $h = \frac{b-a}{n}$, $x_i = a + ih$, $\forall i = 1, 2, \ldots, n-1$.

El mètode de Romberg és una combinació del **mètode d'integració per trapezis** i del **mètode d'extrapolació repetida de Richardson**. La fórmula general és

$$T_j(h) = \frac{4^j T_{j-1}(\frac{h}{2}) - T_{j-1}(h)}{4^j - 1} \ \forall j \geq 1 \ . \tag{2}$$

a) Feu una funció principal `main_trapezis.c` que llegeixi els extrems de l'interval $[a, b]$, la precisió `prec` i un nombre positiu `maxit`. Invocarà la funció `trap` diverses vegades, cada vegada duplicant el valor de subdivisions de l'interval $[a, b]$. S'aturarà el procés quan dues aproximacions consecutives difereixen menys que la precisió desitjada `prec`, o bé quan la quantitat d'invocacions de la funció `trap` sigui superior a `maxit`.

b) Feu una funció

```
double trap(int n, double a, double b)
```

Aquesta funció NO implementarà directament la fórmula dels trapezis (1), sinó que només calcularà la part corresponent a les noves avaluacions de $f(x)$. O sigui, no avaluarà $f(x)$ en les x on ja s'ha avaluat anteriorment.

c) Feu una funció principal `main_romberg.c` que llegeixi els extrems de l'interval $[a, b]$, la precisió `prec` i el nombre màxim de passos d'extrapolació a realitzar `numextrap`. Calcularà l'aproximació de la integral usant el mètode de Romberg fent servir que les avaluacions de (2) es poden estructurar en un esquema triangular

$$\begin{array}{lll} T_0(h) & & \\ T_0(h/2) & T_1(h) & \\ T_0(h/2^2) & T_1(h/2) & T_2(h) \\ \vdots & \vdots & \vdots \end{array}$$

Així s'anirà calculant avançant fila a fila i d'esquerra a dreta. Per a calcular la primera columna de cada fila s'usa la fórmula dels trapezis (amb el mateix estalvi d'avaluacions de $f(x)$ que s'ha implementat en `trap`). Per a la resta de columnes, s'usen els últims dos valors ja coneguts de la columna anterior.

S'atura el procés quan, o bé els dos últims elements d'una fila difereixen menys que una precisió demanada `prec`, o bé s'ha arribat al màxim de files (és a dir, de passos de extrapolació) que volem calcular `numextrap`.