Prof. S. Sánchez-Pedreño

Entrega de prácticas 1
2 de noviembre de 2023

Para poder corregir de forma adecuada las entregas de prácticas, seguiremos el siguiente proceso.

- Se creará una «Tarea» en el Aula Virtual correspondiente a esta entrega.
- Todos los programas de Octave que crees debes colocarlos en la carpeta `Biblioteca`, si son de propósito general, o bien en una carpeta denominada `Entrega1` si corresponden a programas que utilizan las funciones incluidas en `Biblioteca` y que resuelven distintos apartados concretos de la práctica.
- Una vez hayas terminado tu trabajo, debes crear un archivo `zip` de tu carpeta de trabajo con Octave y de forma que contenga al menos las dos carpetas anteriores. Si quieres incluir algún archivo de texto o pdf con comentarios u otras cuestiones puedes hacerlo en el mismo `zip`. También puedes incluir los comentarios en los scripts de Octave o en la casilla destinada para ellos en el envío de la tarea del Aula Virtual.
- Finalmente debes subir este archivo `zip` a la Tarea del Aula Virtual.

Fecha de inicio: 11 de noviembre de 2022
Fecha de entrega: 10 de diciembre de 2022
Se valorarán muy positivamente los comentarios y observaciones acerca de los ejercicios realizados y de los resultados numéricos y gráficos obtenidos, naturalmente siempre que sean adecuados y correctos. También se valorará muy positivamente la eficiencia del código, en particular el uso del cálculo en forma vectorial para el que Octave está muy bien adaptado.

Introducción

El objetivo general de esta práctica es la experimentación con el cálculo del polinomio interpolador en sus distintas formas, y en particular, la estabilidad de la evaluación de dicho polinomio. En el último ejercicio haremos una experiencia acerca de cómo pueden afectar los errores de redondeo de los datos a los resultados finales (seguimos, por tanto, en torno al problema de la estabilidad).

A estas alturas tenemos a nuestra disposición (en teoría) varias formas de calcular y evaluar un polinomio interpolador. Esta formas son (en términos generales):

- Calcular los coeficientes del polinomio en su forma de Newton y evaluarlo mediante el método de Horner.
- Calcular los coeficientes de la forma de Lagrange (que es un polinomio escrito en la forma habitual $a_n x^n + \cdots + a_1 x + a_0$) y evaluarlo mediante el método de Horner.
- Evaluar directamente el polinomio mediante la forma de Lagrange.

La primera de las formas anteriores ya es conocida y ya la tenemos programada; ha sido la forma habitual de cálculo, hasta ahora. El primer ejercicio (ya iniciado en la práctica 4) tiene por objetivo implementar en Octave el segundo procedimiento. El tercer procedimiento lo implementaremos en el segundo ejercicio, a través de la conocida como «fórmula baricéntrica» para el polinomio de Lagrange.

Ejercicio 1

Este ejercicio es prácticamente idéntico al ejercicio 3 de la práctica 4: sólo hay pequeños cambios en el enunciado, aparte del cambio de numeración del mismo y de sus scripts: sólo tienes que copiar a la carpeta `/Entrega1`, o a `/Biblioteca`, los scripts o funciones que ya tienes realizados, cambiándoles la numeración y completando los pequeños cambios.

1. Escribe una función denominada `factLagrange.m`, que guardarás en `/Biblioteca`, que realice la tarea siguiente: a partir de su único argumento `x`, que es el vector de nodos de interpolación ($x = \{x_0, \dots, x_n\}$), devuelve una matriz cuya fila k-ésima contiene los coeficientes del factor de Lagrange ($k = 0, \dots, n$):

   $$L_k(x) = \prod_{j=0,\, j\neq k}^{n} \frac{x - x_j}{x_k - x_j}$$

   Para facilitar la tarea usaremos la función de Octave:

   `conv(p,q)`

   que devuelve el polinomio producto de los polinomios `p` y `q` (¡recuerda cómo se representan los polinomios en Octave!).

2. Crea en `/Biblioteca` la función `interpolLagrange.m` que espera como argumentos la matriz de coeficientes de los factores de Lagrange y el vector de ordenadas a interpolar, y devuelve los coeficientes del polinomio interpolador en la forma de Lagrange:

   $$p(x) = \sum_{k=0}^{n} y_k L_k(x)$$

3. Para verificar que la implementación es correcta: escribe en el script `Ejercicio1_1.m` el cálculo de los coeficientes del polinomio interpolador en los conjuntos de puntos

   $$\{(0,0),(1,2),(2,4),(5,10)\} \quad \text{y} \quad \{(0,0),(1,1),(2,8),(3,27)\}.$$

   Debes saber cuáles son los polinomios resultantes (sin hacer ningún cálculo): si el resultado no coincide exactamente con los polinomios esperados, ¿hay alguna explicación o hay errores en el script?

4. Una vez creadas las funciones anteriores disponemos ya del polinomio interpolador en la forma de Lagrange: para evaluarlo en un vector de abscisas, `z`, utilizaremos la siguiente función de Octave

   `polyval(p,z)`

   que devuelve el vector de las evaluaciones del polinomio `p` en `z`: este cálculo está implementado en Octave mediante el método de Horner (así que no vale la pena que lo implementemos nosotros...)

5. Para comparar los resultados de ambas formas de cálculo utilizaremos la interpolación de la función $\operatorname{sen}(x)$ en el intervalo $[0, 2\pi]$, mediante nodos equidistribuidos: $x_0 = a$, $x_k = a + k\frac{2\pi}{n}$. Para esta función sabemos que los polinomios interpoladores $p_n(x)$ convergen uniformemente a $f(x)$, por lo que si aumentamos considerablemente el número de puntos a interpolar deberíamos aproximar más y más a la función...

   Crea el script `Ejercicio1_2.m` que para un valor de n realice las tareas siguientes:

   - en una primera ventana gráfica incluya las gráficas de f y del polinomio interpolador en la forma de Newton;
   - en una segunda ventana gráfica incluya las gráficas de f y del polinomio interpolador en la forma de Lagrange;
   - en otra ventana incluya la gráfica del valor absoluto de la diferencia de los dos polinomios interpoladores;

- nos dé como salida en la ventana de comandos el máximo del valor absoluto de: la diferencia entre la función f y cada uno de los polinomios, y de la diferencia entre los dos polinomios (calculados sobre una malla de puntos en el intervalo $[0, 2\pi]$ de al menos 1000 puntos).

Ejecuta el script anterior para distintos valores de n, por ejemplo: 5, 10, 15 y otros mayores, por ejemplo 20, 30, 40. ¿Qué observas? Observa que las diferencias grandes aparecen cerca de los extremos del intervalo, ¿tiene esto que ver con el fenómeno de Runge? Comenta lo que consideres adecuado sobre la estabilidad de los cálculos en una u otra forma. Intenta determinar experimentalmente a partir de qué valores de n se producen errores de computación para la evaluación del polinomio de Lagrange.

Ejercicio 2

Como has podido comprobar, el procedimiento empleado en el ejercicio anterior para la evaluación del polinomio de Lagrange es inestable para valores de n no demasiado grandes. En este ejercicio vamos a implementar una forma de evaluación mucho más estable: se trata de utilizar la llamada *forma baricéntrica* para el polinomio de Lagrange.

He aquí un breve cálculo que permite obtener dicha forma. El polinomio de Lagrange es $p(x) = \sum_{i=0}^{n} y_i L_i(x)$, siendo L_i, para $i = 0, \dots, n$, los factores de Lagrange. Escribamos

$$\omega_n(x) = \prod_{j=0}^{n}(x - x_j) \qquad \text{y} \qquad \lambda_i^n = \prod_{j=0,\, j\neq i}^{n} \frac{1}{x_i - x_j}.$$

Observa que $L_i(x) = \frac{\lambda_i^n}{x - x_i}\omega_n(x)$ para x distinto de todos los nodos x_i. Entonces podemos escribir

$$p(x) = \sum_{i=0}^{n} y_i L_i(x) = \sum_{i=0}^{n} y_i \frac{\lambda_i^n}{x - x_i}\omega_n(x)$$

que es válido para todo x distinto de los nodos x_i. Ahora, utilizando que $\sum_{i=0}^{n} L_i(x) = 1$ para todo $x \in \mathbb{R}$, tenemos

$$p(x) = \frac{\displaystyle\sum_{i=0}^{n} y_i \frac{\lambda_i^n}{x - x_i}\omega_n(x)}{\displaystyle\sum_{i=0}^{n} \frac{\lambda_i^n}{x - x_i}\omega_n(x)} = \frac{\displaystyle\sum_{i=0}^{n} y_i \frac{\lambda_i^n}{x - x_i}}{\displaystyle\sum_{i=0}^{n} \frac{\lambda_i^n}{x - x_i}} \tag{1}$$

que es la fórmula baricéntrica buscada (se denomina así de forma un poco abusiva, porque es una combinación lineal de los valores y_i con coeficientes que suman 1, aunque no es una combinación convexa porque dichos coeficientes pueden ser negativos).

El objetivo del ejercicio es implementar esta forma de evaluación y compararla con las otras dos formas de las que ya disponemos.

Para implementar esta forma de cálculo vamos a escribir un algoritmo eficiente (merece una reflexión para convencerse de que lleva a cabo el proceso requerido). Observemos que para $k \geq 1$ se tiene

$$\lambda_i^k = \frac{\lambda_i^{k-1}}{x_i - x_k}, \quad i = 0, 1, \dots, k-1$$

entonces el algoritmo consiste en la construcción de la siguiente matriz triangular (λ_i^k), $i = 0, \dots, k-1$, $k = 0, 1, \dots, n$ construida de la forma siguiente:

- $\lambda_0^0 = 1$.
- Para $k = 1, 2, \dots, n$,

$$\lambda_i^k = \frac{\lambda_i^{k-1}}{x_i - x_k}, \quad i = 0, 1, \dots, k-1$$
$$\lambda_k^k = \frac{1}{\prod_{j=0}^{k-1}(x_k - x_j)}$$

Departamento de Matemáticas. Campus de Espinardo. 30100 Espinardo Murcia
e-mail: pedreno@um.es

Observa que los λ_i^k para $i = 0, \ldots, k-1$ se calculan de forma recurrente, mientras que los λ_k^k se calculan directamente.

Una vez calculados los λ_i^n (los elementos de la última columna de la matriz) ya se puede proceder a la evaluación de la expresión de la derecha en la fórmula (1).

Observa que: el número de operaciones total es similar al necesario para evaluar el polinomio interpolador en la forma de Newton y, además, este procedimiento permite añadir un nodo más al proceso de interpolación sin necesidad de recalcular todo, basta añadir una columna a la matriz λ_i^k (concretamente calcular $\lambda_0^{n+1}, \ldots, \lambda_{n+1}^{n+1}$) y recalcular parte de la expresión en (1).

1. Crea en la carpeta `/Biblioteca` una función denominada `polyLagrange_eval(x,y,s)` que reciba como variables: el vector de nodos (`x`), el vector de ordenadas (`y`), y el vector de puntos donde se evaluará el polinomio (`s`). La función debe devolver el vector de evaluaciones del polinomio en las componentes de `s` utilizando el algoritmo anterior. **Importante: lee la Nota al final del ejercicio**.

2. Realiza las mismas tareas que en el apartado 5 del ejercicio anterior (la misma función y los mismos nodos).

Comenta los resultados: ¿hasta qué valores de n parece ahora estable la evaluación? ¿Los errores de cálculo empiezan para valores de n menores en el cálculo de la forma baricéntrica de Lagrange que en la forma de Newton? (haz ensayos con valores grandes de n, del orden 50, 60...)

Nota

La forma baricéntrica no tiene sentido al evaluar el polinomio en un punto x igual a alguno de los nodos, es decir, si $x = x_i$ para algún $i \in \{0, \ldots, n\}$, pues en la fórmula se debe dividir por $x - x_i$. Si no se toma ninguna precaución es posible que los gráficos tengan algún hueco, pues si alguna componente de `s` coincide con alguna componente de `x` el valor calculado en esa componente será `NaN` (o quizás `Inf`). Esto no es un problema para la gráfica, pues en esos valores no se dibuja (lo que puede dar huecos muy pequeños si `s` es una malla suficientemente fina). Quizás sí podría afectar al cálculo de los máximos de las diferencias (si el valor en algún punto es `Inf`).

Puesto que en los nodos sabemos el valor del polinomio interpolador, se le podría asignar este valor en la componente adecuada de `s`, en caso de coincidir... el problema es que entonces deberíamos hacer una gran cantidad de condicionales `if...` (para comprobar si alguna componente de `s` coincide con algún nodo), a no ser que abordemos este problema de forma vectorial, utilizando la estrategia de la indexación booleana de Octave (recuerda los comentarios al respecto en la práctica 2). Pero esto entraña una cierta dificutad. **Es totalmente voluntario implementar el código necesario para evitar esta indefinición** en el caso citado de coincidencia en las abscisas. En cualquier caso, una forma de evitar las coincidencias (salvo en los extremos del intervalo, 0 y 2π) es hacer una malla de un número de puntos que sea coprimo con el número de nodos.

Ejercicio 3

Una forma de estudiar la inestabilidad del proceso de interpolación con polinomios de grado elevado es la siguiente. Supongamos que la función exacta que deseamos interpolar es $f(x)$ y sea $\tilde{f}(x)$ la función calculada efectivamente, que está afectada por errores de redondeo. Supondremos que $\|f - \tilde{f}\|_\infty < \varepsilon_0$ (por ejemplo, ε_0 puede ser un múltiplo fijo del épsilon de la máquina...)

Si $p(x) = \sum_{i=0}^n f(x_i)L_i(x)$ es el polinomio interpolador de Lagrange, $\tilde{p}(x) = \sum_{i=0}^n \tilde{f}(x_i)L_i(x)$ será el polinomio real que calcularemos, y podemos escribir

$$|f(x) - \tilde{p}(x)| \leq |f(x) - p(x)| + |p(x) - \tilde{p}(x)|.$$

En la expresión anterior el primer sumando es el error de aproximación por interpolación, que ya conocemos teóricamente, mientras que el segundo es el error debido al redondeo y que ahora pretendemos estimar.

Es fácil obtener

$$|p(x) - \tilde{p}(x)| = \left|\sum_{i=0}^{n}(f(x_i) - \tilde{f}(x_i))L_i(x)\right| \leq \varepsilon_0 \sum_{i=0}^{n}|L_i(x)| \leq \varepsilon_0 \sum_{i=0}^{n}\|L_i\|_\infty$$

Por tanto la función $\Lambda_n(x) = \sum_{i=0}^{n}|L_i(x)|$ y el número $M_n = \sum_{i=0}^{n}\|L_i\|_\infty$ (que suelen denominarse *función* y *número de Lebesgue*) nos proporcionan una vía para estudiar la inestabilidad del proceso.

En este ejercicio vamos a evaluar las funciones Λ_n y el número M_n, para distintos valores de n. Así mismo estudiaremos experimentalmente cómo crecen estos valores con n y trataremos de comprobar a través de un modelo sencillo si las estimaciones sobre los errores de redondeo anteriores son ratificadas por la práctica.

Observación. Para calcular correctamente $\Lambda_n(x)$ o M_n deberíamos usar el procedimiento baricéntrico del ejercicio anterior, pero eso significaría reprogramarlo, puesto que necesitamos los valores de $\sum_{i=0}^{n}|L_i(x)|$, no los de $|\sum_{i=0}^{n} L_i(x)|$. Puesto que no vamos a elegir valores grandes de n utilizaremos las funciones del ejercicio 1 para estos cálculos: después de ese primer ejercicio estamos más o menos tranquilos, puesto que esperamos que el cálculo de los factores de Lagrange no genere grandes problemas de inestabilidad para los valores de n que utilizaremos en este ejercicio.

1. Crea en la carpeta `/Biblioteca` una función `lebesgue(nodos,z)`, donde la variable `nodos` es el vector de nodos y `z` es un vector de puntos donde evaluaremos la función de Lebesgue. Esta función debe devolver el vector de evaluaciones, en `z`, de la función $\Lambda_n(x) = \sum_{i=0}^{n}|L_i(x)|$ correspondiente a los `nodos` y el número $M_n = \sum_{i=0}^{n}\|L_i\|_\infty$. Para realizar estos cálculos utilizaremos:

    - la función ya programada `factLagrange` para la evaluación de Λ_n, y
    - los máximos $\|L_i\|_\infty$ se calcularán experimentalmente, tomando el máximo de $|L_i|$ evaluado en el vector de puntos `z`.

    El vector de puntos a evaluar `z` debe tener al menos 500 componentes (puesto que vamos a estimar el máximo de los factores de Lagrange sobre dicho vector).

2. En el script `Ejercicio3_1.m` debes dibujar las gráficas de Λ_n para los nodos consistentes en $n+1$ puntos equiespaciados en el intervalo $[-1, 1]$, y n debe tomar los valores $5, 10, 15, 20$. Cada gráfica debe aparecer en una ventana diferente.

3. En el script `Ejercicio3_2.m` debes dibujar gráficas de M_n respecto a n: en una gráfica para n desde 1 hasta 9, en la segunda desde 1 hasta 19 (con los nodos del apartado anterior). Comenta lo que consideres oportuno sobre el crecimiento de M_n respecto a n.

4. En el script `Ejercicio3_3.m` vamos a simular mediante un modelo la estimación de los errores de redondeo. Para ello perturbaremos los datos para la función seno mediante valores pequeños de forma aleatoria. Consideramos $N + 1$ puntos equiespaciados del intervalo $[-1, 1]$, es decir, $x_k = -1 + \frac{2}{N}k$, $k = 0, \ldots, N$. Debes realizar las tareas siguientes para $N = 8$ y $N = 14$.

    a) Generamos un conjunto de N valores aleatorios u_k, tales que $|u_k| \leq 0.01$. Para ello utilizamos la función de Octave `rand(p,q)` que genera una matriz $p \times q$ de números aleatorios pertenecientes al intervalo $(0, 1)$. En nuestro caso generamos `w=rand(1,N+1)` y para que los números aleatorios verifiquen $|u_k| \leq 0.01$ les aplicamos la transformación $u = -0.01 + 0.02 * w$.

    b) Construimos el polinomio interpolador p_N en los puntos $\{(x_i, \mathrm{sen}(x_i))\}_{i=0}^{N}$ y el polinomio interpolador $\tilde{p}_N$ en los puntos $\{(x_i, z_i)\}_{i=0}^{N}$, siendo $z_j = \mathrm{sen}(x_j) + u_j$; así pues estamos perturbando ligeramente los valores de la función seno en los nodos.

    c) Calcula el máximo de la diferencia $|p_N(x) - \tilde{p}_N(x)|$ en una malla de puntos en el intervalo $[-1, 1]$ suficientemente fina. Compara esta diferencia máxima con el valor $0.01 \cdot M_N$ que es la máxima desviación esperada. Comenta lo que consideres oportuno sobre el ejercicio.

Departamento de Matemáticas. Campus de Espinardo. 30100 Espinardo Murcia
e-mail: pedreno@um.es