UNIVERSIDAD DE MURCIA

Departamento de Matemáticas
Cálculo Numérico en una Variable
Curso 2021/22

Prof. S. Sánchez-Pedreño

Entrega de prácticas 1

21 de noviembre de 2021

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`, pero fuera de la carpeta de trabajo de Octave.
- Finalmente debes subir este archivo `zip` a la Tarea del Aula Virtual.

Fecha de inicio: 9 de noviembre de 2021
Fecha de entrega: 5 de diciembre de 2021
Se valorarán positivamente cualquier tipo de comentarios y observaciones acerca de los ejercicios realizados y de los resultados numéricos y gráficos obtenidos.

He incluido algunos comentarios adicionales y correcciones que algunos estudiantes me habéis comentado (comentarios en rojo). Cualquiera que ya haya programado todo o parte de esta práctica, no es necesario que modifique nada...

Notas sobre Octave

A lo largo del ejercicio necesitarás, o podrá utilizar, diversas funciones de Octave. A continuación tienes una rápida revisión de algunas de ellas.

- Recuerda que para Octave un vector $a = [a_0, \ldots, a_{n+1}]$ representa al polinomio $a_0x^n + \cdots + a_nx + a_{n+1}$.
- `polyval(a,t)`, evalúa el polynomio `a` en el vector de abscisas `t` (que puede ser también un número, no un vector);
- `polyder(a)`, devuelve un vector con los coeficientes del polinomio derivado de `a`.
- Necesitarás, en el ejercicio 3, la función de Octave `conv(a,b)` que devuelve los coeficientes del polinomio producto de los polinomios (vectores) `a` y `b`. La suma de polinomios y el producto por un escalar se realizan como si fueran vectores, naturalmente.
- Otras utilidades de Octave que puede resultar de interés recordar:
  - Puedes inicializar una matriz $m \times n$, con todos sus elementos nulos, con `A=zeros(m,n)`; también con todos sus elementos iguales a 1, con: `A=ones(m,n)`.
  - Si `A` es una matriz, con `A(k,:)` se identifica el vector cuyas componentes son las de la fila `k`-ésima de `A`; así pues, puedes hacer asignaciones del tipo `v=A(k,:)` o `v=A(:,j)`. También puedes seleccionar una parte de las componentes, con `v=A(k,3:2:8)`, que denota el vector formado por las componentes de la fila `k` de `A` situados en las columnas `3, 5` y `7` (como la siguiente columna es la `9`, mayor que `8`, se detiene en `7`).

Ejercicio 1

El objetivo de este ejercicio, y del siguiente, es poner en práctica una de las formas de interpolación polinomial que no hemos desarrollado en el tema: la interpolación a trozos. Supongamos que deseamos interpolar una función $f(x)$ en un intervalo $[a, b]$. Sabemos que si el número de nodos de interpolación es grande aparecen problemas (fenómeno de Runge), mientras que si el número es pequeño la aproximación no será buena. Se trata entonces de seguir una estrategia similar a la de los splines: dividimos el intervalo en un cierto número de subintervalos y procedemos a interpolar en cada uno de estos subintervalos con un polinomio de grado pequeño. El resultado será una función interpoladora que es polinomial a trozos, con un grado en cada subintervalo que depende del número de nodos incluidos en dicho subintervalo.

En este primer ejercicio nos centramos en la interpolación a trozos de Lagrange, mientras que en el segundo ejercicio haremos interpolación a trozos de Hermite.

Centrándonos en la interpolación de Lagrange a trozos. Llamaremos *nudos* a los puntos de una partición del intervalo $[a, b]$: $a = z_0 < z_1 < z_{m-1} < z_m = b$. Estos son los puntos que separan el intervalo $[a, b]$ en los m trozos distintos donde se interpola: $[z_i, z_{i+1}]$, $i = 0, \ldots, m-1$.

Ahora debemos definir los nodos. Aquí tenemos dos posibilidades: que todos los nudos sean nodos o que no. En cada caso, los datos del problema serán:

- en el primer caso debemos proporcionar los nudos, $\{z_0, \ldots, z_m\}$ y los nodos, que serán una colección de puntos $x_{i,k}$, con $k = 0, \ldots, k_i$, siendo $x_{i,k} \in [z_i, z_{i+1}]$;
- en el segundo caso basta con dar una lista de nodos, asumiendo que en esa lista se encuentran los nudos z_i.

Cada una de estas opciones tiene ventajas e inconvenientes. Por ejemplo, la segunda opción no permite interpolar en cada subintervalo con nodos de Chebyshev, pero parece más sencilla de programar, en particular, si decidimos que el número de nodos en cada subintervalo será siempre el mismo (incluyendo sus extremos).

El objetivo del ejercicio es crear una función que calcule estas familias de polinomios interpoladores a trozos en una de las dos posibilidades anteriores, **a tu elección**, y después chequear las aproximaciones que proporcionan.

Para fijar las ideas y simplificar el ejercicio tomaremos el número de nodos a interpolar en todos los subintervalos igual a $d + 1$, así todos los polinomios serán de grado d.

1. Crea una función en la carpeta `Biblioteca` que devuelva una matriz cuyas filas son los coeficientes de los distintos polinomios, en la forma de Newton, de grado d en los subintervalos $[z_i, z_{i+1}]$.

   Puedes optar por cualquiera de las dos opciones comentadas antes. Ten en cuenta que:

   - si optas por la primera opción tendrás que enviar un vector de nudos y una matriz de nodos, la matriz debe tener dimensiones $m \times (d + 1)$;

     Como me habéis hecho ver, no es necesario enviar a la función el vector de los nudos, basta con la matriz de los nodos. De hecho se podría enviar solamente un vector con los nodos, en lugar de hacerlo como matriz, siempre y cuando se envíen ordenados y también se envíe el grado d. De no enviarlos ordenados, la programación se complicaría bastante, porque habría que reagruparlos de forma que cada $d + 1$ se sitúen entre dos nudos. En cualquier caso, a la hora de evaluar la función polinomial a trozos sí es nencesario enviar el vector de nudos.
   - si optas por la segunda opción, basta que envíes un vector de $m \cdot d + 1$ puntos, indicando también los valores de m y d como variables. La salida será una matriz como en el caso anterior.

     Como me habéis hecho ver, no es imprescindible enviar como argumento el valor m, puesto que a partir del número de puntos $n = m \cdot d + 1$ y el valor de d, podemos calcular m. No obstante no hay inconveniente en que paséis m como argumento... podéis hacerlo como prefiráis.

   Aunque si tu ves otra forma de pasar los datos necesarios, puedes optar por ella.

En cualquiera de los casos puedes utilizar, naturalmente, las funciones de las que ya disponemos: `dif_divNewton.m` (si lo necesitas), `interpolNewton.m` y, si quieres, `polyinterpolador.m` y `polyout` (de Octave).

2. Crea una función en la carpeta `Biblioteca` que pueda evaluar, en un vector de abscisas arbitrario, las funciones polinomiales a trozos obtenidas en el apartado anterior. Esto es completamente análogo a la función de evaluación de splines de la que ya disponemos. Recurre al uso de `polyinterpolador_eval.m`.

3. Chequea tus programas realizando interpolación a trozos en los siguientes casos, comprarándolos con la interpolación usual de Lagrange que se indica.

   a) $f(x) = \operatorname{sen}(x)$, $x \in [0, \pi]$, $m = 5, 15, 30$, $d = 2, 4$. Compara con la interpolación en 5, 15 y 30 nodos.

   b) $f(x) = \frac{1}{1+x^2}$, $x \in [-5, 5]$ (ejemplo de Runge), con los mismos nudos y la misma comparación.

   c) $f(x) = x^{5/2}$, $x \in [0, 1]$. Utiliza nudos más próximos entre sí cerca de 0 que cerca de 1 (recuerda por ejemplo el ejercicio 3 de la práctica 4)

   No es preciso que dejes escritos todos los casos, basta que practiques unos casos y dejes escrito en el archivo `.m` algunos de ellos. Utiliza comentarios para orientar en la lectura del código.

4. Haz un gráfico superponiendo la función, el polinomio interpolador y el interpolador a trozos (en los ejemplos que eliges entre los anteriores) y en otro gráfico dibuja la gráfica de la diferencia entre el polinomio interpolador y el interpolador a trozos para poder darte una idea de la diferencia en la aproximación.

5. Finalmente elige uno de los polinomios interpoladores a trozos para el caso de la función de Runge, y dibuja su derivada: comprobarás que la derivada no es continua.

   Si se opta por la primera forma de interpolación a trozos (nodos que no contienen a los nudos) la función definida a trozos ni siquiera tiene por qué ser continua. Para aquellos que aún no habéis hecho este apartado, podéis cambiar y dibujar la gráfica de la función interpoladora, en lugar de su derivada.

Ejercicio 2

Se trata de realizar la misma idea del ejercicio anterior, pero ahora con polinomios interpoladores de Hermite. Para simplificar el trabajo, se propone concretar: fijamos nudos $z_0, \ldots, z_m$ como partición del intervalo $[a, b]$ e interpolamos con polinomios de grado tres en cada subintervalo, pasando como datos los valores de $f(x)$ en cada z_i y z_{i+1}, así como los valores de la derivada en los mismos puntos.

1. Crea una función en la carpeta `Biblioteca` que devuelva una matriz cuyas filas son los coeficientes de los distintos polinomios, en la forma de Newton, de grado 3 que interpola los valores $\{z_i, f(z_i), f'(z_i)\}$ y $\{z_{i+1}, f(z_{i+1}), f'(z_{i+1})\}$ en cada subintervalo $[z_i, z_{i+1}]$.

   Observa que la función `interpolHermite.m`, que debes utilizar en cada subintervalo, devuelve los nodos repetidos, pero eso no lo necesitamos estrictamente, puesto que sabemos a priori que cada uno se repetirá dos veces. Las variables de entrada de la función deben ser: un vector de nodos x (que ahora coinciden con los nudos), un vector de ordenadas $f(x)$ y un vector de derivadas $f'(x)$. Dentro de la función debes convertir los datos de la función y sus derivadas a una variable de tipo `celda`, que es lo que espera `interpolHermite.m`.

2. Crea una función en la carpeta `Biblioteca` que pueda evaluar, en un vector de abscisas arbitrario, las funciones polinomiales a trozos obtenidas en el apartado anterior. En función de cómo captures la salida en el script del apartado anterior es posible que puedas utilizar la misma función de evaluación que en el ejercicio 1. En caso contrario, crea un nuevo script para evaluar adaptado a este caso.

3. Chequea con ejemplos en torno a la función de Runge, como en el ejercicio anterior, dibujando las gráficas correspondientes de las funciones y el error.

4. Compara gráficamente (y numéricamente) la calidad de la aproximación de estos polinomios de Hermite a trozos con la aproximación de un spline sujeto para la misma función, con el mismo número de subdivisiones del intervalo.

5. Realiza una gráfica de la derivada segunda de los polinomios a trozos para constatar que no es una función con derivada segunda continua.

Ejercicio 3

El propósito de este ejercicio es construir funciones que permitan obtener los polinomios interpoladores en la forma de Lagrange, por tanto, el objetivo fundamental es ser capaz de construir los factores de Lagrange asociados a un conjunto de nodos arbitrario.

1. Crea en `Biblioteca` una función que reciba un vector de nodos $[x_0, \ldots, x_n]$ y devuelva una matriz en la que la fila i-ésima contiene los coeficientes del polinomio

$$L_i(x) = \prod_{j=0,\, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}.$$

   Nota importante. En mis pruebas para este ejercicio encontré dificultades considerables, debido a ciertas limitaciones en el uso de los vectores de Octave (o a mi desconocimiento de éste). Recomiendo: no utilizar directamente una fila de la matriz de polinomios que se debe devolver para realizar los cálculos intermedios; en lugar de ello, utilizar una variable auxiliar, incializada a la constante `1`, donde se vayan almacenando los productos parciales y, una vez finalizado el cálculo de un factor de Lagrange, igualar dicha variable a la fila de la matriz correspondiente. Luego recomenzar, con la misma variable auxiliar reiniciada a la constante 1, con el bucle para otro factor.

2. Crea en `Biblioteca` una función que reciba la matriz de factores de Lagrange, los nodos y las ordenadas en dichos nodos, y devuelva los coeficientes del polinomio interpolador en la forma de Lagrange.

3. Experimenta tus funciones con el caso siguiente: $f(x) = \operatorname{sen}(x)$, $x \in [0, 2\pi]$. Los polinomios interpoladores convergen a la función uniformemente, por tanto debemos ver buenas aproximaciones tanto si calculamos los polinomios en la forma de Lagrange como si lo hacemos en la forma de Newton. Realiza comparaciones (dibujando la gráfica de la diferencia entre uno y otro polinomio) para 3, 5 y 15 nodos equiespaciados (también puedes hacerlo con los de Chebyshev). Escribe en la salida el vector diferencia de los coeficientes de ambos polinomios (cuidado: ambos en forma estándar, debes pasar el polinomio de Newton a su forma estándar); es una forma de verificar los cálculos.

4. Haz el mismo experimento que en el apartado anterior con un número de nodos grande: del orden de 40 o 50. ¿Qué ocurre? ¿Alguna explicación?