MA-0501 Análisis Numérico I
Tarea 2

Luis Fernando Amey Apuy (C20470)
Javier Antonio Hernández Navarro (C13674)

Respuesta corta

1. (2 puntos) Describa un algoritmo para determinar los pesos w_i de la fórmula de Newton-Cotes para $n = 3$ en el intervalo $[-1, 1]$, la cual viene dada por

$$\int_{-1}^{1} f(x)\ dx \approx w_0 f(-1) + w_1 f(-1/3) + w_2 f(1/3) + w_3 f(1).$$

No es necesario calcular los valores exactos.

R/ Vamos a definir los nodos de interpolación, equisdistantes.

$$x_0 = -1,\ x_1 = \frac{-1}{3},\ x_2 = \frac{1}{3}\ x_3 = 1$$

Definiendo $w_k = \int_{-1}^{1} L_k(x)\ dx$, tenemos que

$$\int_{-1}^{1} f(x)\ dx \approx \sum_{k=0}^{3} w_k f(x_k)$$

Si queremos que esta función sea exacta para polinomios de grado 0, 1, 2, 3 podemos resolver un sistema de ecuaciones sobre integrales de los mismos polinomios y evaluando en las funciones del lado izquierdo, con los nodos de interpolación.

$$\int_{-1}^{1} 1\ dx = 2 = w_0 + w_1 + w_2 + w_3$$

$$\int_{-1}^{1} x\ dx = 0 = w_0 - \frac{1}{3}w_1 + \frac{1}{3}w_2 + w_3$$

$$\int_{-1}^{1} x^2\ dx = \frac{2}{3} = w_0 + \frac{1}{9}w_1 + \frac{1}{9}w_2 + w_3$$

$$\int_{-1}^{1} x^3\ dx = 0 = w_0 - \frac{1}{27}w_1 + \frac{1}{27}w_2 + w_3$$

Resolviendo este sistema de ecuaciones logramos encontrar los pesos indicados.

2. (2 puntos) Describa un método, con base en lo estudiado en clase, para aproximar la integral impropia

$$I = \int_0^1 \frac{\sin(x)}{x^{3/2}}\, dx$$

de forma tal que pueda garantizar que el error es menor a una tolerancia dada.

R/ Podemos tomar un cambio de variable

$$x = u^2,\ dx = 2udu$$

$$\int_0^1 \frac{\sin(x)}{x^{3/2}}\, dx = \int_0^1 \frac{\sin(u^2)}{u^2}\, du$$

Y tomar la cuadratura de Gauss compuesta con algún n fijo y variar m hasta satisfacer a la tolerancia dada.

```
g = @(x) 2*sin(x^2)/x^2;
disp(gausscomp(0, 1, 100, 2, g))
```

```
1.935154981985296
```

3. (2 puntos) En una regla de cuadratura en el intervalo $[0, 1]$ con $n + 1$ nodos, se fijan 3 nodos internos. ¿Cuál es el mayor grado de un polinomio que se puede integrar de manera exacta? Justifique de forma general.

R/ Al fijar 3 nodos internos en una regla de cuadratura con $n+1$ nodos en el intervalo $[0, 1]$, se pierde la libertad para fijar 3 condiciones. Esto deja $n - 2$ nodos libres, lo que nos permite satisfacer un total de $n+1$ condiciones con los pesos para asegurar la exactitud de la cuadratura. Por lo tanto, el mayor grado de un polinomio que se puede integrar de manera exacta es $n - 2 + n + 1 - 1 = 2n - 2$

Desarrollo

1. (6 puntos) Considere $n+2$ puntos $\{(x_i, y_i)\}_{i=0}^{n+1}$ (donde los x_i son distintos). Sea q el polinomio de Lagrange de grado n que interpola los puntos $\{(x_i, y_i)\}_{i=0}^{n}$ y sea r el polinomio de Lagrange de grado n que interpola los puntos $\{(x_i, y_i)\}_{i=1}^{n+1}$. Defina

$$p(x) := \frac{(x - x_0)r(x) - (x - x_{n+1})q(x)}{x_{n+1} - x_0}.$$

Demuestre que p es el polinomio de grado $n+1$ que interpola todo el conjunto de datos.

Prueba

Para demostrar que p interpola todo el conjunto de datos, basta con probar que $p(x_i) = y_i$ para $i = 0, 1, \ldots, n+1$.

- Si $x = x_0$

$$p(x_0) = \frac{(x_0 - x_0)r(x_0) - (x_0 - x_{n+1})q(x_0)}{x_{n+1} - x_0} = \frac{-(x_0 - x_{n+1})q(x_0)}{x_{n+1} - x_0}$$

Dado que $q(x_0) = y_0$ y que todos los x_i son distintos, se tiene:

$$p(x_0) = \frac{(x_{n+1} - x_0)y_0}{x_{n+1} - x_0} = y_0$$

- Si $x = x_{n+1}$

$$p(x_{n+1}) = \frac{(x_{n+1} - x_0)r(x_{n+1}) - (x_{n+1} - x_{n+1})q(x_{n+1})}{x_{n+1} - x_0} = \frac{(x_{n+1} - x_0)r(x_{n+1})}{x_{n+1} - x_0}$$

Dado que $r(x_{n+1}) = y_{n+1}$ y que todos los x_i son distintos, se tiene:

$$p(x_{n+1}) = \frac{(x_{n+1} - x_0)y_{n+1}}{x_{n+1} - x_0} = y_{n+1}$$

- Si $x = x_i$ para $i = 1, 2, \ldots, n$

  En este caso, tanto $q(x)$ como $r(x)$ interpolan los puntos $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, por lo que:

$$q(x_i) = r(x_i) = y_i$$

  Entonces:

$$p(x_i) = \frac{(x_i - x_0)r(x_i) - (x_i - x_{n+1})q(x_i)}{x_{n+1} - x_0} = \frac{(x_i - x_0)y_i - (x_i - x_{n+1})y_i}{x_{n+1} - x_0}$$

  Simplificando se obtiene que:

$$p(x_i) = \frac{y_i[(x_i - x_0) - (x_i - x_{n+1})]}{x_{n+1} - x_0} = \frac{y_i(x_i - x_0 - x_i + x_{n+1})}{x_{n+1} - x_0}$$

  Finalmente:

$$p(xi) = \frac{y_i(x_{n+1} - x_0)}{x_{n+1} - x_0} = y_i$$

$\therefore$ p es el polinomio de grado $n+1$ que interpola los $n+2$ puntos del conjunto de datos.

■

2. En este ejercicio se demostrará cómo obtener la fórmula de Gauss de una forma diferente a la vista en clase. Dados n natural, un intervalo $[a, b]$ y una función peso $w : (a, b) \to \mathbb{R}^+$, considere el polinomio ortogonal $\phi_{n+1}(x)$ de grado $n + 1$. En clase demostramos que sus $n + 1$ ceros, los cuales denotamos por $\{x_0, \ldots, x_n\}$, son distintos y están en el intervalo (a, b).

   a) (3 puntos) Sea p un polinomio de grado a lo sumo $2n + 1$. Por el algoritmo de la división, es posible escribir $p(x) = \phi_{n+1}(x)q(x)+r(x)$, para polinomios apropiados q, r. Demuestre que

$$\int_a^b p(x)w(x)\ dx = \int_a^b r(x)w(x)\ dx.$$

   Prueba

   Partiendo del algoritmo de la división se tiene que:

$$\int_a^b p(x)w(x)\ dx = \int_a^b \Big(\phi_{n+1}(x)q(x) + r(x)\Big)w(x)\ dx$$

   Ahora, por linealidad de la integral se obtiene que:

$$\int_a^b p(x)w(x)\ dx = \int_a^b \phi_{n+1}(x)q(x)w(x)\ dx + \int_a^b r(x)w(x)\ dx$$

   Para demostrar la igualdad basta probar que:

$$\int_a^b \phi_{n+1}(x)q(x)w(x)\ dx = 0 \tag{1}$$

   Por el algoritmo de la división es fácil ver que q tiene grado n, pues por hipótesis p es un polinomio de grado a lo sumo $2n + 1$, y ϕ_{n+1} tiene grado $n + 1$.

   Por esta razón, q se puede expresar como una combinación lineal de los polinomios ortogonales $\phi_0, \phi_1, \ldots, \phi_n$[1], es decir:

$$q(x) = \sum_{i=0}^{n} a_i\phi_i(x)$$

   con a_i coeficientes constantes.

   Sustituyendo lo anterior en (1) se obtiene:

$$\int_a^b \phi_{n+1}(x) \left(\sum_{i=0}^{n} a_i\phi_i(x)\right) w(x)\ dx = 0$$

   Dado que los polinomios ϕ_{n+1} y ϕ_i son ortogonales para $i = 0, 1, \ldots, n$ se tiene que:

$$\int_a^b \phi_{n+1}(x)\phi_i(x)w(x)\ dx = 0 \quad \text{para todo } i \leq n$$

[1]En clase se demostró que $< \phi_0, \phi_1, \ldots, \phi_n >$ es una base para los polinomios de grado n.

Entonces, podemos concluir que:

$$\int_a^b \phi_{n+1}(x)q(x)w(x)\ dx = 0$$

$\therefore$ Hemos demostrado que:

$$\int_a^b p(x)w(x)\ dx = \int_a^b r(x)w(x)\ dx$$

■

b) (3 puntos) Exprese r como combinación lineal de los polinomios de Lagrange $\{L_i\}_{i=0}^n$ (asociados a los nodos $\{x_0, \ldots, x_n\}$). Justifique por qué esto es posible.

R/ Se sabe que los polinomios de Lagrange $\{L_i\}_{i=0}^n$ son de la forma:

$$L_i(x_j) = \begin{cases} 1 & \text{si } j = i, \\ 0 & \text{si } j \neq i. \end{cases}$$

Estos polinomios forman una base para los polinomios de grado n, lo que significa que cualquier polinomio de grado a lo sumo n puede escribirse como una combinación lineal de ellos.

Por otro lado, dado que dado que r es un polinomio de grado a lo sumo n (lo cual se obtiene a partir del algoritmo de la división), podemos expresar a $r(x)$ como una combinación lineal de los $L_i(x)$ de la siguiente manera:

$$r(x) = \sum_{i=0}^{n} a_i L_i(x)$$

donde a_i son coeficientes constantes.

Ahora bien, por la propiedad de los polinomios de Lagrange, sabemos que $L_i(x_j) = 0$ cuando $i \neq j$ y $L_j(x_j) = 1$, por lo que al evaluar en cada nodo, la sumatoria se reduce a:

$$r(x_i) = a_i$$

Note que si $x \notin \{x_0, x_1, \ldots, x_n\} \implies L_i(x) = 0$, por lo que solo sobreviven los nodos. De esta forma se tiene que:

$$r(x) = \sum_{i=0}^{n} r(x_i) L_i(x)$$

c) (3 puntos) Demuestre que

$$\int_a^b p(x)w(x)\ dx = \sum_{i=0}^{n} w_i p(x_i)$$

para valores apropiados de $\{w_i\}_{i=0}^n$. Dé explícitamente la fórmula para cada w_i. ¿Coincide con la fórmula dada en clase?

Prueba

Por los incisos (a) y (b), podemos escribir $p(x)$ en términos de los polinomios de Lagrange $L_i(x)$, que están constituidos para los nodos $x_0, x_1, \ldots, x_n$, de la siguiente manera:

$$p(x) = \sum_{i=0}^{n} p(x_i)L_i(x)$$

Partiendo del lado izquierdo de la igualdad, note que:

$$\int_a^b p(x)w(x)\ dx = \int_a^b \left(\sum_{i=0}^{n} p(x_i)L_i(x)\right) w(x)\ dx = \sum_{i=0}^{n} p(x_i) \int_a^b L_i(x)w(x)\ dx$$

Ahora, defina los pesos w_i tales que: [2]

$$w_i = \int_a^b L_i(x)w(x)\ dx$$

De esta forma, se tiene que:

$$\int_a^b p(x)w(x)\ dx = \sum_{i=0}^{n} p(x_i)w_i$$

la cual coincide con la fórmula de la la Cuadratura de Gauss vista en clasen.

■

d) (1 punto) Concluya que la cuadratura $\sum_{i=0}^{n} w_i p(x_i)$ es exacta para cualquier polinomio de grado a lo sumo $2n+1$.

R/ En clase se demostró que la cuadratura de Gauss es exacta para polinomios de grado hasta $2n+1$, esto debido a la propiedad de los polinomios ortogonales y a cómo se eligen los nodos y los pesos. Dado que tanto los pesos como la fórmula en sí coinciden con la demostrada en clase, podemos concluir que esta es exacta para polinomios de grado a lo sumo $2n+1$.

[2] Estos pesos coinciden con los definidos en clase.

3. En este ejercicio se deducirá una manera diferente para definir los nodos y pesos de la cuadratura de Gauss. Considere un conjunto de polinomios ortogonales mónicos[3] $\{\phi_j\}_{j=0}^{\infty}$ en el intervalo $[a, b]$, correspondientes a una función peso $w(x)$.

   a) (6 puntos) Demuestre que existen sucesiones $\{a_k\}_{k=1}^{\infty}$, $\{b_k\}_{k=0}^{\infty}$, tales que los polinomios ortogonales satisfacen la relación de recurrencia

$$\begin{aligned}\phi_1(x) &= (x - b_0)\phi_0(x),\\ \phi_{k+1}(x) &= (x - b_k)\phi_k(x) - a_k\phi_{k-1}(x) \ \forall\ k \geq 1.\end{aligned}$$

   Dé fórmulas explícitas para el término general de cada sucesión[4]. Sugerencia: Por algoritmo de la división, para cada $k \geq 1$ se tiene que $\phi_{k+1}(x) - x\phi_k(x)$ es un polinomio de grado k.

   R/ Sabiendo que $\phi_{k+1}(x) - x\phi_k(x) = p_k \in \mathbb{P}_k$ entonces podemos reformular esta equivalencia para poder empezar a buscar los coeficientes. Sabemos que el polinomio de grado k es una combinación lineal de polinomios ortogonales,

$$\phi_{k+1} - x\phi_k = \sum_{i=0}^{k} \alpha_i \phi_i$$

   Vamos a averiguar quiénes son estos coeficientes. Para $0 \leq j \leq k-2$, integramos con el peso y ϕ_j

$$(\phi_{k+1}, \phi_j)_w - (x\phi_k, \phi_j)_w = \sum_{i=0}^{k} \alpha_i (\phi_i, \phi_j)_w$$

   Como hay ortogonalidad

$$\implies 0 - (x\phi_k, \phi_j)_w = \alpha_j (\phi_j, \phi_j)_w$$

$$\implies -(\phi_k, x\phi_j)_w = \alpha_j \|\phi_j\|^2$$

   Sabemos que $x\phi_j = \phi_{j+1} - p_j = \phi_{j+1} - \sum_{i=0}^{j} \alpha_i \phi_i$

$$\implies -(\phi_k, \phi_{j+1})_w + \sum_{i=0}^{j} \alpha_i (\phi_k, \phi_i)_w = \alpha_j \|\phi_j\|^2$$

   Por ortogonalidad LHS es 0 $\implies \alpha_i = 0$

[3] En este caso, primero normalizamos los polinomios de forma tal que el coeficiente del término de mayor grado es 1.

[4] Deben quedar en términos del producto interno $(f, g)_w = \int_a^b fgw$ o la norma asociada $\|f\| = ((f, f)_w)^{1/2}$.

Ahora tome $j = k - 1$. Por ortogonalidad

$$(\phi_{k+1}, \phi_j)_w - (x\phi_k, \phi_j)_w = \sum_{i=0}^{k} \alpha_i (\phi_i, \phi_j)_w$$

$$0 - (x\phi_k, \phi_j)_w = \alpha_j (\phi_j, \phi_j)_w$$

Simplificando y cambiando α_j por $-a_k$ (para que quede similar a la fórmula inicial)

$$-(\phi_k, x\phi_{k-1})_w = -a_k \|\phi_{k-1}\|^2$$

Aplicamos la misma lógica que la fórmula de equivalencia que usamos:

$$-(\phi_k, x\phi_j)_w = -(\phi_k, \phi_{j+1})_w + \sum_{i=0}^{j} \alpha_i (\phi_k, \phi_i)_w$$

$$\implies -(\phi_k, \phi_k)_w + \sum_{i=0}^{k-1} \alpha_i (\phi_k, \phi_i)_w = -a_k \|\phi_{k-1}\|^2$$

La sumatoria es 0 por ortogonalidad. Entonces

$$-\|\phi_k\|^2 = -a_k \|\phi_{k-1}\|^2$$

$$\implies a_k = \frac{\|\phi_k\|^2}{\|\phi_{k-1}\|^2}$$

Por último tome $j = k$

$$(\phi_{k+1}, \phi_j)_w - (x\phi_k, \phi_j)_w = \sum_{i=0}^{k} \alpha_i (\phi_i, \phi_j)_w$$

$$0 - (x\phi_k, \phi_j)_w = \alpha_j (\phi_j, \phi_j)_w$$

Simplificando y cambiando α_j por $-b_k$

$$-(\phi_k, x\phi_k)_w = -b_k \|\phi_k\|^2$$

$$\implies b_k = \frac{(x, \phi_k^2)_w}{\|\phi_k\|^2}$$

Podemos armar las ecuaciones ya que sabemos todos los coeficientes

$$\phi_{k+1} = (x - b_k)\phi_k - a_k \phi_{k-1}$$

Como para $k = 0$, $\phi_{k-1} = 0$ (cuando se trabaja no existe ese polinomio)

$$\phi_1 = (x - b_0)\phi_0$$

b) (4 puntos) Considere ahora polinomios ortonormales $\{\widetilde{\phi}_j\}_{j=0}^{\infty}$. Escribiendo $\phi_k = \|\phi_k\|\widetilde{\phi}_k$ y las fórmulas de los coeficientes del inciso anterior, deduzca que existen sucesiones $\{\alpha_k\}_{k=1}^{\infty}$, $\{\beta_k\}_{k=0}^{\infty}$ tales que

$$\begin{aligned} \alpha_1\widetilde{\phi}_1(x) + \beta_0\widetilde{\phi}_0(x) &= x\widetilde{\phi}_0(x), \\ \alpha_{k+1}\widetilde{\phi}_{k+1}(x) + \beta_k\widetilde{\phi}_k(x) + \alpha_k\widetilde{\phi}_{k-1}(x) &= x\widetilde{\phi}_k(x) \ \forall\ k \geq 1. \end{aligned} \tag{2}$$

R/ Despejando las relaciones de recurrencia

$$\widetilde{\phi}_{k+1}\|\phi_{k+1}\| = (x - b_k)\widetilde{\phi}_k\|\phi_k\| - a_k\widetilde{\phi}_{k-1}\|\phi_{k-1}\|$$

$$\widetilde{\phi}_{k+1}\|\phi_{k+1}\| + a_k\widetilde{\phi}_{k-1}\|\phi_{k-1}\| + b_k\widetilde{\phi}_k\|\phi_k\| = x\widetilde{\phi}_k\|\phi_k\|$$

$$\widetilde{\phi}_{k+1}\frac{\|\phi_{k+1}\|}{\|\phi_k\|} + a_k\widetilde{\phi}_{k-1}\frac{\|\phi_{k-1}\|}{\|\phi_k\|} + b_k\widetilde{\phi}_k = x\widetilde{\phi}_k$$

Sustituyendo $a_k = \dfrac{\|\phi_k\|^2}{\|\phi_{k-1}\|^2}$

$$\widetilde{\phi}_{k+1}\frac{\|\phi_{k+1}\|}{\|\phi_k\|} + \widetilde{\phi}_{k-1}\frac{\|\phi_k\|}{\|\phi_{k-1}\|} + b_k\widetilde{\phi}_k = x\widetilde{\phi}_k$$

Tome $\beta_k = b_k$ y $\alpha_k = \dfrac{\|\phi_k\|}{\|\phi_{k-1}\|}$. Entonces

$$\alpha_{k+1}\widetilde{\phi}_{k+1} + \beta_k\widetilde{\phi}_k + \alpha_k\widetilde{\phi}_{k-1} = x\widetilde{\phi}_k$$

Igual, para $k = 0$, $\widetilde{\phi}_{k-1} = 0$, entonces

$$\alpha_1\widetilde{\phi}_1 + \beta_0\widetilde{\phi}_0 = x\widetilde{\phi}_0$$

c) (4 puntos) Para n fijo, considere los ceros $\{x_0, \ldots, x_n\}$ del polinomio $\widetilde{\phi}_{n+1}$. Evalúe (2) (para $1 \leq k \leq n$) en un cero x_j . Concluya que x_j es un cero del polinomio ortogonal $\widetilde{\phi}_{n+1}$ si y solo si es un valor propio de la matriz T de tamaño $(n+1) \times (n+1)$ dada por

$$T = \begin{bmatrix} \beta_0 & \alpha_1 & & & & \\ \alpha_1 & \beta_1 & \alpha_2 & & & \\ & \alpha_2 & \beta_2 & \alpha_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \alpha_{n-1} & \beta_{n-1} & \alpha_n \\ & & & & \alpha_n & \beta_n \end{bmatrix}.$$

R/ Usando las fórmulas anteriores para un x_j fijo nos damos cuenta de algo

$$\beta_0\widetilde{\phi}_0(x_j) + \alpha_1\widetilde{\phi}_1(x_j) = x_j\widetilde{\phi}_0(x_j)$$

$$\alpha_{k+1}\widetilde{\phi}_{k+1}(x_j) + \beta_k\widetilde{\phi}_k(x_j) + \alpha_k\widetilde{\phi}_{k-1}(x_j) = x_j\widetilde{\phi}_k(x_j)$$

Y por x_j ser cero de $\widetilde{\phi}_{n+1}$

$$\beta_n\widetilde{\phi}_n(x_j) + \alpha_n\widetilde{\phi}_{n-1}(x_j) = x_j\widetilde{\phi}_n(x_j)$$

Podemos expresar estas equivalencias de la siguiente multiplicación de matrices, definiendo a $\widetilde{\phi} = [\widetilde{\phi}_0, \widetilde{\phi}_1, \ldots, \widetilde{\phi}_n]^T$

$$T\widetilde{\phi}(x_j) = x_j\widetilde{\phi}(x_j)$$

Dándonos cuenta que x_j es un valor propio de la matriz dada y $\widetilde{\phi}(x_j)$ su vector propio respectivo. Esta matriz no se logra formar si x_j no es un cero, puesto que ocupamos la fila *n+1* como tal, donde se cancela el término $\widetilde{\phi}(x_j)$ como vimos anteriormente. Por lo tanto x_j es un cero de $\widetilde{\phi}_{n+1}$ si y solo si es un valor propio de T.

d) (4 puntos) Para $i, j \in \{0, \ldots, n\}$, utilice el hecho que la cuadratura de Gauss integra de forma exacta $\widetilde{\phi}_i\widetilde{\phi}_j$ para deducir que

$$\delta_{ij} = \sum_{k=0}^{n} w_k \widetilde{\phi}_i(x_k)\widetilde{\phi}_j(x_k).$$

Concluya que $P^T W P = I$, donde I es la matriz identidad de tamaño $(n+1) \times (n+1)$, y las matrices W, P vienen dadas por

$$W = \begin{bmatrix} w_0 & 0 & \ldots & 0 \\ 0 & w_1 & \ddots & \vdots \\ \vdots & \ddots & & 0 \\ 0 & \ldots & 0 & w_n \end{bmatrix}, P = \begin{bmatrix} \widetilde{\phi}_0(x_0) & \ldots & \widetilde{\phi}_n(x_0) \\ \vdots & \ddots & \vdots \\ \widetilde{\phi}_0(x_n) & \ldots & \widetilde{\phi}_n(x_n) \end{bmatrix}.$$

R/ Sabiendo que el delta de Kroneker es el siguiente y por lo anterior podemos deducir por la cuadratura de Gauss que

$$\delta_{ij} = \int \widetilde{\phi}_i(x)\widetilde{\phi}_j(x)w(x)dx = \sum_{k=0}^{n} w_k \widetilde{\phi}_i(x_k)\widetilde{\phi}_j(x_k)$$

Ahora bien, sabemos que

$$\delta_{ij} \quad = \quad \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

Por lo que podemos crear una matriz identidad con los deltas, y crear multiplicaciones de matrices para poder separar la sumatoria.

$$[w_0\widetilde{\phi}_i(x_0), \ldots, w_k\widetilde{\phi}_i(x_k)] \cdot [\widetilde{\phi}_j(x_0), \ldots, \widetilde{\phi}_j(x_n)]^T = \sum_{k=0}^{n} w_k \widetilde{\phi}_i(x_k)\widetilde{\phi}_j(x_k)$$

Por esto ya podemos tomar a

$$P = \begin{bmatrix} \widetilde{\phi}_0(x_0) & \ldots & \widetilde{\phi}_n(x_0) \\ \vdots & \ddots & \vdots \\ \widetilde{\phi}_0(x_n) & \ldots & \widetilde{\phi}_n(x_n) \end{bmatrix}$$

Lo que nos queda en la sumatoria es un vector horizontal de la matriz restante $[w_0\widetilde{\phi}_i(x_0), \ldots, w_k\widetilde{\phi}_i(x_k)]$. Podemos hacer la matriz restante

$$\begin{bmatrix} w_0\widetilde{\phi}_0(x_0) & \ldots & w_n\widetilde{\phi}_0(x_n) \\ \vdots & \ddots & \vdots \\ w_0\widetilde{\phi}_n(x_0) & \ldots & w_n\widetilde{\phi}_n(x_n) \end{bmatrix}$$

De aquí podemos observar que la matriz restante es igual a $P^T W$, puesto cualquier posición ij que queramos calcular ocupa la transpuesta de P multiplicado por un peso, que se repite en cada columna. Por tanto $P^T W P = I$

e) (2 puntos) Es posible demostrar que $W^{-1} = PP^T$. Deduzca así que

$$\frac{1}{w_j} = \sum_{i=0}^{n} (\widetilde{\phi}_i(x_j))^2.$$

R/ De la multiplicación de matrices y saber que

$$W^{-1} = \begin{bmatrix} \frac{1}{w_0} & 0 & \dots & 0 \\ 0 & \frac{1}{w_1} & \ddots & \vdots \\ \vdots & \ddots & & 0 \\ 0 & \dots & 0 & \frac{1}{w_n} \end{bmatrix}$$

$$\implies \frac{1}{w_j} = (W^{-1})_{jj} = (PP^T)_{jj} = [\phi_0(x_j), \dots, \phi_n(x_j)] \cdot [\phi_0(x_j), \dots, \phi_n(x_j)]^T$$

$$\implies \frac{1}{w_j} = \sum_{i=0}^{n} (\widetilde{\phi}_i(x_j))^2$$

f) (2 puntos) Considere ahora el vector propio unitario $v^{(j)} = [v_1^{(j)}, \dots, v_{n+1}^{(j)}]^T$ de la matriz T asociado al valor propio (nodo) x_j. Demuestre que existe una constante C tal que $v^{(j)} = C[\widetilde{\phi}_0(x_j), \dots, \widetilde{\phi}_n(x_j)]^T$.

R/Sabemos por los incisos anteriores que $[\widetilde{\phi}_0(x_j), \dots, \widetilde{\phi}_n(x_j)]^T = \widetilde{\phi}$ es el vector propio asociado al nodo x_j. Además, todas las soluciones son distintas, entonces los vectores propios no generarán una base. Por lo que entonces tomamos la norma para hacerlo unitario

$$\implies \frac{\widetilde{\phi}}{\|\widetilde{\phi}\|} = v^{(j)} \implies C = \|\widetilde{\phi}\|^{-1}$$

g) (2 puntos) Demuestre que $C = \sqrt{\mu} v_1^{(j)}$, con $\mu = \int_a^b w(x)\, dx$. Sugerencia: utilice el hecho que $(\widetilde{\phi}_0, \widetilde{\phi}_0)_w = 1$.

R/ Sabemos que $\|\phi_0\| = 1$. Al aplicar la constante en el primer valor del vector tenemos entonces que

$$C\phi_0 = v_1^{(j)} \implies C = \frac{v_1^{(j)}}{\phi_0}$$

$$\implies C = \frac{v_1^{(j)}}{\phi_0} \frac{\sqrt{\mu}}{\sqrt{\mu}}$$

Al expandir lo de abajo

$$\sqrt{\mu}\phi_0 = \sqrt{(\mu\phi_0^2)}$$

Puesto $\phi_0(x_j) = \phi_0(x)$ porque es una constante

$$\implies \sqrt{(\mu\phi_0^2)} = \|\phi_0\|^{1/2} = 1$$

$$\implies C = \sqrt{\mu} v_1^{(j)}$$

h) (2 puntos) Concluya que los pesos de la cuadratura se pueden escribir como $w_j = \mu(v_1^{(j)})^2$.

$$\frac{1}{w_j} = \sum_{i=0}^{n} (\widetilde{\phi}_i(x_j))^2 = \sum_{i=0}^{n} \left(\frac{v_i^{(j)}}{\sqrt{\mu} v_1^{(j)}} \right)^2 \implies w_j = \frac{\mu (v_1^{(j)})^2}{\sum_{i=0}^{n} (v_i^{(j)})^2}$$

Sabemos que el vector $v^{(j)}$ es unitario. Por lo que su norma es 1.

$$\sum_{i=0}^{n} (v_i^{(j)})^2 = \|v^{(j)}\|^2 = 1 \implies w_j = \mu (v_1^{(j)})^2$$

i) (8 puntos) [`MATLAB`] Para el caso $w(x) = 1$ en $[-1, 1]$, escriba una función que calcule la matriz T para n dado, y sus valores y vectores propios. Utilice las fórmulas demostradas anteriormente para que la función retorne los pesos y nodos de la cuadratura de Gauss. Sugerencia: para calcular los valores y vectores propios utilice la función `[V,D]=eig(T)`.

R/ Estos polinomios son de Legendre, con la relación

$$P_0 = 1,\ P_1 = x,\ (k+1)P_{k+1} = (2k+1)xP_k - kP_{k-1}$$

También, al calcular μ, obtenemos $\mu = \int_{-1}^{1} w(x)\ dx = 1 - -1 = 2$
Por lo que podemos calcular los pesos de manera fácil.
Tenemos que calcular α_k, puesto $b_k = 0$ según la fórmula de Legendre anterior.
Aproximando por medio de una normalización λ_k, tenemos que

$$\frac{k+1}{2k+1} \frac{\lambda_{k+1}}{\lambda_k} P_{k+1} + \frac{k}{2k+1} \frac{\lambda_{k-1}}{\lambda_k} P_{k-1} = xP_k$$

Tomando a

$$\frac{k+1}{2k+1} \frac{\lambda_{k+1}}{\lambda_k} = \alpha_{k+1},\ \frac{k}{2k+1} \frac{\lambda_{k-1}}{\lambda_k} = \alpha_k$$

Podemos hacer una igualdad

$$\frac{k+1}{2k+1} \frac{\lambda_{k+1}}{\lambda_k} = \frac{k+1}{2k+3} \frac{\lambda_k}{\lambda_{k+1}}$$

$$\lambda_{k+1} = \lambda_k \sqrt{\frac{2k+1}{2k+3}}$$

Sabiendo que $\|P_0\| = \sqrt{\int_{-1}^{1} w(x)dx} = \sqrt{2}$

$$\implies \lambda_{k+1} = \sqrt{2} \sqrt{\frac{1}{2k+3}},\ \lambda_k = \sqrt{\frac{2}{2k+1}}$$

$$\implies \alpha_k = \frac{k}{2k+1} \sqrt{\frac{2k+1}{2k-1}} = \frac{k}{\sqrt{(2k+1)(2k-1)}} = \frac{k}{\sqrt{4k^2-1}}$$

Con estos valores de alpha, ya podemos hacer la función programada.

```
function [T, nod, w] = tortog(n)
    %
    %   Función que recibe el tamaño de la matriz T y calcula
    %   los nodos y pesos de la cuadratura de Gauss
    %
    %   Inputs:
    %       n   - el tamaño de la matriz T
    %
    %   Outputs:
    %       T   - matriz de los alpha y beta demostrada
    %       nod - los nodos de la cuadratura de Gauss
    %       w   - los pesos de la cuadratura de Gauss

    % Iniciamos los valores
    T = zeros(n+1);
    nod = zeros(n+1, 1);
    w = zeros(n+1, 1);

    % Rellenamos los alpha
    for k = 1:n
        T(k,k+1) = k/sqrt(4*k^2 - 1);
        T(k+1,k) = T(k,k+1);
    end

    % Calculamos los valores y vectores propios
    [vec, val] = eig(T);

    % Calculamos los nodos y pesos de la cuadratura de Gauss
    for i = 1:n+1
        w(i) = 2*(vec(1, i))^2;
        nod(i) = val(i,i);
    end
end
```

j) (2 puntos) [MATLAB]Verifique que para $n = 0$ su programa retorna $x_0 = 0, w_0 = 2$, y que para $n = 1$ retorna $x_0 = -1/\sqrt{3}, x_1 = 1/\sqrt{3}, w_0 = w_1 = 1$.

R/ Primero para $n = 0$, tenemos el nodo

```
[~, nod, w] = tortog(0);
disp(nod)
```

```
     0
```

Y su peso

```
disp(w)
```

```
     2
```

Después, respectivamente, tenemos $n = 1$, con sus nodos en orden

```
[~, nod, w] = tortog(1);
disp(nod)
```

```
   -0.5774
    0.5774
```

Y sus pesos

```
disp(w)
```

```
    1.0000
    1.0000
```

k) (4 puntos) [MATLAB] Construya una cuadratura que permita integrar $f(x) = x^8 + 2x^2 + x$ de manera exacta. Utilice el mínimo valor de n posible. Escriba los nodos y pesos que utilizó, y verifique que obtiene el resultado correcto.

R/ Usaremos 5 nodos, ya que la cuadratura integra de forma exacta a 5*2-1 = 9. Sabemos que la integral exacta está dada por

$$\int_{-1}^{1} x^8 + 2x^2 + x \; dx = \left(\frac{x^9}{9} + \frac{2x^3}{3} + \frac{x^2}{2} \right) \Bigg|_{-1}^{1} = \frac{2 + 12}{9} = \frac{14}{9}$$

También la podemos revisar en el código

```
intf = @(x) x^9/9 + 2/3*x^3 + 0.5*x^2;
disp(intf(1) - intf(-1))
```

```
1.5556
```

Construyendo la cuadratura con 5 nodos, entonces podemos calcularla sin la necesidad de integrar, pero con la cuadratura respectiva

```
n = 5;
[T, nod, w] = tortog(n-1);
f = @(x) x^8 + 2*x^2 + x;
int = 0;
for k = 1:n
    int = int + w(k)*f(nod(k));
end
disp(int)
```

```
1.5556
```

l) (6 puntos) [MATLAB] Implemente una función que calcule la fórmula compuesta de la cuadratura de Gauss. Para ello, considere como entrada el intervalo $[a, b]$, el número de subintervalos m, el valor n para la cuadratura de Gauss y la función f. Dicha función debe calcular una partición uniforme $\{x_j\}_{j=0}^{m}$ dada por

$$x_j := a + jh \ (j = 0, \ldots, m), \quad \text{con} \ \ h = \frac{b-a}{m},$$

y calcular la fórmula

$$\int_a^b f(x) \ dx \approx \frac{h}{2} \sum_{j=1}^{m} \sum_{k=0}^{n} W_k f\left(\frac{1}{2}(x_{j-1} + x_j) + \frac{1}{2} h \widetilde{x}_k\right) \ dt,$$

donde $\{\widetilde{x}_k\}_{k=0}^{n}$ son los ceros de un polinomio ortogonal de grado $n+1$ en el intervalo $[-1, 1]$ y w_k los pesos asociados (que calculó anteriormente).

```
function c = gausscomp(a, b, m, n, f)
    %
    %   Función que calcula la cuadratura de Gauss compuesta,
    %   por medio de subintervalos m, con un nivel n, en el
    %   intervalo [a,b] de la función f
    %
    %   Inputs
    %       a - intervalo izquierdo
    %       b - intervalo derecho
    %       m - la cantidad de subintervalos
    %       n - el nivel de las cuadraturas de Gauss
    %       f - la función a integrar
    %
    %   Outputs
    %       c - el valor de la integral

    % Realizamos el ancho de banda y los espaciamientos
    h = (b - a) / m;
    xj = linspace(a, b, m+1);

    % Hacemos la doble sumatoria de la cuadratura compuesta
    [~, nod, w] = tortog(n);
    c = 0;
    for j = 1:m
        for k = 1:n+1
            c = c + w(k)*f((xj(j) + xj(j+1))/2 + (h/2)*nod(k));
        end
    end
    c = c * h / 2;
end
```

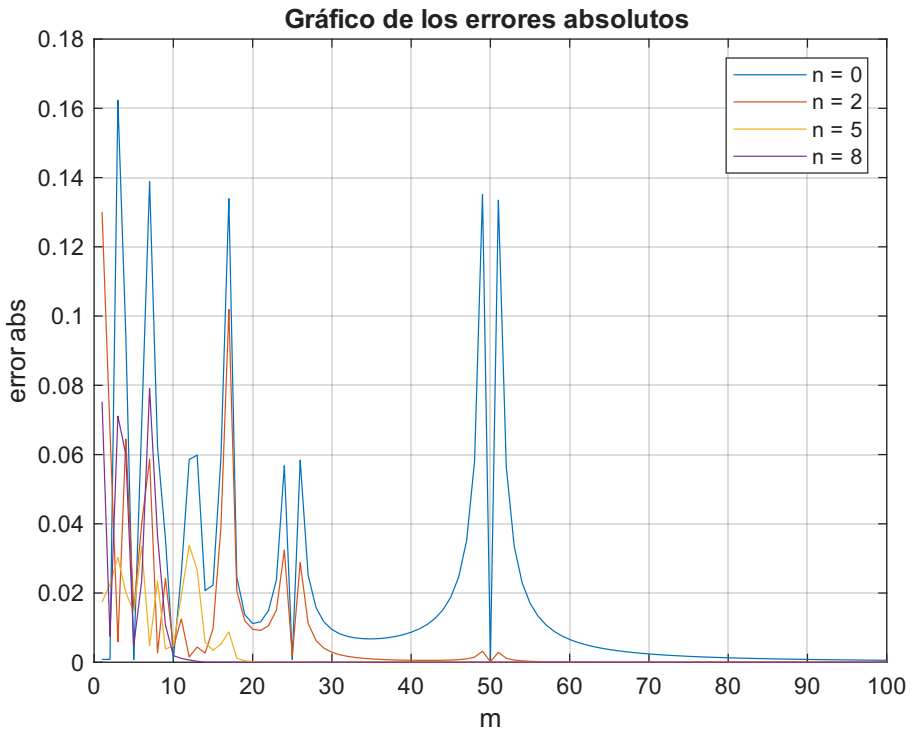
Podemos comprobar la funcionalidad de la función realizando el ejercicio anterior con un $m = 1$

```
disp(gausscomp(-1, 1, 1, 5, f))
```

```
1.5556
```

m) (4 puntos) [MATLAB] Considere $g(x) = \text{sen}(100\pi x)(1-x)^{1/2}\log(1-x)$ en el intervalo $[0, 1]$. Grafique, para $n \in \{0, 2, 5, 8\}$, el error absoluto de esta cuadratura compuesta en función de m (una curva para cada valor de n). Comente sus resultados. Utilice $I_g = 0.0008197612371239843$ como valor exacto.

```
g = @(x) sin(100.*pi.*x).*((1-x).^0.5).*log(1-x);
m = zeros(100, 4);
ind = [0, 2, 5, 8];
for j = 1:4
    for i = 1:100
        m(i,j) = abs(gausscomp(0, 1, i, ind(j), g) - 0.000819...);
    end
end
```



En la ejecución se nota una inconsistencia con $n = 0$, en general no es recomendable ya que inclusive no converge relativamente bien como los otros valores de n. Inclusive es mucho mejor tomar $n = 2$ en términos de eficiencia, puesto que en los valores altos de m igual los tres siguen la misma línea. Claramente es mejor $n = 8$ pero no debe de ser eficiente para algún método de integrales.

4. En este ejercicio se busca construir una aproximación para la derivada de una función, a partir de su polinomio de interpolación de una manera diferente a las fórmulas estudiadas de $n+1$ puntos del Capítulo 3.

   a) (4 puntos) Demuestre que

$$L_k(x) = \frac{\lambda_k}{x - x_k} \Bigg/ \sum_{j=0}^{n} \frac{\lambda_j}{x - x_j}, \text{para } x \notin \{x_0, \ldots, x_n\},$$

   donde $\{L_k\}$ es la base de polinomios de interpolación de Lagrange. Deduzca que

$$L_k(x)s(x) = \lambda_k \frac{x - x_i}{x - x_k},$$

   donde $s(x) = \sum_{j=0}^{n} \lambda_j \frac{x - x_i}{x - x_j}$, $i \in \{0, 1, \ldots, n\}$.

   Prueba
   Por definición se tiene:

$$L_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)}$$

   Observe que:

$$L_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)} \cdot \frac{(x - x_k)}{(x - x_k)} = \frac{L(x)}{L'(x_k)(x - x_k)}$$

   Ahora, defina $\lambda_k = \dfrac{1}{L'(x_k)}$, se sigue que:

$$L_k(x) = \lambda_k \frac{L(x)}{(x - x_k)} \tag{3}$$

   Por otro lado sabemos que:

$$\sum_{k=0}^{n} L_k(x) = 1 \overset{(3)}{\Longrightarrow} \sum_{k=0}^{n} \lambda_k \frac{L(x)}{(x - x_k)} = 1 \Longrightarrow L(x) \sum_{k=0}^{n} \lambda_k \frac{1}{(x - x_k)} = 1$$

   Al dividir (3) por lo anterior se obtiene que:

$$L_k(x) = \frac{\lambda_k \dfrac{L(x)}{(x - x_k)}}{L(x) \displaystyle\sum_{k=0}^{n} \lambda_k \frac{1}{(x - x_k)}}$$

Finalmente, al simplificar se obtiene la igualdad a demostrar:

$$L_k(x) = \frac{\dfrac{\lambda_k}{(x - x_k)}}{\displaystyle\sum_{k=0}^{n} \frac{\lambda_k}{(x - x_k)}}$$

■

Ahora, multiplicando la igualdad por un 1 conveniente se obtiene:

$$L_k(x) = \frac{\dfrac{\lambda_k}{(x - x_k)}}{\displaystyle\sum_j \lambda_j \frac{1}{(x - x_j)}} \cdot \frac{(x - x_i)}{(x - x_i)} = \frac{\lambda_k \dfrac{(x - x_i)}{(x - x_k)}}{\displaystyle\sum_j \lambda_j \frac{(x - x_i)}{(x - x_j)}}$$

Al reemplazar $s(x) = \displaystyle\sum_{j=0}^{n} \lambda_j \frac{x - x_i}{x - x_j}$ se tiene:

$$L_k(x) = \frac{\lambda_k \dfrac{(x - x_i)}{(x - x_k)}}{s(x)}$$

Y finalmente podemos deducir que:

$$L_k(x)s(x) = \lambda_k \frac{(x - x_i)}{(x - x_k)} \tag{4}$$

b) (2 puntos) Del inciso (a), deduzca que

$$L_k'(x)s(x) + L_k(x)s'(x) = \lambda_k \left(\frac{x - x_i}{x - x_k} \right)'.$$

R/ Mediante un cálculo directo al derivar la igualdad (4) se obtiene que:

$$L_k'(x)s(x) + L_k(x)s'(x) = \lambda_k \left(\frac{x - x_i}{x - x_k} \right)'$$

Donde:

$$L_k'(x) = \sum_{j \neq k} \frac{1}{x_k - x_j} \left(\prod_{m \neq j,k} \frac{x - x_m}{x_k - x_m} \right)$$

$$s'(x) = \sum_j \lambda_j \frac{x_j - x_i}{(x - x_j)^2}$$

$$\left(\frac{x - x_i}{x - x_k} \right)' = \frac{x_i - x_k}{(x - x_k)^2}$$

c) (2 puntos) Verifique que $s(x_i) = \lambda_i$ y deduzca que

$$L_k'(x_i) = \frac{\lambda_k}{\lambda_i} \frac{1}{x_i - x_k} (i \neq k).$$

R/ Note que:

$$s(x) = \sum_{j=0}^{n} \lambda_j \frac{x - x_i}{x - x_j} = \lambda_i \frac{x - x_i}{x - x_i} + \sum_{\substack{j=0 \\ j \neq i}}^{n} \lambda_j \frac{x - x_i}{x - x_j}$$

Al reemplazar $x = x_i$ se tiene que:

$$s(x_i) = \lambda_j \frac{x_i - x_i}{x_i - x_i} + \sum_{\substack{j=0 \\ j \neq i}}^{n} \lambda_j \frac{x_i - x_i}{x_i - x_j}$$

Fácilmente se puede ver que $\sum_{\substack{j=0 \\ j \neq i}}^{n} \lambda_j \frac{x_i - x_i}{x_i - x_j} = 0$ para todo $j \neq i$.

Por otro lado, el término $\lambda_j \frac{x_i - x_i}{x_i - x_i}$ presente a priori una indeterminación del tipo $\frac{0}{0}$, por lo que se procede a tomar el límite:

$$\lim_{x \to x_i} \lambda_i \frac{x - x_i}{x - x_i} \;=\; \lim_{x \to x_i} \lambda_i \cdot 1 \;=\; \lambda_i$$

Entonces se tiene que:

$$s(x_i) = \lambda_i$$

Por otro lado, al evaluar $x = x_i$ con $i \neq k$ en la igualdad del inciso (b) se obtiene que:

$$\begin{aligned}
& L_k'(x_i)s(x_i) + L_k(x_i)s'(x_i) = \lambda_k \left(\frac{x_i - x_i}{x - x_k} \right)' \\
\implies & L_k'(x_i)s(x_i) + L_k(x_i)s'(x_i) = 0 \\
\implies & L_k'(x_i)s(x_i) = -L_k(x_i)s'(x_i) \\
\implies & L_k'(x_i) = \frac{-L_k(x_i)s'(x_i)}{s(x_i)} \\
\implies & L_k'(x_i) = \frac{-L_k(x_i)s'(x_i)}{\lambda_i}
\end{aligned}$$

Sustituyendo $L_k(x_i)$ por lo obtenido en el inciso (a) se tiene:

$$L'_k(x_i) = \frac{-\left(\dfrac{\lambda_k}{x_i - x_k} \Big/ \displaystyle\sum_{j=0}^{n} \frac{\lambda_j}{x_i - x_j}\right) s'(x_i)}{\lambda_i}$$

Reacomodando la expresión anterior se llega a que:

$$L'_k(x_i) = \frac{-\lambda_k}{\lambda_i} \cdot \frac{1}{(x_i - x_k)} \cdot \frac{s'(x_i)}{\displaystyle\sum_{j=0}^{n} \frac{\lambda_j}{(x_i - x_j)}} \tag{5}$$

Finalmente, mediante un cálculo directo se obtiene que:

$$\begin{aligned}
s'(x) &= \sum_j \lambda_j \frac{x_j - x_i}{(x - x_j)^2} \\
\implies s'(x_i) &= \sum_{j=0}^{n} \lambda_j \frac{x_j - x_i}{(x_i - x_j)^2} \\
\implies s'(x_i) &= -\sum_{j=0}^{n} \lambda_j \frac{x_i - x_j}{(x_i - x_j)^2} \\
\implies s'(x_i) &= -\sum_{j=0}^{n} \lambda_j \frac{1}{(x_i - x_j)}
\end{aligned}$$

De lo anterior se obtiene que:

$$\frac{s'(x_i)}{\displaystyle\sum_{j=0}^{n} \lambda_j \frac{1}{(x_i - x_j)}} = -1$$

Reemplazando esta última igualdad en (5) se llega a que:

$$L'_k(x_i) = \frac{-\lambda_k}{\lambda_i} \cdot \frac{1}{(x_i - x_k)} \cdot -1$$

Y así se deduce que:

$$L'_k(x_i) = \frac{\lambda_k}{\lambda_i} \cdot \frac{1}{(x_i - x_k)} \quad \text{para } i \neq k$$

d) (2 puntos) Demuestre que

$$L'_k(x_k) = -\sum_{i \neq k} L'_i(x_k).$$

Prueba

Se sabe que:

$$L_0(x) + L_1(x) + \ldots + L_k(x) + \ldots + L_n(x) = 1$$

Derivando ambos lados de la ecuación se llega a:

$$L'_0(x) + L'_1(x) + \ldots + L'_k(x) + \ldots + L'_n(x) = 0$$

Y despejando se obtiene que:

$$L'_k(x) = -\sum_{i \neq k} L'_i(x)$$

Finalmente, al evaluar en $x = x_k$ se obtiene:

$$L'_k(x_k) = -\sum_{i \neq k} L'_i(x_k)$$

■

e) (6 puntos) Denote la matriz $D \in \mathbb{R}^{n+1 \times n+1}$ cuyas entradas vienen dadas por $D_{ik} = L'_k(x_i)$, para $i, j \in \{0, 1, \ldots, n\}$. ¿Qué representa el vector $Df(\hat{\boldsymbol{x}})$, donde $\hat{\boldsymbol{x}}$ es el vector de nodos de Chebyshev y $f(\hat{\boldsymbol{x}})$ el vector con las imágenes de dichos nodos?

R/ Por definición, la matriz $D \in \mathbb{R}^{n+1 \times n+1}$ es de la forma:

$$D = \begin{pmatrix} L'_0(x_0) & L'_1(x_0) & \cdots & L'_n(x_0) \\ L'_0(x_1) & L'_1(x_1) & \cdots & L'_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ L'_0(x_n) & L'_1(x_n) & \cdots & L'_n(x_n) \end{pmatrix}$$

De los incisos (c) y (d) se sabe que:

$$L'_k(x_i) = \begin{cases} \dfrac{\lambda_k}{\lambda_i} \dfrac{1}{x_i - x_k} & \text{si } i \neq k \\ -\displaystyle\sum_{i \neq k} L'_i(x_k) & \text{si } i = k \end{cases}$$

El vector $f(\hat{x})$ es el vector de valores de la función $f(x)$ evaluada en los nodos $\{x_0, x_1, \ldots, x_n\}$ los cuales corresponden a los nodos de Chebyshev. Al multiplicar la matriz D por este vector $f(\hat{x})$ se obtiene:

$$Df(\hat{x}) = \begin{pmatrix} L_0'(x_0) & L_1'(x_0) & \cdots & L_n'(x_0) \\ L_0'(x_1) & L_1'(x_1) & \cdots & L_n'(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ L_0'(x_n) & L_1'(x_n) & \cdots & L_n'(x_n) \end{pmatrix} \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix}$$

$$= \begin{pmatrix} L_0'(x_0)f(x_0) + L_1'(x_0)f(x_1) + \cdots + L_n'(x_0)f(x_n) \\ L_0'(x_1)f(x_0) + L_1'(x_1)f(x_1) + \cdots + L_n'(x_1)f(x_n) \\ \vdots \\ L_0'(x_n)f(x_0) + L_1'(x_n)f(x_1) + \cdots + L_n'(x_n)f(x_n) \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{k=0}^{n} f(x_k)L_k'(x_0) \\ \sum_{k=0}^{n} f(x_k)L_k'(x_1) \\ \vdots \\ \sum_{k=0}^{n} f(x_k)L_k'(x_n) \end{pmatrix} = \begin{pmatrix} p_n'(x_0) \\ p_n'(x_1) \\ \vdots \\ p_n'(x_n) \end{pmatrix}$$

El resultado de $Df(\hat{x})$ es un vector de aproximaciones de las derivadas de f en los nodos $\{x_0, x_1, \ldots, x_n\}$, es decir:

$$Df(\hat{x}) \approx \begin{pmatrix} f'(x_0) \\ f'(x_1) \\ \vdots \\ f'(x_n) \end{pmatrix}$$

Note que el resultado anterior es una aproximación, ahora, si a cada $p_n'(x_i)$ se le sumara el término $e(x_i) = \dfrac{f^{(n+1)}(\xi)}{(n+1)!} \displaystyle\prod_{k \neq i} (x_i - x_k)$ entonces:

$$Df(\hat{x}) + e(x) = \begin{pmatrix} p_n'(x_0) + e(x_0) \\ p_n'(x_1) + e(x_1) \\ \vdots \\ p_n'(x_n) + e(x_n) \end{pmatrix} = \begin{pmatrix} f'(x_0) \\ f'(x_1) \\ \vdots \\ f'(x_n) \end{pmatrix}$$

según el teorema visto en clase.

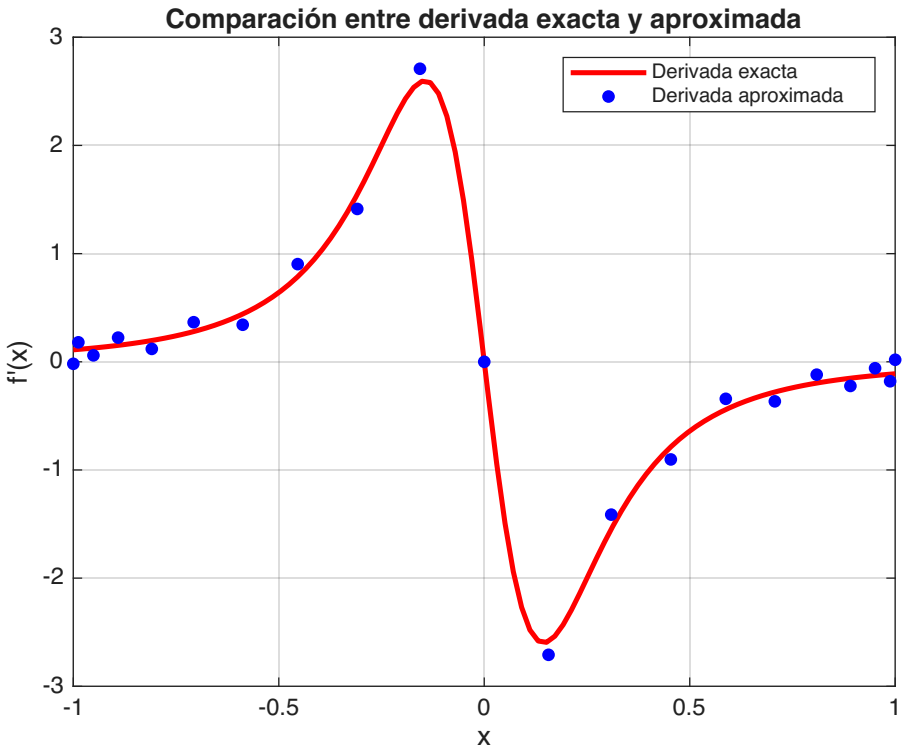
f) (6 puntos) [MATLAB] Implemente una función que reciba n y calcule la matriz D del inciso anterior. Para $n = 20, f(x) = 1/(1 + 16x^2)$, calcule y grafique $f'(x)$, junto a los puntos $(\hat{\boldsymbol{x}}, Df(\hat{\boldsymbol{x}}))$. Cuantifique el error en los nodos de interpolación en esta gráfica.

R/ Se implementó la siguiente función en MATLAB:

```
function D = calcularMatrizD(n)
    %
    %   Función que calcula la matriz D con las derivadas del
    %   polinomio de Lagrange evaluadas en los nodos de
    %   chebyshev.
    %
    %   Inputs:
    %       n - grado de la función.
    %
    %   Outputs:
    %       D - matriz con las derivadas del polinomio de
    %           Lagrange evaluadas sobre los nodos.

    % Calcula los nodos de Chebyshev
    x_chebyshev = cos((0:n)' * pi / n);

    % Calcula los lambdas para todo x nodo de Chebyshev
    lambdas = zeros(n+1, 1);
    for k = 1:n+1
        prod = 1;
        for i = 1:n+1
            if i ~= k
                prod = prod*(x_chebyshev(k) - x_chebyshev(i));
            end
        end
        lambdas(k) = 1 / prod;
    end

    % Inicializa la matriz D
    D = zeros(n+1, n+1);

    % Llena la matriz D en las entradas i!=k
    for i = 1:n+1
        for k = 1:n+1
            if i ~= k
                % Caso i != k
                D(i, k) = lambdas(k) /
                (lambdas(i)*(x_chebyshev(i) - x_chebyshev(k)));
            end
        end
    end

    % Llena la matriz D en las entradas i=k
    for i = 1:n+1
        D(i, i) = -sum(D(i, :));
    end
end
```

Al ejecutar dicha función en $f(x) = 1/(1+x^2)$ con $n = 20$ y graficar los resultados se obtiene:

```
% Ejecución de la función
n = 20;
f = @(x) 1 ./ (1 + 16*x.^2);

% Nodos de Chebyshev
x_chebyshev = cos((0:n)' * pi / n);

% Evaluación de f en los nodos
fx = f(x_chebyshev);

% Calcula la matriz D
D = calcularMatrizD(n);

% Derivada exacta de f
df_exacta = @(x) -32 * x ./ (1 + 16*x.^2).^2;
x_continuo = linspace(-1,1);

% Derivada aproximada
df_aprox = D * fx;
```



Finalmente, se puede cuantificar el error en cada nodo de interpolación de la siguiente manera:

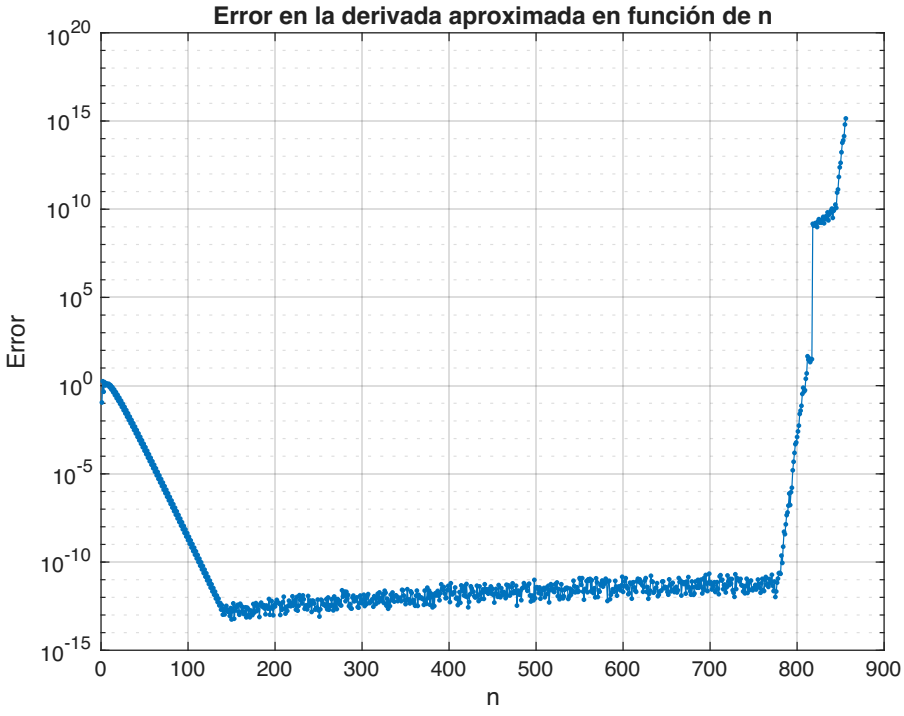
```
error_df = abs(df_exacta(x_chebyshev) - df_aprox);
disp(error_df);
```

```
    0.1294
    0.0654
    0.0676
    0.0715
    0.0776
    0.0864
    0.0991
    0.1162
    0.1345
    0.1237
    0.0000
    0.1237
    0.1345
    0.1162
    0.0991
    0.0864
    0.0776
    0.0715
    0.0676
    0.0654
    0.1294
```

g) (3 puntos) [MATLAB] Grafique $\|Df(\hat{\boldsymbol{x}}) - f'(\hat{\boldsymbol{x}})\|_\infty$ en función de n. ¿Cuál es el menor error posible?

R/ Al graficar $\|Df(\hat{\boldsymbol{x}}) - f'(\hat{\boldsymbol{x}})\|_\infty$ se obtiene:

```
n = 1000;
errores = zeros(1, n);

for k = 1:n
    % Calcula la matriz D para cada k
    D = calcularMatrizD(k);
    x_chebyshev = cos((0:k)' * pi / k);

    % Evalúa la función y la derivada
    fx = f(x_chebyshev);
    df_aprox = D * fx;
    dfx_exacta = df_exacta(x_chebyshev);

    % Norma infinito
    errores(k) = max(abs(df_aprox - dfx_exacta));
end
```



De lo anterior se puede calcular el menor error posible, el cual corresponde a:

```
[val_min, pos_min] = min(errores);
disp(val_min);
```

```
5.5886e-14
```

Y se alcanza cuando n toma el valor de:

```
disp(pos_min);
```

```
150
```

5. Es posible escribir el polinomio de interpolación de Lagrange p en la base de los polinomios de Chebyshev,

$$p(x) = \sum_{k=0}^{n} c_k T_k(x), \quad |x| \leq 1,$$

donde $T_k(x) = \cos(k \arccos x)$. Sabemos que los polinomios de Chebyshev satisfacen la relación de recurrencia

$$T_{k+1}(x) = \alpha_k(x) T_k(x) + \beta_k(x) T_{k-1}(x),$$

para $\alpha_k(x) = 2x$ y $\beta_k(x) = -1$. Deseamos calcular $p(x_0)$ para x_0 dado (asumiendo que lo tenemos expresado en esta base). Para ello, considere el siguiente algoritmo:

Alg.1– Defina la sucesión $\{b_k(x_0)\}_{k=0}^{n+2}$ mediante la recurrencia hacia atrás:

$$\begin{aligned} b_{n+2}(x_0) &= b_{n+1}(x_0) = 0, \\ b_k(x_0) &= c_k + \alpha_k(x_0) b_{k+1}(x_0) + \beta_{k+1}(x_0) b_{k+2}(x_0), \quad k = n, n-1, \ldots, 1. \end{aligned}$$

Se tiene entonces que el valor deseado es

$$p(x_0) = T_0(x_0) c_0 + T_1(x_0) b_1(x_0) + \beta_1(x_0) T_0(x_0) b_2(x_0). \tag{6}$$

a) (2 puntos) Determine el número de operaciones (sumas y multiplicaciones) que se deben realizar en el Algoritmo 1.

R/ Primero, dado que los valores de $\alpha_k(x)$ y $\beta_k(x)$ son conocidos, resulta pertinente sustituirlos, lo que nos simplifica el algoritmo a:

$$\begin{aligned} b_{n+2}(x_0) &= b_{n+1}(x_0) = 0, \\ b_k(x_0) &= c_k + 2x_0 b_{k+1}(x_0) - b_{k+2}(x_0), \quad k = n, n-1, \ldots, 1. \end{aligned}$$

Y para el valor del polinomio:

$$p(x_0) = T_0(x_0) c_0 + T_1(x_0) b_1(x_0) - T_0(x_0) b_2(x_0)$$

Ahora, note que para calcular el valor de $b_k(x_0)$ se necesitan realizar 2 multiplicaciones y 2 sumas, dado que esa recurrencia se lleva a cabo n veces, se necesitan un total de $4n$ operaciones para calcular los $b_k(x_0)$.

Por otro lado, al evaluar el polinomio se están realizando 3 multiplicaciones y 2 sumas, lo que nos da un total de 5 operaciones para realizar esta acción.

Por lo tanto, se necesitan un total de $4n+5$ operaciones para obtener el resultado deseado. Lo importante acá es notar que dicho algoritmo sigue un orden lineal para ejecutarse.

b) (4 puntos) Escriba una función `y0 = evalCheb(c,x0)` en `MATLAB` que calcule $y_0 = p(x_0)$ mediante la fórmula (6), donde la entrada es el vector de coeficientes $\boldsymbol{c}$ y el valor x_0.

```
function y0 = evalCheb(c, x0)
    %
    %   Función que calcula el valor del polinomio evaluado
    %   en x0 de manera recursiva definida por el Algoritmo 1,
    %   mediante los coeficientes de Chebyshev.
    %
    %   Inputs:
    %       c - vector de coeficientes de Chebyshev
    %       x0 - punto donde se evalúa el polinomio p(x)
    %
    %   Outputs:
    %       y0 - valor del polinomio en x0 según el Algoritmo 1
    %

    % Valores iniciales
    n = length(c)-1;
    b_n2 = 0;
    b_n1 = 0;

    % Recurrencia
    for k = n:-1:1
        b_k = c(k+1) + 2 * x0 * b_n1 - b_n2;
        b_n2 = b_n1;
        b_n1 = b_k;
    end

    % Evaluación de p(x0)
    y0 = c(1) + x0 * b_n1 - b_n2;
end
```

c) (3 puntos) Considere el polinomio $p(x) = x^3 - 3x^2 + 1$. Exprese $p(x)$ como combinación lineal de polinomios de Chebyshev

$$p(x) = c_0T_0(x) + c_1T_1(x) + c_2T_2(x) + c_3T_3(x)$$

y verifique el comportamiento del algoritmo para $x_0 = 1$.

R/ Los polinomios de Chebyshev son:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \end{aligned}$$

Despejando x^3 y x^2 se obtiene:

$$x^3 = \frac{T_3(x) + 3T_1(x)}{4},\ x^2 = \frac{T_2(x) + T_0(x)}{2}$$

Sustituyendo en $p(x)$:

$$p(x) = x^3 - 3x^2 + 1 = \frac{T_3(x) + 3T_1(x)}{4} - 3\left(\frac{T_2(x) + T_0(x)}{2}\right) + 1.$$

Agrupando los términos:

$$p(x) = \frac{1}{4}T_3(x) + \frac{3}{4}T_1(x) - \frac{3}{2}T_2(x) - \frac{1}{2}T_0(x).$$

Finalmente, reordenando se tiene que:

$$p(x) = -\frac{1}{2}T_0(x) + \frac{3}{4}T_1(x) - \frac{3}{2}T_2(x) + \frac{1}{4}T_3(x).$$

Ahora, usando el código programado en el inciso anterior:

```
c = [-1/2, 3/4, -3/2, 1/4];
x0 = 1;

% Verificación con el algoritmo
disp(evalCheb(c, x0));
```

```
-1
```

Note que el algoritmo coincide con el valor de $p(x)$ con $x = 1$.

d) (3 puntos) Utilice `c = rand(n,1)`, $n = 10^8$, $x_0 = 0.1$ y reporte el tiempo de ejecución (puede realizar varias corridas y mostrar un tiempo promedio). Compare con el tiempo que requiere `polyval` para evaluar un polinomio del mismo grado. Comente sus resultados.

R/ Se realizaron 10 iteraciones para tomar el promedio en el tiempo de ejecución de ambos algoritmos, a continuación se presentan los resultados:

```
% Valores
n = 10^8;
c = rand(n,1);
x0 = 0.1;

% Número de ejecuciones
num_iteraciones = 10;

% evalCheb
tiempos_cheb = zeros(num_iteraciones, 1);
for i = 1:num_iteraciones
    tic;
    evalCheb(c, x0);
    tiempos_cheb(i) = toc;
end
t_cheb = mean(tiempos_cheb);
disp(t_cheb)
```

```
0.4487
```

```
% polyval
tiempos_polyval = zeros(num_iteraciones, 1);
for i = 1:num_iteraciones
    tic;
    polyval(c, x0);
    tiempos_polyval(i) = toc;
end
t_polyval = mean(tiempos_polyval);
disp(t_polyval)
```

```
1.2700
```

La diferencia en los tiempos de ejecución entre los algoritmos `evalCheb` y `polyval` se debe principalmente a la optimización del primero para trabajar con polinomios representados mediante coeficientes de Chebyshev. Además, el algoritmo de `evalCheb` utiliza una recurrencia eficiente, la cual aprovecha la estructura especial

de estos polinomios para minimizar el número de operaciones. Este método recursivo permite reutilizar cálculos previos, evitando multiplicaciones innecesarias y reduciendo así el costo computacional. Por esta razón, `evalCheb` es más rápido al evaluar polinomios de Chebyshev en comparación con un enfoque más general.

Por otro lado, `polyval` es una función diseñada para evaluar cualquier tipo de polinomio sin asumir propiedades especiales en los coeficientes. Utiliza el método de Horner, que si bien es eficiente para polinomios generales, no está especializado para polinomios de Chebyshev. Esto hace que, aunque polyval sea una herramienta versátil, no sea tan rápida como `evalCheb` en este caso específico, donde la estructura recursiva de los polinomios de Chebyshev ofrece una ventaja clara en términos de rendimiento.