PRACTICA CALIFICADA N°1- CM431A

Dadas las siguientes indicaciones que a continuación se detallan resolver los ejercicios planteados y presentar la respuesta en digital, adjuntando también los archivos de MATLAB.

1 Algunos comandos en MATLAB relacionados con los polinomios

Véase el toolbox *polyfun* de MATLAB para ampliar la información de la tabla 1. Recuérdese que los polinomios se representan como vectores $p = (p(1), \ldots, p(n))$ cuyas componentes son los coeficientes del polinomio en potencias decrecientes de x, es decir $p(x) = x^4 - 3x^3 + 2x + 1$ queda representado por el vector $p = [1 \ \ -3 \ \ 0 \ \ 2 \ \ 1]$.

Comando	Explicación
$y = polyval(p, x)$	y guarda los valores $p(x)$
$z = roots(p)$	z guarda las raíces de p ($p(z) = 0$)
$p = conv(p1, p2)$	p guarda los coeficientes del polinomio $p1(x)p2(x)$
$[q, r] = deconv(p1, p2)$	q, r guardan los coeficientes de los polinomios de la division $p1(x) = q(x)p2(x) + r(x)$
$y = polyder(p)$	y guarda los coeficientes de $p'(x)$
$y = polyint(p, c)$	y guarda los coeficientes de $\int p(t)dt = c + \int_0^x p(t)dt$
$p = polyfit(x, y, n)$	p guarda los coeficientes del polinomio interpolador de los $n+1$ datos $(x, y) = \{(x(i), y(i))\}_{i=1}^{n+1}$

Tabla 1: Algunos comandos de MATLAB relacionados con los polinomios.

Ejercicio 1. Evalúa el polinomio $p(x) = x^4 - 3x^3 + 2x + 1$ en las abscisas equiespaciadas $x_k = -1 + k/4, \quad k = 0, 1, \ldots, 8$.

Ejercicio 2. Calcula los ceros de los polinomios $p(x) = x^3 - 6x^2 + 11x - 6$, $p(x) = (x+1)^7$. ¿Qué ocurre con el segundo polinomio?

Ejercicio 3. Calcula el producto y el cociente de los polinomios $p_1(x) = x^4 - 1$ y $p_2(x) = x^3 - 1$.

Ejercicio 4. Calcula la derivada y una primitiva de los polinomios $p(x) = x^3 + 2x^2 + 3x + 4, p(x) = x^6 - x^2 - 3x$.

Ejercicio 5. Calcula, en potencias de x, el polinomio p de grado tres que interpola los datos $(0, 0), (1, 2), (2, -1), (3, 0)$ y evalúalo en los nodos $x_k = 3k/16, \quad k = 0, 1, \ldots, 16$.

2 El fenómeno de Runge

Consideremos la llamada función de Runge

$$f(x) = \frac{1}{1+x^2}, \quad x \in [-5, 5].$$

Fijemos $N + 1$ nodos equiespaciados

$$x_j = a + jh, h = (b-a)/N, j = 0, \ldots, N, \tag{1}$$

en $[a, b] = [-5, 5]$, así como los correspondientes valores a interpolar $y_j = f(x_j), j = 0, \ldots, N$. Vamos primero a elaborar un programa en MATLAB

function frunge(N)

que, dado N, realice lo siguiente:

PRACTICA CALIFICADA N°1- CM431A

Dadas las siguientes indicaciones que a continuación se detallan resolver los ejercicios planteados y presentar la respuesta en digital, adjuntando también los archivos de MATLAB.

- Construya el polinomio interpolador de Lagrange de f en los nodos x_j en su forma de Newton, utilizando diferencias divididas.
- Evalúe el polinomio y la función en los puntos del intervalo $[-5, 5]$ dados por

$$s_n = -5 + nk, k = 10^{-3}, n = 0, \ldots, 1000, \tag{2}$$

  y realice la correspondiente gráfica conjunta de ambas funciones (que deben distinguirse con diferente trazo).

Para el primer apartado, utilizaremos el programa **difd.m** de diferencias divididas, presentado en este tema.

```
function [y,p]=difd(y,x,t)
% Algoritmo de diferencias divididas y evaluación
% x: nodos de interpolación   y: valores a interpolar
% t: nodos de evaluación del polinomio
n = length(x);
for j = 2 : n
  for k = n : -1 : j
    y(k) = (y(k) - y(k - 1))/(x(k) - x(k - j + 1));
  end
end
b = zeros(size(y)); m = length(t);
for j = 1 : m
  b(n) = y(n);
  for k = n - 1 : -1 : 1
    b(k) = y(k) + (t(j) - x(k)) * b(k + 1);
  end
  p(j) = b(1);
end
```

para los nodos x_j y los valores de la función de Runge, generada en el fichero **f1.m**.

```
function f=f1(x)
% Evaluación de la función de Runge
% x: puntos de evaluación (vector)
% f: valores de la función de Runge en x
f = 1./(1 + x.^2);
end
```

El programa calculará los coeficientes del polinomio en forma de Newton y lo evaluará en los puntos s_n. (Importante: no hay que confundir s_n en (2) con los nodos de interpolación x_j de (1).) La estructura sería

```
a = -5; b = 5; h = (b - a)/N;
x = a : h : b; f = feval('f1', x);
dt = 1e - 03; s = a : dt : b;
[f, p] = difd(f, x, s);
```

PRACTICA CALIFICADA N°1- CM431A

Dadas las siguientes indicaciones que a continuación se detallan resolver los ejercicios planteados y presentar la respuesta en digital, adjuntando también los archivos de MATLAB.

Ahora, los coeficientes del polinomio interpolador de grado N de $f(x)$ en forma de Newton están almacenados en el vector f mientras que el vector p guarda la evaluación del polinomio en las componentes del vector s de componentes (2). Completamos el programa evaluando la función de Runge en s y dibujando función y polinomio interpolador en una gráfica conjunta. El programa completo queda entonces de este modo:

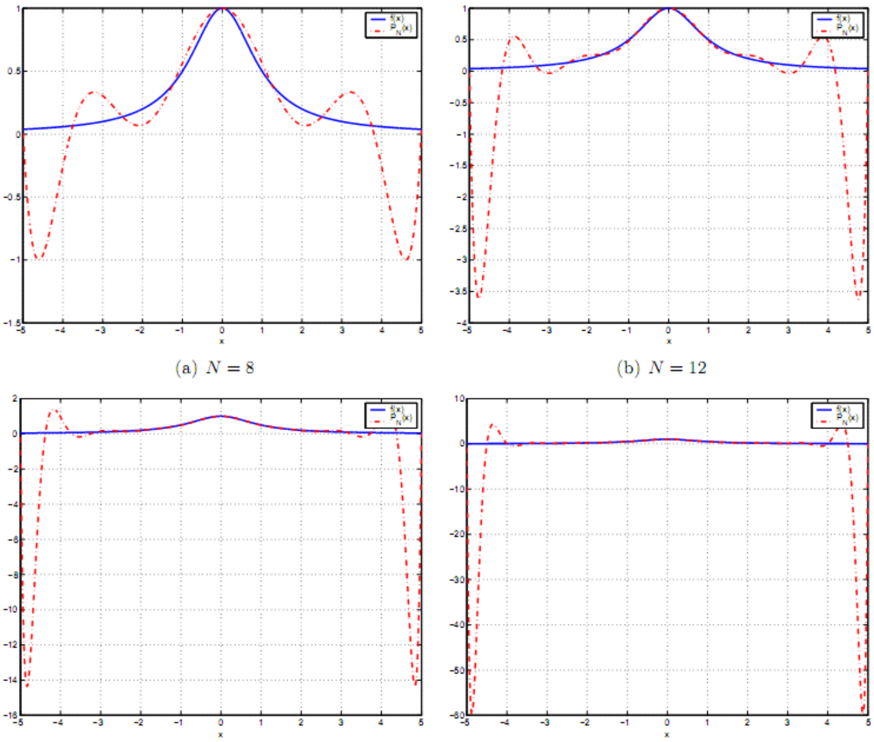
Ejercicio 6. ¿Se puede elaborar una alternativa al programa **frunge1.m** utilizando los comandos **polyfit** y **polyval**?

Ejecutamos el programa para $N = 8, 12, 16, 20$, valores que corresponden, respectivamente, a las figuras 2(a), (b), (c) y (d). En ellas se observar la formación de oscilaciones, mayores cuanto mayor es N, en la gráfica del polinomio interpolador con respecto a la de la función de Runge, especialmente cerca de los extremos del intervalo. Ello indica que, a medida que N crece, el polinomio aproxima peor a la función. Este hecho se conoce como fenómeno de Runge y refleja dos aspectos importantes del problema de interpolación:

- La distribución de los nodos de interpolación es relevante.

```
function frunge1(N)
% Polinomio interpolador de la función de Runge de grado N
% Forma de Newton
a = -5; b = 5; h = (b - a)/N;
x = a : h : b; f = feval('f1', x);
dt = 1e - 03; s = a : dt : b;
[f, p] = difd(f, x, s);
fs = feval('f1', s);
plot(s, fs, 'b-', 'Linewidth', 2)
hold on
plot(s, p, 'r - .', 'Linewidth', 2)
xlabel('x')
legend('f(x)', 'P_N(x)')
grid on
end
```

Dadas las siguientes indicaciones que a continuación se detallan resolver los ejercicios planteados y presentar la respuesta en digital, adjuntando también los archivos de MATLAB.



(a) $N = 8$ (b) $N = 12$

(c) $N = 16$ (d) $N = 20$

Figura 1: Interpolación de Lagrange de la función de Runge en forma de Newton y con los nodos (1).

- Aumentar el grado del polinomio interpolador no siempre mejora la aproximación. Los polinomios de grado alto suelen ser largos de construir y además presentar oscilaciones. Observemos que para un polinomio $P_N(x)$, su derivada segunda $P_N''(x)$ es un polinomio de grado $N-2$. Si las $N-2$ raíces de $P_N''(x)$ son reales, entonces $P_N(x)$ tendrá en general $N-2$ cambios de curvatura, alternando concavidad y convexidad; así, en general, el número de oscilaciones de un polinomio crece con su grado.

3 Nodos de Chebyshev

Tratamos ahora de evitar el fenómeno de Runge modificando el programa **frunge1.m** con respecto a los dos puntos anteriores. En primer lugar, podemos mejorar la distribución de los nodos. Recordemos que la estimación del error de interpolación establece que si $f : [a,b] \to \mathbb{R}$ es de clase C^N en $[a,b]$ y su derivada $f^{N+1)}$ existe en (a,b), entonces, para cada $x \in [a,b]$,

$$|f(x) - P_N(x)| \leq \frac{|x-x_0|\cdots|x-x_N|}{(N+1)!} \max_{\xi\in[a,b]} |f^{N+1)}(\xi)|$$

De esta manera, la cota no sólo depende de la regularidad de f sino también del tamaño del factor $|x-x_0|\cdots|x-x_N|$. Puede comprobarse entonces que la función $F(x_0,\ldots,x_N) = |x-x_0|\cdots|x-x_N|$, para $x \in [a,b]$ fijo, alcanza su valor mínimo cuando

$$x_j = x_j^* = \frac{a+b}{2} - \frac{b-a}{2}\cos\left(\frac{\pi j}{N}\right), \quad j = 0,\ldots,N \tag{3}$$

Los valores (3) se llaman nodos de Chebyshev en $[a,b]$. Las figuras 3(a), (b) y (c) muestran la distribución de estos nodos para el intervalo $[-1,1]$ y diferentes valores de N. Tomando (3) como nodos de interpolación, se puede comprobar que si la función a interpolar es diferenciable en $[a,b]$, entonces el polinomio interpolador de grado N converge en norma del máximo en $[a,b]$ a la función a medida que N crece.

PRACTICA CALIFICADA N°1- CM431A

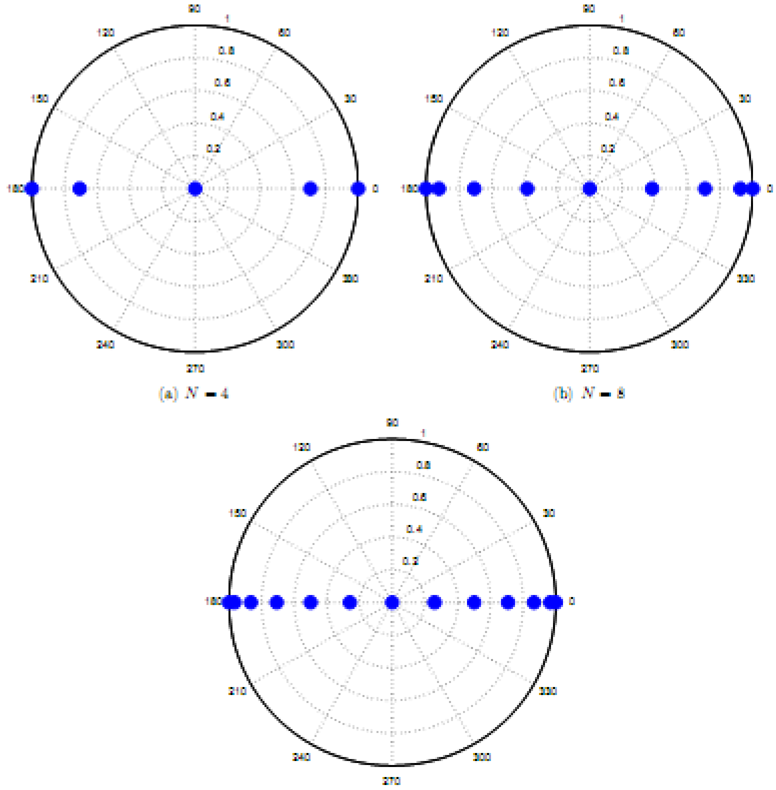
Dadas las siguientes indicaciones que a continuación se detallan resolver los ejercicios planteados y presentar la respuesta en digital, adjuntando también los archivos de MATLAB.

Modificamos el programa **frunge1.m** incorporando los nuevos nodos de interpolación.

```
function frunge2(N)
% Polinomio interpolador de la función de Runge
% de grado N
% Nodos de Chebyshev
a = -5; b = 5; h = pi/N;
y = 0 : h : pi; xm = - cos(y);
x = (a + b)/2 + ((b - a)/2) * xm;
f = feval('f1', x);
dt = 1e - 03; s = a : dt : b;
[f, p] = difd(f, x, s);
fs = feval('f1', s);
plot(s, fs, 'b-', 'Linewidth', 2)
hold on
plot(s, p, 'r-.', 'Linewidth', 2)
xlabel('x')
legend('f(x)', 'P_N(x)')
grid on
end
```

PRACTICA CALIFICADA N°1- CM431A

Dadas las siguientes indicaciones que a continuación se detallan resolver los ejercicios planteados y presentar la respuesta en digital, adjuntando también los archivos de MATLAB.



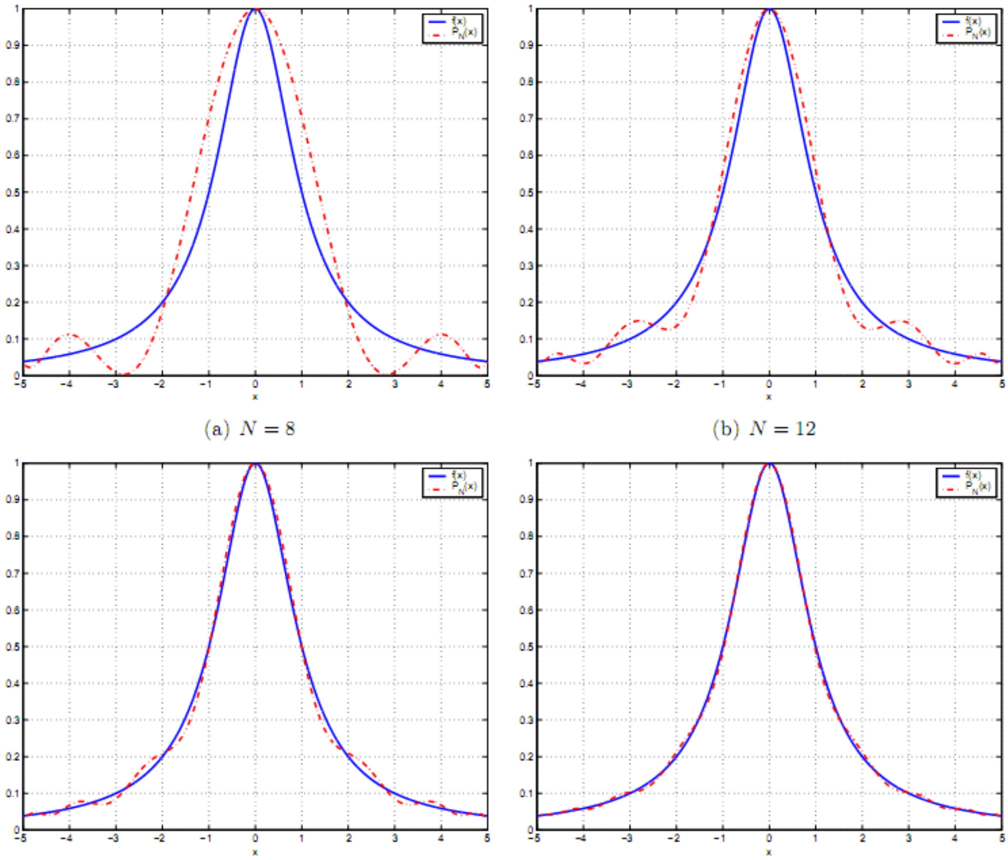
(a) $N = 4$ (b) $N = 8$

(c) $N = 12$

Figura 2: Nodos de Chebyshev (3).

PRACTICA CALIFICADA N°1- CM431A

Dadas las siguientes indicaciones que a continuación se detallan resolver los ejercicios planteados y presentar la respuesta en digital, adjuntando también los archivos de MATLAB.



(a) $N = 8$ (b) $N = 12$

(c) $N = 16$ (d) $N = 20$

Figura 3: Interpolación de Lagrange de la función de Runge en forma de Newton y con los nodos (3).

Los resultados para $N = 8, 12, 16, 20$ se muestran, respectivamente, en las figuras 3(a), (b), (c) y (d). Puede observarse la desaparición progresiva de las oscilaciones y la mejor aproximación del polinomio cuando N crece.

Ejercicio 7. Modifica el programa anterior, utilizando los nodos de interpolación

$$x_j = \frac{a+b}{2} - \frac{b-a}{2}\cos\left(\frac{2j+1}{N+1}\frac{\pi}{2}\right), \quad j = 0, \dots, N.$$

Ejecuta el nuevo programa con $N = 8, 12, 16, 20$ y comenta los resultados, comparándolos con los de las figuras 2 y 3.

Ejercicio 8. Modifica los programas **frunge1.m** y **frunge2.m** para incorporar:

- Una medición del tiempo de ejecución del programa.

- El cálculo de los errores, en norma del máximo, entre los valores de la función y el polinomio interpolador evaluados en las componentes del vector (2).

Ejecuta las modificaciones con $N = 8, 12, 16, 20$ y completa las tablas siguientes, comparando los resultados.

frunge1.m			frunge2.m		
N	CPU(N)	$\|\|f - P_N\|\|_\infty$	N	CPU(N)	$\|\|f - P_N\|\|_\infty$
8			8		
12			12		
16			16		
20			20		

PRACTICA CALIFICADA N°1- CM431A

Dadas las siguientes indicaciones que a continuación se detallan resolver los ejercicios planteados y presentar la respuesta en digital, adjuntando también los archivos de MATLAB.

4 Interpolación a trozos

En referencia al segundo punto, si se mantienen los nodos iniciales de interpolación (1), se pueden evitar las oscilaciones con la llamada interpolación compuesta o a trozos. Se trata de dividir el intervalo $[a, b]$ en subintervalos e interpolar en ellos con polinomios de grado bajo. (Si los polinomios tienen grado uno, se habla de interpolación lineal; si son de grado dos, interpolación cuadrática, etc. Por supuesto, no es necesario interpolar siempre con el mismo grado.) Ilustramos la interpolación a trozos en nuestro ejemplo con el caso cuadrático. Supongamos que N es par, $N = 2M, M > 1$. Dividimos el intervalo $[-5, 5]$ en M subintervalos, con tres nodos cada uno. En cada subintervalo interpolamos la función de Runge con el polinomio de grado dos correspondiente a los tres nodos contenidos en él. Para elaborar la implementación en MATLAB, podemos hacer uso de la estrategia anterior, pero aplicada a cada subintervalo.

```
function frunge3(N)
% Interpolación cuadrática a trozos de la función de Runge
% Se supone N = 2M.
a = -5; b = 5; h = (b - a)/N; M = N/2;
x = a : h : b; f = feval('f1', x);
for j = 1 : M
y = [x(2 * j - 1) x(2 * j) x(2 * j + 1)]
faux = [f(2 * j - 1) f(2 * j) f(2 * j + 1)]
dt = 1e - 03; s = x(2 * j - 1) : dt : x(2 * j + 1);
[faux, p] = difd(faux, y, s);
fs = feval('f1', s);
plot(s, fs, 'b-', 'Linewidth', 2)
hold on
plot(s, p, 'r - .', 'Linewidth', 2)
drawnow
end
xlabel('x')
legend('f(x)', 'P_N(x)')
grid on
end
```

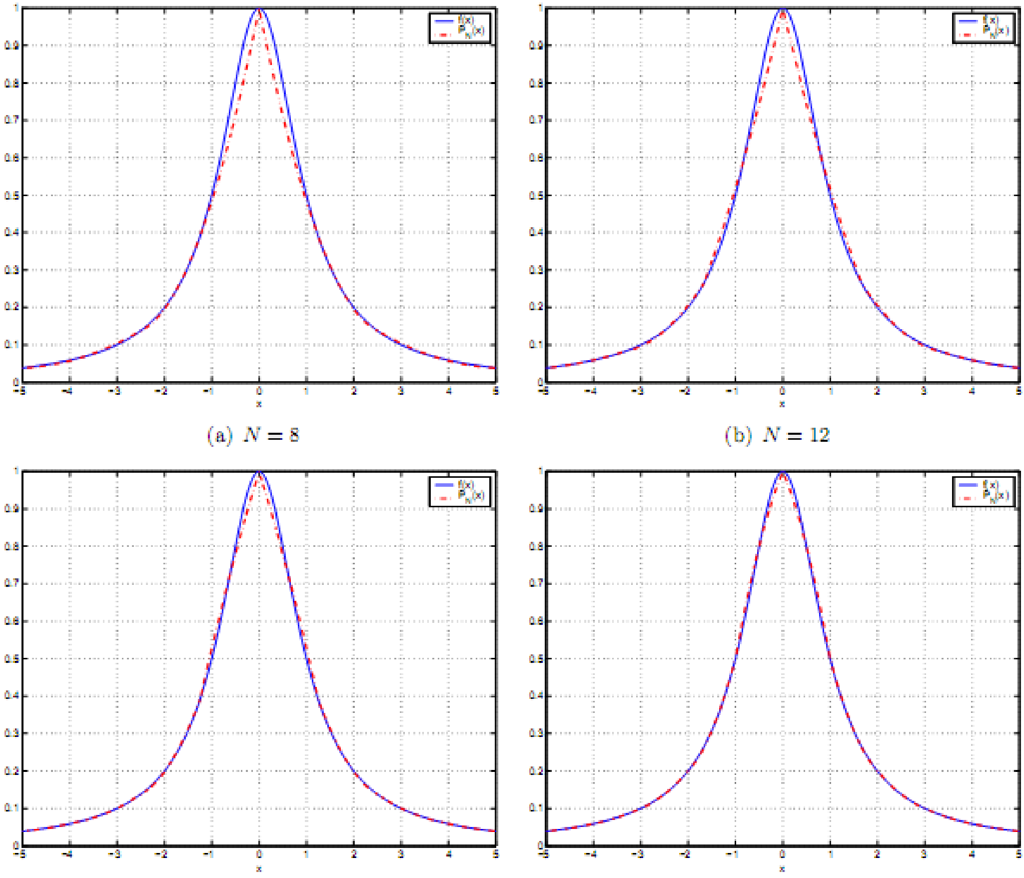
La ejecución de **frunge3.m** con $N = 8, 12, 16, 20$ se muestra, respectivamente, en las figuras 4(a), (b), (c) y (d).

La interpolación a trozos es también necesaria cuando se sabe que la función a interpolar presenta un comportamiento diferente según la región del intervalo (diferente regularidad, por ejemplo).

Ejercicio 9. Siguiendo la estrategia del programa **frunge3.m**, elabora un programa en MATLAB que implemente la interpolación lineal a trozos para la función de Runge en $[-5, 5]$ con nodos equiespaciados. (En este caso no es necesario suponer que N es par, ¿por qué?) Ejecuta el programa con $N = 8, 12, 16, 20$ y compara los resultados con los de **frunge1.m** y **frunge3.m**. ¿Cómo se implementaría una interpolación cúbica a trozos?

PRACTICA CALIFICADA N°1- CM431A

Dadas las siguientes indicaciones que a continuación se detallan resolver los ejercicios planteados y presentar la respuesta en digital, adjuntando también los archivos de MATLAB.



(a) $N = 8$ (b) $N = 12$

(c) $N = 16$ (d) $N = 20$

Figura 4: Interpolación cuadrática a trozos de la función de Runge. Forma de Newton y con los nodos (1).

Ejercicio 10

Sea $f \in \mathscr{C}^n[a,b]$ y $x_0,\ldots,x_n \in [a,b]$. Entonces $\exists \xi \in [a,b]$ tal que

$$f[x_0,\ldots,x_n] = \frac{f^{(n)}(\xi)}{n!}.$$

Demuestre que:

Si $f \in \mathscr{C}^{n+1}[a,b]$ entonces

$$E(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\Pi(x). \quad \text{donde} \quad \Pi(x) = \prod_{i=0}^{n}(x - x_i).$$

Presentación: Viernes 10 de Abril 2015 16-18-R1-215A.