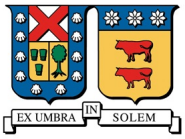


UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Tarea N°3

Interpolación e Integración Numérica

Métodos Numéricos

MEC270 - 2022

Departamento de Ing. Mecánica UTFSM

Martín Achondo Mercado

Rol: 201860005-9

Catalina Santibáñez Mercado

Rol: 201804108-4

Profesor: Romain Jean-Michel Gers

11 de Julio 2022

Resumen

En el presente trabajo se intentó resolver 2 problemas para comparar métodos numéricos. El primero consistió en interpolar las funciones $\sin(x)$ y $\frac{1}{1+x^2}$ por medio de polinomios de Lagrange, diferencias divididas de Newton, interpolación con nodos de Tchebyshev y Splines Lineales. De esta forma, se obtuvo que la primera función genera una buena representación con polinomios de orden igual o mayor a 6. Por otra parte, la segunda función presenta oscilaciones en los bordes del intervalo conocidas como fenómeno de Runge. Esta característica se vio minimizada con la utilización de nodos de Tchebyshev. Por último, la interpolación por Splines Lineales obtuvo una representación bastante precisa de ambas funciones para una cantidad mayor a 34 nodos. Para el segundo problema, se comparó el error al calcular la integral $\int_0^1 e^{-x^2} dx$ por medio de los siguientes métodos: rectángulos, punto medio, trapecios, Simpson, trapecios corregidos y Gauss Legendre. Estos métodos nos entregan buenas aproximaciones del valor de la integral original, dependiendo de la cantidad de nodos utilizados. Por lo tanto se utilizó una cantidad de nodos de 14, 22 y 33 para demostrar que el error obtenido será menor al aumentar la cantidad de nodos para los métodos de Newton-Cotes, y en el caso particular de Gauss-Legendre se utilizó una cantidad de nodos de 1, 2, 3 ,4 , 5 y 6 para demostrar que es un método eficiente, ya que necesita menos cantidad de nodos para obtener un menor error.

Índice

1. Introducción

1.1. Problema 1: Interpolación

En este problema se pide interpolar las funciones:

$$\sin(x) \quad x \in [0, 3\pi] \tag{1}$$

$$\frac{1}{1+x^2} \quad x \in [-5, 5] \tag{2}$$

con polinomios de Lagrange de orden 2 a 8. Con esto:

- Se pide comparar la velocidad de resolución al utilizar el método de Lagrange con el método de diferencias divididas de Newton
- Además, se pide utilizar los polinomios de Tchebyshev para así comparar el polinomio interpolante y el error de interpolación.

Por último, se pide interpolar ambas funciones con Splines Lineales y así encontrar el número mínimo de puntos para obtener una representación precisa.

1.2. Problema 2: Integración Numérica

La finalidad de este problema es comparar el error en la integración numérica de la función

$$\int_0^1 e^{-x^2} dx \tag{3}$$

por el método de:

- los rectángulos
- del punto medio
- de los trapecios
- de Simpson
- de los trapecios corregidos
- de Gauss-Legendre

Los cálculos serán programados en matlab y subdivididos por métodos.

2. Metodología

2.1. Problema 1: Interpolación

Para este trabajo se interpolarán las funciones:

$$f(x) = \sin(x) \quad x \in [0, 3\pi] \tag{4}$$

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

Las cuales como se notan están bien definidas en un intervalo acotado. Ambas funciones serán interpoladas por polinomios en bases de Lagrange y Newton, de tal forma que se pueda comparar el tiempo de cálculo. Además, se comparará el error de interpolación utilizando ahora nodos de Tchebyshev. Por último ambas funciones serán interpoladas por Splines Lineales para así verificar la cantidad de nodos necesarios para una representación precisa.

En un problema de interpolación se tienen $n+1$ pares reales $\{(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)\}$, y el objetivo es buscar una función g de tal forma que:

$$g(x_j) = y_j \quad j = 0, 1, \cdots, n \tag{6}$$

Los métodos se diferencian en la forma que adopta esta función interpolante g. Es preciso detallar que si esta función es un polinomio de grado n, $p_n(x)$, este es único independiente de la base utilizada. Así, p_n puede ser escrito como:

$$p_n(x) = \sum_{k=0}^{n} \lambda_k \ell_k(x) \tag{7}$$

Para el trabajo se utilizará la base de Lagrange y la base de Newton, las cuales se detallarán en la siguiente sección. Además, la función interpolante puede estar definida a trozos (splines), de la cual en este caso se utilizarán los splines lineales.

2.1.1. Interpolación con Base de Lagrange

Teniendo $n+1$ pares (x_j, y_j), se busca obtener un polinomio interpolante de grado n:

$$p_n(x) = \sum_{j=0}^{n} y_j L_j(x) \tag{8}$$

En donde el conjunto $\{L_0, L_1, \cdots, L_n\}$ es la base de Lagrange para $\mathbf{P}_n$, la cual se calcula como:

$$L_j(x) = \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k} \tag{9}$$

Notar que se cumple:

$$L_j(x_k) = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases} \tag{10}$$

2.1.2. Interpolación con Base de Newton

Teniendo $n+1$ pares (x_j, y_j), se busca obtener un polinomio interpolante de grado n:

$$p_n(x) = \sum_{j=0}^{n} \lambda_j N_j(x) \tag{11}$$

En donde el conjunto $\{N_0, N_1, \cdots, N_n\}$ es la base de Newton para $\mathbf{P}_n$, la cual se calcula como:

$$\begin{aligned} N_0(x) &= 1 \\ N_j(x) &= \prod_{k=0}^{j-1} (x - x_k) \end{aligned} \tag{12}$$

Para el cálculo de los coeficientes λ_j, se utiliza las diferencias divididas de Newton, dadas por:

$$\begin{aligned} y[m] &:= y_j \\ y[i, \cdots, m] &:= \frac{y[i+1, \cdots, m] - y[i, \cdots, m-1]}{x_m - x_i} \end{aligned} \tag{13}$$

De esta manera los coeficientes de la base de Newton λ_j están dados por:

$$\lambda_j = y[0, \cdots, j] \tag{14}$$

2.1.3. Interpolación con Nodos de Tchebyshev

La interpolación utilizando nodos de Tchebyshev es un método para minimizar el error de interpolación que puede surgir por utilizar nodos arbitrarios o equiespaciados. De esta forma, los nodos para la interpolación son impuestos. Los nodos antes mencionados, para un intervalo $[a, b]$ pueden ser obtenidos de la siguiente manera:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(2(n-i)+1)\pi}{2n+2}\right) \tag{15}$$

Notar entonces que el polinomio interpolante debe cumplir que:

$$p_n(x_i) = y_i \quad i = 0, 1, \cdots, n \tag{16}$$

Con x_i nodos de Tchebyshev. Por último, vale destacar que se puede cualquier base dicha anteriormente bajo estos nodos, pero en este caso se utilizará la base de Lagrange.

2.1.4. Interpolación por Splines Lineales

Teniendo $n+1$ pares (x_j, y_j), se busca obtener n splines lineales que interpolan los datos a trozos. De esta manera, los n splines se definen de la siguiente manera:

$$\begin{aligned} s_0(x) &= y_0 + m_0(x - x_0) \quad x \in [x_0, x_1] \\ s_1(x) &= y_1 + m_1(x - x_1) \quad x \in [x_1, x_2] \\ &\vdots \\ s_{n-1}(x) &= y_{n-1} + m_{n-1}(x - x_{n-1}) \quad x \in [x_{n-1}, x_n] \end{aligned} \tag{17}$$

En donde la pendiente de cada spline m_j se puede calcular como:

$$m_j = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} \tag{18}$$

Con esto cada spline lineal queda definido y pueden ser encontrados para interpolar las funciones presentadas.

2.1.5. Error en Interpolación

Para cada interpolación presentada se graficará el error de interpolación como la diferencia entre la función real y la función interpolante, dado por:

$$r(x) = f(x) - g(x) \tag{19}$$

En donde f es la función a interpolar y g la función interpolante. Es importante notar que también se trabajará con el valor máximo de r para identificar el error global, dado por:

$$R = \max_x |r(x)| \tag{20}$$

De esta manera se podrán comparar los métodos de manera mas sencilla.

2.2. Integración Numérica

En este trabajo se integrará la siguiente función:

$$f(x) = e^{-x^2} \quad x \in [0, 1]$$

Donde la integración numérica puede ser representada por:

$$\int_a^b f(x)dx \tag{21}$$

Siendo $f(x)$ una función integrable en el intervalo acotado $[a, b]$. Matemáticamente el cálculo de integrales (las cuales son asociadas con el calculo de áreas) se denomina cuadratura, donde la integral definida en una suma de integrales sobre sub-intervalos, es decir, sean $a = x1 < x2 < ... < xn = b$ un conjunto de n puntos en el intervalo [a, b] y la integral descompuesta como:

$$\int_a^b f(x)dx = \int_{x_1}^{x_2} f(x)dx + \int_{x_2}^{x_3} f(x)dx \cdots + \int_{x_{n-1}}^{x_n} f(x)dx \tag{22}$$

En este problema se nos solicita trabajar con los métodos de Cuadraturas de Newton-Cotes y Cuadratura de Gauss, que serán detallados a continuación.

El método de Cuadraturas de Newton-Cotes utiliza la interpolación para calcular el área bajo de la curva de la integral, donde el grado del polinomio dependerá de la cantidad de nodos a utilizar. Ahora bien, supondremos una base del rectángulo constante h y una altura variable.

2.2.1. Método del rectángulo

Corresponde al método mas simple de interpolación, donde se sustituye la función por un polinomio constante. Por lo tanto, la altura puede depender del punto del intervalo que se encuentre a la izquierda, como a la derecha. Siendo representada por:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n-1}(x_{i+1}-x_i)f\left(x_i\right) = h\sum_{i=1}^{n-1} f\left(x_i\right) \quad (Lado\, Izquierdo) \tag{23}$$

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n-1}(x_{i+1}-x_i)f\left(x_{i+1}\right) = h\sum_{i=1}^{n-1} f\left(x_{i+1}\right) \quad (Lado\, derecho) \tag{24}$$

2.2.2. Método del Punto medio

Este método es similar al anterior, la diferencia es que la altura depende del punto medio de dicho intervalo. Siendo representada por:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n-1}(x_{i+1}-x_i)f\left(\frac{x_{i+1}+x_i}{2}\right) = h\sum_{i=1}^{n-1} f\left(\frac{x_{i+1}+x_i}{2}\right) \tag{25}$$

2.2.3. Método del trapecio

Este método sustituye la función por un polinomio lineal de grado 1, donde el trapecio se obtiene al unir dos puntos $(x_i, f(x_i))$ y $(x_{i+1}, f(x_{i+1}))$ mediante una recta. Siendo representada por:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n-1}(x_{i+1}-x_i)\frac{f(x_{i+1})+f(x_i)}{2} = h\sum_{i=1}^{n-1}\frac{f(x_{i+1})+f(x_i)}{2} \tag{26}$$

2.2.4. Método de Simpson

Consiste en usar polinomios de grado superior para unir los puntos del intervalo, es decir, el polinomio sera de grado n-1, donde n corresponde a la cantidad de nodos.

El metodo de simpson esta compuesto por: Simpson 1/3 y Simpson 3/8. Los cuales se basan en las siguientes expresiones:

El método de Simpson 1/3 es utilizado para números par de nodos.

$$\int_a^b f(x)dx \approx \frac{h}{3}(f(x_0)+4\sum_{i=1,3,5}^{n-1} f(x_i)+2\sum_{i=2,4,6}^{n-2} f(x_j)+f(x_n)) \tag{27}$$

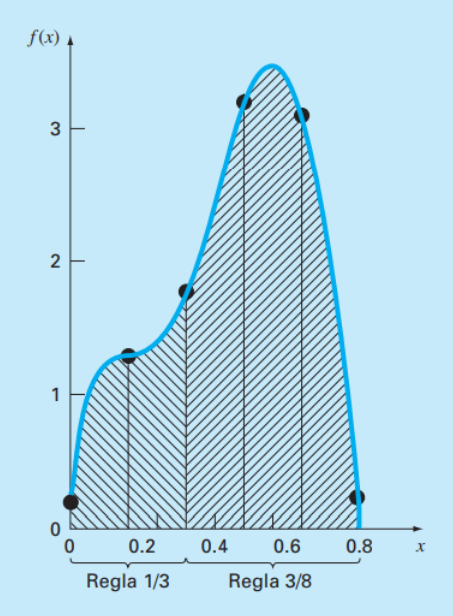


Figura 1: Método de Simpson

El método de simpson 3/8 es utilizado para números impar de nodos. Además este método utiliza polinomios de lagrange de tercer orden.

$$\int_a^b f(x)dx \approx \frac{h}{8}(f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)) \tag{28}$$

Por lo tanto el método de Simpson permite ingresar nodos pares e impares, ya que sus sistema funciona de la siguiente manera: Se verifica si el numero es impar o par, si es impar se toma los últimos 4 números y se aplica el método de simpson 3/8, los demás números son calculados por el método de simpson 1/3, ya que se obtiene un numero de nodos pares. Si se obtiene un numero par se aplica directamente el método de Simpson 1/3.

2.2.5. Método del trapecio Corregido

Es el mismo que el método del Trapecio, sólo con una pequeña corrección, la cual se puede ver claramente a continuación:

$$\int_a^b f(x)dx \approx \underbrace{\sum_{i=1}^{n-1}(x_{i+1} - x_i)\frac{f(x_{i+1}) + f(x_i)}{2}}_{\text{Trapecio}} + \underbrace{\sum_{i=1}^{n-1}(x_{i+1} - x_i)^2\frac{f^{(1)}(x_i) - f^{(1)}(x_{i+1})}{12}}_{\text{Corrección}} \tag{29}$$

Donde f^1 corresponde a la primera derivada de $f(x)$. También puede ser representada como:

$$\int_a^b f(x)dx \approx h[\frac{1}{2}(f(x_0)+f(x_n)) + \sum_{i=1}^{n-1} f_i + h_i^2 \frac{f^{(1)}(a) - f^{(1)}(b)}{12} \tag{30}$$

2.2.6. Método de Gauss-Legendre

Pertenece a las cuadratura Gaussiana, la cual selecciona las abscisas concretas basándose en las propiedades de unos polinomios ortogonales que aumentan el grado de la aproximación.

La fórmula de cuadratura de Gauss-Legendre es la más empleada en la integración en el método de los elementos finitos. Se basa en los polinomios de Legendre, que son ortogonales en el intervalo [1, 1] con respecto a la función peso $W(x) = 1$. Por otro lado, los pesos como las abscisas (que son las raíces de los polinomios de Legendre) pueden ser calculados sistemáticamente, sin embargo, se utilizara los valores tabulados.

Este método puede ser expresado como:

$$\int_a^b f(x)dx \approx \frac{(b-a)}{2} \sum_{i=1}^{n} w_i f\left(\frac{a+b}{2} + \frac{b-a}{2} z_i\right) \tag{31}$$

donde $[a, b]$ corresponde al intervalo de integración, w_i los pesos de la fórmula de cuadratura de Gauss-Legendre y z_i son las abscisas.

Tabla 1: Tabla de pesos y abscisas para $n = 4$

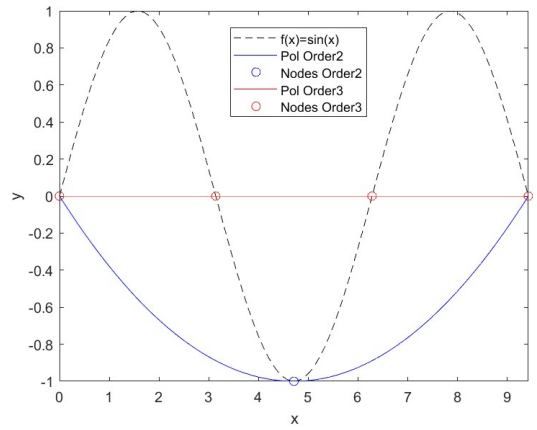
i	w_i	z_i
1	0.3478548	–0.861136312
2	0.6521452	–0.339981044
3	0.6521452	0.339981044
4	0.3478548	0.861136312

3. Resultados

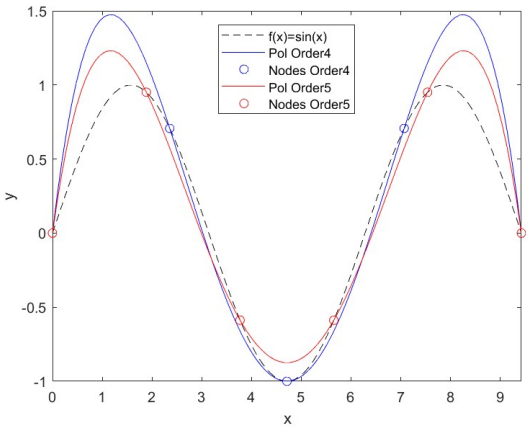
3.1. Problema 1: Interpolación

Se presentan los resultados para las interpolaciones de ambas funciones. Además, se graficará la diferencia entre el polinomio interpolante y la función a interpolar.

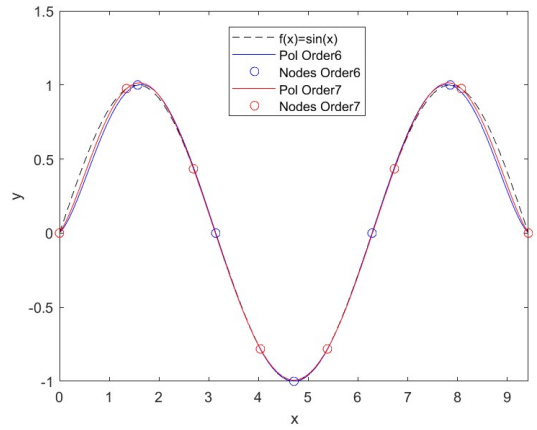
3.1.1. Interolación de Lagrange para $\sin(x)$



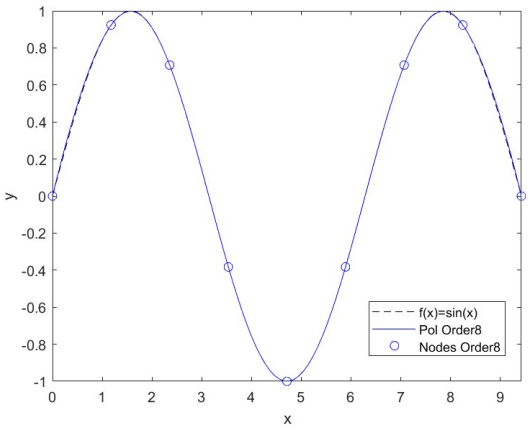
(a) Orden 2 y 3



(b) Orden 4 y 5



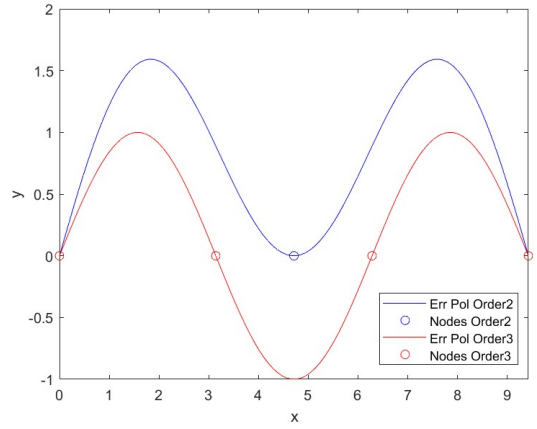
(c) Orden 6 y 7



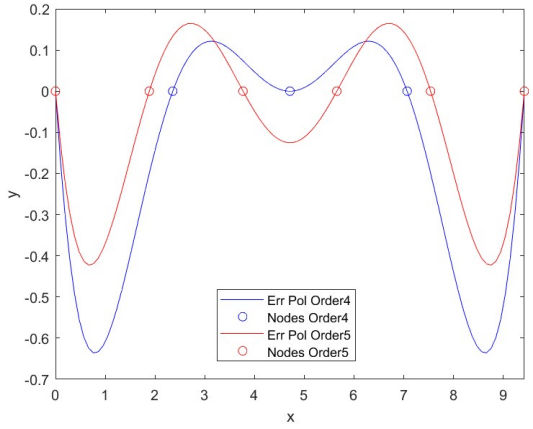
(d) Orden 8

Figura 2: Interpolación de Lagrange de orden 2 a 8 para $\sin(x)$

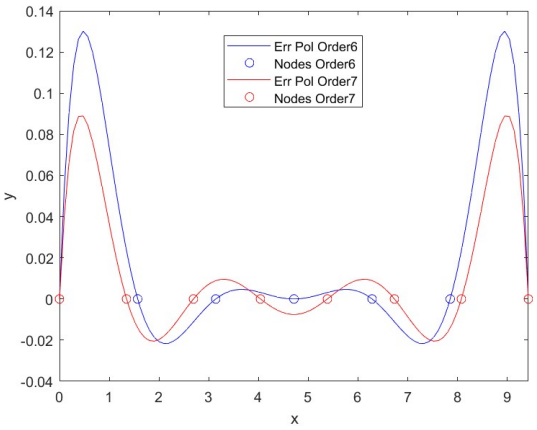
3.1.2. Error en Interolación de Lagrange para $\sin(x)$



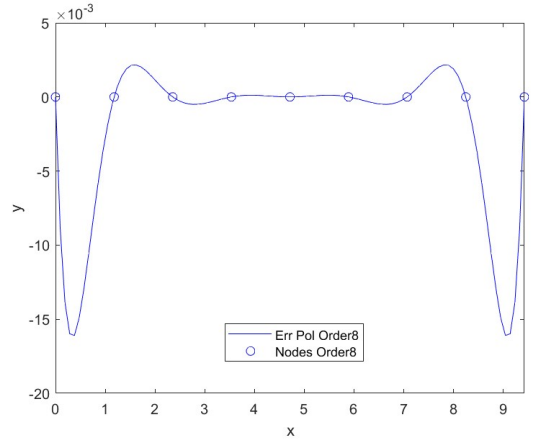
(a) Orden 2 y 3



(b) Orden 4 y 5



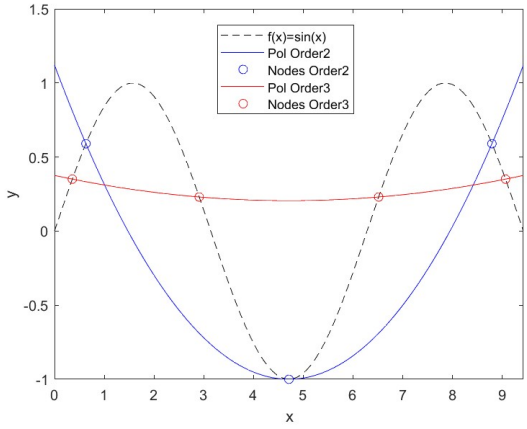
(c) Orden 6 y 7



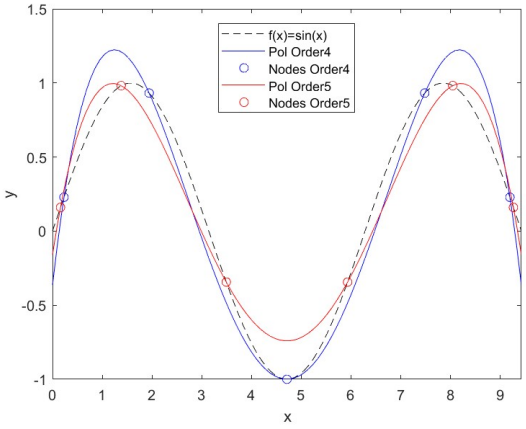
(d) Orden 8

Figura 3: Error en nterpolación de Lagrange de orden 2 a 8 para $\sin(x)$

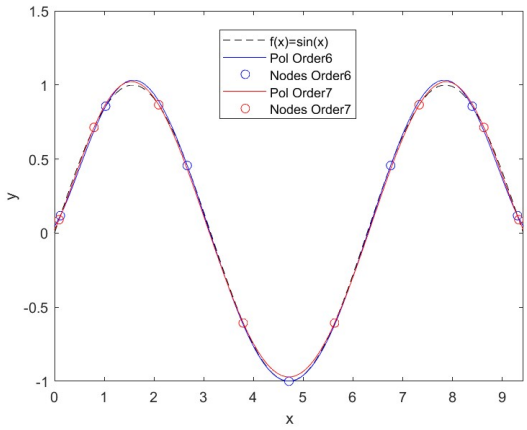
3.1.3. Interolación de Lagrange para $\sin(x)$ con nodos de Tchebyshev



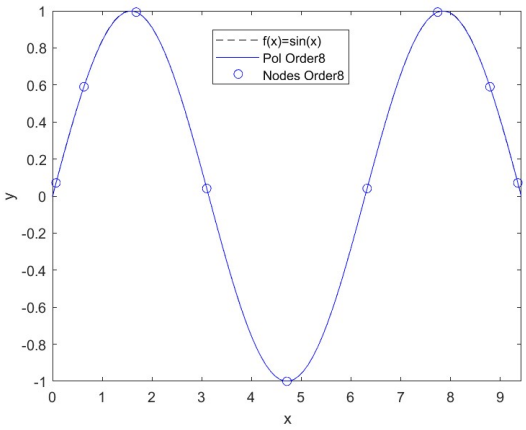
(a) Orden 2 y 3



(b) Orden 4 y 5



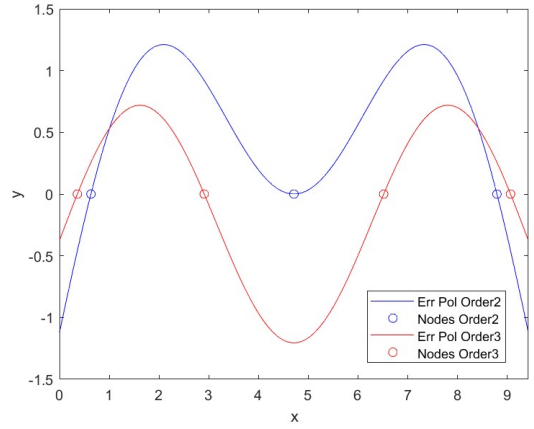
(c) Orden 6 y 7



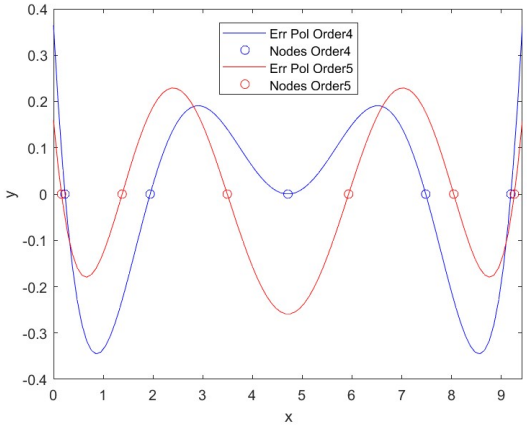
(d) Orden 8

Figura 4: Interpolación de Lagrange con nodos de Tchebyshev de orden 2 a 8 para $\sin(x)$

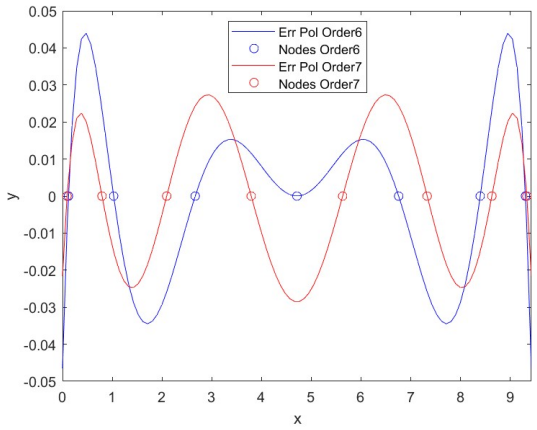
3.1.4. Error en Interolación de Lagrange para $\sin(x)$ con nodos de Tchebyshev



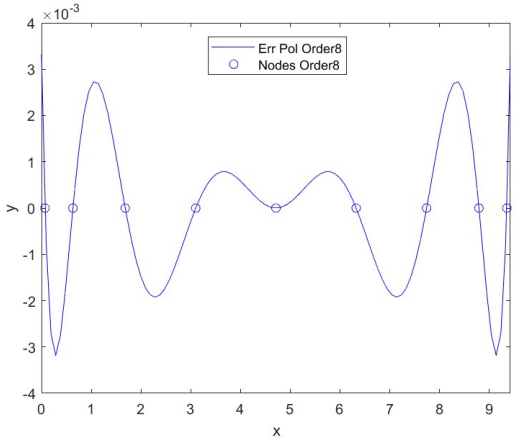
(a) Orden 2 y 3



(b) Orden 4 y 5



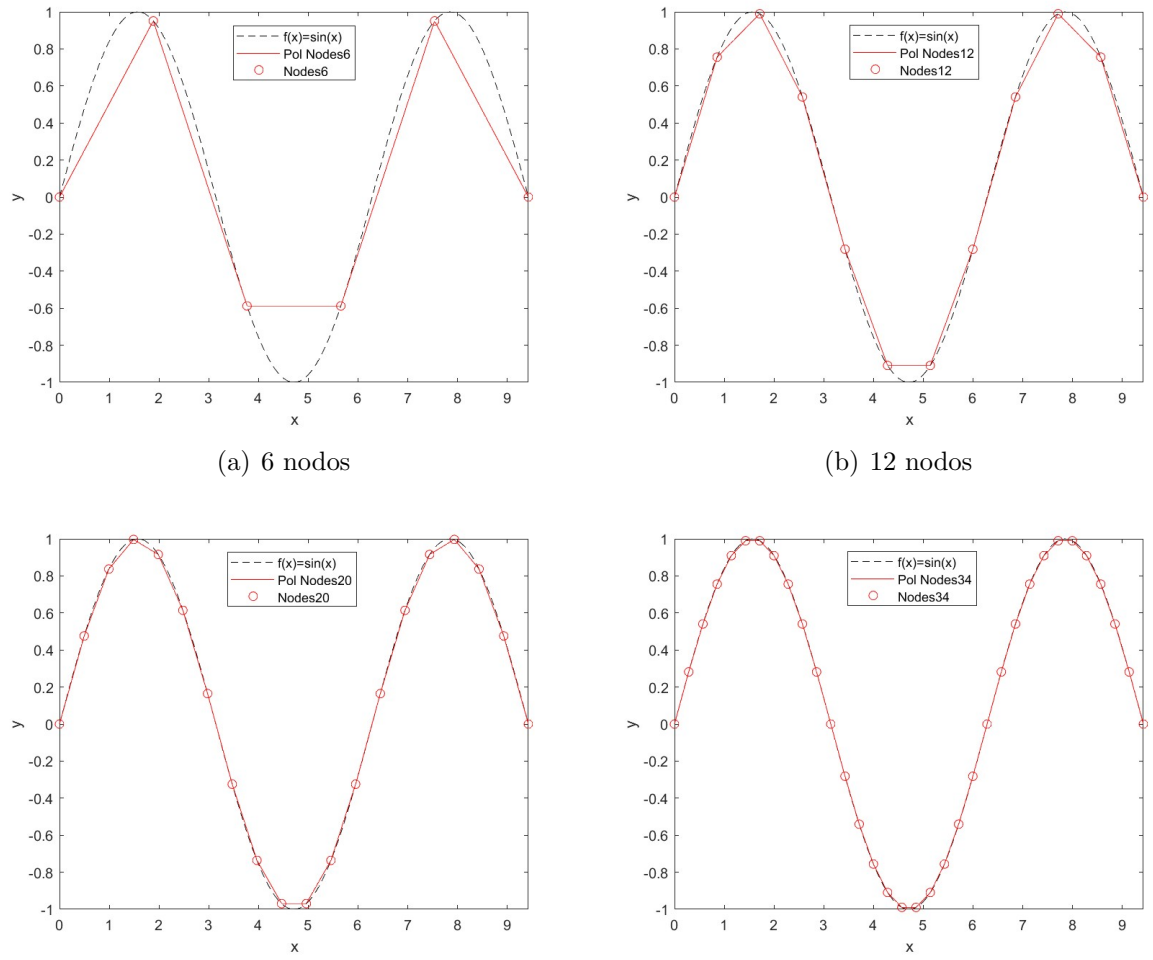
(c) Orden 6 y 7



(d) Orden 8

Figura 5: Error en nterpolación de Lagrange con nodos de Tchebyshev de orden 2 a 8 para $\sin(x)$

3.1.5. Interolación por Splines Lineales para $\sin(x)$



(a) 6 nodos

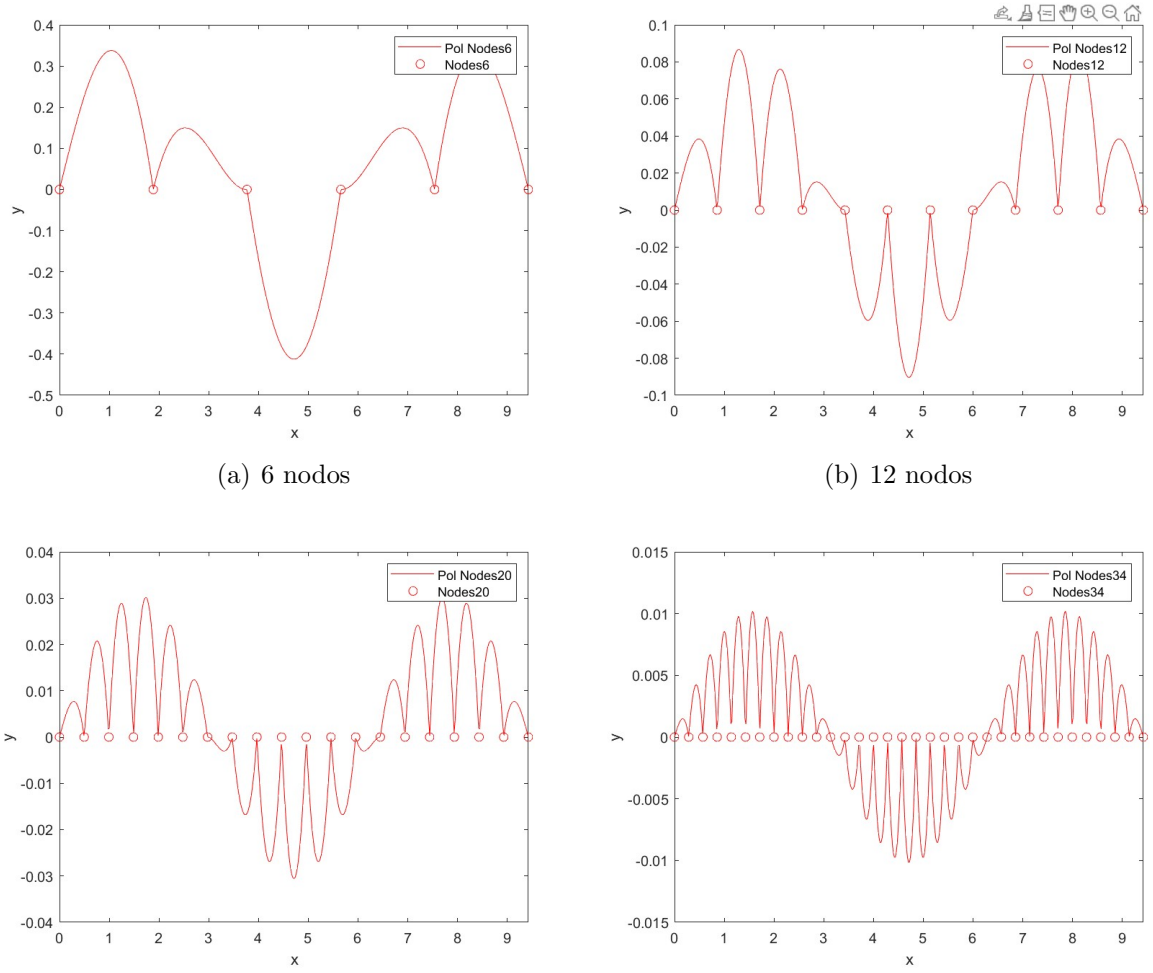
(b) 12 nodos

(c) 20 nodos

(d) 34 nodos

Figura 6: Interpolación por splines lineales de 6, 12, 20 y 34 nodos para $\sin(x)$

3.1.6. Error en Interolación por Splines Lineales para $\sin(x)$



(a) 6 nodos

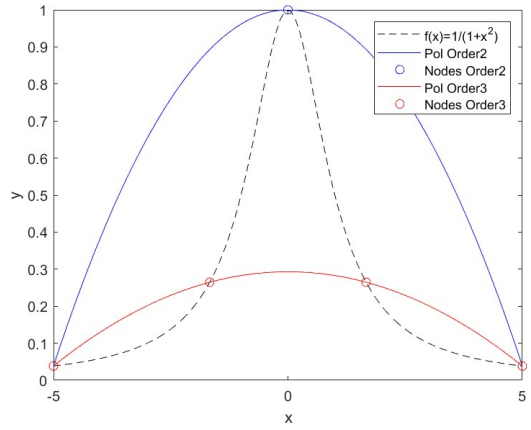
(b) 12 nodos

(c) 20 nodos

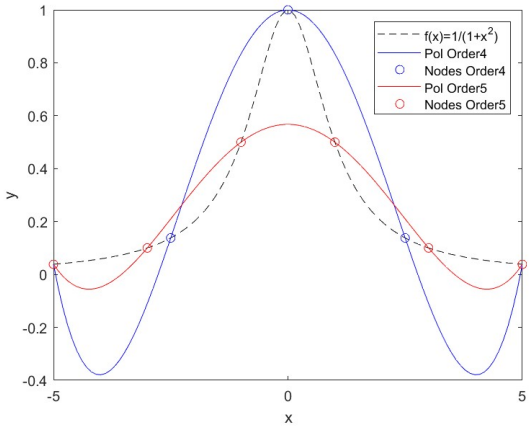
(d) 34 nodos

Figura 7: Interpolación por splines lineales de 6, 12, 20 y 34 nodos para $\sin(x)$

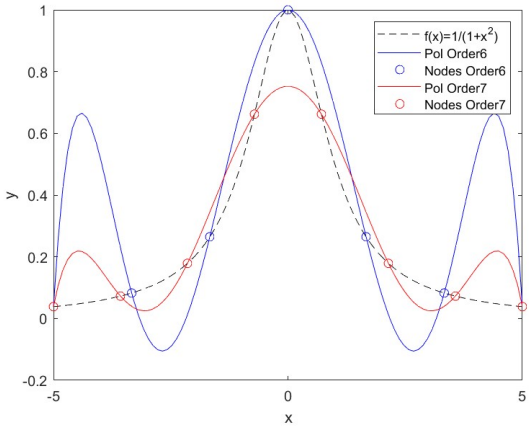
3.1.7. Interolación de Lagrange para $\frac{1}{1+x^2}$



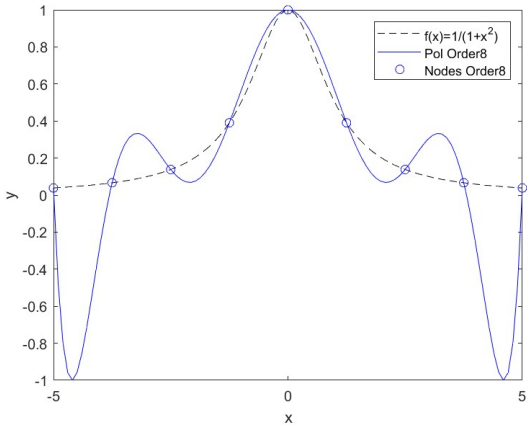
(a) Orden 2 y 3



(b) Orden 4 y 5



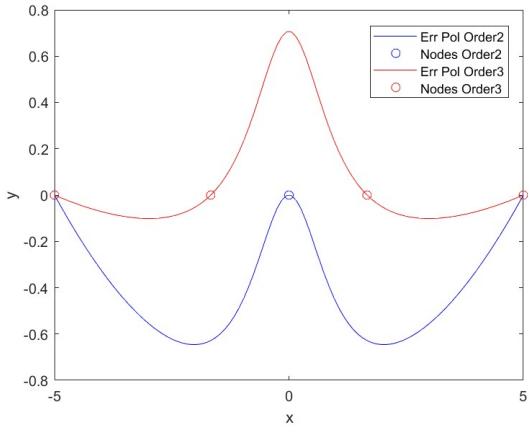
(c) Orden 6 y 7



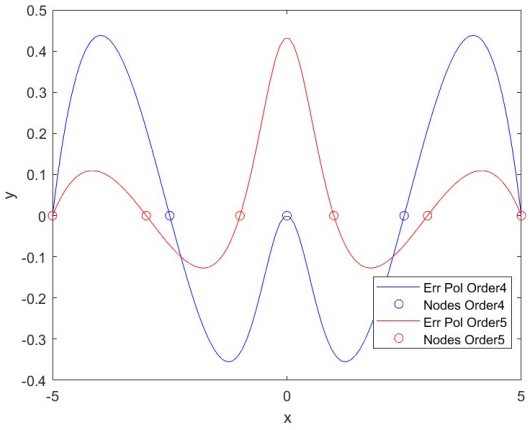
(d) Orden 8

Figura 8: Interpolación de Lagrange de orden 2 a 8 para $\frac{1}{1+x^2}$

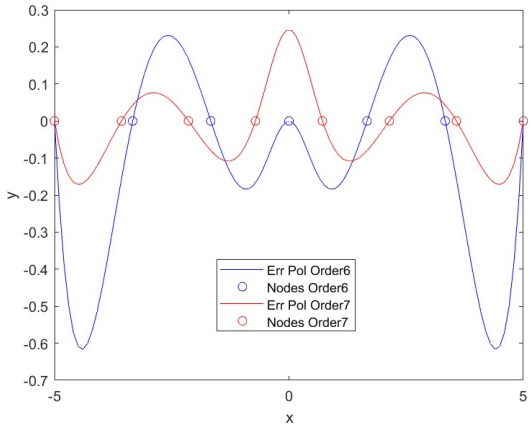
3.1.8. Error en Interolación de Lagrange para $\frac{1}{1+x^2}$



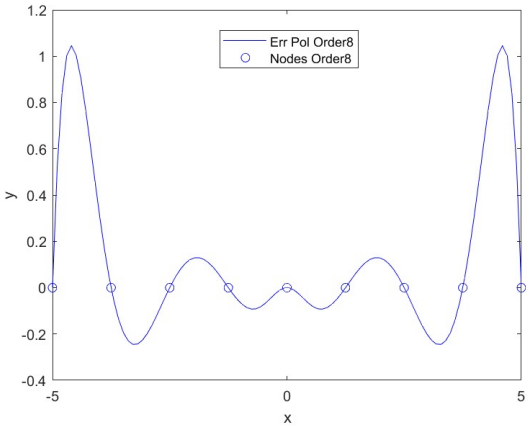
(a) Orden 2 y 3



(b) Orden 4 y 5



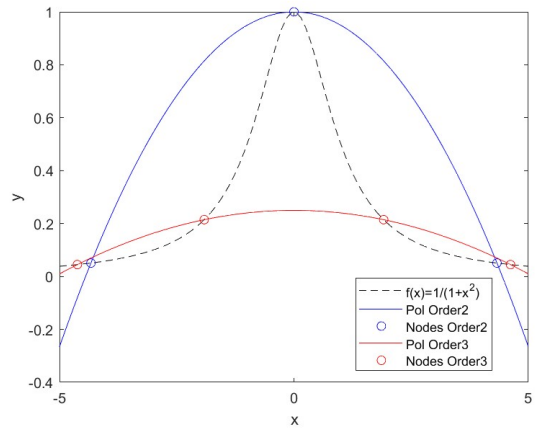
(c) Orden 6 y 7



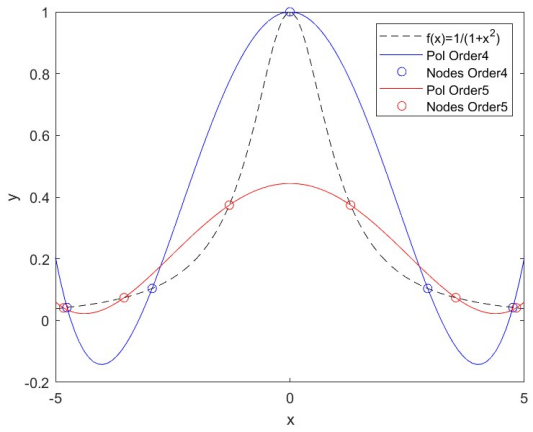
(d) Orden 8

Figura 9: Error en nterpolación de Lagrange de orden 2 a 8 para $\frac{1}{1+x^2}$

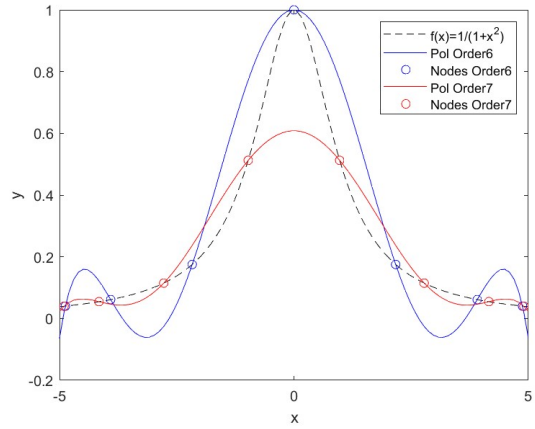
3.1.9. Interolación de Lagrange para $\frac{1}{1+x^2}$ con nodos de Tchebyshev



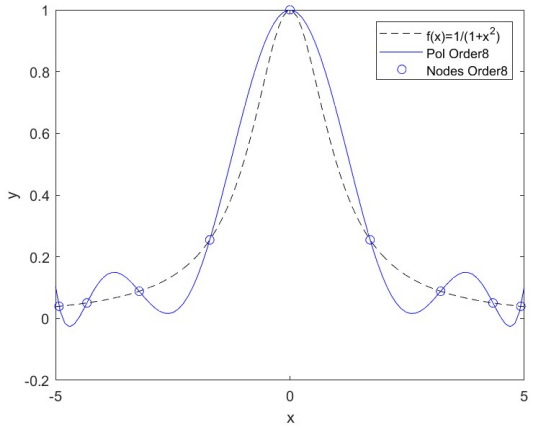
(a) Orden 2 y 3



(b) Orden 4 y 5



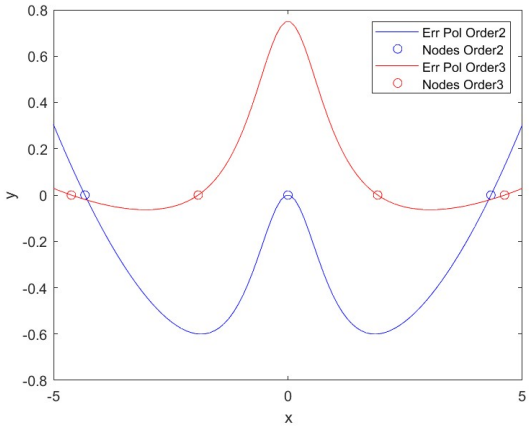
(c) Orden 6 y 7



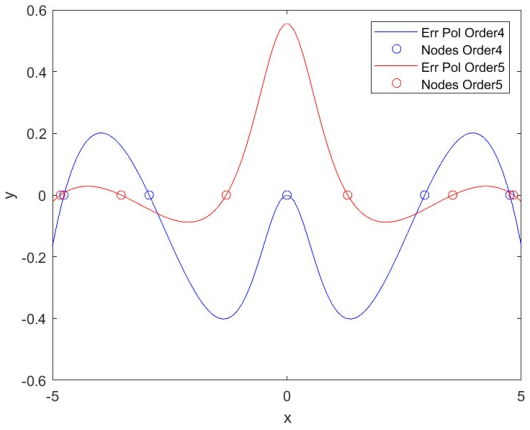
(d) Orden 8

Figura 10: Interpolación de Lagrange con nodos de Tchebyshev de orden 2 a 8 para $\frac{1}{1+x^2}$

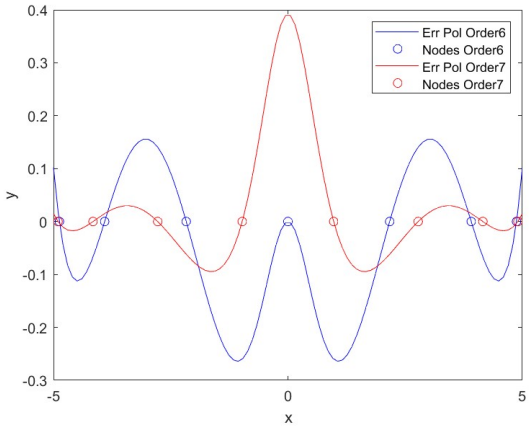
3.1.10. Error en Interolación de Lagrange para $\frac{1}{1+x^2}$ con nodos de Tchebyshev



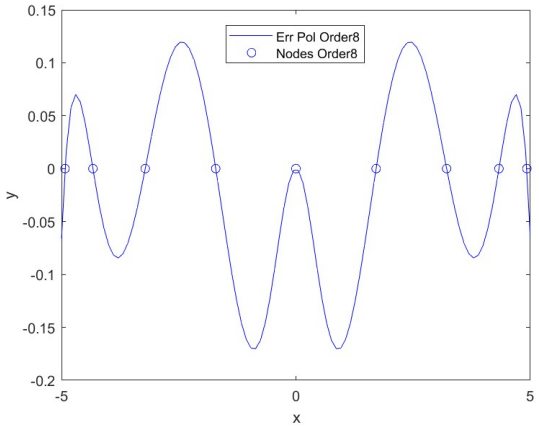
(a) Orden 2 y 3



(b) Orden 4 y 5



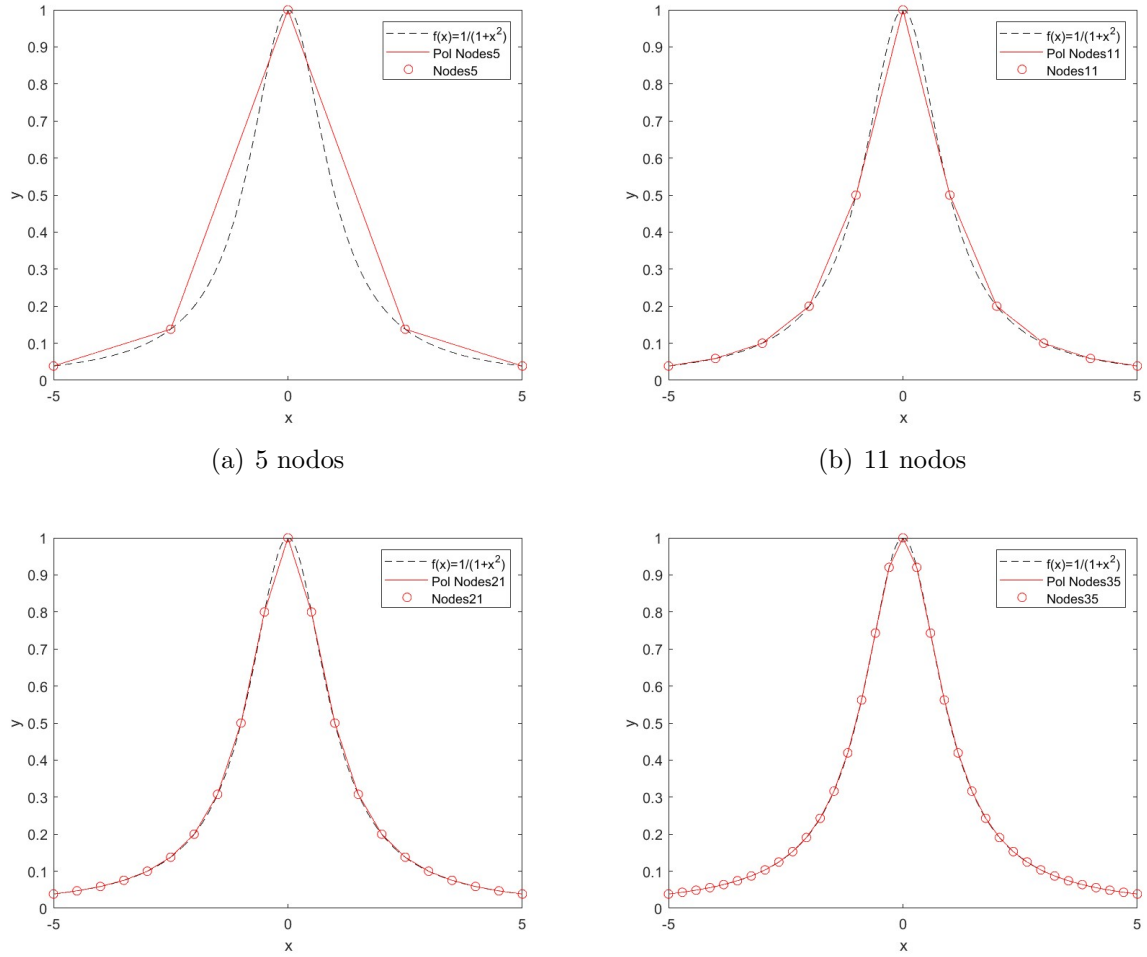
(c) Orden 6 y 7



(d) Orden 8

Figura 11: Error en nterpolación de Lagrange con nodos de Tchebyshev de orden 2 a 8 para $\frac{1}{1+x^2}$

3.1.11. Interolación por Splines Lineales para $\frac{1}{1+x^2}$



(a) 5 nodos

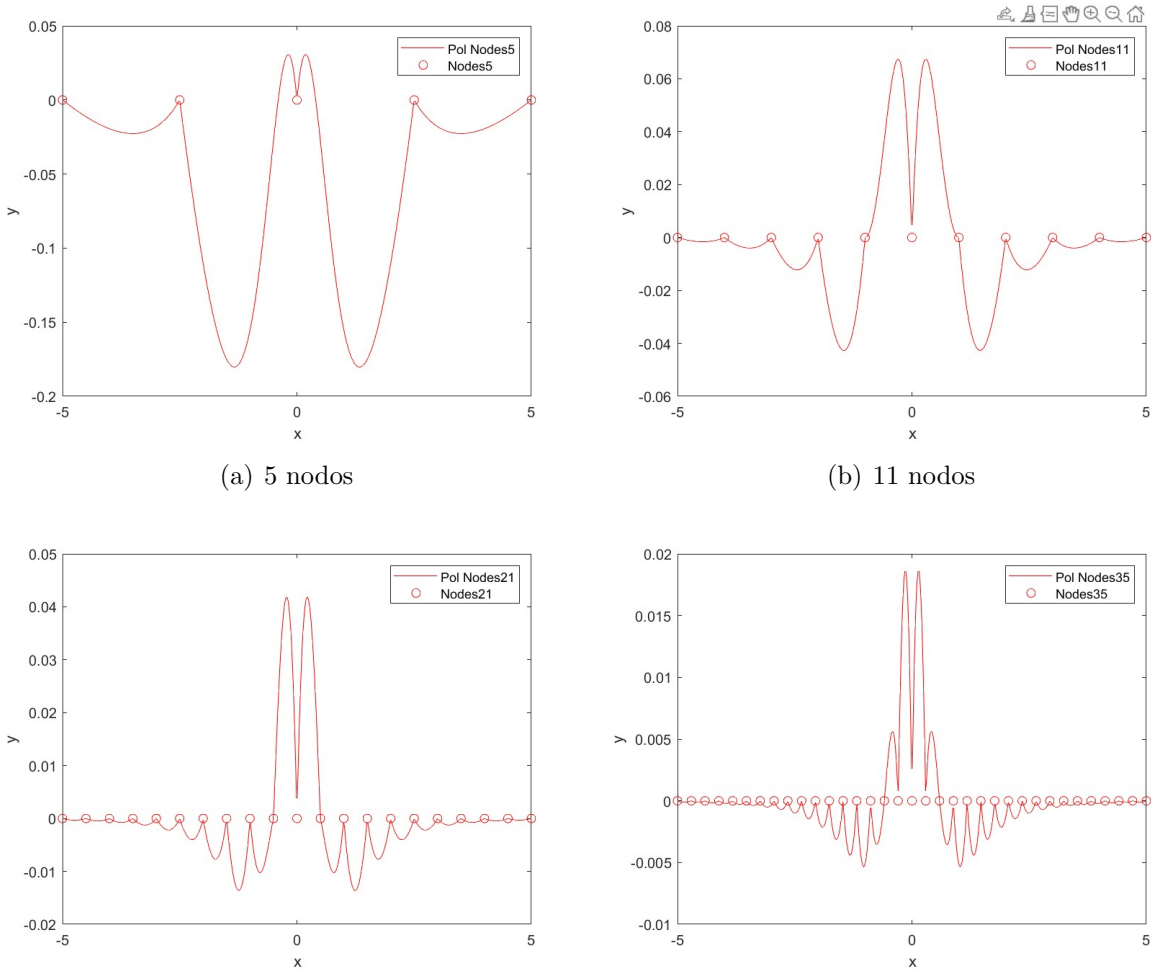
(b) 11 nodos

(c) 21 nodos

(d) 35 nodos

Figura 12: Interpolación por splines lineales de 5, 11, 21 y 35 nodos para $\frac{1}{1+x^2}$

3.1.12. Error en Interolación por Splines Lineales para $\frac{1}{1+x^2}$



(a) 5 nodos

(b) 11 nodos

(c) 21 nodos

(d) 35 nodos

Figura 13: Interpolación por splines lineales de 5, 11, 21 y 35 nodos para $\frac{1}{1+x^2}$

3.1.13. Errores globales en interpolación de Lagrange

Tabla 2: Error global para $\sin(x)$

Nodos	Orden 2	Orden 3	Orden 4	Orden 5	Orden 6	Orden 7	Orden 8
Equiespaciados	1.60	0.99	0.65	0.42	0.13	0.088	0.022
Tchebyshev	1.20	1.20	0.36	0.25	0.046	0.028	0.003

Tabla 3: Error global para $\frac{1}{1+x^2}$

Nodos	Orden 2	Orden 3	Orden 4	Orden 5	Orden 6	Orden 7	Orden 8
Equiespaciados	0.64	0.70	0.43	0.43	0.61	0.24	1.00
Tchebyshev	0.60	0.74	0.40	0.55	0.26	0.38	0.17

3.1.14. Comparación de métodos de interpolación

Se comparan las sumas de los tiempos y flops de interpolar las funciones mediante polinomios de orden 2 a 8.

Tabla 4: Tiempos y flops para ambas funciones

$\sin(x)$	Lagrange	Newton
Tiempos [ms]	0.6	0.4
Flops Asint.	1295	203

$\frac{1}{1+x^2}$	Lagrange	Newton
Tiempos [ms]	0.9	0.6
Flops Asint.	1295	203

3.2. Problema 2: Integración numérica

El problema propuesto nos solicita comparar el error en la integración numérica de la función respecto a los métodos de rectángulos, Punto medio, Trapecio, Simpson , Trapecio corregido y Gaus-Legendre. Por lo tanto el valor numérico de la integral es:

$$\int_0^1 e^{-x^2} dx = \frac{\sqrt{\pi} \cdot erf(1)}{2} = 0.7468$$

Es importante mencionar que el programa Matlab creado entrego el valor de 0.7468 en 2.985383 segundos. Para poder realizar la comparar el error de integración, se realizara 4 tablas, donde 3 de ellas serán para los métodos de Newton-Cotes y una para el método de Gauss-Legendre, donde se tomara distintas cantidades de nodos, con la finalidad de comprender como influencia la subdivisión del intervalo en los métodos de integración.

Para el calculo del error se utilizo la diferencia entre ambos resultados.

$$error = |valor\ integral - valor\ metodo|$$

Tabla 5: Tabla resumen de los resultados para $N = 14$ nodos

Método	Rectangulo Izquierda	Punto medio	Trapecio	Simpson	Trapecio corregido
Nodos	14	14	14	14	14
Resultado	0.7691	0.7470	0.7465	0.7468	0.7468
Error	0.0223	1.5646e-04	3.1288e-04	2.1254e-07	5.3188e-08
Tiempo [s]	3.009927	3.163536	3.232872	3.202020	4.058305

Tabla 6: Tabla resumen de los resultados para $N = 22$ nodos

Método	Rectangulo Izquierda	Punto medio	Trapecio	Simpson	Trapecio corregido
Nodos	22	22	22	22	22
Resultado	0.7611	0.7469	0.7467	0.7468	0.7468
Error	0.0142	6.3348e-05	1.2669e-04	3.4881e-08	8.7237e-09
Tiempo [s]	3.797048	3.682476	3.745879	3.750374	4.067669

Tabla 7: Tabla resumen de los resultados para $N = 30$ nodos

Método	Rectangulo Izquierda	Punto medio	Trapecio	Simpson	Trapecio corregido
Nodos	30	30	30	30	30
Resultado	0.7573	0.7469	0.7468	0.7468	0.7468
Error	0.0105	3.4065e-05	6.8128e-05	1.0090e-08	2.5230e-09
Tiempo [s]	3.225735	3.292598	3.687293	3.326182	4.187077

Para el caso de Gauss-Legendre se trabaja hasta 6 nodos, la finalidad es comprender el comportamiento del método, ya que se esta trabajando con valores de peso y abscisas tabuladas.

Tabla 8: Tabla resumen para Gauss-Legendre

Nodos	1	2	3	4	5	6
Resultado	0.7788	0.7465	0.7468	0.7468	0.7468	0.7468
Error	0.03197	2.2944e-04	9.5122e-06	3.3797e-07	3.0899e-08	1.4684e-09
Tiempo [s]	3.203877	3.192937	3.053790	3.259203	3.403604	3.197885

4. Análisis de Resultados

4.1. Problema 1

De los resultados presentados, ambas funciones pudieron ser interpoladas con los métodos propuestos. De todas formas, ambas funciones presentan resultados diferentes en torno a la precisión de la representación por la interpolación. Primero se nota que la interpolación con la base de Newton (diferencias divididas) es más rápida que utilizar el método de Lagrange. Esto se debe a que el método es más barato en total (encontrar el polinomio y evaluarlo). Asíntoticamente para el metodo de Newton es de $\mathcal{O}(n^2)$ y para el método de Lagrange es de $\mathcal{O}(n^3)$. Así, el costo asintótico es menor y se ve reflejado en el tiempo de cálculo. De todas formas, al ser una cantidad relativamente pequeña de nodos, la diferencia es de 0.3 [ms] aproximadamente.

Por otra parte, se nota que la función $\sin(x)$ es representada de manera bastante precisa con nodos equiespaciados y polinomios de orden mayor o igual a 6. Esto se debe a que la forma de la función sinusoidal es el intervalo acotado tiene un comportamiento similar a un polinomio de mayor orden. Considerando ahora la función $\frac{1}{1+x^2}$, se nota que no ocurre lo mismo. Al esta función ser parecido a un pulso, una representación polinómica no le queda muy bien, en donde se ve reflejado en los gráficos. Los nodos equiespaciados generan oscilaciones en los bordes del dominio, los cuales aumentan el error. Este comportamiento es llamado *Fenómeno de Runge*. Lo interesante es que utilizando nodos de Tchebyshev para la interpolación, este fenómeno se puede minimizar y así representar mejor los bordes del dominio. Sin embargo, la interpolación sigue siendo no muy precisa.

Se notó que el utilizar nodos de Tchebyshev disminuye el error en ambas funciones, dado que los nodos son impuestos para lograr esto.

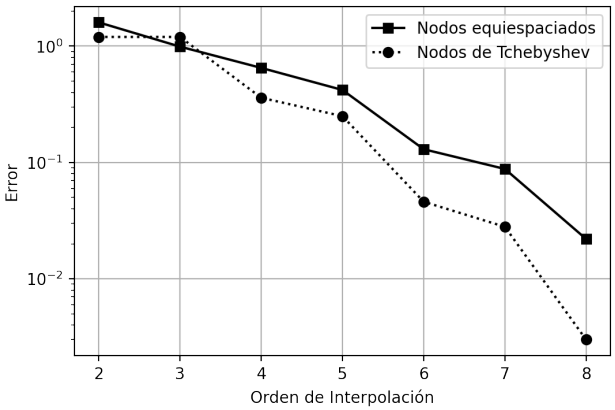


Figura 14: Error global para $\sin(x)$

En la función $\sin(x)$ se la imposición de nodos disminuye el error en las interpolaciones de orden mayor a 3.

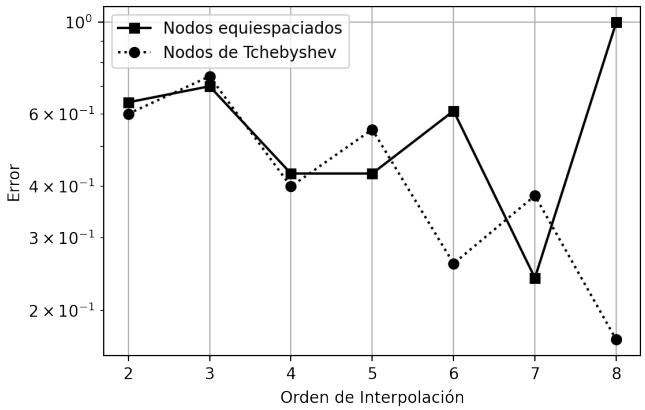


Figura 15: Error global para $\frac{1}{1+x^2}$

Por otra parte, en la función $\frac{1}{1+x^2}$, se nota como se generan oscilaciones en el error dado el Fenómeno de Runge. De todas formas, se nota como en mayores ordenes, los nodos impuestos de Tchebyshev logran disminuir el error. La imposición de nodos puede ser ventajosa por lo visto, pero en un problema real que se quiera interpolar una tabla de datos, por ejemplo, puede ser difícil obtener justo puntos que coincidan con los nodos de Tchebyshev. Dado esto, su gran ventaja se puede ver minimizada en la práctica.

Por último, ambas funciones se interpolaron utilizando Splines Lineales. Ambas funciones lograron buenas representaciones con una cantidad mayor de 20 nodos. Se considera que en 34 para $\sin(x)$ y 35 para $\frac{1}{1+x^2}$ se tiene una representación bastante precisa. Incluso, es válido destacar que se representa mejor la función en comparación a un solo polinomio interpolante. De todas formas, para grandes dominios este método puede ser mas costoso dado que hay que almacenar un polinomio por cada intervalo. Mejoras a esta última interpolación puede ser la utilización de Splines Cúbicos, al cuales son altamente utilizadas en la práctica dada su gran representación de funciones.

4.2. Problema 2

Luego de realizar un programa para poder calcular error de integración numérica de la función con respecto a los métodos mediante el programa MATLAB, se pudo tabular los resultados para una cantidad de nodos de 14, 22 y 30 para los métodos de Newton-Cotes y nodos de 1, 2, 3, 4, 5 y 6 para el método de Gauss-Legendre, esto con la finalidad de poder comparar la variación de los resultados y su rendimiento.

Se puedo observar que el método con menor error es el de Trapecio corregido, donde su error corresponde a $5.3188e-08$ para $n = 14$, $8.7237e-09$ para $n = 22$ y $2.5230e-09$ para $n = 30$, si bien el resultado es prácticamente similar, el tiempo de calculo es mayor que los otros métodos.

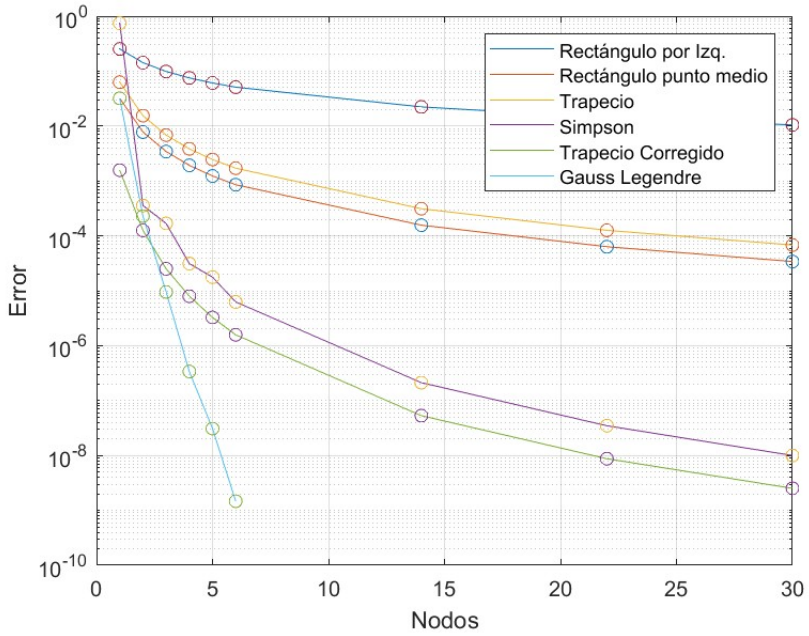


Figura 16: Errores los diferentes métodos

Por otra parte el segundo método con menor error es el de Simpson equivalente a $2.1254e-07$ para $n = 14$, $3.4881e-08$ para $n = 22$ y $1.0090e-08$ para $n = 30$, pues esto se debe a que el método es subdivido en dos partes, para números pares que se trabaja con el método de Simpson 1/3 y para los pares que trabaja con el método de Simpson 3/8, esto permite obtener un resultado mas exacto, sin embargo entre más aumenta la cantidad de nodos, se demora mas en obtener el resultado y a su vez es más costoso computacionalmente.

Una observación importante es el método de Gauss-Legendre, ya que a medida que se va aumentando la cantidad de nodos, el error disminuye rápidamente, pues al observar el gráfico que contiene todos los métodos respecto a sus nodos y errores, podemos concluir que se necesita una

menor cantidad de nodos para obtener un error menor, demostrando ser un método sumamente eficiente. Esto se debe a que trabaja con pesos y abscisas tabuladas de los polinomios de legendre, permitiendo ahorrar tiempo en el calculo y disminuyendo los costos computacionales que esto implica.

5. Conclusión

Mediante el análisis realizado para ambos problemas resueltos, se puede concluir que existe una gran variedad de métodos dependiendo del problema a resolver. Por ejemplo para interpolar e integrar funciones se utilizaron los métodos presentados en este trabajo. Es importante destacar que cada método tiene sus ventajas dependiendo de la función a trabajar, y por ende, hay que ser cuidadoso con su elección.

Para el problema 1 de interpolación, se noto que la función $\sin(x)$ es factible interpolarla con polinomios de grado mayor o igual a 6, entregando errores bastante bajos. Por otra parte, la función $\frac{1}{1+x^2}$ presenta problemas al interpolarse con polinomios, dado que se evidencia el fenómeno de Runge. Esta problemática se minimiza utilizando nodos de Tchebyshev, lo cual reduce el error de interpolación. De todas formas, al ser los nodos impuestos, se hace difícil de aplicar en la práctica. Por último, una representación por Splines tiende a ser la mejor dado que interpola la función a tramos, lo que evita oscilaciones. En el trabajo se obtuvo que con alrededor de 34 nodos ambas funciones tienen una buena representación.

Para el problema 2 se puede ver que al subdividir el intervalo [a,b] se obtiene un error menor, pero el error dependerá netamente del método utilizado. Respecto al método de Newton – Cotes, se puede observar que los menores errores que se encuentran dentro de esta categoría, corresponden a los método de Trapecio corregido que se relaciona con la derivada de la función a integrar y el método de Simpson, en el cual acepta cualquier nodo, ya que está diseñado para identificar si el nodo es par o impar, para proceder a ser calculado mediante los métodos de Simpson 1/3 o 3/8, el cual entrega una buena aproximación, pero requiere de un costo computacional alto. Por otra parte, el método de Gauss-legendre se encuentra dentro de los métodos mas eficientes y que tienen menores errores, ya que las cuadraturas de Gauss son mucho mejores y más precisas que las fórmulas de Newton-Cotes.

6. Referencias

[1] Chapra, S; Canale, R.(2007). Métodos numéricos para ingenieros, 5^a Edición.

[2] Gers, R. Apuntes de clases : MEC270-Métodos numéricos en ingeniería mecánica.

7. Anexos

7.1. Códigos Elaborados

Se crearon dos archivos main para cada problema en el cual se importan las funciones programadas para cada método. Todos los archivos se adjuntan a continuación.

7.1.1. Códigos Principales Pregunta 1

Principal para interpolación de Lagrange

El código crea los gráficos presentados para la interpolación con polinomios de Lagrange.

```
1
2  format compact
3
4
5  plot_lagrange(0,3*pi,@f1,1,'f(x)=sin(x)')
6
7  plot_lagrange(-5,5,@f2,5,'f(x)=1/(1+x^2)')
8
9  plot_lagrange_error(0,3*pi,@f1,9,'sin(x)')
10
11 plot_lagrange_error(-5,5,@f2,13,'f(x)=1/(1+x^2)')
12
13 function [y] = f1(x)
14     y = sin(x);
15 end
16
17 function [y] = f2(x)
18     y = 1./(1+x.^2);
19 end
20
21
22 function [flag] = plot_lagrange(a,b,f,pp,lab)
23
24     pts = 100;
25     x_Lagrange = linspace(a,b,pts);
26     Y_Lagrange = zeros(7,pts);
27     j = pp;
28     for i=1:7
29         order = i + 1;
30         n = order + 1;
31         x = linspace(a,b,n);
32         y = f(x);
33         [y_inter,~] = inter_lagrange(x,y,pts);
34         Y_Lagrange(n-2,:) = y_inter;
35
```

```
36          if (mod(order,2)~=0)
37              x_real = x_Lagrange;
38              figure(j)
39              plot(x_real,f(x_real),'k--','DisplayName',lab)
40              hold on
41              plot(x_real,Y_Lagrange(i-1,:),'b','DisplayName',strcat('Pol
                    Order ',int2str(order-1)))
42              hold on
43              x1 = linspace(a,b,n-1);
44              x2 = linspace(a,b,n);
45              scatter(x1,f(x1),'b','DisplayName',strcat('Nodes Order ',
                    int2str(order-1)))
46              hold on
47              plot(x_real,Y_Lagrange(i,:),'r','DisplayName',strcat('Pol
                    Order ',int2str(order)))
48              hold on
49              xlim([a b])
50              xlabel('x')
51              ylabel('y')
52              scatter(x2,f(x2),'r','DisplayName',strcat('Nodes Order ',
                    int2str(order)))
53              hold on
54              legend
55              j = j + 1;
56          end
57
58          if(i==7)
59              figure(j)
60              plot(x_real,f(x_real),'k--','DisplayName',lab)
61              hold on
62              plot(x_real,Y_Lagrange(i,:),'b','DisplayName',strcat('Pol
                    Order ',int2str(order)))
63              hold on
64              xlim([a b])
65              xlabel('x')
66              ylabel('y')
67              x1 = linspace(a,b,n);
68              scatter(x1,f(x1),'b','DisplayName',strcat('Nodes Order ',
                    int2str(order)))
69              hold on
70              legend
71          end
72
73      end
74      flag = true;
75
76  end
77
```

```
78  function [flag] = plot_lagrange_error(a,b,f,pp,lab)
79
80      pts = 100;
81      x_Lagrange = linspace(a,b,pts);
82      Y_Lagrange = zeros(7,pts);
83      j = pp;
84      for i=1:7
85          order = i + 1;
86          n = order + 1;
87          x = linspace(a,b,n);
88          y = f(x);
89          [y_inter,~] = inter_lagrange(x,y,pts);
90          Y_Lagrange(n-2,:) = y_inter;
91
92          if (mod(order,2)~=0)
93              x_real = x_Lagrange;
94              figure(j)
95              err = f(x_real)-Y_Lagrange(i-1,:);
96              disp(strcat('Error of Pol Order',int2str(order-1)))
97              e = max(abs(err))
98              plot(x_real,err,'b','DisplayName',strcat('Err Pol Order ',
                    int2str(order-1)))
99              hold on
100             x1 = linspace(a,b,n-1);
101             x2 = linspace(a,b,n);
102             scatter(x1,f(x1)*0,'b','DisplayName',strcat('Nodes Order ',
                    int2str(order-1)))
103             hold on
104             err = f(x_real)-Y_Lagrange(i,:);
105             disp(strcat('Error of Pol Order',int2str(order)))
106             e = max(abs(err))
107             plot(x_real,err,'r','DisplayName',strcat('Err Pol Order ',
                    int2str(order)))
108             hold on
109             xlim([a b])
110             xlabel('x')
111             ylabel('y')
112             scatter(x2,f(x2)*0,'r','DisplayName',strcat('Nodes Order ',
                    int2str(order)))
113             hold on
114             legend
115             j = j + 1;
116         end
117
118         if(i==7)
119             figure(j)
120             err = f(x_real)-Y_Lagrange(i,:);
121             disp(strcat('Error of Pol Order',int2str(order)))
```

```
122             e = max(abs(err))
123             plot(x_real,err,'b','DisplayName',strcat('Err Pol Order ',
                    int2str(order)))
124             hold on
125             xlim([a b])
126             xlabel('x')
127             ylabel('y')
128             x1 = linspace(a,b,n);
129             scatter(x1,f(x1)*0,'b','DisplayName',strcat('Nodes Order ',
                    int2str(order)))
130             hold on
131             legend
132         end
133
134     end
135     flag = true;
136
137 end
```

Principal para interpolación con nodos de Tchebyshev

El código crea los gráficos presentados para la interpolación con polinomios de Lagrange en nodos de Tchebyshev.

```
1
2  format compact
3
4
5  plot_tchebyshev(0,3*pi,@f1,1,'f(x)=sin(x)')
6
7  plot_tchebyshev(-5,5,@f2,5,'f(x)=1/(1+x^2)')
8
9  plot_tchebyshev_error(0,3*pi,@f1,9,'sin(x)')
10
11 plot_tchebyshev_error(-5,5,@f2,13,'f(x)=1/(1+x^2)')
12
13 function [y] = f1(x)
14     y = sin(x);
15 end
16
17 function [y] = f2(x)
18     y = 1./(1+x.^2);
19 end
20
21
22 function [flag] = plot_tchebyshev(a,b,f,pp,lab)
23
24     pts = 100;
25     x_chebs = linspace(a,b,pts);
```

```
26      Y_Chebs = zeros(7,pts);
27      j = pp;
28      for i=1:7
29          order = i + 1;
30          n = order + 1;
31          x = linspace(a,b,n);
32          y = f(x);
33          [y_inter,x2,yi2,xi2] = inter_chebs(x,pts,f);
34          Y_Chebs(n-2,:) = y_inter;
35
36          if(mod(order,2)==0)
37              xi1 = xi2;
38              yi1 = yi2;
39              x1 = x2;
40          end
41
42          if (mod(order,2)~=0)
43              x_real = x_chebs;
44              figure(j)
45              plot(x_real,f(x_real),'k--','DisplayName',lab)
46              hold on
47              plot(x1,Y_Chebs(i-1,:),'b','DisplayName',strcat('Pol Order
                    ',int2str(order-1)))
48              hold on
49              scatter(xi1,f(xi1),'b','DisplayName',strcat('Nodes Order ',
                    int2str(order-1)))
50              hold on
51              plot(x2,Y_Chebs(i,:),'r','DisplayName',strcat('Pol Order ',
                    int2str(order)))
52              hold on
53              xlim([a b])
54              xlabel('x')
55              ylabel('y')
56              scatter(xi2,f(xi2),'r','DisplayName',strcat('Nodes Order ',
                    int2str(order)))
57              hold on
58              legend
59              j = j + 1;
60          end
61
62          if(i==7)
63              figure(j)
64              plot(x1,f(x1),'k--','DisplayName',lab)
65              hold on
66              plot(x1,Y_Chebs(i,:),'b','DisplayName',strcat('Pol Order ',
                    int2str(order)))
67              hold on
68              xlim([a b])
```

```
69              xlabel('x')
70              ylabel('y')
71              x1 = linspace(a,b,n);
72              scatter(xi1,f(xi1),'b','DisplayName',strcat('Nodes Order ',
                    int2str(order)))
73              hold on
74              legend
75          end
76
77      end
78      flag = true;
79
80  end
81
82  function [flag] = plot_tchebyshev_error(a,b,f,pp,lab)
83
84      pts = 100;
85      x_chebs = linspace(a,b,pts);
86      Y_Chebs = zeros(7,pts);
87      j = pp;
88      for i=1:7
89          order = i + 1;
90          n = order + 1;
91          x = linspace(a,b,n);
92          y = f(x);
93          [y_inter,x2,yi2,xi2] = inter_chebs(x,pts,f);
94          Y_Chebs(n-2,:) = y_inter;
95
96          if(mod(order,2)==0)
97              xi1 = xi2;
98              yi1 = yi2;
99              x1 = x2;
100         end
101
102         if (mod(order,2)~=0)
103             x_real = x_chebs;
104             figure(j)
105             err = f(x1)-Y_Chebs(i-1,:);
106             disp(strcat('Error of Pol Order',int2str(order-1)))
107             e = max(abs(err))
108             plot(x1,err,'b','DisplayName',strcat('Err Pol Order ',
                    int2str(order-1)))
109             hold on
110             scatter(xi1,f(xi1)*0,'b','DisplayName',strcat('Nodes Order
                    ',int2str(order-1)))
111             hold on
112             err = f(x2)-Y_Chebs(i,:);
113             disp(strcat('Error of Pol Order',int2str(order)))
```

```
114             e = max(abs(err))
115             plot(x2,err,'r','DisplayName',strcat('Err Pol Order ',
                    int2str(order)))
116             hold on
117             xlim([a b])
118             xlabel('x')
119             ylabel('y')
120             scatter(xi2,f(xi2)*0,'r','DisplayName',strcat('Nodes Order
                    ',int2str(order)))
121             hold on
122             legend
123             j = j + 1;
124         end
125
126         if(i==7)
127             figure(j)
128             err = f(x1)-Y_Chebs(i,:);
129             disp(strcat('Error of Pol Order',int2str(order)))
130             e = max(abs(err))
131             plot(x1,err,'b','DisplayName',strcat('Err Pol Order ',
                    int2str(order)))
132             hold on
133             xlim([a b])
134             xlabel('x')
135             ylabel('y')
136             scatter(xi1,f(xi1)*0,'b','DisplayName',strcat('Nodes Order
                    ',int2str(order)))
137             hold on
138             legend
139         end
140
141     end
142     flag = true;
143
144 end
```

Principal para interpolación por splines lineales

El código crea los gráficos presentados para la interpolación con Splines Lineales.

```
1
2 format compact
3
4
5 plot_lineal(0,3*pi,@f1,1,'f(x)=sin(x)',[6 12 20 34])
6
7 plot_lineal(-5,5,@f2,5,'f(x)=1/(1+x^2)',[5 11 21 35])
8
9 plot_lineal_error(0,3*pi,@f1,9,'sin(x)',[6 12 20 34])
```

```
10
11 plot_lineal_error(-5,5,@f2,13,'f(x)=1/(1+x^2)',[5 11 21 35])
12
13
14 function [y] = f1(x)
15     y = sin(x);
16 end
17
18 function [y] = f2(x)
19     y = 1./(1+x.^2);
20 end
21
22
23 function [flag] = plot_lineal(a,b,f,pp,lab,nodes)
24
25     pts = 500;
26     x_lineal = linspace(a,b,pts);
27     Y_Lineal = zeros(7,pts);
28     j = pp;
29     for r=1:4
30         i = nodes(r);
31         n = i;
32         x = linspace(a,b,n);
33         y = f(x);
34         [y_inter,~] = inter_lineal(x,y,pts);
35         Y_Lineal(r,:) = y_inter;
36
37         x_real = x_lineal;
38         figure(j)
39         plot(x_real,f(x_real),'k--','DisplayName',lab)
40         hold on
41         x2 = linspace(a,b,n);
42         plot(x_real,Y_Lineal(r,:),'r','DisplayName',strcat('Pol Nodes
               ',int2str(n)))
43         hold on
44         xlim([a b])
45         xlabel('x')
46         ylabel('y')
47         scatter(x2,f(x2),'r','DisplayName',strcat('Nodes  ',int2str(n))
               )
48         hold on
49         legend
50         j = j + 1;
51     end
52     flag = true;
53
54 end
55
```

```
function [flag] = plot_lineal_error(a,b,f,pp,lab,nodes)

    pts = 500;
    x_lineal = linspace(a,b,pts);
    Y_Lineal = zeros(7,pts);
    j = pp;
    for r=1:4
        i = nodes(r);
        n = i;
        x = linspace(a,b,n);
        y = f(x);
        [y_inter,~] = inter_lineal(x,y,pts);
        Y_Lineal(r,:) = y_inter;

        x_real = x_lineal;
        figure(j)
        x2 = linspace(a,b,n);
        err = f(x_real)-Y_Lineal(r,:);
        disp(strcat('Error of Pol Nodes',int2str(n)))
        e = max(abs(err))
        plot(x_real,err,'r','DisplayName',strcat('Pol Nodes ',int2str(n
           )))
        hold on
        xlim([a b])
        xlabel('x')
        ylabel('y')
        scatter(x2,f(x2)*0,'r','DisplayName',strcat('Nodes  ',int2str(n
           )))
        hold on
        legend
        j = j + 1;


    end
    flag = true;

end
```

Principal para medición de tiempo en interpolación

El código obtiene los tiempos de cálculo.

```

format compact

tic
lf1 = all_lagrange(0,3*pi,@f1);
toc

```

```
8  tic
9  nf1 = all_newton(0,3*pi,@f1);
10 toc
11
12 tic
13 lf2 = all_lagrange(-5,5,@f2);
14 toc
15
16 tic
17 nf2 = all_newton(-5,5,@f2);
18 toc
19
20 tic
21 all_lineal(0,3*pi,@f1,[4,10,15,34]);
22 toc
23
24 tic
25 all_lineal(-5,5,@f2,[4,10,15,35]);
26 toc
27
28 function [y] = f1(x)
29     y = sin(x);
30 end
31
32 function [y] = f2(x)
33     y = 1./(1+x.^2);
34 end
35
36
37 function [Y_Lagrange] = all_lagrange(a,b,f)
38
39     pts = 100;
40     Y_Lagrange = zeros(7,pts);
41     for i=1:7
42         order = i + 1;
43         n = order + 1;
44         x = linspace(a,b,n);
45         y = f(x);
46         [y_inter,~] = inter_lagrange(x,y,pts);
47         Y_Lagrange(n-2,:) = y_inter;
48     end
49 end
50
51 function [Y_Newton] = all_newton(a,b,f)
52
53     pts = 100;
54     Y_Newton = zeros(7,pts);
55     for i=1:7
```

```
56          order = i + 1;
57          n = order + 1;
58          x = linspace(a,b,n);
59          y = f(x);
60          [y_inter,~] = inter_newton(x,y,pts);
61          Y_Newton(n-2,:) = y_inter;
62      end
63  end
64
65  function [Y_Lineal] = all_lineal(a,b,f,nodes)
66
67      pts = 100;
68      Y_Lineal = zeros(7,pts);
69      for r=1:4
70          i = nodes(r);
71          n = i;
72          x = linspace(a,b,n);
73          y = f(x);
74          [y_inter,~] = inter_lineal(x,y,pts);
75          Y_Lineal(r,:) = y_inter;
76      end
77  end
```

7.1.2. Interpolación con base de Lagrange

```
1
2  function [y_inter,x_inter] = inter_lagrange(x,y,pts)
3
4      n = length(x);
5      x_inter = linspace(min(x),max(x),pts);
6      y_inter = zeros(1,pts);
7
8
9      for k=1:pts
10         P = 0;
11         for i=1:n
12             L = L_lagrange(x_inter(k),x,n,i);
13             P = P + L*y(i);
14         end
15         y_inter(k) = P;
16     end
17
18
19     function [L] = L_lagrange(x_eval,x_data,n,i)
20         L = 1;
21         if(x_eval==x_data(i))
22             L = 1;
23         else
24             for j=1:n
25                 if (j ~= i)
26                     L = L * (x_eval-x_data(j))/(x_data(i)-x_data(j));
27                 end
28             end
29         end
30     end
31 end
```

7.1.3. Interpolación con base de Newton

```
1
2  function [y_inter,x_inter] = inter_newton(x,y,pts)
3
4      n = length(x);
5      lam = fin_dif(x,y);
6      x_inter = linspace(min(x),max(x),pts);
7      y_inter = zeros(1,pts);
8
9      for z=1:pts
10         P = 0;
11         for l=1:n
12             Nj = N_newton(x_inter(z),x,l);
13             P = P + lam(l)*Nj;
14         end
```

```
15          y_inter(z) = P;
16      end
17
18      function [N] = N_newton(x_eval,x_data,l)
19          N = 1;
20          if (l==1)
21              N = 1;
22          else
23              for q=1:l-1
24                  N = N * (x_eval - x_data(q));
25              end
26          end
27
28      end
29
30
31      function [lambda] = fin_dif(x,y)
32
33          n = length(x);
34          fn = zeros(n,n);
35          for j=1:n
36              fn(j,j) = y(j);
37          end
38
39          for i=2:n
40              for k=1:n-i+1
41                  fn(k,i+k-1) = (fn(k+1,i+k-1)-fn(k,i+k-2))/(x(i+k-1)-x(k
                        ));
42              end
43          end
44
45          lambda = fn(1,:);
46
47      end
48
49  end
```

7.1.4. Interpolación con nodos de Tchebyshev

```
1
2   function [y_inter,x_inter,y_chebs,x_chebs] = inter_chebs(x,pts,f_eval)
3
4       n = length(x) - 1;
5       x_chebs = nodos_chebs(min(x),max(x),n);
6       y_chebs = f_eval(x_chebs);
7       [y_inter,x_inter] = interpol_lagrange(x_chebs,y_chebs,pts,min(x),
            max(x));
8
9       function [x_n] = nodos_chebs(a,b,n)
```

```
10          x_n = zeros(1,n+1);
11          for i=0:n
12              x_n(i+1) = (a+b)/2 + (b-a)/2*cos(pi*(2*(n-i)+1)/(2*n+2));
13          end
14      end
15
16      function [y_inter,x_inter] = interpol_lagrange(x,y,pts,a,b)
17
18          n = length(x);
19          x_inter = linspace(a,b,pts);
20          y_inter = zeros(1,pts);
21          for k=1:pts
22              P = 0;
23              for i=1:n
24                  L = L_lagrange(x_inter(k),x,n,i);
25                  P = P + L*y(i);
26              end
27              y_inter(k) = P;
28          end
29
30          function [L] = L_lagrange(x_eval,x_data,n,i)
31              L = 1;
32              if(x_eval==x_data(i))
33                  L = 1;
34              else
35                  for j=1:n
36                      if (j ~= i)
37                          L = L * (x_eval-x_data(j))/(x_data(i)-x_data(j)
                                );
38                      end
39                  end
40              end
41          end
42      end
43  end
```

7.1.5. Interpolación con Splines Lineales

```
1
2   function [y_inter,x_inter] = inter_lineal(x,y,pts)
3
4       n = length(x);
5       x_inter = linspace(min(x),max(x),pts);
6       y_inter = zeros(1,pts);
7
8       [m,b] = splines_lin(x,y,n);
9
10      i = 1;
11      for k=1:pts
```

```
12          if(x_inter(k)<=x(i+1))
13              y_inter(k) = b(i) + m(i)*(x_inter(k)-x(i));
14          else
15              y_inter(k) = b(i+1) + m(i+1)*(x_inter(k)-x(i+1));
16              i = i + 1;
17          end
18      end
19
20      function [mr,br] = splines_lin(x,y,n)
21
22          mr = zeros(1,n);
23          br = zeros(1,n);
24
25          for l=1:n-1
26              mr(l) = (y(l+1)-y(l))/(x(l+1)-x(l));
27              br(l) = y(l);
28          end
29      end
30  end
```

7.1.6. Código Principal Pregunta 2

Se presenta el código principal de la pregunta 2.

```
clc
clear all
tic
a=0;
b=1;
syms k;
f= [exp(-k^{2})];
g= [-2*k*(exp(-k^{2}))];
I=int(f,a,b);
n=14;
h=(b-a)/n;
x=a:h:b;

%calculo de los metodos
[rectang] = rectangulo(n,x,f,k,h)
[pmedioo] = pmedio(n,x,f,k,h)
[trap] = trapecio(n,x,f,k,h)
[simps] = simpson(n,x,f,k,h)
[trap_co]=trapecio_corregido(n,x,f,g,k,h,a,b)
[Gauss_leg]= gauss_legendre(n,x,f,k,h,a,b)

%Calculo de errores
RI=eval(I);
error_Rectangulo= abs(RI-rectang)
error_Pmedio= abs(RI-pmedioo)
error_trapecio= abs(RI-trap)
error_simpson= abs(RI-simps)
error_trapecio_Corregido = abs(RI-trap_co)
error_Gauss_legendre= abs(RI-Gauss_leg)
toc
```

7.1.7. Método de Rectángulo

```

%Metodo del rectangulo izquierda
function [valor1] = rectangulo(n,x,f,k,h)
    suma=0;
    for i=1:n
        y=eval(subs(f,k,x(i)));
        suma=suma + h*y;
    end
    valor1=suma;
end
```

7.1.8. Método de Punto Medio

```
1
2  %Metodo del punto medio
3  function [valor2] = pmedio(n,x,f,k,h)
4      suma=0;
5      for i=1:n;
6          j=(x(i)+x(i+1))/2;
7          y=eval(subs(f,k,j));
8          suma=suma + h*y;
9      end
10     valor2=suma;
11 end
```

7.1.9. Método de Trapecio

```
1  %Metodo del trapecio
2  function [valor3] = trapecio(n,x,f,k,h)
3      suma=0;
4      for i=1:n
5          y1=eval(subs(f,k,x(i)));
6          y2=eval(subs(f,k,x(i+1)));
7          suma=suma + h*((y2+y1)/2);
8      end
9      valor3=suma;
10 end
```

7.1.10. Método de Trapecio Corregido

```
1
2  % Metodo del trapecio corregido
3  function [valor5] = trapecio_corregido(n,x,f,g,k,h,a,b)
4      valorr= trapecio(n,x,f,k,h);
5      fa=eval(subs(g,k,a));
6      fb=eval(subs(g,k,b));
7      valor5= valorr + ((h^2)/12)*(fa-fb);
8  end
```

7.1.11. Método de Simpson

```
1  %Metodo de simpson
2  function [valor4] = simpson(n,x,f,k,h)
3      v=n;
4      suma=0;
5      dat1=n/2;
6      dat2=floor(n/2);
7      if (dat1-dat2)>0 & n>1;
8          f1=eval(subs(f,k,x(n-2)));
9          f2=eval(subs(f,k,x(n-1)));
10         f3=eval(subs(f,k,x(n)));
11         f4=eval(subs(f,k,x(n+1)));
```

```
12          [val1]= simpson38(h,f1,f2,f3,f4);
13          suma = suma + val1;
14          v = n-3;
15      end
16      if v >1;
17          [val2]= simpson13(h,v,f,k,x);
18          suma = suma + val2;
19      end
20      valor4=suma;
21
22      %Metodo simpson 3/8
23      function [val1]= simpson38(h,f1,f2,f3,f4);
24          val1= 3*h*((f1+3*(f2+f3)+f4)/8);
25      end
26
27      % Metodo simpson 1/3
28      function [val2]= simpson13(h,v,f,k,x)
29          n=v;
30          suma= eval(subs(f,k,x(1)));
31          for i=1:2:n-2;
32              suma = suma+ 4*eval(subs(f,k,x(i+1)))+ 2*eval(subs(f,k,x(i
                    +2)));
33          end
34          suma = suma + 4*eval(subs(f,k,x(n))) + eval(subs(f,k,x(n+1)));
35          val2 = (h*suma)/3;
36      end
37  end
```

7.1.12. Método de Gauss Legendre

```
1
2   %Metodo de gauss-legendre
3   function [valor6] = gauss_legendre(n,f,k,h,a,b)
4       suma=0;
5       [z,w]=gauleg(n);
6       for i=1:length(z)
7           y=(b+a)/2 + (b-a)*z(i)/2;
8           suma=suma+ w(i)*(eval(subs(f,k,y)));
9       end
10      valor6 =((b-a)/2)*suma;
11      end
12
13      %Calculo de w y z para gauss-legendre
14      function [z,w] = gauleg(n)
15      if n==1
16          x= [0] ;
17          c=[2];
18      end
19      if n==2
```

```
20          x=[-0.577350269 0.577350269];
21          c=[1 1];
22      end
23      if n==3
24          x=[-0.774596669 0.0 0.774596669];
25          c=[0.5555556 0.8888889 0.5555556];
26      end
27      if n==4
28          x=[-0.861136312 -0.339981044 0.339981044 0.861136312];
29          c=[0.3478548 0.6521452 0.6521452 0.3478548];
30      end
31      if n==5
32          x=[-0.906179846 -0.538469310 0.0 0.538469310 0.906179846];
33          c=[0.2369269 0.4786287 0.5688889 0.4786287 0.2369269];
34      end
35      if n==6
36          x=[-0.932469514 -0.661209386 -0.238619186 0.238619186
               0.661209386 0.932469514];
37          c=[0.1713245 0.3607616 0.4679139 0.4679139 0.3607616
               0.1713245];
38      end
39      z=x;
40      w=c;
41      end
```