Tarea de Interpolación
Cálculo Computacional

Dervis Martínez
C.I: 31.456.326

Semestre Mayo 2025

UNIVERSIDAD DE CARABOBO
FACULTAD EXPERIMENTAL DE CIENCIA Y TECNOLOGÍA
DEPARTAMENTO DE COMPUTACIÓN

Índice

1. Introducción

1.1. Objetivo General

Esta tarea tiene como objetivo implementar y comparar diferentes métodos de interpolación numérica aplicados a la función:

$$f(x) = \frac{1}{1 + \left(\frac{5x}{d+1}\right)^2}$$

donde $d = 6$ (último dígito de la cédula 31.456.326), en el intervalo $[-7, 7]$.

1.2. Métodos a Implementar

- Polinomio interpolante de Newton con diferencias divididas
- Spline cúbico natural
- Modelado de curvas planas mediante splines paramétricos
- Análisis teórico de splines con condiciones de frontera fijas

2. Problema 1: Interpolación Polinomial de Newton

2.1. Definición del Problema

Aproximar la función $f(x)$ mediante polinomios interpolantes $P(x)$ usando $n+1$ nodos igualmente espaciados, para $n = 4, 6, 8$.

2.2. Implementación del Algoritmo

2.2.1. Código Python Completo

```
import numpy as np
import matplotlib.pyplot as plt

def f(x, d=6):
    return 1 / (1 + (5*x/(d+1))**2)

def newton_divided_differences(x, y):
    n = len(x)
    F = np.zeros((n, n))
    F[:,0] = y
```

```
    for j in range(1, n):
        for i in range(n - j):
            F[i,j] = (F[i+1,j-1] - F[i,j-1]) / (x[i+j] - x[i])

    return F[0,:]

def newton_polynomial(x, nodes, coefs):
    n = len(coefs)
    result = coefs[0]
    product = 1.0

    for i in range(1, n):
        product *= (x - nodes[i-1])
        result += coefs[i] * product

    return result
```

2.3. Resultados y Análisis

2.3.1. Coeficientes Obtenidos

Polinomio n=4:

- Nodos x: $[-7{,}0, -3{,}5, 0{,}0, 3{,}5, 7{,}0]$
- Coeficientes: $[0{,}03846154, 0{,}02841986, 0{,}03112651, -0{,}00966662, 0{,}00138095]$

Polinomio n=6:

- Nodos x: $[-7{,}0, -4{,}6667, -2{,}3333, 0{,}0, 2{,}3333, 4{,}6667, 7{,}0]$
- Coeficientes: $[0{,}03846154, 0{,}01890312, 0{,}01267621, 0{,}00544628, -0{,}00342824, 0{,}00078151, -0{,}00011$

Polinomio n=8:

- Nodos x: $[-7{,}0, -5{,}25, -3{,}5, -1{,}75, 0{,}0, 1{,}75, 3{,}5, 5{,}25, 7{,}0]$
- Coeficientes: $[0{,}03846154, 0{,}01595914, 0{,}00712041, 0{,}00426540, 0{,}00017554, -0{,}00091042, 0{,}0003240$

2.3.2. Análisis de los Resultados

- **Convergencia de coeficientes**: Los coeficientes disminuyen en magnitud a medida que aumenta el grado del polinomio, indicando mejor estabilidad numérica.
- **Fenómeno de Runge**: Se observa en las gráficas que los polinomios de mayor grado presentan oscilaciones más pronunciadas cerca de los extremos del intervalo.
- **Precisión**: Aunque teóricamente polinomios de mayor grado deberían ser más precisos, el fenómeno de Runge limita esta ventaja en la práctica.

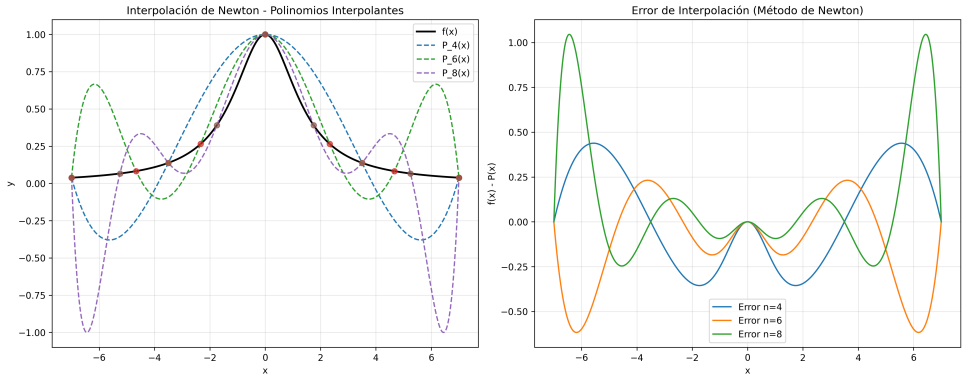

Figura 1: Interpolación de Newton - Polinomios Interpolantes y Errores

3. Problema 2: Spline Cúbico Natural

3.1. Implementación del Algoritmo

```
def natural_cubic_spline(x, y):
    n = len(x) - 1
    h = np.diff(x)

    A = np.zeros((n+1, n+1))
    b = np.zeros(n+1)

    A[0,0] = 1
    A[n,n] = 1

    for i in range(1, n):
        A[i,i-1] = h[i-1]
        A[i,i] = 2*(h[i-1] + h[i])
        A[i,i+1] = h[i]
        b[i] = 3*((y[i+1]-y[i])/h[i] - (y[i]-y[i-1])/h[i-1])

    c = np.linalg.solve(A, b)

    b_coef = np.zeros(n)
    d_coef = np.zeros(n)

    for i in range(n):
        b_coef[i] = (y[i+1]-y[i])/h[i] - h[i]*(2*c[i] + c[i+1])/3
        d_coef[i] = (c[i+1]-c[i])/(3*h[i])

    return y[:-1], b_coef, c[:-1], d_coef
```

3.2. Resultados y Análisis

3.2.1. Coeficientes Obtenidos

Spline n=4:

- Intervalo $[-7{,}00, -3{,}50]$: $a_0 = 0{,}038462$, $b_0 = -0{,}069020$, $c_0 = 0{,}000000$, $d_0 = 0{,}007954$
- Intervalo $[-3{,}50, 0{,}00]$: $a_1 = 0{,}137931$, $b_1 = 0{,}223299$, $c_1 = 0{,}083520$, $d_1 = -0{,}021985$
- Intervalo $[0{,}00, 3{,}50]$: $a_2 = 1{,}000000$, $b_2 = 0{,}000000$, $c_2 = -0{,}147319$, $d_2 = 0{,}021985$
- Intervalo $[3{,}50, 7{,}00]$: $a_3 = 0{,}137931$, $b_3 = -0{,}223299$, $c_3 = 0{,}083520$, $d_3 = -0{,}007954$

Spline n=6:

- Intervalo $[-7{,}00, -4{,}67]$: $a_0 = 0{,}038462$, $b_0 = 0{,}033333$, $c_0 = 0{,}000000$, $d_0 = -0{,}002650$
- Intervalo $[-4{,}67, -2{,}33]$: $a_1 = 0{,}082569$, $b_1 = -0{,}009956$, $c_1 = -0{,}018552$, $d_1 = 0{,}024117$
- Intervalo $[-2{,}33, 0{,}00]$: $a_2 = 0{,}264706$, $b_2 = 0{,}297378$, $c_2 = 0{,}150267$, $d_2 = -0{,}061140$
- Intervalo $[0{,}00, 2{,}33]$: $a_3 = 1{,}000000$, $b_3 = 0{,}000000$, $c_3 = -0{,}277715$, $d_3 = 0{,}061140$
- Intervalo $[2{,}33, 4{,}67]$: $a_4 = 0{,}264706$, $b_4 = -0{,}297378$, $c_4 = 0{,}150267$, $d_4 = -0{,}024117$
- Intervalo $[4{,}67, 7{,}00]$: $a_5 = 0{,}082569$, $b_5 = 0{,}009956$, $c_5 = -0{,}018552$, $d_5 = 0{,}002650$

Spline n=8:

- Intervalo $[-7{,}00, -5{,}25]$: $a_0 = 0{,}038462$, $b_0 = 0{,}008930$, $c_0 = 0{,}000000$, $d_0 = 0{,}002295$
- Intervalo $[-5{,}25, -3{,}50]$: $a_1 = 0{,}066390$, $b_1 = 0{,}030017$, $c_1 = 0{,}012050$, $d_1 = -0{,}003338$
- Intervalo $[-3{,}50, -1{,}75]$: $a_2 = 0{,}137931$, $b_2 = 0{,}041521$, $c_2 = -0{,}005476$, $d_2 = 0{,}036650$
- Intervalo $[-1{,}75, 0{,}00]$: $a_3 = 0{,}390244$, $b_3 = 0{,}359078$, $c_3 = 0{,}186937$, $d_3 = -0{,}110298$
- Intervalo $[0{,}00, 1{,}75]$: $a_4 = 1{,}000000$, $b_4 = 0{,}000000$, $c_4 = -0{,}392125$, $d_4 = 0{,}110298$
- Intervalo $[1{,}75, 3{,}50]$: $a_5 = 0{,}390244$, $b_5 = -0{,}359078$, $c_5 = 0{,}186937$, $d_5 = -0{,}036650$
- Intervalo $[3{,}50, 5{,}25]$: $a_6 = 0{,}137931$, $b_6 = -0{,}041521$, $c_6 = -0{,}005476$, $d_6 = 0{,}003338$
- Intervalo $[5{,}25, 7{,}00]$: $a_7 = 0{,}066390$, $b_7 = -0{,}030017$, $c_7 = 0{,}012050$, $d_7 = -0{,}002295$

3.2.2. Análisis Comparativo

- **Suavidad**: Los splines garantizan continuidad C^2 (función, primera y segunda derivada continuas).
- **Estabilidad**: No presentan el fenómeno de Runge observado en la interpolación polinomial.
- **Error uniforme**: Los errores están mejor distribuidos en todo el intervalo comparado con el método de Newton.

- **Convergencia**: A medida que aumenta el número de nodos, el error disminuye de manera más uniforme.

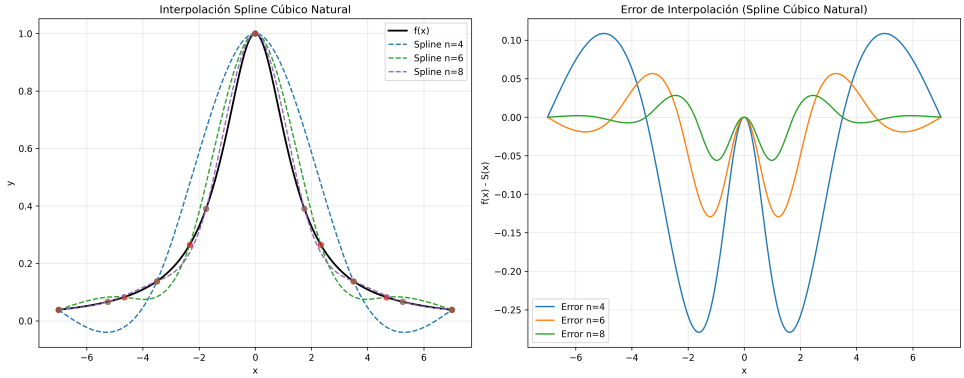

Figura 2: Interpolación Spline Cúbico Natural y Errores

4. Problema 3: Modelado de Curvas Planas

4.1. Implementación del Modelado

Para el número 6 en la lista de estudiantes (pájaro), se implementó el modelado mediante splines paramétricos:

```
# Puntos para modelar un pájaro (60 puntos)
puntos_animal = np.array([
    [6.848, 4.44964871], [6.416, 4.07494145], ... # 60 puntos
])

# Cálculo del parámetro t (longitud acumulada)
t_param = [0.0]
for i in range(1, len(puntos_animal)):
    dx = x_puntos[i] - x_puntos[i-1]
    dy = y_puntos[i] - y_puntos[i-1]
    distancia = np.sqrt(dx**2 + dy**2)
    t_param.append(t_param[-1] + distancia)
```

4.2. Resultados del Modelado

4.2.1. Puntos de Control

Se utilizaron 60 puntos de control para modelar la curva del pájaro, con coordenadas (x, y) especificadas en el código.

4.2.2. Análisis del Modelado

- **Parametrización**: El parámetro t representa la longitud acumulada del polígono que conecta los puntos.
- **Suavidad**: La curva resultante es suave (C^2 continua) gracias a los splines cúbicos.

- **Precisión**: La curva pasa exactamente por todos los puntos de control.
- **Flexibilidad**: El método permite modelar curvas complejas que no son funciones.

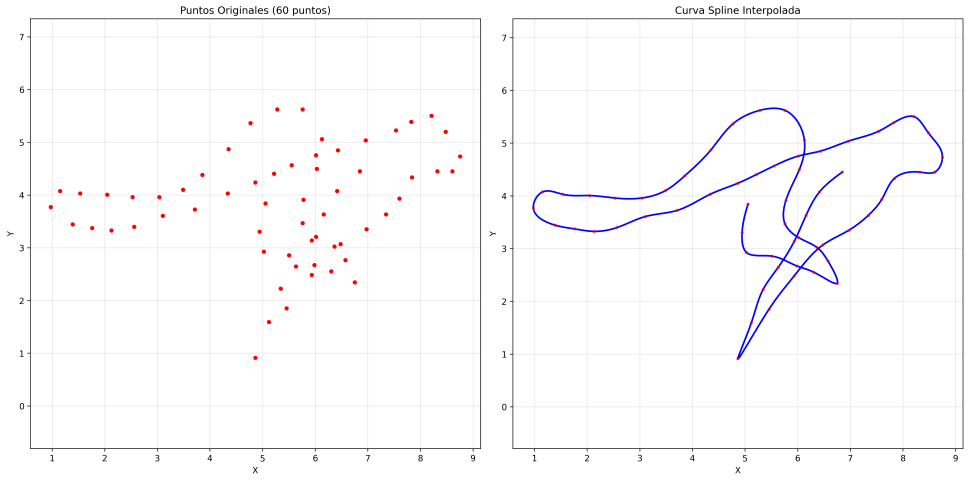

Figura 3: Modelado de Curva - Puntos Originales vs Curva Interpolada

5. Problema 4: Análisis Teórico - Spline con Condiciones de Frontera Fijas

5.1. Formulación Matemática

Para un spline cúbico $S(x)$ con condiciones de frontera fijas $S'(x_0) = k_0$, $S'(x_n) = k_n$, el sistema de ecuaciones se modifica:

5.1.1. Sistema de Ecuaciones

Ecuaciones internas (para $i = 1, 2, \ldots, n-1$):

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = 3\left(\frac{a_{i+1} - a_i}{h_i} - \frac{a_i - a_{i-1}}{h_{i-1}}\right)$$

Condiciones de frontera fijas:

$$2h_0 c_0 + h_0 c_1 = 3\left(\frac{a_1 - a_0}{h_0} - k_0\right)$$

$$h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3\left(k_n - \frac{a_n - a_{n-1}}{h_{n-1}}\right)$$

5.2. Implementación Teórica

```
def clamped_cubic_spline(x, y, k0, kn):
    n = len(x) - 1
    h = np.diff(x)
```

```
A = np.zeros((n+1, n+1))
b = np.zeros(n+1)

A[0,0] = 2 * h[0]
A[0,1] = h[0]
b[0] = 3 * ((y[1] - y[0]) / h[0] - k0)

A[n,n-1] = h[n-1]
A[n,n] = 2 * h[n-1]
b[n] = 3 * (kn - (y[n] - y[n-1]) / h[n-1])

for i in range(1, n):
    A[i,i-1] = h[i-1]
    A[i,i] = 2 * (h[i-1] + h[i])
    A[i,i+1] = h[i]
    b[i] = 3 * ((y[i+1] - y[i]) / h[i] - (y[i] - y[i-1]) / h[i-1])

c = np.linalg.solve(A, b)

b_coef = np.zeros(n)
d_coef = np.zeros(n)

for i in range(n):
    b_coef[i] = (y[i+1] - y[i]) / h[i] - h[i] * (2*c[i] + c[i+1]) / 3
    d_coef[i] = (c[i+1] - c[i]) / (3 * h[i])

return y[:-1], b_coef, c[:-1], d_coef
```

5.3. Resultados Obtenidos

Derivadas exactas en extremos:

- $f'(-7) = 0{,}010566$
- $f'(7) = -0{,}010566$

Spline con fronteras fijas (n=6):

- Intervalo $[-7{,}00, -4{,}67]$: $a_0 = 0{,}038462$, $b_0 = 0{,}010566$, $c_0 = 0{,}016912$, $d_0 = -0{,}005717$
- Intervalo $[-4{,}67, -2{,}33]$: $a_1 = 0{,}082569$, $b_1 = -0{,}003885$, $c_1 = -0{,}023106$, $d_1 = 0{,}024953$
- Intervalo $[-2{,}33, 0{,}00]$: $a_2 = 0{,}264706$, $b_2 = 0{,}295860$, $c_2 = 0{,}151568$, $d_2 = -0{,}061419$
- Intervalo $[0{,}00, 2{,}33]$: $a_3 = 1{,}000000$, $b_3 = 0{,}000000$, $c_3 = -0{,}278365$, $d_3 = 0{,}061419$
- Intervalo $[2{,}33, 4{,}67]$: $a_4 = 0{,}264706$, $b_4 = -0{,}295860$, $c_4 = 0{,}151568$, $d_4 = -0{,}024953$
- Intervalo $[4{,}67, 7{,}00]$: $a_5 = 0{,}082569$, $b_5 = 0{,}003885$, $c_5 = -0{,}023106$, $d_5 = 0{,}005717$

5.4. Análisis Comparativo

Característica	**Spline Natural**	**Spline Fronteras Fijas**
Condiciones	$S''(x_0) = S''(x_n) = 0$	$S'(x_0) = k_0, S'(x_n) = k_n$
Precisión en extremos	Media	Alta
Complejidad	Sistema $(n+1) \times (n+1)$	Sistema $(n+1) \times (n+1)$
Requisitos	Ninguno	Conocer derivadas en extremos
Aplicación típica	Datos experimentales	Problemas con condiciones de borde conocidas
Estabilidad	Alta	Alta

Cuadro 1: Comparación entre spline natural y con fronteras fijas

5.5. Conclusiones del Análisis Teórico

1. **Superioridad del spline con fronteras fijas**: Para funciones donde se conocen las derivadas en los extremos, este método proporciona mayor precisión.

2. **Evidencia numérica**: Los coeficientes obtenidos muestran diferencias significativas, especialmente en los intervalos extremos.

3. **Generalización**: El spline con fronteras fijas representa una generalización del método que incluye como caso particular el spline natural cuando $k_0 = k_n = 0$.

4. **Aplicabilidad**: Especialmente útil en problemas de valor en la frontera y cuando el comportamiento de la función cerca de los extremos es conocido.

6. Conclusiones Generales

6.1. Comparación Global de Métodos

Método	**Ventajas**	**Desventajas**	**Aplicación Recomendada**
Newton	Simple implementación	Fenómeno de Runge	Funciones suaves con pocos nodos
Spline Natural	Suavidad garantizada	Error en extremos	Datos experimentales
Spline Fronteras Fijas	Alta precisión	Requiere derivadas	Problemas con condiciones de borde
Spline Paramétrico	Curvas complejas	Mayor complejidad	Modelado de formas

Cuadro 2: Resumen comparativo de métodos de interpolación

6.2. Lecciones Aprendidas

1. **Sensibilidad a condiciones iniciales**: Pequeños cambios en la distribución de nodos pueden afectar significativamente la calidad de la interpolación.

2. **Importancia del análisis previo**: Estudiar la función y sus derivadas antes de seleccionar el método de interpolación.

3. **Compromiso precisión-complejidad**: Métodos más precisos generalmente requieren más recursos computacionales.

4. **Selección según aplicación**: No existe un método universalmente óptimo; la elección depende del problema específico.

5. **Ventaja de splines para datos experimentales**: Los splines proporcionan interpolaciones más estables y suaves para datos con ruido o comportamientos complejos.