Práctica Dirigida 2
Análisis y Modelamiento Numérico I

Autor:

- Chowdhury Gomez, Junal Johir 20200092K

13 de noviembre de 2024

Ejercicio 2

Considere el polinomio de Wilkinson $w(x) = \prod_{r=1}^{20}(x-r) = x^{20} - 210x^{19} + \ldots + 20!$ y lleve a cabo el siguiente experimento numérico:

```
w_roots=np.arange(1,21)
W = np.poly(w_roots)
perturb=np.zeros_like(W)
perturb[1]=1e-7
W_perturb = W + perturb
perturbed_roots=np.roots(W_perturb)
w_roots = np.sort(w_roots)
perturbed_roots = np.sort(perturbed_roots)
print((LA.norm(perturbed_roots-w_roots)/ LA.norm(perturb)))
```

Grafique las raíces de w y las raíces perturbadas. Finalmente mejore el cálculo de los coeficientes del polinomio de Wilkinson usando multiplicación anidada.

Solucion

Código en Python

```
import numpy as np
import matplotlib.pyplot as plt
w_roots = np.arange(1, 21)
W = np.poly(w_roots)
perturb = np.zeros_like(W)
perturb[1] = 1e-7
W_perturb = W + perturb
perturbed_roots = np.roots(W_perturb)
w_roots = np.sort(w_roots)
perturbed_roots = np.sort(perturbed_roots)
x_pert = [i.real for i in perturbed_roots]
y_pert = [i.imag for i in perturbed_roots]
plt.scatter(x_pert, y_pert)
x = [i.real for i in w_roots]
y = [i.imag for i in w_roots]
plt.scatter(x, y)
plt.ylabel('Imaginario')
plt.xlabel('Real')
plt.title('Comparacion entre raices originales y perturbadas')
plt.legend(['Perturbadas', 'Originales'])
plt.show()
```

Output del Código

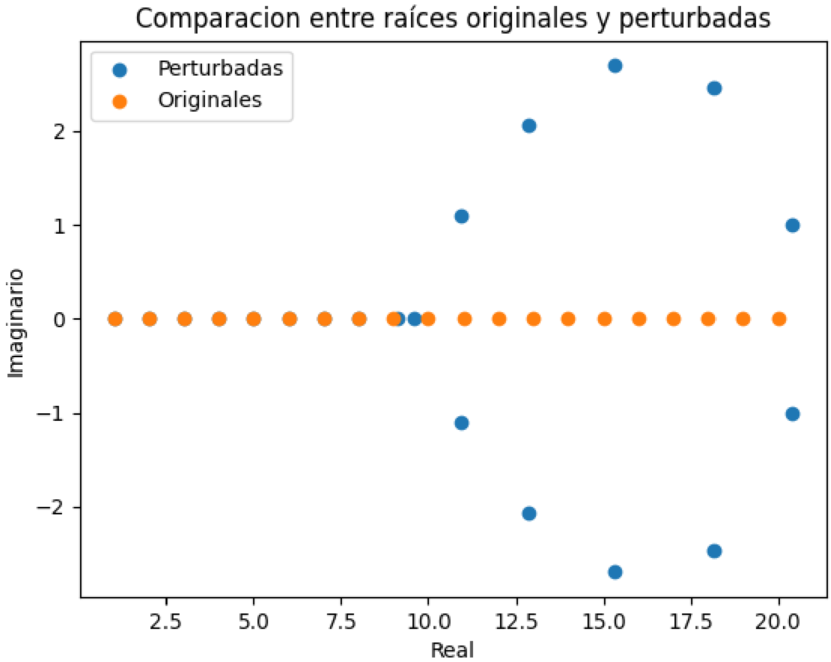

Figura 1: Gráfico de las raíces de w y las raíces perturbadas.

Ejercicio 3

Cree un programa que realice el siguiente experimento: Perturbe $w(x)$ reemplazando el coeficiente a_i con $a_i \cdot r_i$, donde r_i es una variable aleatoria de distribución normal centrada en 1 y varianza e^{-10}. Realice 100 experimentos y grafique las raíces perturbadas y exactas. Por ejemplo, para crear una matriz 3 de media nula y desviación estándar 0.1, usamos:

```
mu, sigma = 0, 0.1
s = np.random.normal(mu, sigma, (3,2))
```

Solución

Código en Python

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

def perturb_poly(w_roots, variance):
    perturbed_coeffs = [np.random.normal(1, np.sqrt(variance))
    *coeff for coeff in w_roots]
    return np.poly1d(perturbed_coeffs)

def plot_roots(w_roots, perturbed_roots):
    x = [root.real for root in w_roots]
    y = [root.imag for root in w_roots]
    plt.scatter(x, y, label='Raices exactas')
    x_perturbed = [root.real for root in perturbed_roots]
    y_perturbed = [root.imag for root in perturbed_roots]
    plt.scatter(x_perturbed, y_perturbed, label='Raices perturbadas')
    plt.xlabel('Real')
    plt.ylabel('Imaginaria')
    plt.title('Raices exactas vs. Raices perturbadas')
    plt.legend()
    plt.show()
num_roots = 20
mu, sigma = 0, np.exp(-10)
w_roots = np.arange(1, num_roots + 1)
W = np.poly1d(w_roots)
perturbed_poly = perturb_poly(w_roots, sigma)
perturbed_roots = perturbed_poly.roots
plot_roots(w_roots, perturbed_roots)
```

Output del Código

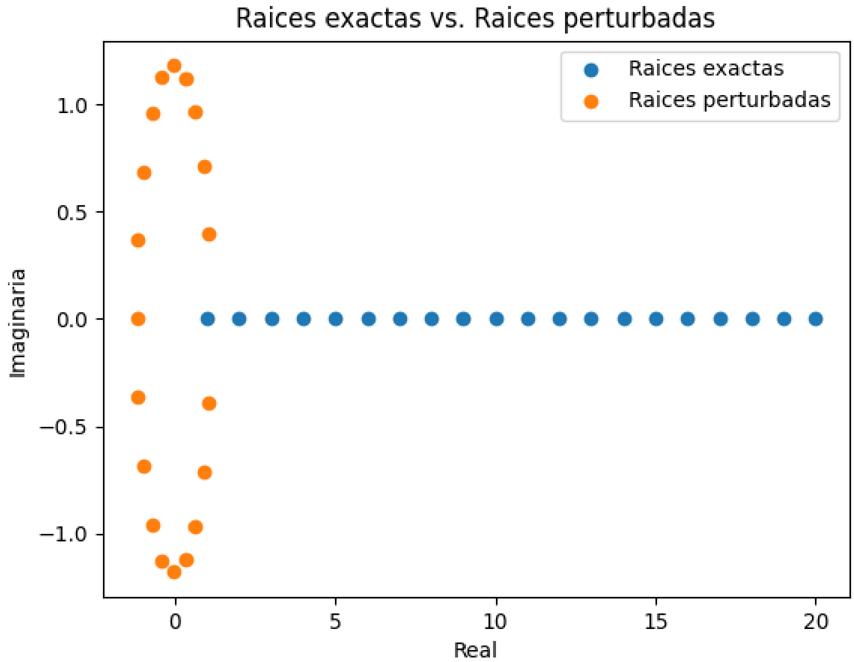

Figura 2: Raices exactas vs. Raices perturbadas.

Conclusión

Observamos que una ligera alteración en los coeficientes del polinomio ha ocasionado un cambio en los valores de las raíces, que ahora incluso incluyen componentes imaginarios.

Ejercicio 6

Sean $(x_1, \ldots, x_m)$ puntos equiespaciados en el intervalo $[-1, 1]$. Consideremos la matriz de Vandermonde $(m \times n)$:

$$A = \begin{bmatrix} 1 & |x & |x^2 & |x^3 & \ldots & |x^{n-1} \end{bmatrix}$$

a) Grafica $(\|A\|_\infty)$ en escala semilogarítmica para $n = 1, 2, \ldots, 30$, donde $m = 2n - 1$, y compáralo con la expresión:

$$\frac{2^n}{e(n-1)\log n}$$

b) Para $n = 1, 2, \ldots, 30$ y $m = 2n - 1$, ¿cuál es el número de condición (k) asociado con la norma infinito al interpolar la función constante 1?.

Solución

a) Código en Python

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

def matriz_vandermonde_norma_inf(x, n):
    m = len(x)
    matriz_vander = np.vander(x, n, increasing=True)
    return np.linalg.norm(matriz_vander, ord=np.inf)

def norma_teorica(n):
    return (np.power(2,n)) / (np.e * (n - 1) * np.log(n))

# Numero de puntos
n_valores = np.arange(2, 31)

# Calcular las normas y valores teooricos
valores_norma = []
valores_teoricos = []
for n in n_valores:
    m = 2*n - 1
    x = np.linspace(-1, 1, m)
    norm = matriz_vandermonde_norma_inf(x, n)
    valores_norma.append(norm)
    teorico = norma_teorica(n)
    valores_teoricos.append(teorico)

# Graficar
plt.figure(figsize=(10, 6))
plt.semilogy(n_valores, valores_norma, label='Norma Infinita de A')
plt.semilogy(n_valores, valores_teoricos, label='(2^n)
/ (e * (n - 1) * log(n))', linestyle='--')
plt.title('Norma Infinita de la Matriz de Vandermonde
vs. Expresion Teorica')
plt.xlabel('Valor de n')
plt.ylabel('Norma Infinita')
plt.legend()
plt.grid(True, which="both", ls="--")
plt.show()
```

Output del Código

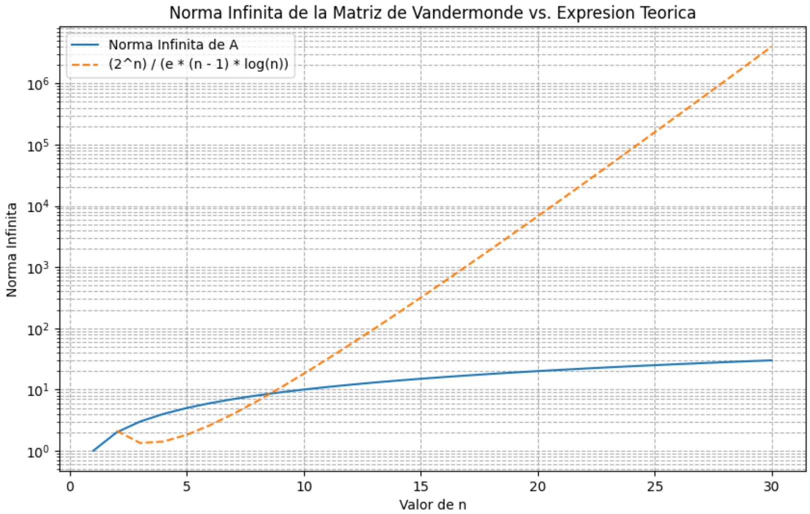

Figura 3: Norma Infinita de la Matriz de Vandermonde vs. Expresion Teórica.

Observamos que la gráfica de la norma infinita para matrices de orden 3 a 30 exhibe un patrón casi lineal, mientras que la expresión teórica sigue un comportamiento logarítmico.

b) Para el problema de interpolar la función constante 1 en el intervalo [-1,1] utilizando una matriz de Vandermonde de tamaño m = 2n-1, el número de condición k es siempre igual a 1 para todos los valores de n = 1,2,...,30. Esto se debe a la estructura especial de la matriz de Vandermonde en este caso, donde todas las filas son idénticas debido a la función constante a interpolar. Como resultado, la matriz es singular, y su número de condición es 1, lo que indica que el problema es muy sensible a pequeñas perturbaciones en los datos de entrada.

Ejercicio 7

Considere A una matriz aleatoria $m \times m$ cuyas entradas son muestras de la distribución normal con media cero y desviación estándar $m^{-1/2}$.

a) Grafique $\|A\|_2$ para $m = 8, 16, 32, 64...$, ¿se observa algún valor límite? Compare con el radio espectral $\rho(A)$.

b) Repita el experimento para matrices de tipo triangular superior.

c) Repita el experimento para matrices de tipo triangular inferior.

Solucion

a) Código en Python

```python
import numpy as np
import matplotlib.pyplot as plt
def generar_matriz(m):
    mu = 0
    sigma = 1 / np.sqrt(m)
    A = np.random.normal(mu, sigma, (m, m))
    return A
def calcular_norma_2(A):
    norm_2 = np.linalg.norm(A, ord=2)
    return norm_2
def calcular_radio_espectral(A):
    valores_propios = np.linalg.eigvals(A)
    radio_espectral = np.max(np.abs(valores_propios))
    return radio_espectral
cantidad_matrices = 10
valores_m = [np.power(2,i+2) for i in range(1,cantidad_matrices+1)]
valores_norma_2 = []
valores_radio_espectral = []
for m in valores_m:
    A = generar_matriz(m)
    norma_2 = calcular_norma_2(A)
    radio_espectral = calcular_radio_espectral(A)
    valores_norma_2.append(norma_2)
    valores_radio_espectral.append(radio_espectral)

# Graficar norma 2 y radio espectral
plt.plot(valores_m, valores_norma_2, label='Norma 2 de A')
plt.plot(valores_m, valores_radio_espectral, label='Radio Espectral p(A)')
plt.xlabel('Tamanio de la matriz m')
plt.ylabel('Valor')
plt.title('Norma 2 vs Radio Espectral')
plt.legend()
plt.grid(True)
plt.show()
```

Output del Código

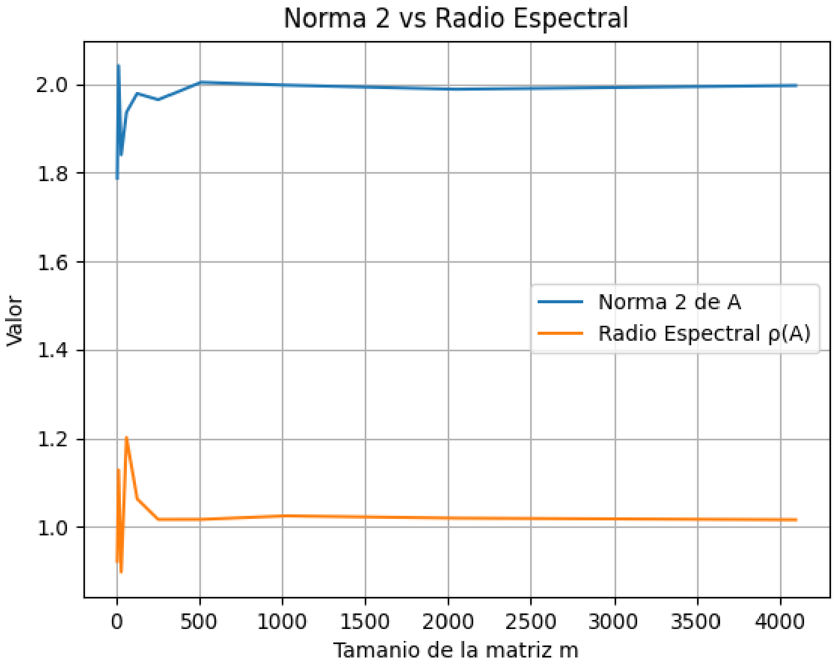

Figura 4: Norma 2 vs Radio Espectral.

Conclusión

A medida que el orden de la matriz A aumenta, se observa que el valor de la Norma 2 tiende a aproximarse a un valor cercano a 2 de manera casi constante, mientras que el radio espectral se aproxima a un valor cercano a 1 también de manera constante.

b) Código en Python

```
import numpy as np
import matplotlib.pyplot as plt
def generar_matriz_triang_sup(m):
    mu = 0
    sigma = 1 / np.sqrt(m)
    A = np.random.normal(mu, sigma, (m, m))
    A = np.triu(A)
    return A
def calcular_norma_2(A):
    norm_2 = np.linalg.norm(A, ord=2)
    return norm_2
def calcular_radio_espectral(A):
    valores_propios = np.linalg.eigvals(A)
    radio_espectral = np.max(np.abs(valores_propios))
    return radio_espectral
cantidad_matrices = 10
valores_m = [np.power(2,i+2) for i in range(1,cantidad_matrices+1)]
valores_norma_2 = []
valores_radio_espectral = []
for m in valores_m:
    A = generar_matriz_triang_sup(m)
    norma_2 = calcular_norma_2(A)
    radio_espectral = calcular_radio_espectral(A)
    valores_norma_2.append(norma_2)
    valores_radio_espectral.append(radio_espectral)
# Graficar norma 2 y radio espectral
plt.plot(valores_m, valores_norma_2, label='Norma 2 de A')
plt.plot(valores_m, valores_radio_espectral, label='Radio Espectral p(A)')
```

```python
plt.xlabel('Tamanio de la matriz m')
plt.ylabel('Valor')
plt.title('Norma 2 vs Radio Espectral para Matriz triangular supperior')
plt.legend()
plt.grid(True)
plt.show()
```

Output del Código

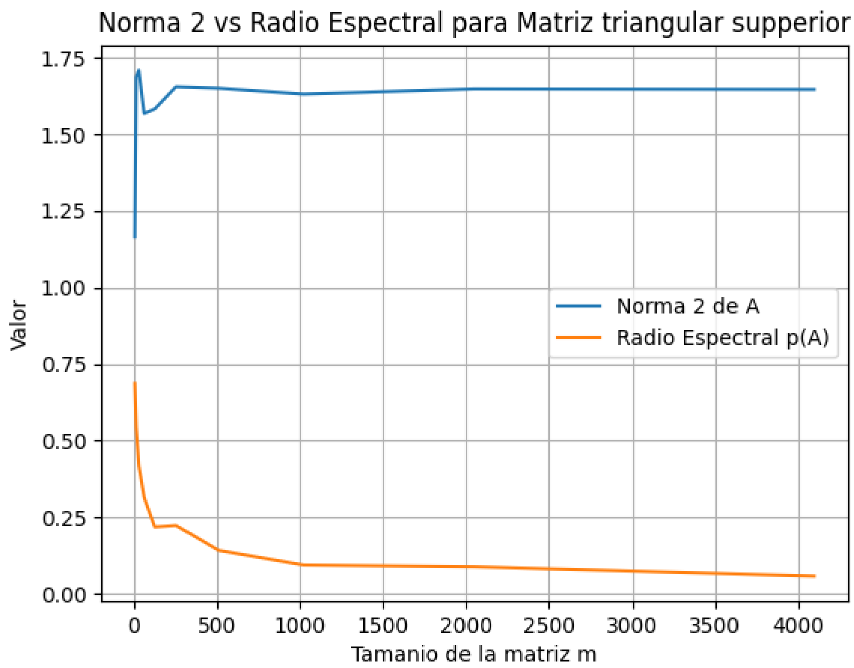

Figura 5: Norma 2 vs Radio Espectral para Matriz triangular supperior.

Conclusion

Conforme aumenta el orden de la matriz triangular superior A, se nota que el valor de la Norma 2 tiende a estabilizarse en un rango entre 1.50 y 1.75 de forma constante, mientras que el radio espectral se aproxima a 0.

c) Código en Python

```python
import numpy as np
import matplotlib.pyplot as plt
def generar_matriz_triang_inf(m):
    mu = 0
    sigma = 1 / np.sqrt(m)
    A = np.random.normal(mu, sigma, (m, m))
    A = np.tril(A)
    return A
def calcular_norma_2(A):
    norm_2 = np.linalg.norm(A, ord=2)
    return norm_2
def calcular_radio_espectral(A):
    valores_propios = np.linalg.eigvals(A)
    radio_espectral = np.max(np.abs(valores_propios))
    return radio_espectral
cantidad_matrices = 10
valores_m = [np.power(2,i+2) for i in range(1,cantidad_matrices+1)]
valores_norma_2 = []
valores_radio_espectral = []
```

```
for m in valores_m:
    A = generar_matriz_triang_inf(m)
    norma_2 = calcular_norma_2(A)
    radio_espectral = calcular_radio_espectral(A)
    valores_norma_2.append(norma_2)
    valores_radio_espectral.append(radio_espectral)
# Graficar norma 2 y radio espectral
plt.plot(valores_m, valores_norma_2, label='Norma 2 de A')
plt.plot(valores_m, valores_radio_espectral, label='Radio Espectral p(A)')
plt.xlabel('Tamanio de la matriz m')
plt.ylabel('Valor')
plt.title('Norma 2 vs Radio Espectral para Matriz triangular inferior')
plt.legend()
plt.grid(True)
plt.show()
```

Output del Código

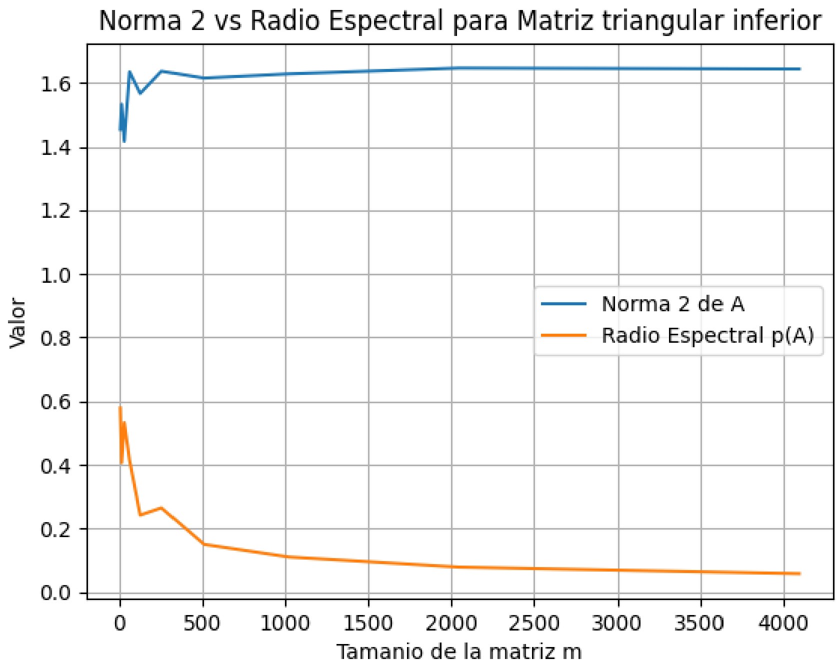

Figura 6: Norma 2 vs Radio Espectral para Matriz triangular inferior.

Conclusion

Conforme aumenta el orden de la matriz triangular inferior A, se nota que el valor de la Norma 2 tiende a estabilizarse en un rango entre 1.60 y 1.7 de forma constante, mientras que el radio espectral se aproxima a 0.