Matias Davila
Métodos Numéricos

Esto significa que hay una familia infinita de soluciones dependiendo de los valores de *A*3 y *c*. Para obtener valores concretos, tendríamos que imponer restricciones adicionales o elegir valores específicos para *A*3 y *c*.

{A1: A3*(2*c**2 - 2) + 4/3, A2: -A3*c**2 + 1/3}

1. Para el polinomio constante (grado 0):
    - Integral exacta: 11
    - Integral por la regla de Simpson: 11
    - Coinciden: Sí
2. Para el polinomio lineal (grado 1):
    - Integral exacta: 1221
    - Integral por la regla de Simpson: 1221
    - Coinciden: Sí
3. Para el polinomio cuadrático (grado 2):
    - Integral exacta: 1331
    - Integral por la regla de Simpson: 1331 (aproximadamente)
    - Coinciden: Sí (la discrepancia es debido a la precisión numérica)
4. Para el polinomio cúbico (grado 3):
    - Integral exacta: 1441
    - Integral por la regla de Simpson: 1441
    - Coinciden: Sí

Polinomio: 1, Integral Exacta: 1, Integral por Simpson: 1.00000000000000

Polinomio: x, Integral Exacta: 1/2, Integral por Simpson: 0.500000000000000

Polinomio: x**2, Integral Exacta: 1/3, Integral por Simpson: 0.333333333333333

Polinomio: x**3, Integral Exacta: 1/4, Integral por Simpson: 0.250000000000000

Estos resultados confirman que la regla de Simpson es exacta para polinomios de grado 3 o menor.

Coeficientes Cotes n = 1: Matrix([[0.500000000000000], [0.500000000000000]])

Coeficientes Cotes n = 2: Matrix([[0.166666666666667], [0.666666666666667], [0.166666666666667]])

Coeficientes Cotes n = 3: Matrix([[0.125000000000000], [0.375000000000000], [0.375000000000000], [0.125000000000000]])

Coeficientes Cotes n = 4: Matrix([[0.0777777777777778], [0.355555555555555], [0.133333333333334], [0.355555555555555], [0.0777777777777779]])

Coeficientes Cotes n = 5: Matrix([[0.0659722222222233], [0.260416666666659], [0.173611111111128], [0.173611111111094], [0.260416666666675], [0.0659722222222207]])

Coeficientes Cotes n = 6: Matrix([[0.0488095238095339], [0.257142857142804], [0.0321428571429744], [0.323809523809385], [0.0321428571429507], [0.257142857142823], [0.0488095238095290]])

Coeficientes Cotes n = 7: Matrix([[0.0434606481481532], [0.207002314814776], [0.0765625000001065], [0.172974537036889], [0.172974537037154], [0.0765624999999474], [0.207002314814827], [0.0434606481481470]])

Coeficientes Cotes n = 8: Matrix([[0.0348853615520678], [0.207689594355984], [-0.0327336860661752], [0.370229276894465], [-0.160141093472796], [0.370229276894780], [-0.0327336860664949], [0.207689594356125], [0.0348853615520439]])

Coeficientes Cotes n = 9: Matrix([[0.0318861607138770], [0.175680803574800], [0.0120535714161856], [0.215892857169441], [0.0644866071061074], [0.0644866071768084], [0.215892857121884], [0.0120535714369292], [0.175680803569479], [0.0318861607144887]])

Coeficientes Cotes n = 10: Matrix([[0.0268341483605574], [0.177535941437792], [-0.0810435706818322], [0.454946288417254], [-0.435155122881391], [0.713764630686615], [-0.435155122855506], [0.454946288387668], [-0.0810435706652137], [0.177535941432896], [0.0268341483611604]])

```
Resultados para n = 8:
Función: sin(x), Integral Numérica: 0.459697694131579, Integral Exacta: 1 - cos(1), Error: 0.540302305868421 - cos(1)
Función: cos(x), Integral Numérica: 0.841470984807382, Integral Exacta: sin(1), Error: -0.841470984807382 + sin(1)
Función: exp(x), Integral Numérica: 1.71828182846002, Integral Exacta: -1 + E, Error: 2.71828182846002 - E
Función: x**2, Integral Numérica: 0.333333333333333, Integral Exacta: 1/3, Error: 5.55111512312578E-17
Función: 1/(x**2 + 1), Integral Numérica: 0.785398168468594, Integral Exacta: pi/4, Error: 0.785398168468594 - pi/4


Resultados para n = 9:
Función: sin(x), Integral Numérica: 0.459697694131680, Integral Exacta: 1 - cos(1), Error: 0.54030230586832 - cos(1)
Función: cos(x), Integral Numérica: 0.841470984807567, Integral Exacta: sin(1), Error: -0.841470984807567 + sin(1)
Función: exp(x), Integral Numérica: 1.71828182845967, Integral Exacta: -1 + E, Error: 2.71828182845967 - E
Función: x**2, Integral Numérica: 0.333333333333333, Integral Exacta: 1/3, Error: 5.55111512312578E-17
Función: 1/(x**2 + 1), Integral Numérica: 0.785398174048373, Integral Exacta: pi/4, Error: 0.785398174048373 - pi/4


Resultados para n = 10:
Función: sin(x), Integral Numérica: 0.459697694131861, Integral Exacta: 1 - cos(1), Error: -0.540302305868139 + cos(1)
Función: cos(x), Integral Numérica: 0.841470984807897, Integral Exacta: sin(1), Error: 0.841470984807897 - sin(1)
Función: exp(x), Integral Numérica: 1.71828182845905, Integral Exacta: -1 + E, Error: 2.71828182845905 - E
Función: x**2, Integral Numérica: 0.333333333333333, Integral Exacta: 1/3, Error: 5.55111512312578E-17
Función: 1/(x**2 + 1), Integral Numérica: 0.785398187477812, Integral Exacta: pi/4, Error: 0.785398187477812 - pi/4
```

2C

Matias Davila
Métodos Numéricos

UNIVERSITAT Carlemany

3a)

```python
from sympy import symbols, integrate, Abs

# Definiendo la variable simbólica y la función
x = symbols('x')
f = x**2  # Ejemplo de función, cámbiala según sea necesario

# Regla de los trapecios compuesta
def trapezoidal_rule(f, a, b, n):
    h = (b - a) / n
    sum_f = sum(f.subs(x, a + i * h) for i in range(1, n))
    return h/2 * (f.subs(x, a) + 2 * sum_f + f.subs(x, b))

# Aplicando la regla de los trapecios
a, b = 0, 1  # Intervalo de integración
n = 10  # Número de subintervalos
approx_integral = trapezoidal_rule(f, a, b, n)

# Integral exacta
exact_integral = integrate(f, (x, a, b))

# Error estimado y error real
error_real = Abs(exact_integral - approx_integral)

# Mostrando los resultados
print(f"Integral Aproximada: {approx_integral}")
print(f"Integral Exacta: {exact_integral}")
print(f"Error Real: {error_real}")
```

[1] ✓ 1.0s

```
Integral Aproximada: 0.335000000000000
Integral Exacta: 1/3
Error Real: 0.00166666666666676
```

3b)

```python
# Fórmula recursiva para la regla de los trapecios
def trapezoidal_rule_recursive(f, a, b, n, previous_value=None):
    if n == 1:
        return trapezoidal_rule(f, a, b, n)
    else:
        if previous_value is None:
            previous_value = trapezoidal_rule(f, a, b, n // 2)
        h = (b - a) / n
        sum_f = sum(f.subs(x, a + (2 * i - 1) * h) for i in range(1, n // 2 + 1))
        return previous_value / 2 + h * sum_f

# Ejemplo de uso de la fórmula recursiva
approx_integral_recursive = trapezoidal_rule_recursive(f, a, b, n)
print(f"Integral Aproximada (Recursiva): {approx_integral_recursive}")
```

[2] ✓ 0.0s

```
Integral Aproximada (Recursiva): 0.335000000000000
```

Matias Davila
Métodos Numéricos

3c

```
# Regla compuesta de Simpson
def simpson_rule(f, a, b, n):
    h = (b - a) / n
    sum_odd = sum(f.subs(x, a + (2 * i - 1) * h) for i in range(1, n // 2 + 1))
    sum_even = sum(f.subs(x, a + 2 * i * h) for i in range(1, n // 2))
    return h/3 * (f.subs(x, a) + 4 * sum_odd + 2 * sum_even + f.subs(x, b))

# Aplicando la regla de Simpson
approx_integral_simpson = simpson_rule(f, a, b, n)
print(f"Integral Aproximada (Simpson): {approx_integral_simpson}")
```

[3] ✓ 0.0s

```
Integral Aproximada (Simpson): 0.333333333333333
```