Tarea 2 Errores Numéricos

Pasquel Johann

Tabla de Contenidos

CONJUNTO DE EJERCICIOS 1

1. Calcule los errores absoluto y relativo en las aproximaciones de p por p*

a) $p = \pi, p* = \frac{22}{7}$

$E_a = |\pi - \frac{22}{7}|$

$E_r = \frac{|p-p^*|}{|p|}$

$E_r = \frac{|\pi-\frac{22}{7}|}{|\pi|}$

```
import math

p = math.pi
p_estrella = 22 / 7

error_absoluto = abs(p - p_estrella)

error_relativo = error_absoluto / abs(p)
print("Error absoluto:", error_absoluto)
print("Error relativo:", error_relativo)
```

```
Error absoluto: 0.0012644892673496777
Error relativo: 0.0004024994347707008
```

b) $p = \pi, p* = 3.1416$

$E_a = |p - p^*|$

$E_a = |\pi - 3.1416|$

$E_r = \frac{|p-p^*|}{|p|}$

$E_r = \frac{|\pi-3.1416|}{|\pi|}$

```
import math

p = math.pi
p_estrella = 3.1416

error_absoluto = abs(p - p_estrella)

error_relativo = error_absoluto / abs(p)
print("Error absoluto:", error_absoluto)
print("Error relativo:", error_relativo)
```

```
Error absoluto: 7.346410206832132e-06
Error relativo: 2.3384349967961744e-06
```

c) $p = e, p* = 2.718$

$E_a = |p - p^*|$

$E_a = |e - 2.718|$

$E_r = \frac{|p-p^*|}{|p|}$

$E_r = \frac{|e-2.718|}{|e|}$

```
import math

p = math.e
p_estrella = 2.718

error_absoluto = abs(p - p_estrella)

error_relativo = error_absoluto / abs(p)
print("Error absoluto:", error_absoluto)
print("Error relativo:", error_relativo)
```

```
Error absoluto: 0.0002818284590451192
Error relativo: 0.00010367889601972718
```

d) $p = \sqrt{2}, p* = 1.414$

$E_a = |p - p^*|$

$E_a = |\sqrt{2} - 1.414|$

$E_r = \frac{|p-p^*|}{|p|}$

$E_r = \frac{|\sqrt{2}-1.414|}{|\sqrt{2}|}$

```
import math

p = math.sqrt(2)
p_estrella = 1.414

error_absoluto = abs(p - p_estrella)

error_relativo = error_absoluto / abs(p)
print("Error absoluto:", error_absoluto)
print("Error relativo:", error_relativo)
```

```
Error absoluto: 0.00021356237309522186
Error relativo: 0.00015101140222192286
```

2. Calcule los errores absoluto y relativo en las aproximaciones de p por p*

a) $p = e^{10}, p* = 22000$

$E_a = |p - p^*|$

$E_a = |e^{10} - 22000|$

$E_r = \frac{|p-p^*|}{|p|}$

$E_r = \frac{|e^{10}-22000|}{|e^{10}|}$

```
import math

p = (math.e)**10
p_estrella = 22000
```

```
error_absoluto = abs(p - p_estrella)

error_relativo = error_absoluto / abs(p)
print("Error absoluto:", error_absoluto)
print("Error relativo:", error_relativo)
```

```
Error absoluto: 26.465794806703343
Error relativo: 0.0012015452253326688
```

b) $p = 10^{\pi}, p* = 1400$

$E_a = |p - p^*|$

$E_a = |10^{\pi} - 1400|$

$E_r = \frac{|p-p^*|}{|p|}$

$E_r = \frac{|10^{\pi}-1400|}{|10^{\pi}|}$

```
import math

p = 10**(math.pi)
p_estrella = 1400

error_absoluto = abs(p - p_estrella)

error_relativo = error_absoluto / abs(p)
print("Error absoluto:", error_absoluto)
print("Error relativo:", error_relativo)
```

```
Error absoluto: 14.544268632989315
Error relativo: 0.010497822704619136
```

c) $p = 8!, p* = 39900$

$E_a = |p - p^*|$

$E_a = |8! - 39900|$

$E_r = \frac{|p-p^*|}{|p|}$

$E_r = \frac{|8!-39900|}{|8!|}$

```python
import math

p = math.factorial(8)
p_estrella = 39900

error_absoluto = abs(p - p_estrella)

error_relativo = error_absoluto / abs(p)
print("Error absoluto:", error_absoluto)
print("Error relativo:", error_relativo)
```

```
Error absoluto: 420
Error relativo: 0.010416666666666666
```

d) $p = 9!,\ p^* = \sqrt{18\pi}\left(\frac{9}{e}\right)^9$

$E_a = |p - p^*|$

$E_a = |9! - \sqrt{18\pi}(\frac{9}{e})^9|$

$E_r = \frac{|p-p^*|}{|p|}$

$E_r = \frac{|9!-\sqrt{18\pi}(\frac{9}{e})^9|}{|9!|}$

```python
import math

p = math.factorial(9)
p_estrella = (math.sqrt(18 * math.pi))*((9/math.e)**9)

error_absoluto = abs(p - p_estrella)

error_relativo = error_absoluto / abs(p)
print("Error absoluto:", error_absoluto)
print("Error relativo:", error_relativo)
```

```
Error absoluto: 3343.1271580516477
Error relativo: 0.009212762230080598
```

3. Encuentre el intervalo más largo en el que se debe encontrar * para aproximarse a con error relativo máximo de 10^{-4} para cada valor de .

$E_r = \frac{|p-p^*|}{|p|}$

$E_r : \frac{|p^*-p|}{|p|} \leq 10^{-4}$

$p - 10^{-4} \cdot p \leq p^* \leq p + 10^{-4} \cdot p$

a) π

$p = \pi$

```
import math

p = math.pi
error_absoluto_max = 10**-4 * p

p_estrella_min = p - error_absoluto_max
p_estrella_max = p + error_absoluto_max

rango = [p_estrella_min, p_estrella_max]

print("p* debe estar entre:", rango)
```

```
p* debe estar entre: [3.141278494324434, 3.141906812855152]
```

b) e

$p = e$

```
import math

p = math.e
error_absoluto_max = 10**-4 * p

p_estrella_min = p - error_absoluto_max
p_estrella_max = p + error_absoluto_max

rango = [p_estrella_min, p_estrella_max]

print("p* debe estar entre:", rango)
```

```
p* debe estar entre: [2.718010000276199, 2.718553656641891]
```

c) $\sqrt{2}$

$p = \sqrt{2}$

```
import math

p = math.sqrt(2)
error_absoluto_max = 10**-4 * p

p_estrella_min = p - error_absoluto_max
p_estrella_max = p + error_absoluto_max

rango = [p_estrella_min, p_estrella_max]

print("p* debe estar entre:", rango)
```

```
p* debe estar entre: [1.4140721410168577, 1.4143549837293325]
```

d) $\sqrt[3]{7}$

$p = \sqrt[3]{7}$

```
import math

p = 7**(1/3)
error_absoluto_max = 10**-4 * p

p_estrella_min = p - error_absoluto_max
p_estrella_max = p + error_absoluto_max

rango = [p_estrella_min, p_estrella_max]

print("p* debe estar entre:", rango)
```

```
p* debe estar entre: [1.9127398896541117, 1.9131224758906662]
```

4. Use la aritmética de redondeo de tres dígitos para realizar lo siguiente. Calcule los errores absoluto y relativo con el valor exacto determinado para por lo memos cinco dígitos.

a) $\frac{\frac{13}{14}-\frac{5}{7}}{2e-5.4}$

```python
import math
import pandas as pd

real_values = [
    13 / 14,
    5 / 7,
    2 * math.e,
    5.4
]
operations = ["13/14", "5/7", "2e", "5.4"]
approx_values = [round(val, 3) for val in real_values]
rel_errors = [abs(real - approx) / abs(real) for real, approx in zip(real_values, approx_valu

data = {
    "Real": real_values,
    "Aproximado": approx_values,
    "Error Relativo": rel_errors
}
df = pd.DataFrame(data, index=operations)
df
```

	Real	Aproximado	Error Relativo
13/14	0.928571	0.929	0.000462
5/7	0.714286	0.714	0.000400
2e	5.436564	5.437	0.000080
5.4	5.400000	5.400	0.000000

```python
real_values = [
    round(13 / 14, 5),
    round(5 / 7, 5),
    round(2 * math.e, 5),
    round(5.4, 5)
]

operations = ["13/14", "5/7", "2e", "5.4"]

approx_values = [round(val, 3) for val in real_values]

abs_errors = [abs(real - approx) for real, approx in zip(real_values, approx_values)]
rel_errors = [abs(real - approx) / abs(real) for real, approx in zip(real_values, approx_valu
```

```
numerador_real = real_values[0] - real_values[1]
denominador_real = real_values[2] - real_values[3]

numerador_aprox = approx_values[0] - approx_values[1]
denominador_aprox = approx_values[2] - approx_values[3]

fraccion_real = numerador_real / denominador_real
fraccion_aprox = numerador_aprox / denominador_aprox

abs_error_numerador = abs(numerador_real - numerador_aprox)
rel_error_numerador = abs_error_numerador / abs(numerador_real)

abs_error_denominador = abs(denominador_real - denominador_aprox)
rel_error_denominador = abs_error_denominador / abs(denominador_real)

abs_error_fraccion = abs(fraccion_real - fraccion_aprox)
rel_error_fraccion = abs_error_fraccion / abs(fraccion_real)

data_final = {
    "Real": [numerador_real, denominador_real, fraccion_real],
    "Aproximado": [numerador_aprox, denominador_aprox, fraccion_aprox],
    "Error Absoluto": [abs_error_numerador, abs_error_denominador, abs_error_fraccion],
    "Error Relativo": [rel_error_numerador, rel_error_denominador, rel_error_fraccion]
}

operaciones_finales = ["Numerador", "Denominador", "Fracción completa"]

df_final = pd.DataFrame(data_final, index=operaciones_finales)

print("\nTabla de resultados finales:")
df_final
```

```
Tabla de resultados finales:
```

	Real	Aproximado	Error Absoluto	Error Relativo
Numerador	0.21428	0.215000	0.00072	0.003360
Denominador	0.03656	0.037000	0.00044	0.012035
Fracción completa	5.86105	5.810811	0.05024	0.008572

b) $-10\pi + 6e - \frac{3}{61}$

```python
import math
import pandas as pd

real_values = [
    round(-10 * math.pi, 5),
    round(6 * math.e, 5),
    round(3 / 61, 5)
]

operations = ["-10 ", "6e", "3/61"]

approx_values = [round(val, 3) for val in real_values]

abs_errors = [abs(real - approx) for real, approx in zip(real_values, approx_values)]
rel_errors = [abs(real - approx) / abs(real) for real, approx in zip(real_values, approx_valu

data_initial = {
    "Real": real_values,
    "Aproximado": approx_values,
    "Error Absoluto": abs_errors,
    "Error Relativo": rel_errors
}
df_initial = pd.DataFrame(data_initial, index=operations)

print("Tabla inicial de errores:")
df_initial
```

```
Tabla inicial de errores:
```

	Real	Aproximado	Error Absoluto	Error Relativo
-10	-31.41593	-31.416	0.00007	0.000002
6e	16.30969	16.310	0.00031	0.000019
3/61	0.04918	0.049	0.00018	0.003660

```python
expresion_real = real_values[0] + real_values[1] - real_values[2]
expresion_aprox = approx_values[0] + approx_values[1] - approx_values[2]

abs_error_expresion = abs(expresion_real - expresion_aprox)
rel_error_expresion = abs_error_expresion / abs(expresion_real)
```

```
data_final = {
    "Real": [expresion_real],
    "Aproximado": [expresion_aprox],
    "Error Absoluto": [abs_error_expresion],
    "Error Relativo": [rel_error_expresion]
}

operaciones_finales = ["Expresión completa"]

df_final = pd.DataFrame(data_final, index=operaciones_finales)

print("\nTabla de resultados finales:")
df_final
```

```
Tabla de resultados finales:
```

	Real	Aproximado	Error Absoluto	Error Relativo
Expresión completa	-15.15542	-15.155	0.00042	0.000028

c) $(\frac{2}{9})(\frac{9}{11})$

```
import pandas as pd

real_values = [
    round(2 / 9, 5),
    round(9 / 11, 5)
]

operations = ["2/9", "9/11"]

approx_values = [round(val, 3) for val in real_values]

abs_errors = [abs(real - approx) for real, approx in zip(real_values, approx_values)]
rel_errors = [abs(real - approx) / abs(real) for real, approx in zip(real_values, approx_valu

data_initial = {
    "Real": real_values,
    "Aproximado": approx_values,
    "Error Absoluto": abs_errors,
```

```
    "Error Relativo": rel_errors
}
df_initial = pd.DataFrame(data_initial, index=operations)

print("Tabla inicial de errores:")
df_initial
```

```
Tabla inicial de errores:
```

	Real	Aproximado	Error Absoluto	Error Relativo
2/9	0.22222	0.222	0.00022	0.00099
9/11	0.81818	0.818	0.00018	0.00022

```
multiplicacion_real = real_values[0] * real_values[1]
multiplicacion_aprox = approx_values[0] * approx_values[1]

abs_error_multiplicacion = abs(multiplicacion_real - multiplicacion_aprox)
rel_error_multiplicacion = abs_error_multiplicacion / abs(multiplicacion_real)

data_final = {
    "Real": [multiplicacion_real],
    "Aproximado": [multiplicacion_aprox],
    "Error Absoluto": [abs_error_multiplicacion],
    "Error Relativo": [rel_error_multiplicacion]
}

operaciones_finales = ["Resultado"]

df_final = pd.DataFrame(data_final, index=operaciones_finales)

print("\nTabla de resultados finales:")
df_final
```

```
Tabla de resultados finales:
```

	Real	Aproximado	Error Absoluto	Error Relativo
Resultado	0.181816	0.181596	0.00022	0.00121

d) $\frac{\sqrt{13}+\sqrt{11}}{\sqrt{13}-\sqrt{11}}$

```
import math
import pandas as pd

real_values = [
    round(math.sqrt(13), 5),
    round(math.sqrt(11), 5)
]

operations = ["√13", "√11"]
approx_values = [round(val, 3) for val in real_values]

abs_errors = [abs(real - approx) for real, approx in zip(real_values, approx_values)]
rel_errors = [abs(real - approx) / abs(real) for real, approx in zip(real_values, approx_valu

data_initial = {
    "Real": real_values,
    "Aproximado": approx_values,
    "Error Absoluto": abs_errors,
    "Error Relativo": rel_errors
}
df_initial = pd.DataFrame(data_initial, index=operations)

print("Tabla inicial de errores:")
df_initial
```

```
Tabla inicial de errores:
```

	Real	Aproximado	Error Absoluto	Error Relativo
√13	3.60555	3.606	0.00045	0.000125
√11	3.31662	3.317	0.00038	0.000115

```
numerador_real = real_values[0] + real_values[1]
denominador_real = real_values[0] - real_values[1]
```

```
numerador_aprox = approx_values[0] + approx_values[1]
denominador_aprox = approx_values[0] - approx_values[1]

fraccion_real = numerador_real / denominador_real
fraccion_aprox = numerador_aprox / denominador_aprox

abs_error_numerador = abs(numerador_real - numerador_aprox)
rel_error_numerador = abs_error_numerador / abs(numerador_real)

abs_error_denominador = abs(denominador_real - denominador_aprox)
rel_error_denominador = abs_error_denominador / abs(denominador_real)

abs_error_fraccion = abs(fraccion_real - fraccion_aprox)
rel_error_fraccion = abs_error_fraccion / abs(fraccion_real)

data_final = {
    "Real": [numerador_real, denominador_real, fraccion_real],
    "Aproximado": [numerador_aprox, denominador_aprox, fraccion_aprox],
    "Error Absoluto": [abs_error_numerador, abs_error_denominador, abs_error_fraccion],
    "Error Relativo": [rel_error_numerador, rel_error_denominador, rel_error_fraccion]
}

operaciones_finales = ["Numerador", "Denominador", "Fracción completa"]

df_final = pd.DataFrame(data_final, index=operaciones_finales)

print("\nTabla de resultados finales:")
df_final
```

```
Tabla de resultados finales:
```

	Real	Aproximado	Error Absoluto	Error Relativo
Numerador	6.922170	6.923000	0.000830	0.000120
Denominador	0.288930	0.289000	0.000070	0.000242
Fracción completa	23.957948	23.955017	0.002931	0.000122

5.Los primeros tres términos diferentes a cero de la serie de Maclaurin para la función arcotangente son: $x - (\frac{1}{3})x^3 + (\frac{1}{5})x^5$

Calcule los errores absoluto y relativo en las siguientes aproximaciones de π mediante el polinomio en lugar del arcotangente:

a. $4\left[\arctan\left(\frac{1}{2}\right)+\arctan\left(\frac{1}{3}\right)\right]$

$x_1=\frac{1}{2}$, $x_2=\frac{1}{3}$

```
import math

pi_exact = math.pi

x1 = 1 / 2
arctan_1_2_approx = x1 - (x1**3) / 3 + (x1**5) / 5
x2 = 1 / 3
arctan_1_3_approx = x2 - (x2**3) / 3 + (x2**5) / 5

pi_approx = 4 * (arctan_1_2_approx + arctan_1_3_approx)

error_absoluto = abs(pi_exact - pi_approx)
error_relativo = error_absoluto / abs(pi_exact)

print("Aproximación de pi:", pi_approx)
print("Error absoluto:", error_absoluto)
print("Error relativo:", error_relativo)
```

```
Aproximación de pi: 3.1455761316872426
Error absoluto: 0.003983478097449478
Error relativo: 0.0012679804598147663
```

b. $16\arctan\left(\frac{1}{5}\right)-4\arctan\left(\frac{1}{239}\right)$

$x_1=\frac{1}{5}$, $x_2=\frac{1}{239}$

```
import math

pi_exact = math.pi

x1 = 1 / 5
arctan_1_5_approx = x1 - (x1**3) / 3 + (x1**5) / 5
x2 = 1 / 239
arctan_1_239_approx = x2 - (x2**3) / 3 + (x2**5) / 5

pi_approx = 16 * arctan_1_5_approx - 4 * arctan_1_239_approx
```

```
error_absoluto = abs(pi_exact - pi_approx)
error_relativo = error_absoluto / abs(pi_exact)

print("Aproximación de pi:", pi_approx)
print("Error absoluto:", error_absoluto)
print("Error relativo:", error_relativo)
```

```
Aproximación de pi: 3.1416210293250346
Error absoluto: 2.837573524150372e-05
Error relativo: 9.032277055104427e-06
```

6. El número e se puede definir por medio de $\sum_{n=0}^{\infty}(\frac{1}{n!})$, donde $! = n(− 1) 2 $ 1 para $ $ 0 $ y 0!$ = 1. Calcule los errores absoluto y relativo en la siguiente aproximación de e.

a. $\sum_{n=0}^{5}(\frac{1}{n!})$

```
import math

approx_e = sum(1 / math.factorial(n) for n in range(6))
e_exact = math.e

error_absoluto = abs(e_exact - approx_e)
error_relativo = error_absoluto / abs(e_exact)

print("Aproximación de e:", approx_e)
print("Valor exacto de e:", e_exact)
print("Error absoluto:", error_absoluto)
print("Error relativo:", error_relativo)
```

```
Aproximación de e: 2.7166666666666663
Valor exacto de e: 2.718281828459045
Error absoluto: 0.0016151617923787498
Error relativo: 0.0005941848175817597
```

b. $\sum_{n=0}^{10}(\frac{1}{n!})$

```python
import math

approx_e = sum(1 / math.factorial(n) for n in range(11))
e_exact = math.e

error_absoluto = abs(e_exact - approx_e)
error_relativo = error_absoluto / abs(e_exact)

print("Aproximación de e:", approx_e)
print("Valor exacto de e:", e_exact)
print("Error absoluto:", error_absoluto)
print("Error relativo:", error_relativo)
```

```
Aproximación de e: 2.7182818011463845
Valor exacto de e: 2.718281828459045
Error absoluto: 2.7312660577649694e-08
Error relativo: 1.0047766310211053e-08
```

7. Suponga que dos puntos (x_0,y_0) y (x_1,y_1) se encuentran en línea racta con $y_1 \neq y_0$. Existen dos fórmulas para encontrar la intersección x de la línea:

$x = \frac{x_0 y_1 - x_1 y_0}{y_1 - y_0}$ & $x = x_0 - \frac{(x_1 - x_0) y_0}{y_1 - y_0}$

a. Use los datos (x_0, y_0)=(1.31,3.24) y $(x_1, y_1) = (1.93,5.76)$ y la aritmética de redondeo de tres dígitos para calcular la intersección con x de ambas manera ¿Cuál método es mejor y por qué?

```python
x0, y0 = 1.31, 3.24
x1, y1 = 1.93, 5.76

y_diff = round(y1 - y0, 3)

x0_y1 = round(x0 * y1, 3)
x1_y0 = round(x1 * y0, 3)

numerator_formula1 = round(x0_y1 - x1_y0, 3)

x_formula1 = round(numerator_formula1 / y_diff, 3)
x_diff = round(x1 - x0, 3)

numerator_formula2 = round(x_diff * y0, 3)
```

```python
x_formula2 = round(x0 - numerator_formula2 / y_diff, 3)

print("Primera fórmula: x =", x_formula1)
print("Segunda fórmula: x =", x_formula2)

if abs(x_formula1 - x_formula2) < 0.01:
    print("Ambos métodos son cercanos, pero la segunda fórmula es ligeramente más precisa del
else:
    print("La segunda fórmula es preferible por precisión numérica.")
```

```
Primera fórmula: x = 0.513
Segunda fórmula: x = 0.513
Ambos métodos son cercanos, pero la segunda fórmula es ligeramente más precisa debido a menos
```