Tarea 02: Ejercicios Unidad 01-A

Métodos Numéricos

Christopher Criollo

2025-11-06

Tabla de Contenidos

1 CONJUNTO DE EJERCICIOS

1.1 Resuelva los siguientes ejercicios, tome en cuenta que debe mostrar el desarrollo completo del ejercicio.

1.1.1 Ejercicio de Cálculo de Errores

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

a.

$$p = \pi, p^* = 22/7$$

```python
import math

p = math.pi
p_star = 22/7

E_a = abs(p - p_star)
E_r = E_a / abs(p)

print(f"1a) E_a = {E_a:.6f}, E_r = {E_r:.6f}")
```

```
1a) E_a = 0.001264, E_r = 0.000402
```

b.

$$p = \pi, p^* = 3.1416$$

```python
p_star = 3.1416
E_a = abs(p - p_star)
E_r = E_a / abs(p)

print(f"1b) E_a = {E_a:.6f}, E_r = {E_r:.6f}")
```

```
1b) E_a = 0.000007, E_r = 0.000002
```

c.

$$p = e, p^* = 2.718$$

```python
p = math.e
p_star = 2.718

E_a = abs(p - p_star)
E_r = E_a / abs(p)

print(f"1c) E_a = {E_a:.6f}, E_r = {E_r:.6f}")
```

```
1c) E_a = 0.000282, E_r = 0.000104
```

d.

$$p = \sqrt{2}, p^* = 1.414$$

```python
p = math.sqrt(2)
p_star = 1.414

E_a = abs(p - p_star)
E_r = E_a / abs(p)

print(f"1d) E_a = {E_a:.6f}, E_r = {E_r:.6f}")
```

```
1d) E_a = 0.000214, E_r = 0.000151
```

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

a.

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

```
p = math.exp(10)
p_star = 22000

E_a = abs(p - p_star)
E_r = E_a / abs(p)

print(f"2a) E_a = {E_a:.3f}, E_r = {E_r:.6f}")
```

```
2a) E_a = 26.466, E_r = 0.001202
```

b.

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

```
p = 10 ** math.pi
p_star = 1400

E_a = abs(p - p_star)
E_r = E_a / abs(p)

print(f"2b) E_a = {E_a:.3f}, E_r = {E_r:.6f}")
```

```
2b) E_a = 14.544, E_r = 0.010498
```

c.

$$p = 8!, p^* = 39900$$

```
import math

p = math.factorial(8)
p_star = 39900

E_a = abs(p - p_star)
E_r = E_a / abs(p)

print(f"2c) E_a = {E_a:.0f}, E_r = {E_r:.6f}")
```

```
2c) E_a = 420, E_r = 0.010417
```

d.

$$p = 9!, p^* = \sqrt{18\pi}(9/e)^9$$

```python
p = math.factorial(9)
p_star = math.sqrt(18 * math.pi) * (9 / math.e) ** 9

E_a = abs(p - p_star)
E_r = E_a / abs(p)

print(f"2d) E_a = {E_a:.0f}, E_r = {E_r:.6f}")
```

```
2d) E_a = 3343, E_r = 0.009213
```

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

a.

$$p = \pi$$

```python
p = math.pi
a = p - 1e-4 * p
b = p + 1e-4 * p
print(f"3a) [{a:.5f}, {b:.5f}]")
```

```
3a) [3.14128, 3.14191]
```

b.

$$p = e$$

```python
p = math.e
a = p - 1e-4 * p
b = p + 1e-4 * p
print(f"3b) [{a:.5f}, {b:.5f}]")
```

```
3b) [2.71801, 2.71855]
```

c.

$$p = \sqrt{2}$$

```python
p = math.sqrt(2)
a = p - 1e-4 * p
b = p + 1e-4 * p
print(f"3c) [{a:.5f}, {b:.5f}]")
```

```
3c) [1.41407, 1.41435]
```

d.

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

```python
p = 7 ** (1/3)
a = p - 1e-4 * p
b = p + 1e-4 * p
print(f"3d) [{a:.5f}, {b:.5f}]")
```

```
3d) [1.91274, 1.91312]
```

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 menos cinco dígitos.

a.

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

```python
import math

# Valor exacto
num_exact = (13/14) - (5/7)
den_exact = 2*math.e - 5.4
exacto = num_exact / den_exact

# Redondeo a 3 dígitos
num1 = 13/14
num2 = 5/7
num = num1 - num2

e_3 = 2.72
den = 2*e_3 - 5.4

approx = num / den

E_a = abs(exacto - approx)
E_r = E_a / abs(exacto)

print(f"4a) Exacto: {exacto:.6f}, Aprox: {approx:.3f}")
print(f"E_a = {E_a:.6f}, E_r = {E_r:.6f}")
```

```
4a) Exacto: 5.860620, Aprox: 5.357
E_a = 0.503478, E_r = 0.085909
```

b.

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

```
# Valor exacto
exacto = -10*math.pi + 6*math.e - 3/61

# Redondeo a 3 dígitos
pi_3 = 3.14
e_3 = 2.72
t1 = -10 * pi_3
t2 = 6 * e_3
t3 = 3/61
suma = t1 + t2 - t3

E_a = abs(exacto - suma)
E_r = E_a / abs(exacto)

print(f"4b) Exacto: {exacto:.6f}, Aprox: {suma:.3f}")
print(f"E_a = {E_a:.4f}, E_r = {E_r:.6f}")
```

```
4b) Exacto: -15.155416, Aprox: -15.129
E_a = 0.0262, E_r = 0.001731
```

c.

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

```
# Valor exacto
exacto = (2/9) * (9/11)  # = 2/11

# Redondeo a 3 dígitos
a1 = 2/9
a2 = 9/11
prod = a1 * a2

E_a = abs(exacto - prod)
E_r = E_a / abs(exacto)

print(f"4c) Exacto: {exacto:.6f}, Aprox: {prod:.3f}")
print(f"E_a = {E_a:.6f}, E_r = {E_r:.6f}")
```

```
4c) Exacto: 0.181818, Aprox: 0.182
E_a = 0.000000, E_r = 0.000000
```

d.

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

```
# Valor exacto (forma racionalizada)
exacto = 12 + math.sqrt(143)

# Redondeo a 3 dígitos
sqrt13 = 3.61
sqrt11 = 3.32
num = sqrt13 + sqrt11
den = sqrt13 - sqrt11
div = num / den

E_a = abs(exacto - div)
E_r = E_a / abs(exacto)

print(f"4d) Exacto: {exacto:.6f}, Aprox: {div:.3f}")
print(f"E_a = {E_a:.4f}, E_r = {E_r:.6f}")
```

```
4d) Exacto: 23.958261, Aprox: 23.897
E_a = 0.0617, E_r = 0.002576
```

5. Los primeros tres términos diferentes a cero de la serie de Maclaurin para la función arcotangente son:

$$x-(1/3)x^3+(1/5)x^5$$

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

a.

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

```
def arctan_approx(x):
    return x - x**3/3 + x**5/5

x1 = 1/2
x2 = 1/3

approx = 4 * (arctan_approx(x1) + arctan_approx(x2))
exacto = math.pi

E_a = abs(exacto - approx)
E_r = E_a / exacto

print(f"5a) E_a = {E_a:.6f}, E_r = {E_r:.6f}")
```

```
5a) E_a = 0.003983, E_r = 0.001268
```

b.

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

```python
x1 = 1/5
x2 = 1/239

approx = 16 * arctan_approx(x1) - 4 * arctan_approx(x2)
exacto = 16 * math.atan(x1) - 4 * math.atan(x2)

E_a = abs(exacto - approx)
E_r = E_a / abs(exacto)

print(f"5b) E_a = {E_a:.6f}, E_r = {E_r:.6f}")
```

```
5b) E_a = 0.000028, E_r = 0.000009
```

6. El número e se puede definir por medio de:

$$e = \sum_{n=0}^{\infty} \left(\frac{1}{n!}\right)$$

donde $n! = n(n-1)\cdots 2 \cdot 1$ para $n \neq 0$ y $0! = 1$. Calcule los **errores absoluto y relativo** en la siguiente aproximación de e:

a.

$$\sum_{n=0}^{5} \left(\frac{1}{n!}\right)$$

```python
def sum_factorial(n_max):
    s = 0
    for n in range(n_max + 1):
        s += 1 / math.factorial(n)
    return s

approx = sum_factorial(5)
exacto = math.e

E_a = abs(exacto - approx)
E_r = E_a / exacto

print(f"6a) E_a = {E_a:.6f}, E_r = {E_r:.6f}")
```

```
6a) E_a = 0.001615, E_r = 0.000594
```

b.

$$\sum_{n=0}^{10} \left(\frac{1}{n!}\right)$$

```python
approx = sum_factorial(6)

E_a = abs(exacto - approx)
E_r = E_a / exacto

print(f"6b) E_a = {E_a:.6f}, E_r = {E_r:.6f}")
```

```
6b) E_a = 0.000226, E_r = 0.000083
```

7. Suponga que dos puntos (x_0, y_0) y (x_1, y_1) se encuentran en línea recta 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

$$(x_0, y_0) = (1.31, 3.24) \text{ y } (x_1, y_1) = (1.93, 5.76)$$

y la **aritmética de redondeo de tres dígitos** para calcular la intersección con x de ambas maneras. **¿Cuál método es mejor y por qué?**

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

dy = y1 - y0
dx = x1 - x0

m = dy / dx
x_exacto = x0 - y0 / m

x0_3, y0_3 = 1.31, 3.24
x1_3, y1_3 = 1.93, 5.76
dy_3 = 2.52
dx_3 = 0.620

num1 = x0_3 * y1_3
```

```
num2 = x1_3 * y0_3
x1_approx = (num1 - num2) / dy_3

# Fórmula 2
num = dx_3 * y0_3
x2_approx = x0_3 - num / dy_3

error1 = abs(x_exacto - x1_approx)
error2 = abs(x_exacto - x2_approx)

print(f"Exacto: {x_exacto:.4f}")
print(f"Fórmula 1: {x1_approx:.3f}, Error: {error1:.4f}")
print(f"Fórmula 2: {x2_approx:.3f}, Error: {error2:.4f}")
print("La fórmula 2 es mejor por menor error ¨(aunque el codigo lo muestre igual a la fo
```

```
Exacto: 0.5129
Fórmula 1: 0.513, Error: 0.0000
Fórmula 2: 0.513, Error: 0.0000
La fórmula 2 es mejor por menor error ¨(aunque el codigo lo muestre igual a la formula 1
```