ps3_ipynb

October 11, 2025

[11]:

```python
# p2a
import numpy as np
# we want to compute different values for error
# approximation is \frac{F \eta}{RT}
# exact is \exp \left( \frac{\alpha_a F \eta}{RT} \right) - \exp \left(␣
↪-\frac{\alpha_c F \eta}{RT} \right) =
F = 96485.3329  # C/mol
R = 8.3145  # J/(mol*K)
T = 298.15  # K
transfer = [(0.5, 0.5), (0.1, 0.9)]
taylor_eta_values = [0.01, 0.02, 0.05] # V
for alpha_a, alpha_c in transfer:
     for eta in taylor_eta_values:
          approximation = (F * eta) / (R * T)
          exact = np.exp((alpha_a * F * eta) / (R * T)) - np.exp((-alpha_c * F␣
↪* eta) / (R * T))
          error = (exact - approximation) / exact * 100
          print(f"Alpha_a: {alpha_a}, Alpha_c: {alpha_c}, Eta: {eta} V, Error:␣
↪{error:.2f}%")
print("----")
# p2b
tafel_eta_values = [0.05, 0.1, 0.2] # V
tafel_eta_values = [-x for x in tafel_eta_values]  # Correct way
for alpha_a, alpha_c in transfer:
     for eta in tafel_eta_values:
          approximation = -np.exp(-(alpha_c * F * eta) / (R * T))
          exact = np.exp((alpha_a * F * eta) / (R * T)) - np.exp((-alpha_c * F␣
↪* eta) / (R * T))
          error = (exact - approximation) / exact * 100
          print(f"Alpha_a: {alpha_a}, Alpha_c: {alpha_c}, Eta: {eta} V, Error:␣
↪{error:.2f}%")
```

```
Alpha_a: 0.5, Alpha_c: 0.5, Eta: 0.01 V, Error: 0.63%
Alpha_a: 0.5, Alpha_c: 0.5, Eta: 0.02 V, Error: 2.48%
Alpha_a: 0.5, Alpha_c: 0.5, Eta: 0.05 V, Error: 14.20%
Alpha_a: 0.1, Alpha_c: 0.9, Eta: 0.01 V, Error: -16.11%
Alpha_a: 0.1, Alpha_c: 0.9, Eta: 0.02 V, Error: -33.14%
Alpha_a: 0.1, Alpha_c: 0.9, Eta: 0.05 V, Error: -86.89%
```

```
----
Alpha_a: 0.5, Alpha_c: 0.5, Eta: -0.05 V, Error: -16.66%
Alpha_a: 0.5, Alpha_c: 0.5, Eta: -0.1 V, Error: -2.08%
Alpha_a: 0.5, Alpha_c: 0.5, Eta: -0.2 V, Error: -0.04%
Alpha_a: 0.1, Alpha_c: 0.9, Eta: -0.05 V, Error: -16.66%
Alpha_a: 0.1, Alpha_c: 0.9, Eta: -0.1 V, Error: -2.08%
Alpha_a: 0.1, Alpha_c: 0.9, Eta: -0.2 V, Error: -0.04%
```

[ ]:
```python
#p1h --- IGNORE ----
import sympy as sp
# I want to evaluate \frac{RT \ln 10}{(n+\beta q) F} for beta = 1/2
n, q = sp.symbols('n q')
beta = 1/2
R = 8.314  # J/(mol*K)
T = 298    # K
F = 96485  # C/mol
expr = (R * sp.ln(10)) / ((n + beta * q) * F)
print(expr.simplify())
exp2 = R*T*sp.ln(10)/F
print(exp2.simplify())
```

```
19.1436924631525/(96485*n + 48242.5*q)
0.0591265000157479
```