**Exercício de aplicação para o capítulo 8**

O Potencial de Helmholtz para o etanol, de acordo com o artigo *“A Fundamental Equation of State for Ethanol”* é obtido pela expressão

|  |  |
| --- | --- |
|  | (1) |

Em que é energia de Helmholtz, a energia de Helmholtz adimensional, a contribuição de gás ideal da energia adimensional de Helmholtz e a porção residual da energia. Onde

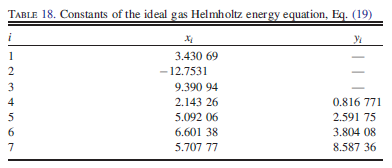
|  |  |
| --- | --- |
|  | (2) |

|  |  |
| --- | --- |
|  | (3) |

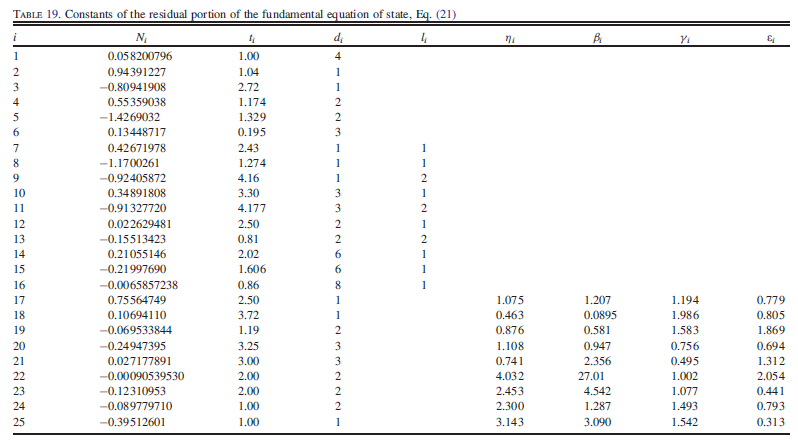
Sendo a densidade crítica e a temperatura crítica. Os termos e são escritos como

|  |  |
| --- | --- |
|  | (4) |
|  | (5) |

As constantes e são dadas pela tabela 18.



E e pela tabela 19.



A condição de estabilidade para o potencial de Helmholtz é uma função côncava da temperatura de acordo com a equação

|  |  |
| --- | --- |
|  | (6) |

O desenvolvimento analítico foi realizado até a primeira derivada.

Utilizando a regra da cadeia

|  |  |
| --- | --- |
|  | (7) |

|  |  |
| --- | --- |
|  | (8) |

|  |  |
| --- | --- |
|  | (9) |

|  |  |
| --- | --- |
|  | (10) |

Aplicando em e

|  |  |
| --- | --- |
|  | (11) |

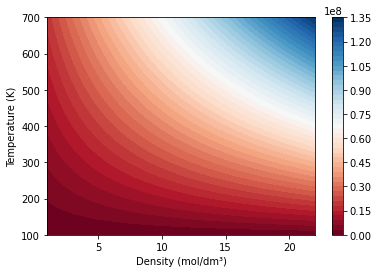
|  |  |
| --- | --- |
|  | (12) |

Substituindo na equação 12

|  |  |
| --- | --- |
|  | (13) |

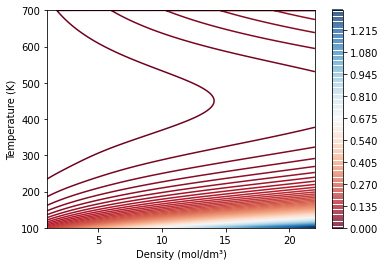
Devido à complexidade dos termos, a tentativa de estimação das regiões de estabilidade foi realizada por meio da utilização de ferramentas numéricas. Para o problema, fez-se o uso de um algoritmo em Python.

A função na formulação do potencial de Helmholtz dada pela equação 1 descreve a superfície abaixo, com unidade de .



A equação 6 determina a condição para que determinado sistema termodinâmico seja estável. Nas demais regiões, o sistema apresenta inomogeneidades internas, que são uma característica básica das transições de fase e identificam os estados de não equilíbrio.

A região limitada pela equação 4 obtida é demonstrada abaixo.



Os resultados foram obtidos numericamente utilizando uma discretização de 4000 elementos para cada variável independente. Nota-se que quanto maior o número de elementos utilizados, mais as curvas de nível se aproximam do ponto crítico nas coordenadas .

Código utilizado:

|  |
| --- |
| #%% |
|  |

|  |
| --- |
| import pandas as pd |
|  |

|  |
| --- |
| import numpy as np |
|  |

|  |
| --- |
| import matplotlib.pyplot as plt |
|  |

|  |
| --- |
| # %% |
|  |

|  |
| --- |
| table18 = pd.read\_csv('table18.txt', index\_col = 'i', sep=' ', dtype={'xi': 'float', 'yi': 'float'}) |
|  |

|  |
| --- |
| table19 = pd.read\_csv('table19.txt', index\_col = 'i', sep=' ', dtype={'Ni': 'float', 'ti': 'float', 'di': 'float', 'li': 'float', 'hi': 'float','bi': 'float', 'gi': 'float', 'ei': 'float'}) |
|  |

|  |
| --- |
| table19.columns = ['Ni','ti','di','li','hi','bi','gi','ei'] |
|  |

|  |
| --- |
| # %% |
|  |

|  |
| --- |
| # Constants |
|  |

|  |
| --- |
| Tc = 514.71 # K |
|  |

|  |
| --- |
| rho\_c = 5.93 # mol/dm³ |
|  |

|  |
| --- |
| R = 8.31446262 #J/(mol.K) |
|  |

|  |
| --- |
| # %% |
|  |

|  |
| --- |
| T = np.linspace(100,700,4000) |
|  |

|  |
| --- |
| rho = np.linspace(1, 22, 4000) |
|  |

|  |
| --- |
| tau = Tc/T |
|  |

|  |
| --- |
| delta = rho/rho\_c |
|  |

|  |
| --- |
| # %% |
|  |

|  |
| --- |
| xdelta, xtau = np.meshgrid(delta,tau) |
|  |

|  |
| --- |
| # %% |
|  |

|  |
| --- |
| def alpha\_0(delta, tau): # J/mol |
|  |

|  |
| --- |
| sum\_alpha0 = np.cumsum([table18['xi'][i]\*(1-np.log(1-np.exp(-table18['yi'][i]\*tau))) for i in range(4,8)], axis=1)[-1] |
|  |

|  |
| --- |
| other\_terms0 = np.log(delta) + table18['xi'][1]\*np.log(tau) + table18['xi'][2] + table18['xi'][3]\*tau |
|  |

|  |
| --- |
| return (other\_terms0 + sum\_alpha0)\*T\*R |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| def alpha\_r(delta, tau): |
|  |

|  |
| --- |
| sum\_r1 = np.cumsum([table19['Ni'][i]\*delta\*\*(table19['di'][i])\*tau\*\*(table19['ti'][i]) for i in range(1,7)], axis=1)[-1] |
|  |

|  |
| --- |
| sum\_r2 = np.cumsum([table19['Ni'][i]\*delta\*\*(table19['di'][i])\*tau\*\*(table19['ti'][i])\*np.exp(-delta\*\*(table19['li'][i])) for i in range(7,17)], axis=1)[-1] |
|  |

|  |
| --- |
| sum\_r3 = np.cumsum([table19['Ni'][i]\*delta\*\*(table19['di'][i])\*tau\*\*(table19['ti'][i])\*np.exp((-table19['hi'][i]\*(delta-table19['ei'][i])\*\*2-table19['bi'][i]\*(tau-table19['gi'][i])\*\*2)) for i in range(17,26)], axis=1)[-1] |
|  |

|  |
| --- |
| return sum\_r1 + sum\_r2 + sum\_r3 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| def helm(delta, tau): |
|  |

|  |
| --- |
| return alpha\_0(delta, tau) + alpha\_r(delta,tau) |
|  |