Revista Brasileira de Ensino de Física
https://doi.org/10.1590/1806-9126-RBEF-2025-0043

Material Suplementar para "Moléculas formadas por átomos hidrogenóides como poços de potencial finito: solução numérica da equação de Schröndiger e proposta de extensão didática via modelos mentais"

Apêndice B - Script em Cuba Python usado para gerar as funções de onda das moléculas diatômicas relacionadas nas tabelas (1) e (2).

```
    # -*- coding: utf-8 -*-
"""
Created on Mon Jul 22 11:59:18 2024
David L. Azevedo used ChatGPT 4o's skeleton proposals,
although the author had to make several adjustments and changes.
Please do not remove any author comments, and if you use this code
cite this paper.
"""

import cupy as cp
import numpy as np
import matplotlib.pyplot as plt
from timeit import default_timer as timer

start = timer()

# Parameters
L = 40.0  # Width of the integration interval
box_width = 2.80057  # Width of the potential well
V0 = 0.84797  # Depth of the potential well
n = 5000  # Number of grid points
h = L / (n + 1)  # Grid spacing

# Define the potential function for a finite potential well
def potential(x, box_width, V0):
    if -box_width/2 <= x <= box_width/2:
        return -V0
    else:
        return 0.0

# Initialize the tridiagonal matrix
d = cp.zeros(n, dtype=cp.float64)  # Main diagonal
e = cp.ones(n-1, dtype=cp.float64) * (-1.0 / (2*h * h))  # Off-diagonal

# Fill the main diagonal with potential values and kinetic term
x = cp.linspace(-L/2, L/2, n)
for i in range(n):
    d[i] = 1.0 / (h * h) + potential(x[i], box_width, V0)

# Convert to dense matrix form for eigenvalue calculation
H = cp.diag(d) + cp.diag(e, k=1) + cp.diag(e, k=-1)

# Solve the eigenvalue problem
eigenvalues, eigenvectors = cp.linalg.eigh(H)

# Convert results back to host for printing
eigenvalues_host = cp.asnumpy(eigenvalues)
eigenvectors_host = cp.asnumpy(eigenvectors)
```

```
# Print the first few eigenvalues
print("First 10 eigenvalues:")
print(eigenvalues_host[:10])

# ...
end = timer()
print(end - start) # Time in seconds,

# Plot the first few eigenvectors
plt.figure(figsize=(10, 6))
x_host = cp.asnumpy(x)
for i in range(3):
    plt.plot(x_host, eigenvectors_host[:, i], label=f'n={i}')
plt.xlabel('Position x')
plt.ylabel('Wavefunction Psi(x)')
plt.title('First 3 Wavefunctions of Finite Well')
plt.legend()
plt.show()
```