Física Computacional
Tarea 3

Pedro Porras Flores
Efraín Ossmar Díaz Pérez

30 de septiembre del 2025

Instrucciones

Resuelva los siguientes ejercicios implementando soluciones en Python. Utilice variables adecuadas, estructuras de control de flujo y funciones. Incluya comentarios en su código y muestre los resultados obtenidos.

1. Considere la ecuación de vibraciones de una viga en voladizo:

$$f(\omega) = \cos(\omega)\cosh(\omega) + 1.$$

a) Encuentre la *menor raíz positiva* de $f(\omega)$ usando el **método de bisección** con tolerancia 10^{-5}.

b) Grafique $f(\omega)$ en $\omega \in [0, 4]$. Marque en la gráfica el intervalo inicial seleccionado y el punto raíz aproximado.

c) Grafique $|f(\omega_n)|$ vs. iteración n en escala semilog (eje vertical logarítmico) para evidenciar la convergencia.

2. Modelo de enfriamiento de Newton:

$$T(t) = T_a + (T_0 - T_a)\, e^{-kt}, \quad T_a = 20°\text{C},\ T_0 = 90°\text{C},\ k = 0{,}07\ \text{min}^{-1}.$$

a) Encuentre t tal que $T(t) = 50°\text{C}$ resolviendo

$$f(t) = T_a + (T_0 - T_a)e^{-kt} - 50 = 0,$$

con el **método de Newton-Raphson**, usando por ejemplo $t_0 = 10$.

b) Grafique $T(t)$ en $t \in [0,\ 30]$ junto con la línea horizontal $T = 50°\text{C}$. Indique el t^* encontrado.

c) Grafique $|f(t_n)|$ vs. iteración n en escala semilog. Comente el régimen de convergencia observado.

3. Resuelva

$$f(x) = e^{x^2}\ \ln(x^2) - x$$

mediante el **método de la secante** con condiciones iniciales $x_0 = 0{,}5$, $x_1 = 1{,}5$.
Nota: f tiene dos raíces reales, una negativa y una positiva.

a) Grafique $f(x)$ en $x \in [-2,\ 2]$. Marque el par inicial (x_0, x_1) y la raíz encontrada.

b) Grafique $|f(x_n)|$ vs. iteración n en escala semilog. Discuta la sensibilidad a los puntos iniciales.

4. Programa una versión segura del método de Newton, combinándolo con el método de bisección. Específicamente, comienza con un intervalo de encuadre y realiza un paso de Newton: si la iterada "quiere" salir del intervalo, realiza en su lugar un paso de bisección. Repite.

5. Vamos a estudiar la solución de la ecuación de Schrödinger (ES) para un sistema compuesto por un neutrón y un protón (el deuterón) que se mueven dentro de un potencial de caja simple.

Comenzamos nuestra discusión de la ES con el sistema neutrón-protón (deuterón) con un potencial de caja $V(r)$. Definimos la parte radial de la función de onda como $R(r)$ e introducimos la definición $u(r) = rR(r)$. La parte radial de la ES para dos partículas en su sistema de centro de masa y con momento orbital $\ell = 0$ es entonces:

$$-\frac{\hbar^2}{2m}\frac{d^2u(r)}{dr^2} + V(r)u(r) = Eu(r),$$

con

$$m = 2\frac{m_p m_n}{m_p + m_n},$$

donde m_p y m_n son las masas del protón y del neutrón, respectivamente. Usamos aquí $m = 938$ MeV. Nuestro potencial se define como:

$$V(r) = \begin{cases} -V_0 & \text{para } 0 \leq r < a \\ 0 & \text{para } r \geq a \end{cases}$$

Los estados ligados corresponden a una energía E negativa, y los estados de dispersión vienen dados por energías positivas. La ES adopta la siguiente forma (sin especificar el signo de E):

$$\frac{d^2u(r)}{dr^2} + \frac{2m}{\hbar^2}(V_0 + E)u(r) = 0 \quad \text{para } r < a,$$

y

$$\frac{d^2u(r)}{dr^2} + \frac{2m}{\hbar^2}Eu(r) = 0 \quad \text{para } r > a.$$

Ahora vamos a buscar eventuales estados confinados, es decir, con $E < 0$. El deuterón tiene solo un estado confinado a una energía $E = -2{,}223$ MeV. Discuta las condiciones de contorno sobre la función de onda y utilícelas para mostrar que la solución a la ES es:

$$u(r) = A\sin(kr) \quad \text{para } r < a,$$

y

$$u(r) = B\exp(-\beta r) \quad \text{para } r > a,$$

donde A y B son constantes. También hemos definido:

$$k = \sqrt{\frac{m(V_0 - |E|)}{\hbar^2}},$$

y

$$\beta = \sqrt{\frac{m|E|}{\hbar^2}}.$$

A continuación, muestre que, utilizando el requisito de continuidad de la función de onda en $r = a$, se obtiene la **ecuación trascendental**:

$$k \cot(ka) = -\beta\,. \tag{1}$$

Inserte los valores de $V_0 = 60\text{MeV}$ y $a = 1{,}45$ fm (1 fm $= 10^{-15}$ m) y realice un gráfico de la Ecuación (1) en función de la energía E para encontrar los eventuales autovalores. Compruebe si estos valores resultan en un estado ligado para E.

Una vez que haya localizado en su gráfico el punto o puntos donde se satisface la Ecuación (1), obtenga un valor numérico para E utilizando el **método de Newton-Raphson**, el **método de bisección** y el **método de la secante**. Realice un análisis de estos tres métodos y discuta cuántas iteraciones son necesarias para encontrar una solución estable. ¿Cuál es el valor más pequeño posible de V_0 que da un estado confinado?

Notas importantes

1. **Criterio de tolerancia.** Todas las implementaciones deben utilizar como criterio de convergencia el *épsilon de máquina*. En Python se obtiene, por ejemplo, con:

   ```
   import sys   TOL = sys.float_info.epsilon
   ```

   El criterio de parada debe escribirse como:

   ```
   if abs(x_new - x) <TOL * max(1.0, abs(x_new)).
   ```

2. **Uso de librerías reales de física.** En los ejercicios donde se usen constates físicas importe la librería, `from scipy.constants`, que cuenta con constantes físicas reales:

   ```
   from scipy.constants import hbar, c, m_p, m_n
   ```

   para que los cálculos sean consistentes en unidades y magnitudes físicas.

3. **Manejo de errores obligatorio.** Cada programa debe incluir de manera explícita bloques de manejo de errores utilizando:

   `try: ... except: ... finally: ...`

   Esto significa que ningún código puede entregarse sin estas estructuras. Los estudiantes deben prever posibles errores (por ejemplo, división entre cero, intervalos inválidos, derivadas nulas, etc.), capturarlos con `except`, y asegurarse con `finally` de liberar recursos o mostrar un mensaje final apropiado. La ausencia de manejo de errores reducirá la calificación del ejercicio.

4. **Funciones auxiliares.** En caso de requerir funciones matemáticas (por ejemplo, cot, sech, etc.), impleméntelas en el archivo `misFunciones.py` que se encuentra en el repositorio de la clase. Después, importe dichas funciones en su programa principal. Esto fomenta la modularidad y la reutilización del código.