Contenido de archivos .m

```
% main.m

clear; clc; close all;
addpath('functions');
addpath('potentials');

L = 5;            % M ximo valor de L
Nx = 250;         % N mero de intervalos
m = 1;            % Masa en unidades at micas
num_states = 10;  % N mero de estados a considerar

[x, hx] = discretize_space(L, Nx);

potentials = {
    'Pozo Finito', @(x) potential_finite_well(x, 5, L);
    'Potencial de Kronig-Penney', @(x)
        potential_kronig_penney(x, 7.5, 1.6, 0.9);
    'Potencial Arm nico', @(x) potential_harmonic(x, m, 1);
    'Dos Pozos Cercanos', @(x) potential_double_well(x, 0.1,
        2);
};

for i = 1:size(potentials, 1)
    potential_name = potentials{i, 1};
    potential_func = potentials{i, 2};
    process_potential(potential_name, potential_func, L, Nx,
        m, num_states, x, hx);
end
% run_potential.m

clear; clc; close all;
addpath('functions');
addpath('potentials');

L = 5;            % M ximo valor de L
Nx = 250;         % N mero de intervalos
m = 1;            % Masa en unidades at micas
num_states = 10;  % N mero de estados a considerar

[x, hx] = discretize_space(L, Nx);

potentials = {
    'Pozo Finito', @(x) potential_finite_well(x, 5, L);
    'Potencial de Kronig-Penney', @(x)
        potential_kronig_penney(x, 30, 0.6, 0.1);
    'Potencial Arm nico', @(x) potential_harmonic(x, m, 1);
    'Dos Pozos Cercanos', @(x) potential_double_well(x, 0.1,
        2.5);
};

fprintf('Lista de potenciales disponibles:\n');
for i = 1:size(potentials, 1)
    fprintf('%d. %s\n', i, potentials{i, 1});
end

```

```
idx = input('Ingrese el  ndice  del potencial (1-4): ');

if idx >= 1 && idx <= size(potentials, 1)
    potential_name = potentials{idx, 1};
    potential_func = potentials{idx, 2};
    fprintf('Ejecutando: %s\n', potential_name);
    process_potential(potential_name, potential_func, L, Nx,
        m, num_states, x, hx);
else
    error(' ndice  de potencial no v lido.');
end
% calculate_energies.m

function [EK, EV, E_total] = calculate_energies(D2, U, V, m,
    hx, num_states)

    EK = zeros(num_states, 1);
    EV = zeros(num_states, 1);
    E_total = zeros(num_states, 1);

    for i = 1:num_states
        phi = V(:, i);
        EK(i) = kinetic_energy(D2, phi, m, hx);
        EV(i) = potential_energy(U, phi, hx);
        E_total(i) = EK(i) + EV(i);
    end
end
% discretize_space.m

function [x, hx] = discretize_space(L, Nx)
    x = linspace(-L, L, Nx+1);
    hx = x(2) - x(1);
end
% hamiltonian.m

function H = hamiltonian(D2, U, m)
    H = -(1/(2*m)) * D2 + diag(U);
end
% kinetic_energy.m

function EK = kinetic_energy(D2, phi, m, hx)
    EK = (1/(2*m)) * (phi' * D2 * phi) * hx;
end
% plot_wavefunctions.m

function plot_wavefunctions(x, V, U, num_states,
    potential_name, E)
    figure('Color', 'white', 'Position', [100, 100, 800,
        600]);
    hold on;

    scale_factor = 4;

    colors = jet(num_states) * 0.9;
    plot(x, U, 'k--', 'LineWidth', 1.5, 'DisplayName',
        '$U(x)$');

```

```
    for i = 1:num_states
        plot(x, scale_factor * V(:, i) + E(i), 'LineWidth',
            2, 'Color', colors(i, :), 'DisplayName',
            ['$\Psi_{' num2str(i) '}(x)$']);
        plot([min(x), max(x)], [E(i), E(i)], '--', 'Color',
            colors(i, :), 'LineWidth', 1.5, 'DisplayName',
            ['Nivel $E_{' num2str(i) '}$']);
    end

    ylabel('$\Psi_{i}(x)$, $E$ (a.u.)', 'Interpreter',
        'latex', 'FontSize', 14);
    xlabel('$x$ (a.u.)', 'Interpreter', 'latex', 'FontSize',
        14);
    title(['Funciones de Onda y Potencial: ',
        potential_name], 'Interpreter', 'latex', 'FontSize',
        16);
    legend('Interpreter', 'latex', 'Location',
        'northeastoutside', 'FontSize', 12);
    grid on;

    filename = sprintf('img/%s.png', strrep(potential_name,
        ' ', '_'));
    print(gcf, filename, '-dpng', '-r300');

    hold off;
end
% potential_energy.m

function EV = potential_energy(U, phi, hx)
    U = U(:);
    EV = sum(U .* (phi.^2)) * hx;
end
% process_potential.m

function process_potential(potential_name, potential_func,
    L, Nx, m, num_states, x, hx)
    disp(['Resolviendo para ', potential_name, '...']);
    U = potential_func(x);
    D2 = second_derivative_matrix(Nx, hx);
    H = hamiltonian(D2, U, m);
    [E, V] = solve_eigen(H);
    [EK, EV, E_total] = calculate_energies(D2, U, V, m, hx,
        num_states);
    plot_wavefunctions(x, V, U, num_states, potential_name,
        E(1:num_states));
end
% second_derivative_matrix.m

function D2 = second_derivative_matrix(Nx, hx)
    D2 = (diag(ones(Nx, 1), 1) - 2 * eye(Nx+1) +
        diag(ones(Nx, 1), -1)) / hx^2;
end
% solve_eigen.m

function [E, V] = solve_eigen(H)
    [V, E_matrix] = eig(H);
    E = diag(E_matrix);
```

```
end
% potential_double_well.m

function U = potential_double_well(x, a, b)
    U = a * x.^4 - b * x.^2;
end
% potential_finite_well.m

function U = potential_finite_well(x, depth, width)
    U = zeros(size(x));
    U(abs(x) <= width/2) = -depth;
    U(abs(x) > width/2) = depth;
end
% potential_harmonic.m

function U = potential_harmonic(x, m, w)
    U = (1/2) * m * w^2 * x.^2;
end
% potential_kronig_penney.m

function U = potential_kronig_penney(x, V0, a, b)
    U = zeros(size(x));
    n_max = floor(max(x) / a);
    for n = -n_max:n_max
        pozo_centro = n * a;
        indices = abs(x - pozo_centro) <= (b / 2);
        U(indices) = -V0;
    end
end
```