

Contenido de archivos .m

```
1 % main.m
2
3 clear; clc; close all;
4 addpath('functions');
5 addpath('potentials');
6
7 L = 5;           % Mximo valor de L
8 Nx = 250;        % Nmero de intervalos
9 m = 1;           % Masa en unidades atmicas
10 num_states = 10; % Nmero de estados a considerar
11
12 [x, hx] = discretize_space(L, Nx);
13
14 potentials = {
15     'Pozo Finito', @(x) potential_finite_well(x, 5, L);
16     'Potencial de Kronig-Penney', @(x)
17         potential_kronig_penney(x, 7.5, 1.6, 0.9);
18     'Potencial Armnico', @(x) potential_harmonic(x, m, 1);
19     'Dos Pozos Cercanos', @(x) potential_double_well(x, 0.1,
20         2);
21 };
22
23 for i = 1:size(potentials, 1)
24     potential_name = potentials{i, 1};
25     potential_func = potentials{i, 2};
26     process_potential(potential_name, potential_func, L, Nx,
27         m, num_states, x, hx);
28 end
29
30 % run_potential.m
31
32 clear; clc; close all;
33 addpath('functions');
34 addpath('potentials');
35
36 L = 5;           % Mximo valor de L
37 Nx = 250;        % Nmero de intervalos
38 m = 1;           % Masa en unidades atmicas
39 num_states = 10; % Nmero de estados a considerar
40
41 [x, hx] = discretize_space(L, Nx);
42
43 potentials = {
44     'Pozo Finito', @(x) potential_finite_well(x, 5, L);
45     'Potencial de Kronig-Penney', @(x)
46         potential_kronig_penney(x, 30, 0.6, 0.1);
47     'Potencial Armnico', @(x) potential_harmonic(x, m, 1);
48     'Dos Pozos Cercanos', @(x) potential_double_well(x, 0.1,
49         2.5);
50 };
51
52 fprintf('Lista de potenciales disponibles:\n');
53 for i = 1:size(potentials, 1)
54     fprintf('%d. %s\n', i, potentials{i, 1});
55 end
```

```

51 idx = input('Ingrese el ndice del potencial (1-4): ');
52
53 if idx >= 1 && idx <= size(potentials, 1)
54     potential_name = potentials{idx, 1};
55     potential_func = potentials{idx, 2};
56     fprintf('Ejecutando: %s\n', potential_name);
57     process_potential(potential_name, potential_func, L, Nx,
58         m, num_states, x, hx);
59 else
60     error(' ndice de potencial no v lido. ');
61 end
62 % calculate_energies.m
63 function [EK, EV, E_total] = calculate_energies(D2, U, V, m,
64     hx, num_states)
65
66     EK = zeros(num_states, 1);
67     EV = zeros(num_states, 1);
68     E_total = zeros(num_states, 1);
69
70     for i = 1:num_states
71         phi = V(:, i);
72         EK(i) = kinetic_energy(D2, phi, m, hx);
73         EV(i) = potential_energy(U, phi, hx);
74         E_total(i) = EK(i) + EV(i);
75     end
76 end
77 % discretize_space.m
78 function [x, hx] = discretize_space(L, Nx)
79     x = linspace(-L, L, Nx+1);
80     hx = x(2) - x(1);
81 end
82 % hamiltonian.m
83
84 function H = hamiltonian(D2, U, m)
85     H = -(1/(2*m)) * D2 + diag(U);
86 end
87 % kinetic_energy.m
88
89 function EK = kinetic_energy(D2, phi, m, hx)
90     EK = (1/(2*m)) * (phi' * D2 * phi) * hx;
91 end
92 % plot_wavefunctions.m
93
94 function plot_wavefunctions(x, V, U, num_states,
95     potential_name, E)
96     figure('Color', 'white', 'Position', [100, 100, 800,
97         600]);
98     hold on;
99
100     scale_factor = 4;
101
102     colors = jet(num_states) * 0.9;
103     plot(x, U, 'k--', 'LineWidth', 1.5, 'DisplayName',
104         '$U(x)$');

```

```

103     for i = 1:num_states
104         plot(x, scale_factor * V(:, i) + E(i), 'LineWidth',
105             2, 'Color', colors(i, :), 'DisplayName',
106             ['$\Psi_{' num2str(i) '} (x)$']);
107         plot([min(x), max(x)], [E(i), E(i)], '--', 'Color',
108             colors(i, :), 'LineWidth', 1.5, 'DisplayName',
109             ['Nivel $E_{' num2str(i) '} $']);
110     end
111     ylabel('$\Psi_{' i '} (x)$, $E$ (a.u.)', 'Interpreter',
112         'latex', 'FontSize', 14);
113     xlabel('$x$ (a.u.)', 'Interpreter', 'latex', 'FontSize',
114         14);
115     title(['Funciones de Onda y Potencial: ',
116         potential_name], 'Interpreter', 'latex', 'FontSize',
117         16);
118     legend('Interpreter', 'latex', 'Location',
119         'northeastoutside', 'FontSize', 12);
120     grid on;
121
122     filename = sprintf('img/%s.png', strrep(potential_name,
123         ' ', '_'));
124     print(gcf, filename, '-dpng', '-r300');
125
126     hold off;
127 end
128 % potential_energy.m
129
130 function EV = potential_energy(U, phi, hx)
131     U = U(:);
132     EV = sum(U .* (phi.^2)) * hx;
133 end
134 % process_potential.m
135
136 function process_potential(potential_name, potential_func,
137     L, Nx, m, num_states, x, hx)
138     disp(['Resolviendo para ', potential_name, '...']);
139     U = potential_func(x);
140     D2 = second_derivative_matrix(Nx, hx);
141     H = hamiltonian(D2, U, m);
142     [E, V] = solve_eigen(H);
143     [EK, EV, E_total] = calculate_energies(D2, U, V, m, hx,
144         num_states);
145     plot_wavefunctions(x, V, U, num_states, potential_name,
146         E(1:num_states));
147 end
148 % second_derivative_matrix.m
149
150 function D2 = second_derivative_matrix(Nx, hx)
151     D2 = (diag(ones(Nx, 1), 1) - 2 * eye(Nx+1) +
152         diag(ones(Nx, 1), -1)) / hx^2;
153 end
154 % solve_eigen.m
155
156 function [E, V] = solve_eigen(H)
157     [V, E_matrix] = eig(H);
158     E = diag(E_matrix);

```

```

146 end
147 % potential_double_well.m
148
149 function U = potential_double_well(x, a, b)
150     U = a * x.^4 - b * x.^2;
151 end
152 % potential_finite_well.m
153
154 function U = potential_finite_well(x, depth, width)
155     U = zeros(size(x));
156     U(abs(x) <= width/2) = -depth;
157     U(abs(x) > width/2) = depth;
158 end
159 % potential_harmonic.m
160
161 function U = potential_harmonic(x, m, w)
162     U = (1/2) * m * w^2 * x.^2;
163 end
164 % potential_kronig_penney.m
165
166 function U = potential_kronig_penney(x, V0, a, b)
167     U = zeros(size(x));
168     n_max = floor(max(x) / a);
169     for n = -n_max:n_max
170         pozo_centro = n * a;
171         indices = abs(x - pozo_centro) <= (b / 2);
172         U(indices) = -V0;
173     end
174 end

```