

Oscilador armónico

Juan Barbosa, 201325901

Febrero 23, 2017

$$-\frac{\hbar^2}{2m}\nabla^2\Psi(x) + V(x)\Psi(x) = E\Psi(x)$$

Para un oscilador armónico la energía potencial se escribe:

$$V(x) = \frac{1}{2}kx^2 = \frac{1}{2}m\omega^2x^2 \quad (1)$$

La ecuación de Schrödinger se escribe de la forma:

$$\frac{d^2}{dx^2}\Psi(x) = \ddot{\Psi}(x) = \frac{2m}{\hbar^2}\left(\frac{1}{2}m\omega^2x^2 - E\right)\Psi(x) \quad (2)$$

La solución a la ecuación anterior debe cumplir que para $x = \pm\infty$, $\Psi(x) = 0$. Las soluciones analíticas usan los polinomios de Hermite.

$$\Psi_n(x) = \frac{1}{\sqrt{2^n n!}} \left(\frac{m\omega}{\pi\hbar}\right)^{1/4} e^{-\frac{m\omega x^2}{2\hbar}} H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right) \quad \text{donde} \quad H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{-x^2} \quad (3)$$

Usando el método de Euler para resolver numéricamente se obtiene:

$$\begin{aligned} \dot{\Psi}_n &= \dot{\Psi}_{n-1} + \ddot{\Psi}_{n-1}\Delta x \\ \Psi_n &= \Psi_{n-1} + \dot{\Psi}_n\Delta x \end{aligned} \quad (4)$$

El sistema de ecuaciones diferenciales es resuelto en C usando $N = 6000001$ puntos, $dx = 0.000001$, para $\omega = 1, 5$. Las unidades usadas son arbitrarias tales que $\hbar^2 = m = 1$. Posteriormente se usa un algoritmo en Python para graficar las soluciones.

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int N = 100000;
double dx = 0.0001, E, omega;
double *psi;
double psi_prime;

double *solver();
double *leapfrog();
double *kutta();
double force(double x, double y);

int main(int argc, char **argv)
{
    double acceptance = 0.5, current = 100;
    psi = malloc(N*sizeof(double));
    int i = 0, j = 0;

    FILE *functions = fopen("functions.dat", "w");
    FILE *energies = fopen("energies.dat", "w");
    FILE *omegas = fopen("omegas.dat", "w");
    omega = 0.4;
    for(omega; omega<=1.0; omega+=0.2)
    {
        for(j=0; j<7; j++)
        {
            current = 100;
            E = omega*(1.4+j) - 0.1;
            while (current > acceptance)
            {
                E += 0.001/omega;
                if (j%2 == 0)
                {
                    psi[0] = 0;
                    psi_prime = 1;
                }
                else
                {
                    psi[0] = 1;
                    psi_prime = 0;
                }
                psi = solver();
                current = fabs(psi[N*5/10-1]);
            }
            printf("%f_ %f\n", omega, E);
            for(i = 0; i<N; i += 10)
            {
                fprintf(functions, "%f\n", psi[i]);
            }
            fprintf(energies, "%f\n", E);
        }
        fprintf(omegas, "%f\n", omega);
    }
    return 0;
}

double *solver()
{
    int i = 0;
    double x = 0, U = 0;
    for(i = 1; i < N; i++)
    {
        U = 0.5*pow(omega*x, 2);
        x = x + dx;
        psi_prime = psi_prime + 2*(U-E)*psi[i-1]*dx;
    }
}

```

```

        psi[i] = psi[i-1] + psi_prime*dx;
    }
    return psi;
}

double *kutta()
{
    double k1, k2, k3, k4, k, l1, l2, l3, l4, l, x=0;
    int i = 0;
    double temp;
    temp = psi_prime;
    for(i = 1; i < N; i++)
    {
        l1 = force(x, psi[i-1]);
        l2 = force(x + 0.5*dx, psi[i-1] + 0.5*l1);
        l3 = force(x + 0.5*dx, psi[i-1] + 0.5*l2);
        l4 = force(x + dx, psi[i-1] + l3);
        l = (l1 + 2*l2 + 2*l3 + l4)*dx/6.0;
        psi_prime += l;

        k1 = psi_prime;
        k2 = k1 + 0.5*dx*k1;
        k3 = k1 + 0.5*dx*k2;
        k4 = k1 + dx*k3;
        k = (k1 + 2*k2 + 2*k3 + k4)*dx/6.0;

        psi[i] = psi[i-1] + k;
        x += dx;
    }
    return psi;
}

double *leapfrog()
{
    double half, x = 0;
    int i = 0, U = 0;
    for(i = 1; i < N; i++)
    {
        half = psi_prime + 0.5*force(x, psi[i-1])*dx;
        psi[i] = psi[i-1] + dx*half;
        psi_prime = half + 0.5*force(x, psi[i])*dx;
        x += dx;
    }
    return psi;
}

double force(double x, double y)
{
    double U;
    U = 0.5*pow(omega*x, 2);
    return 2*(U - E)*y;
}

//double *adams()
//{
//    //int i = 0;
//    //double x = 0, U = 0;
//    //for(i = 2; i < N; i++)
//    //{
//        //psi[i] = psi[i-1] + 1.5*dx*force(x, psi[i-1]) - 0.5*dx*force(x-dx, psi[i-2])
//        //x += dx
//    //}
//    //return psi;
//}
//}

```

```

import numpy as np
import matplotlib.pyplot as plt

data = np.genfromtxt("functions.dat")

```

```

energies = np.genfromtxt("energies.dat")
omegas = np.genfromtxt("omegas.dat")

N = len(omegas)
n = 10000
dx = 0.001
x = np.linspace(0, (n-1)*dx, n)
data_split = np.split(data, N)
energies_split = np.split(energies, N)
answers = int(data_split[0].shape[0]/n)

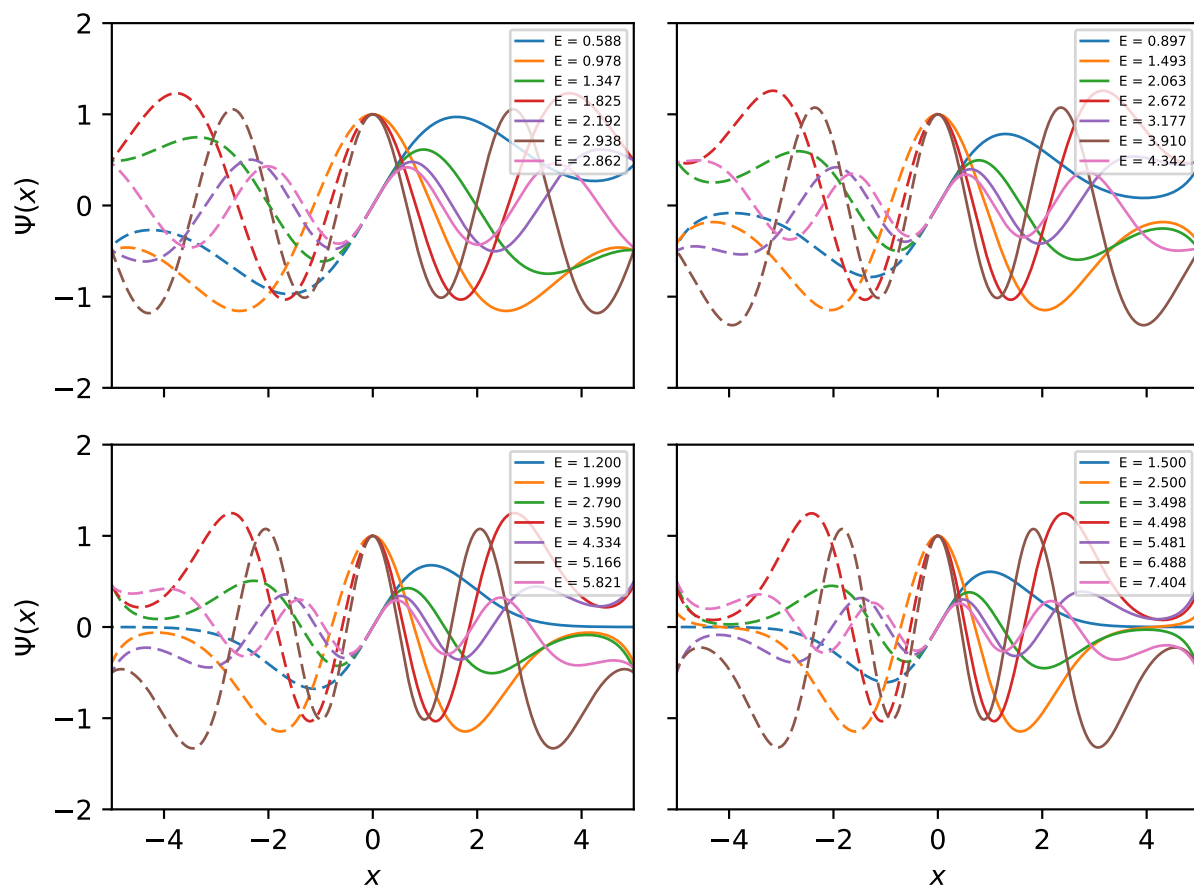
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
axes = axes.reshape(4)
for j in range(N):
    data = np.split(data_split[j], answers)
    for i in range(answers):
        p = axes[j].plot(x, data[i], label="E_=%0.3f"%energies_split[j][i], lw = 1)[0]
        if i%2 == 0:
            axes[j].plot(-x, -data[i], "--", lw = 1, c = p.get_color())
        else:
            axes[j].plot(-x, data[i], "--", lw = 1, c = p.get_color())
    axes[j].set_ylim(-2, 2)
    axes[j].set_xlim(-5, 5)
    axes[j].legend(loc = 1, fontsize=5)
    if j%2 == 0:
        axes[j].set_ylabel("$\Psi(x)$")
    if j > 1:
        axes[j].set_xlabel("$x$")
fig.tight_layout()
plt.savefig("plot.pdf")

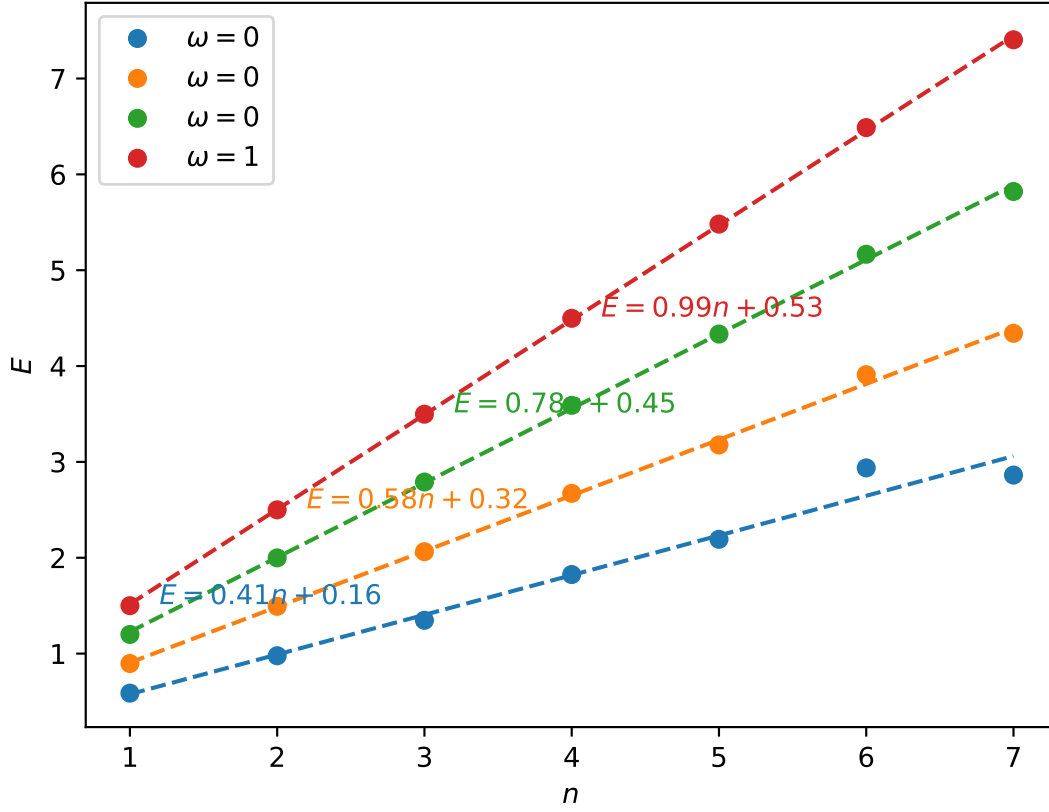
fig = plt.figure()

for i in range(N):
    energies = energies_split[i]
    n = np.arange(1, answers + 1)
    A, B = np.polyfit(n, energies, 1)
    x = np.linspace(1, 7, 10)
    y = A*x + B
    p = plt.plot(n, energies, "o", label = "$\omega_{%d}$"%omegas[i])[0]
    plt.plot(x, y, "--", c=p.get_color())
    plt.text(i+1.2, i+1.5, "$E_{%0.2f}+{%0.2f}$"%(A, B), color=p.get_color())

plt.xlabel("$n$")
plt.ylabel("$E$")
plt.legend()
plt.savefig("energies.pdf")

```





La energía depende del cuadrado de n como número cuántico. Resultado análogo al de un pozo de potencial infinito. La solución analítica a la ecuación es de la forma:

$$\begin{aligned} k_{even} &= \kappa \tan \frac{\kappa L}{2} = \sqrt{\beta^2 - \kappa^2} \\ k_{odd} &= -\frac{\kappa}{\tan \frac{\kappa L}{2}} = \sqrt{\beta^2 - \kappa^2} \end{aligned} \quad (5)$$

$$\begin{aligned} \Psi_{II_{even}}(x) &= F \cos(\kappa x) \\ \Psi_{II_{odd}}(x) &= F \sin(\kappa x) \end{aligned} \quad (6)$$

n	E
1	2.030
2	8.120
3	18.170
4	32.150
5	49.780
6	70.520

Finalmente para el caso con $E > V_0$ se obtienen soluciones oscilantes para toda región del espacio. Para las regiones con $V = V_0$ la amplitud se ve reforzada.