Homework 03

William Oquendo, woquendo@gmail.com

1 Oscilador anarmónico cuántico

(1.cpp, 5.0/5.0) El objetivo de este problema es encontrar los valores límite de la energía del estado base y del primer estado excitado como funciones del tamaño del sistema para un oscilador armónico cuántico anarmónico. Para esto se usará la representación matricial del hamiltoniano y su usará la librería `eigen` para resolver el problema de valores propios como función del tamaño del sistema y de la intensidad de la perturbación. Se le recomienda compilar con la bandera de optimización `-O3`

Un oscilador anarmónico cuántico puede modelarse como una perturbación al oscilador armónico cuántico de la forma

$$H(\lambda) = H_0 + \lambda x^4, \tag{1}$$

donde H denota el hamiltoniano y λ es un número pequeño que modela la perturbación al hamiltoniano original H_0.

Si se desea encontrar los valores propios perturbados, se debe resolver la ecuación $H(\lambda)|n\rangle = E_n|n\rangle$. En la representación matricial, se obtiene que

$$H_{0,nm} = \left(n + \frac{1}{2}\right)\delta_{n,m}, \tag{2}$$

$$x_{nm} = \frac{1}{\sqrt{2}}\sqrt{m+1}\,\delta_{n,m+1} + \frac{1}{\sqrt{2}}\sqrt{m}\,\delta_{n,m-1}, \tag{3}$$

$$H_{nm}(\lambda) = H_{0,nm} + \lambda(x^4)_{nm}, \tag{4}$$

donde δ es el delta de Kronecker (1 si $n = m$, 0 en cualquier otro caso), $x^4 = x \times x \times x \times x$, y los valores propios dependen del tamaño de la matriz y de λ, $E_n = E_n(N, \lambda)$. Los valores propios de la matriz $H_{nm}(\lambda)$ representan las energías de los estados propios del sistema. El menor valor corresponde al estado base y está dado por $E_0(N, \lambda)$. En este punto, usted explorará cómo depende $E_0(N, \lambda)$ con N, para $\lambda = 0.2$ fijo.

Cree un programa que calcule los valores propios del hamiltoniano perturbado, le permita extraer el de menor valor, y al final imprima el valor de ese valor propio en función del inverso del tamaño del sistema, $1/N$, para $\lambda = 0.2$. Tome $N = 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024$. La salida debe ser algo como

```
5.0000000000000000e-01   5.5000000000000004e-01
2.5000000000000000e-01   5.9470546427531512e-01
1.2500000000000000e-01   6.0248643158149484e-01
⋮                        ⋮
1.9531250000000000e-03   6.0240516368327490e-01
9.7656250000000000e-04   6.0240516362982810e-01
```

Y se debe tardar alrededor de 25 segundos si compilaron con -O3 (aunque esto también depende del procesador que esté usando, puede demorarse mucho más) . . Use como plantilla el código que sigue:

```cpp
#include <iostream>
#include <cmath>
#include <algorithm>
#include <Eigen/Dense>
#include <Eigen/Eigenvalues>

void set_H0(Eigen::MatrixXd & M);
void set_X(Eigen::MatrixXd & M);
double eigen_energy(Eigen::MatrixXd & H, Eigen::MatrixXd & X, const double lambda, const int index);

int main(int argc, char **argv)
{
  std::cout.precision(16);
  std::cout.setf(std::ios::scientific);

  double lambda = 0.2;
  for (int N = 2; N <= 1024; N *= 2) {
    Eigen::MatrixXd X(N, N), H0(N, N);
```

```
    set_H0(H0);
    set_X(X);
    std::cout << 1.0/N << " " << eigen_energy(H0, X, lambda, 0) << std::endl;
  }

  return 0;
}


void set_H0(Eigen::MatrixXd & M)
{
  M.setZero();
  // Escriba aca el codigo que crea la matriz H0
  // Puede guiarse por la funcion set_X
}

void set_X(Eigen::MatrixXd & M)
{
  M.setZero();
  for (int n = 0; n < M.cols(); ++n){
    for (int m = 0; m < M.cols(); ++m){
      if (n == m+1) M(n, m) += std::sqrt((m+1.0)/2.0);
      if (n == m-1) M(n, m) += std::sqrt((m)/2.0);
    }
  }
}

double eigen_energy(Eigen::MatrixXd & H, Eigen::MatrixXd & X, const double lambda, const int index)
{
  // Implemente aca el calculo de los valores propios, usando la libreria eigen

  // - Calculo de Hlambda :

  // - Calculo de los valores propios (y vectores propios) :

  // - Extraer los valores propios al vector Eigen::VectorXd evals, solamente
  //   la parte real (cuando pida los valores propios escriba .eigenvalues().real()) :
  Eigen::VectorXd evals = // ... ??

  // ordenar los valores propios :
  std::sort(evals.data(), evals.data() + evals.size());
  // retornar el valor propio (para el estado base index == 0) :
  return evals(index);
}
```

Como puede ver, este programa también le es útil para calcular la energía de estados excitados cuando el índice es diferente a cero. Trate de hacer las cosas lo más sencillas posibles. De hecho, prácticamente cada instrucción que se pide en `eigen_energy` se hace en una sola línea de código.