Project 2

Jane Kim
Physics 480: Computational Physics
(Dated: March 8, 2018)

Abstract

I. INTRODUCTION

Many problems in physics are difficult, if not impossible, to solve analytically. However, in many cases, they can be reduced to relatively simple eigenvalue problems which can be solved numerically. In this project, we developed an eigenvalue and eigenvector solver using Jacobi's algorithm for real, symmetric matrices in order to solve three different examples of such problems with varying levels of complexity.

A. Buckling Beam

For the simplest of the three problems, we consider a beam of length L, whose vertical displacement in the y-direction is $u(x)$ for $x \in [0, L]$. If R is some constant which reflects the properties, such as the rigidity, of the beam and a force F is applied at $x = L$ towards the origin, the vertical displacement $u(x)$ satisfies the differential equation

$$R\frac{d^2u(x)}{dx^2} = -Fu(x), \tag{1}$$

with homogenous boundary conditions $u(0) = u(L) = 0$. Since different combinations of the physical paramters F, R, and L can yield the same solution, it is beneficial to introduce the dimensionless quantity $\rho = \frac{x}{L}$ to scale the differential equation. Then (1) becomes

$$\frac{d^2u(\rho)}{d\rho^2} = -\lambda u(\rho), \tag{2}$$

where $\lambda = FL^2/R$ and $\rho \in [0, 1]$. For a given number of mesh points N, we can obtain a discretized approximation to $u(\rho)$ by rewriting (2) as an eigenvalue problem given by

$$A\vec{u} = \lambda\vec{u},$$

$$\begin{bmatrix} d & a & & & \\ a & d & a & & \\ & \ddots & \ddots & \ddots & \\ & & a & d & a \\ & & & a & d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix}, \tag{3}$$

$$u_i = u(\rho_i),$$
$$\rho_i = ih,$$

where $d = 2/h^2$, $a = -1/h^2$, and $h = 1/N$ is the step size. At the endpoints, $u_0 = u_N = 0$. This Toeplitz matrix has analytic eigenvalues

$$\lambda_i = d + 2a\cos\left(\frac{i\pi}{N+1}\right), \quad i = 1, 2, ..., N-1, \tag{4}$$

which we used to check the accuracy of the eigenvalue solver.

B. One Electron Harmonic Oscillator

Jacobi's algorithm is applicable for any real, symmetric matrix, so the same program we developed to solve the buckling beam problem can be used to solve the time-independent Schrödinger equation for one and two electrons in a three-dimensional harmonic oscillator potential. For one electron with $l = 0$, we substitute $R(r) = u(r)/r$ into the radial equation and introduce a dimensionless variable $\rho = r/\alpha$ to obtain

$$\left(-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2} + \frac{1}{2}m\omega^2\alpha^2\rho^2\right)u(\rho) = Eu(\rho). \tag{5}$$

By fixing $\alpha = (\hbar/m\omega)^{1/2}$ and letting $\lambda = 2m\alpha^2E/\hbar^2 = 2E/\hbar\omega$, (5) becomes

$$\left(-\frac{d^2}{d\rho^2} + \rho^2\right)u(\rho) = \lambda u(\rho). \tag{6}$$

The boundary conditions are $u(0) = \lim_{\rho\to\infty} u(\rho) = 0$, but due to the finite nature of discretizing the differential equation, we must define a window $[\rho_{min}, \rho_{max}]$ on which to compute the eigenvalues and eigenvectors. Then the discretized equation is given by

$$-\frac{1}{h^2}u_{i-1} + \left(\frac{2}{h^2} + \rho_i^2\right)u_i - \frac{1}{h^2}u_{i+1} = \lambda u_i, \tag{7}$$

where $h = (\rho_{max} - \rho_{min})/N$, $\rho_i = \rho_{min} + ih$, $i = 1, ..., N-1$. Therefore, the only difference between this eigenvalue problem and the buckling beam problem is that the diagonal elements have an additional non-constant term ρ_i^2. This problem also has analytic eigenvalues given by $\lambda = 3, 7, 11, 15, ...$.

EIGENVALUE CALCULATIONS ARE MOST ACCURATE WHEN WINDOW CONTAINS VALUES OF RHO FOR WHICH THE CORRESPONDING EIGENVECTOR (OR U(RHO)) IS NOT IDENTICALLY ≈ 0

C. Two Electron Harmonic Oscillator

When the repulsive Coulomb interaction is ignored, the Hamiltonian H_0 for two electrons in a harmonic oscillator can be written with respect to the center-of-mass coordinate $\mathbf{R} = (\mathbf{r}_1 + \mathbf{r}_2)/2$ and the relative coordinate $\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2$:

$$\begin{aligned} H_0(r, R) &= H_0(r) + H_0(R) \\ H_0(r) &= -\frac{\hbar^2}{m}\frac{d^2}{dr^2} + \frac{1}{4}kr^2 \\ H_0(R) &= -\frac{\hbar^2}{4m}\frac{d^2}{dR^2} + kR^2 \end{aligned} \tag{8}$$

where $k = m\omega^2$. Then with the Coulomb interaction

$$V(r) = \frac{\beta e^2}{r}, \quad \beta e^2 = 1.44 \text{ eV} \cdot \text{nm} \tag{9}$$

the r-dependent Schrödinger equation is given by

$$H(r)\psi(r) = [H_0(r) + V(r)]\psi(r) = E_r\psi(r) \tag{10}$$

As with one electron case, we substitute the dimensionless variable $\rho = r/\alpha$ in (10) to obtain

$$\left(-\frac{d^2}{d\rho^2} + \omega_r^2\rho^2 + \frac{1}{\rho}\right)\psi(\rho) = \lambda\psi(\rho). \tag{11}$$

Here, we have fixed $\alpha = \hbar^2/m\beta e^2$ and defined $\lambda = m\alpha^2 E/\hbar^2$. Thus the two electron case simply adds another term $1/\rho_i$ to the diagonal elements of A from the one electron case.

II. METHOD

Jacobi's rotation algorithm is an iterative method for approximating the eigenvalues and eigenvectors of a real, symmetric, $N \times N$ matrix. Say we have such a matrix A, and it has a non-zero element $a_{k\ell}$ for some $k, \ell = 1, 2, ..., N$ and $k \neq \ell$. Let $J(k, l, \theta)$ be a rotation matrix of the form

$$J(k,l,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}, \tag{12}$$

where $J_{kk} = J_{\ell\ell} = \cos\theta = c$ and $J_{k\ell} = -J_{\ell k} = \sin\theta = s$. This matrix is orthogonal, so for some arbitrary angle θ, conjugation of A by $J(k, \ell, \theta)$ is given by

$$A' = J(k,\ell,\theta)^T A J(k,\ell,\theta), \tag{13}$$

where A' has the entries

$$a'_{kk} = c^2 a_{kk} + s^2 a_{\ell\ell} - 2cs a_{k\ell} \tag{14}$$
$$a'_{\ell\ell} = s^2 a_{kk} + c^2 a_{\ell\ell} + 2cs a_{k\ell} \tag{15}$$
$$a'_{ik} = a'_{ki} = c a_{ik} - s a_{i\ell}, \quad i \neq k, i \neq \ell \tag{16}$$
$$a'_{il} = a'_{li} = c a_{i\ell} + s a_{ik}, \quad i \neq k, i \neq \ell \tag{17}$$
$$a'_{k\ell} = a'_{\ell k} = (c^2 - s^2)a_{k\ell} + sc(a_{kk} - a_{\ell\ell}) \tag{18}$$

We can calculate the special angle θ for which the elements $a'_{k\ell}$ and $a'_{\ell k}$ become zero using equation (18). In this way, Jacobi's algorithm diagonalizes the matrix A by repeatedly applying similarity transformations of the form (13) to "zero out" the off-diagonal elements $a_{k\ell}$. The largest off-diagonal element is chosen to increase efficiency. The eigenvalues of A are the diagonal elements of the resulting matrix.

To calculate θ, we set (18) equal to zero to obtain

$$\frac{a_{\ell\ell} - a_{kk}}{a_{k\ell}} = \frac{c^2 - s^2}{cs} = \frac{c}{s} - \frac{s}{c} = \frac{1}{t} - t, \tag{19}$$

where $\tan\theta = t = s/c$. We define $\tau = (a_{\ell\ell} - a_{kk})/2a_{kl}$ so that (19) can be rewritten as the quadratic equation

$$t^2 + 2\tau t - 1 = 0, \tag{20}$$

which has the analytic solutions $t = -\tau \pm \sqrt{1 + \tau^2}$. Once t is known, c and s are calculated using

$$c = \frac{1}{\sqrt{1 + t^2}}, \quad s = tc. \tag{21}$$

In addition, the products c^2, s^2, and cs are precalculated before applying equations (14) and (15).

Suppose the matrix A is (nearly) diagonal after n rotations, i.e.

$$D = J_n^T \cdots J_2^T J_1^T A J_1 J_2 \cdots J_n \tag{22}$$

for some rotation matrices J_i and diagonal matrix D. Then the eigenvectors are simply given by the columns of the product $J_1 J_2 \cdots J_n$.

III. IMPLEMENTATION

A. Jacobi's Algorithm

The bulk of Jacobi's algorithm can be broken down into two parts: the search for the largest off-diagonal element $a_{k\ell}$, called a pivot, followed by the corresponding rotation of the form (13). Accordingly, we developed a function called `get_pivot` which stores the indices k, ℓ of the largest element of A above the diagonal (since A is symmetric).

```
// jacobi.cpp
void get_pivot(mat& A, int N, int& k, int& l){
   double Aij,max_offdiag = 0.0;
   for(int i = 0; i < N-1; i++){
      for(int j = i+1; j < N; j++){
         Aij = fabs(A(i,j));
         if(Aij > max_offdiag){
            max_offdiag = Aij;
            k = i;
            l = j;
         }
      }
   }
}
```

Meanwhile, another function called `rotate` carries out the transformation $A' = J(k,\ell,\theta)^T AJ(k,\ell,\theta)$ and right-multiplies a matrix V by $J(k,\ell,\theta)$. V is initialized as the identity matrix before the first rotation so that the columns of the resulting matrix are the eigenvectors of A.

```
// jacobi.cpp
void rotate(mat& A, mat& V, int k, int l, int N){

   // zero out A(k,l)=A(l,k)
   if( A(k,l) != 0.0 ){

      double c, s, t, tau;
      double cc, ss, cs;
      double Aik, Ail, Vik, Vil;
      double Akk = A(k,k), All = A(l,l), Akl =
          A(k,l);

      // calculate angle of rotation
      tau = 0.5*(All-Akk)/Akl;
      if(tau >= 0.0){ t = -tau+sqrt(1.0+tau*tau); }
      else{ t = -tau-sqrt(1.0+tau*tau); }

      cc = 1.0/(1.0+t*t);
      ss = 1.0-cc;
      cs = t*cc;
      c = sqrt(cc);
      s = t*c;

      // perform rotation
      A(k,l) = 0.0;
      A(l,k) = 0.0;
      A(k,k) = cc*Akk+ss*All-2.0*cs*Akl;
      A(l,l) = ss*Akk+cc*All+2.0*cs*Akl;
      for(int i = 0; i < N; i++){

         if( (i!=k) && (i!=l) ){
            Aik = A(i,k);
            Ail = A(i,l);
            A(i,k) = c*Aik-s*Ail;
            A(i,l) = c*Ail+s*Aik;
            A(k,i) = A(i,k);
            A(l,i) = A(i,l);
         }

         // rotate eigenvectors
         Vik = V(i,k);
         Vil = V(i,l);
         V(i,k) = c*Vik-s*Vil;
         V(i,l) = s*Vik+c*Vil;
      }
   }

   else{ cout << "ERROR: These elements are already
        zero!" << endl; }
}
```

A is considered diagonalized after applying `get_pivot` and `rotate` until the sum of squares of the off-diagonal elements is smaller than some number ϵ.

IV. RESULTS

Data was collected for five different temperatures below the lambda point and the speed of second sound was calculated using equation (6). The resonance peaks were found manually by searching for local maxima in the lock-in amplitude data. The average frequency difference Δf between adjacent resonance peaks and the speed of second sound u_2 for each temperature is shown in the table below. The results are shown in FIG. 5. alongside published data from R.J. Donnelly[?]

T (K)	δT (K)	Δf (Hz)	$\delta(\Delta f)$ (K)	u_2 (m/s)	δu_2 (m/s)
1.6727	0.0013	251.3	1.414	20.11	11.06
1.9602	0.0026	232.5	1.414	18.60	10.23
2.0889	0.0034	185.0	1.414	14.80	8.141
2.1451	0.0130	110.7	1.414	8.856	4.872
2.1624	0.0064	65.50	1.414	5.240	2.884

The horizontal error bars (δT) correspond to the variation in temperature during each sweep. The vertical error bars (δu_2) were calculated with the following:

$$u_2 = 2L\Delta f = 2L(f_n - f_{n-1}) \tag{23}$$

$$\frac{\delta u_2}{u_2} = \sqrt{\left(\frac{\delta L}{L}\right)^2 + \left(\frac{\delta(f_n - f_{n-1})}{\Delta f}\right)^2} \tag{24}$$

The error in resonance frequency measurements was determined to be the frequency step size of 1 Hz. So, $\delta(f_n - f_{n-1}) = \sqrt{2}$ Hz. δL was estimated as the expected change in the length L due to thermal contraction. The thermal expansion coefficient for brass[2] is $\alpha \approx 18.5 \cdot 10^{-6} K^{-1}$, so

$$\begin{aligned}\delta L &\approx \Delta L \\ &= \alpha L \Delta T \\ &\approx (18.5 \cdot 10^{-6}\ K^{-1})(4.0\text{ cm})(300\ K - 2.17\ K) \\ &= 0.022\text{ cm}\end{aligned} \tag{25}$$

All five data points were consistent with the published data[1].

V. CONCLUSION

The speed of second sound was measured in HeII for five different temperatures below T_λ. They agreed with the published data from R.J. Donnelly within the margins of error. We are certain that the peaks we identified were second sound peaks because they appeared only after cooling the liquid nitrogen below the lambda point (FIG. 4).

Jacobi algorithm is a brute force way to get eigenvalues, but not very efficient.

VI. ACKNOWLEDGEMENTS

I would like to thank my lab partner, Antonius Torode, for taking an entire day to correct our crappy data with me. Special thanks to Jaideep Singh and Johannes Pollanen for providing guidance and insight into the sources of error in our initial attempt of this experiment.

[1] R.J. Donnelly. "Experimental Superfluidity". University of Chicago Press. 1967.

[2] "Coefficients of Linear Thermal Expansion". The Engineering ToolBox. EngineeringToolBox.com.

[3] Otis Chodosh, Jeremy Hiatt, Samir Shah, and Ning Yan. "Second Sound in HeII". Department of Physics, Stanford University. March 21, 2008.

[4] "Second Sound in Superfluid Liquid He". Department of Physics and Astronomy, Michigan State University. PHY 451.

[5] "Superfluidity". New World Encyclopedia. October 28, 2015.