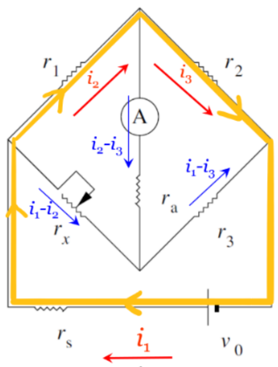
1. Prove that the complexity of the Gaussian elimination algorithm is $O(N^3)$.

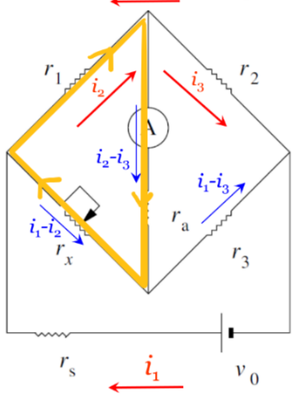
For a $N \times N$ matrix, when we turn $\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \dots & A_{NN} \end{pmatrix}$ into $\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ 0 & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & A_{N2} & \dots & A_{NN} \end{pmatrix}$, it takes us $N \times (N-1)$ additions and $N \times (N-1)$ multiplications. Then we turn $\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ 0 & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & A_{N2} & \dots & A_{NN} \end{pmatrix}$ into $\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ 0 & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_{NN} \end{pmatrix}$ and it takes us $(N-1) \times (N-2)$ additions and $(N-1) \times (N-2)$ multiplications. So, the times of additions we take are $sum = \sum_{n=2}^{N} n(n-1) = \frac{1}{3}(N-1)N(N+1)$, and it's same to multiplications. So, the times of operations we take in total are $\frac{2}{3}(N-1)N(N+1)$. When N is a large number, it tends to $\frac{2}{3}N^3 \sim O(N^3)$.

2. Solve the unbalanced Wheatstone bridge.

From the Kirchhoff Equations we know that:



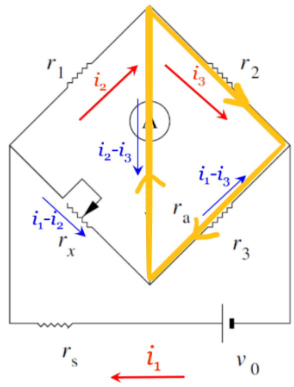
$$r_s i_1 + r_1 i_2 + r_2 i_3 = v_0$$



$$r_1 i_2 + r_a(i_2 - i_3) - r_x(i_1 - i_2)$$

$$= -r_x i_1 + (r_1 + r_x + r_a)i_2 - r_a i_3 = 0$$

$$r_2 i_3 - r_3(i_1 - i_3) - r_a(i_2 - i_3)$$
$$= -r_3 i_1 - r_a i_2 + (r_2 + r_3 + r_a) i_3 = 0$$



Hence, we get the equations: $\begin{cases} r_s i_1 + r_1 i_2 + r_2 i_3 = v_0 \\ -r_x i_1 + (r_1 + r_x + r_a) i_2 - r_a i_3 = 0 \\ -r_3 i_1 - r_a i_2 + (r_2 + r_3 + r_a) i_3 = 0 \end{cases}$

What we need to do is to solve the equations through its augmented matrix by using Gaussian elimination and backward substitution.

The augmented matrix of the equations:

$$\begin{pmatrix} r_s & r_1 & r_2 & v_0 \\ -r_x & r_1 + r_x + r_a & -r_a & 0 \\ -r_3 & -r_a & r_2 + r_3 + r_a & 0 \end{pmatrix}$$

To solve this question, we need to define 3 functions. Firstly, we use function get_parameter() to get the input. Inside this function, we set the voltage of the power $v_0 = 5.0V$. Actually, it doesn't matter because we will divide it anyway when we calculate the resistance $R = \frac{v_0}{i_1}$.

Then we define the function Gauss(x,n), which needs 2 inputs, x is the matrix which needs to be Gaussian eliminated and n is the order of the coefficient matrix. Inside the function, we firstly need a for-loop, i starts from 0 to n-1. When i=0, it means we are turning:

$$\begin{pmatrix} A_{00} & A_{01} & \cdots & A_{0,n-1} \\ A_{10} & A_{11} & \cdots & A_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n-1,0} & A_{n-1,1} & \cdots & A_{n-1,n-1} \end{pmatrix} \rightarrow \begin{pmatrix} A_{00} & A_{01} & \cdots & A_{0,n-1} \\ 0 & A_{11} & \cdots & A_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & A_{n-1,1} & \cdots & A_{n-1,n-1} \end{pmatrix}$$

Inside the for-loop, we firstly need to make sure that the diagonal element is not 0 (we check A_{ii} in i-th loop). When $A_{ii} = 0$, we need a for-loop to search for a non-zero element in i-th column, and then we interchange the two rows. After $A_{ii} \neq 0$, we use another for-loop to eliminate the $A_{ki}\ (for\ k > i)$, and the ratio is $-\frac{A_{ki}}{A_{ii}}$. We multiply the i-th row by the ratio and add it to the k-th row. After this process, we will get a row-echelon matrix, from which we can easily derive the solutions of the equations.

At last, we define the function Back_sub(x,n), which use backward substitutions to solve the equations. The function needs 2 inputs, x is the row-echelon matrix we derive from Gauss(x,n), and n is the order of the coefficient matrix. Firstly, we need a for-loop to go through the last row to the first row. Inside i-th loop, we need another loop to calculate the i-th solution. $solution[i] = (A_{i,n} - \sum_{j=i+1}^{n-1} A_{i,j} \times solution[j])/A_{i,i}$. The function returns a matrix which stores the solutions.

The question is easily solved after we combine the 3 functions together.

Input: $r_s, r_x, r_a, r_1, r_2, r_3$ /* Actually, we input a matrix of r*/

Output: The effective resistance of the circuit

Function Gauss(matrix, n)

For $i \leftarrow 0$ to $(n-1)$ **Do**

    For $j \leftarrow (i+1)$ to $(n-1)$ **Do**

        $ratio \leftarrow -\frac{matrix[j][i]}{matrix[i][i]}$

        For $k \leftarrow 0$ to n **Do**

            $matrix[j][k] \leftarrow matrix[j][k] + ratio \times matrix[i][k]$

Return $matrix$

Function Back_sub(matrix, n)

$solution \leftarrow \{0 \ldots 0\}$ /*n zeros in the set*/

For $i \leftarrow (n-1)$ to 0 **Do**

    $solution[i] \leftarrow matrix[i][n]$

    For $j \leftarrow (i+1)$ to $(n-1)$ **Do**

        $solution[i] \leftarrow matrix[i] - matrix[i][j] \times solution[j]$

    $solution[i] \leftarrow \frac{solution[i]}{matrix[i][i]}$

Return $solution$

```
3_2_17307110134.py ×
C: > Users > 11765 > Desktop > 学习 > 物理 > 计算物理 > 作业 > homework3 > 3_2_17307110134.py > ...
import numpy as np

def main():
    matrix = get_parameter()         # Firstly, we use a function to get the input.
    equation = Gauss(matrix,3)       # The function Gauss() can Gaussian eliminate the matrix.
    solution = Back_sub(equation,3) # The function Back_sub() finishes the backward substitutions.
    resistance = 5.0/solution[0]     # Solution is a matrix which stores the values of the current i1, i2 and i3.
    print('The effective resistance of the circuit is %s' % resistance)

def Gauss(x,n):
# The Gaussian elimination function needs 2 parameters, x is the matrix that we need to solve, n is the dimension of the matrix.
    equation = np.asarray(x,dtype=float)

    for i in range(n):
        if equation[i][i] == 0.0:
        # We need to make sure that the diagonal elements are not 0, or we need to change the row.
            for j in range(i+1,n):
                if equation[j][i] != 0.0:
                    equation[i],equation[j] = equation[j],equation[i]
                    break

        for k in range(i+1,n):
            ratio = -(equation[k][i] / equation[i][i])
            # Ratio is the number we need to multiply before we add the elements of 2 rows together.
            for l in range(n+1):
                equation[k][l] = equation[k][l] + ratio*equation[i][l]

    # After these operations, we turn the coefficient matrix into an upper-triangular matrix.
    return equation

def Back_sub(x,n):
# Backward substitutions
    solution = np.zeros(n)  # We initialize a matrix to store the values of i1, i2 and i3 we've solved from the equations.
    matrix = x.copy()

    for i in range(n-1,-1,-1):
    # We start the backward substitution from the last row, so the index starts from n-1 to 0.
        solution[i] = matrix[i][n]

        for j in range(i+1,n):
            solution[i] = solution[i] - matrix[i][j]*solution[j]
```

Source code.

```
PS C:\Users\11765\Desktop\学习\物理\计算物理\作业\homework3>
launcher' '60419' '--' 'c:\Users\11765\Desktop\学习\物理\计算
Please input the value of resistor r_s:1
Please input the value of resistor r_x:2
Please input the value of resistor r_a:3
Please input the value of resistor r_1:4
Please input the value of resistor r_2:5
Please input the value of resistor r_3:6
The effective resistance of the circuit is 5.136752136752139
PS C:\Users\11765\Desktop\学习\物理\计算物理\作业\homework3>
```

We input the values of the resistor, and we get the effective resistance of the circuit.

3.（题目太复杂，用英语解释不清）第三题的核心思路是利用 QR 分解计算矩阵本征值，而 QR 分解又首先需要对矩阵进行 Gram-Schmidt 正交化。首先定义一些常用的线性代数功能函数：

dot()点积，对两个维度相等的向量，将索引相等的元素相乘后再将乘积求和，即可得到点积。

norm()求模，将向量与自身点积，再开方即可得到向量的模。

求逆，对矩阵求逆可以利用第二题已经完成的高斯消元法的思路。将单位阵作为辅助矩阵扩充原先的矩阵，再利用高斯消元法和回代将原先的矩阵变为单位阵，此时辅助矩阵即变为逆矩阵。为了节约时间并增加代码的可读性，在第三题内调用 numpy.linalg.inv()函数来求矩阵的逆矩阵。

Schmidt()将矩阵 Gram-Schmidt 正交化。由于是将矩阵的列向量 Schmidt 正交化，而矩阵对行进行遍历操作更方便，所以先将矩阵转置得到 x_T。初始化一个与 x_T 行数列数相等的零矩阵 init。利用 for 循环进入 x_T 的 i-th 行，内部再利用 for 循环进入 j-th 列，即 x_T 第 i 行第 j 列的元素，令临时变量 temp=x_T[i][j]。对于第 j 列的元素，对 x_T[i][j]之前的每一个元素，$temp = temp - \frac{(x\_T[i], init[k])}{(init[k], init[k])} \times init[k][j]$，再将 temp 的值赋给 init[i][j]。这样 init 矩阵就变为 x_T 矩阵在 Schmidt 正交化之后的矩阵。再把第 i 行的每一个元素 init[i][j]除以第 i 行的模 norm(init[i][j])，即完成单位化的步骤。

QR(x)将矩阵 x 进行 QR 分解$x = QR$。如果 x 是 m×n 矩阵，那么 Q 也是 m×n 矩阵，且 Q 的每一列都是 x 矩阵列的正交基；R 是 n×n 的上三角矩阵。利用 Schmidt 正交化，$\mathrm{Q} = \mathrm{Schmidt}(\mathrm{x}), R = Q^T x$。

eigenvalue()求本征值。QR 分解，$A_k = Q_k R_k$，由于 Q 是正交矩阵，$A_{k+1} = R_k Q_k = {Q_k}^{-1} Q_k R_k Q_k = {Q_k}^{-1} A_k Q_k$。经过不断迭代即可得到一个对角矩阵，其对角元是$A$的本征值，迭代的次数由函数中的 time 决定。

eigenvector()求本征向量。通过 eigenvalue(A)求得本征值后，我们得到矩阵 A 以及对角元为 A 的本征值的对角矩阵$B = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix}$，那么本征向量即可通过$C = (A - B)^{-1}$求得，C 矩阵的每一个列向量都是一个 A 的本征向量。

接下来计算H矩阵的矩阵元$H_{jk} = \int_{-\infty}^{\infty} \Phi^*{}_j(x) H \Phi_k(x) dx$，由于高斯基不正交，还需要计算 S 矩阵的矩阵元$S_{jk} = \int_{-\infty}^{\infty} \Phi^*{}_j(x) \Phi_k(x) dx$。通过 Mathematica 计算得到：

```
In[∘]:= ϕk = Sqrt[vk / Pi] * Exp[-vk * (x - sk) ^2]
             平方根      …     指数形式
        ϕj = Sqrt[vj / Pi] * Exp[-vj * (x - sj) ^2]
             平方根      …     指数形式
        Integrate[ϕj * (α * D[D[ϕk, x], x] + β * x^2 * ϕk), {x, -Infinity, Infinity}]
        积分                ·· 偏导                              无穷大     无穷大
```

Out[∘]= $\dfrac{e^{-vk\,(-sk+x)^2}\sqrt{vk}}{\sqrt{\pi}}$

Out[∘]= $\dfrac{e^{-vj\,(-sj+x)^2}\sqrt{vj}}{\sqrt{\pi}}$

Out[∘]= ConditionalExpression$\left[\dfrac{e^{-\frac{(sj-sk)^2\,vj\,vk}{vj+vk}}\sqrt{vj}\sqrt{vk}\left(4\,vj\,vk\left(-vk+vj\left(-1+2\,(sj-sk)^2\,vk\right)\right)\alpha+\left(vj+2\,sj^2\,vj^2+vk+4\,sj\,sk\,vj\,vk+2\,sk^2\,vk^2\right)\beta\right)}{2\sqrt{\pi}\,(vj+vk)^{5/2}},\ \mathrm{Re}[vj+vk]\geq 0\right]$

In[∘]:= Simplify$\left[\dfrac{e^{-\frac{(sj-sk)^2\,vj\,vk}{vj+vk}}\sqrt{vj}\sqrt{vk}\left(4\,vj\,vk\left(-vk+vj\left(-1+2\,(sj-sk)^2\,vk\right)\right)\alpha+\left(vj+2\,sj^2\,vj^2+vk+4\,sj\,sk\,vj\,vk+2\,sk^2\,vk^2\right)\beta\right)}{2\sqrt{\pi}\,(vj+vk)^{5/2}}\right]$
化简

Out[∘]= $\dfrac{e^{-\frac{(sj-sk)^2\,vj\,vk}{vj+vk}}\sqrt{vj}\sqrt{vk}\left(4\,vj\,vk\left(-vk+vj\left(-1+2\,(sj-sk)^2\,vk\right)\right)\alpha+\left(vj+2\,sj^2\,vj^2+vk+4\,sj\,sk\,vj\,vk+2\,sk^2\,vk^2\right)\beta\right)}{2\sqrt{\pi}\,(vj+vk)^{5/2}}$

```
In[∘]:= Integrate[ϕj * ϕk, {x, -Infinity, Infinity}]
        积分                  无穷大     无穷大
```

ConditionalExpression$\left[\dfrac{e^{-\frac{(sj-sk)^2\,vj\,vk}{vj+vk}}\sqrt{vj}\sqrt{vk}}{\sqrt{\pi}\sqrt{vj+vk}},\ \mathrm{Re}[vj+vk]>0\right]$
条件表达式 实部

可以看到，当我们保持高斯函数Φ_j, Φ_k中的v_k, s_k, v_j, s_j均为变量时，积分得到的结果已经过于复杂。我们需要设定一些常数以简化问题。令$v_k = v_j = 2$，再次用 Mathematica 积分得到：

```
In[3]:= ϕ2j = Sqrt[2 / Pi] * Exp[-2 * (x - sj) ^2]
              平方根      …    指数形式
```

Out[3]= $e^{-2\,(-sj+x)^2}\sqrt{\dfrac{2}{\pi}}$

```
In[2]:= ϕ2k = Sqrt[2 / Pi] * Exp[-2 * (x - sk) ^2]
              平方根      …    指数形式
```

Out[2]= $e^{-2\,(-sk+x)^2}\sqrt{\dfrac{2}{\pi}}$

```
In[6]:= Integrate[ϕ2j * (α * D[D[ϕ2k, x], x] + β * x^2 * ϕ2k), {x, -Infinity, Infinity}]
        积分                  ·· 偏导                                  无穷大     无穷大
```

Out[6]= $\dfrac{e^{-(sj-sk)^2}\left(16\left(-1+2\,(sj-sk)^2\right)\alpha+\beta+2\,(sj+sk)^2\,\beta\right)}{8\sqrt{\pi}}$

```
In[4]:= Integrate[ϕ2j * ϕ2k, {x, -Infinity, Infinity}]
        积分                    无穷大     无穷大
```

Out[4]= $\dfrac{e^{-(sj-sk)^2}}{\sqrt{\pi}}$

虽然结果仍然有点复杂，但已经大大简化。$H_{jk} = \frac{\alpha e^{-(s_j-s_k)^2}\left(16\left(2(s_j-s_k)^2-1\right)\right)+\beta+2(s_j+s_k)^2\beta}{8\sqrt{\pi}}$,其中$\alpha = -\frac{\hbar^2}{2m}, \beta = \frac{1}{2}m\omega^2, S_{jk} = \frac{e^{-(s_j-s_k)^2}}{\sqrt{\pi}}$。

取合适的m,ω使$\alpha = -1, \beta = 1$，得到：$H_{jk} = \frac{-e^{-(s_j-s_k)^2}\left(16\left(2(s_j-s_k)^2-1\right)\right)+1+2(s_j+s_k)^2}{8\sqrt{\pi}}$。

定义 calculat_H(x,y)函数计算 H 矩阵中的元素。在函数中：$temp1 = e^{-(x-y)^2}, temp2 = 16(2(x-y)^2-1), temp3 = 2(x+y)^2, temp4 = 8\sqrt{\pi}$。将四个临时变量组合在一起即可得到$H(x,y) = \frac{temp1\times temp2+1+temp3}{temp4}$。

定义 calculat_S(x,y)函数计算 S 矩阵中的元素。同样的，在函数中：$temp1 = e^{-(x-y)^2}, temp2 = \sqrt{\pi}$。将临时变量组合在一起得到$S(x,y) = \frac{temp1}{temp2} = \frac{e^{-(x-y)^2}}{\sqrt{\pi}}$。

我们还需要一个函数来完成以下过程：

$$HC = ESC \Longrightarrow \underset{H'}{(S^{-1/2}HS^{-1/2})}\underset{C'}{(S^{1/2}C)} \underset{=}{=} \underset{EC'}{E(S^{1/2}C)}$$

定义 new_H(x,y)函数计算 H’矩阵。$temp\_mat = \left(y^{\frac{1}{2}}\right)^{-1} = y^{-\frac{1}{2}}$, 因此$H' = y^{-\frac{1}{2}}xy^{-\frac{1}{2}} = S^{-\frac{1}{2}}xS^{-\frac{1}{2}}$。

定义函数 list_s(x)为s_j, s_k赋值，选取j,k的数量为 100，H，S 以及 H’将是 100×100 的矩阵。令s_j, s_k为等差数列，差值即步长由输入函数的 x 控制，生成的等差数列s_j, s_k储存在列表中返回。

定义 generator(x,n)函数，x 即为s_j, s_k的步长，n 是矩阵的行数/列数，在上文中我们已经定为 100。通过 calculat_H(x,y)函数和 calculat_S(x,y)函数算出 H 矩阵和 S 矩阵的每一个矩阵元,并通过 new_H(x,y)函数求出$H' = S^{-\frac{1}{2}}HS^{-\frac{1}{2}}$。

<u>#刚得知可以使用库，所以改用 numpy 库来求解矩阵本征值，源代码主要看 main()函数的部分!!：</u>

由于用 numpy 的 linalg.eig()函数求解本征值会返回一个元素全为本征值

的列表，在这个列表中寻找实数本征值，即为我们所要找的 E。由于计算机进行的是数值计算，所以我们需要寻找的是虚部特别小的本征值，这些本征值可近似认为是实数。

在程序中取步长从 0.001—0.01，在所有本征值中查找实数本征值，本征值在 0.004 附近取得极小值，$\lambda = 1.12999$, 可近似认为是 1。而通过量子力学方法求得的$E_0 = \frac{1}{2}\hbar\omega$。上文中为计算简便，我设定了$\alpha = -\frac{\hbar^2}{2m} = -1, \beta = \frac{1}{2}m\omega^2 = 1$，通过这样设定的常数计算得到$\hbar\omega = 2$，第一个本征值恰好为 1，符合程序的结果。输出最后一个本征值对应的本征向量：

```
[-0.0289899 -0.03758006j  0.04816188-0.00542474j  0.04975687+0.02871325j
 -0.02200209+0.01056744j -0.05247667-0.04725808j  0.0595562 +0.01168456j
  0.00917092+0.01655222j  0.03839836+0.04768383j  0.08702035+0.07010918j
 -0.05186959-0.06831658j -0.07843137+0.02522146j -0.0288709 +0.05855456j
 -0.00568356-0.05087341j  0.07307395+0.04285388j  0.07490321-0.04197436j
 -0.04904062+0.00543843j  0.02532955-0.05007408j -0.06345521-0.11094469j
 -0.06770045+0.06503411j -0.00117193-0.04842331j  0.02028493-0.01153665j
 -0.04657557-0.08545115j  0.03937    +0.01276646j -0.02634882+0.05671191j
  0.04940478+0.04530479j  0.12487895+0.04005837j -0.04133209-0.00064578j
 -0.06247341-0.13049761j -0.09566759-0.11295219j -0.04789592-0.09481608j
 -0.04647953-0.00801349j  0.00928324+0.02134774j -0.04998651-0.10792887j
  0.02299625-0.01656454j -0.02026618+0.07122859j  0.00341922-0.03764679j
  0.01673087-0.01150612j -0.01077509-0.04117768j  0.08014544+0.07479922j
  0.05389867+0.05372473j  0.03010918-0.06081477j -0.00669829+0.01567109j
 -0.03103677-0.00315144j  0.09452797+0.02476705j  0.0105507 -0.02415321j
  0.06744642+0.10792444j -0.06500045+0.06463308j  0.05347822-0.00086762j
  0.00865943+0.03826947j -0.05066388-0.07269376j -0.02678506-0.01410612j
 -0.01041171+0.01574742j -0.07583256+0.03507661j  0.0364618 +0.01510685j
 -0.03148484+0.0266038j   0.00969481+0.05797885j  0.00432608+0.0217473j
  0.01090302+0.01167847j -0.05850173-0.04295407j  0.10508975+0.02471655j
  0.24526685+0.j         -0.07030084+0.02375595j -0.00292901-0.03506524j
  0.07148794-0.01868239j -0.00764768-0.01937002j  0.00411889-0.01286223j
  0.0746774 -0.06315108j  0.1126031 +0.01667213j -0.01663403+0.00953474j
 -0.00887345-0.01063119j  0.0894862 +0.02896751j -0.09363479+0.07679199j
 -0.07487073+0.07517074j -0.0434931 -0.06303731j -0.01338464-0.00092495j
  0.01461868-0.00968609j  0.00397689-0.00879931j  0.00051084+0.00712938j
  0.04559763-0.05500809j -0.03722695-0.10514115j  0.06099622-0.05128554j
 -0.24861528+0.08512897j -0.03851861-0.03366928j  0.07878591-0.0087414j
 -0.02473575-0.0151515j   0.17136081+0.05818863j  0.08838769-0.00375214j
  0.00359035+0.06265193j -0.01679901+0.05582031j -0.00483769-0.01382183j
 -0.052105  +0.05793621j -0.2999263 -0.00515482j -0.0202119 -0.0062881j
 -0.01532408-0.05056375j -0.01743795-0.00818931j  0.04688981+0.01082894j
 -0.00301287+0.01756036j  0.13912698-0.0076761j  -0.01224988+0.02863834j
  0.13258589+0.02306631j]
```

$$HC = ESC \Longrightarrow (S^{-1/2}HS^{-1/2})(S^{1/2}C) = E(S^{1/2}C)$$

$$\underset{H'}{} \qquad\qquad \underset{C'}{} \quad = \quad \underset{EC'}{}$$

这个本征向量即为C'，我们需要解得C以构造波函数，$C = S^{-\frac{1}{2}}C'$。

```python
#求S矩阵及S^(-1/2)矩阵
list1 = list_s(0.004)
mat_S = np.zeros([100,100])
for i in range(100):
    for j in range(100):
        mat_S[i][j] = calculate_S(list1[i],list1[j])

temp_mat = np.linalg.inv(sqrtm(mat_S))
new_eigenvector = np.matmul(temp_mat,eigenvector1)
```

这样就求得了本征向量C，这样即可画出波函数$\Psi = \sum_i c_i \Phi_i$。

由于源代码的大部分内容都是在写点积、模、Gram-Schmidt 正交化、QR 分解和 eigenvalue()求本征值，这部分内容伪代码不再展示。

Function generator(x, n)

$list1 \leftarrow [0, x, 2x, \ldots, (n-1)x]$

$mat\_H \leftarrow \begin{pmatrix} 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{pmatrix}$ /*$n \times n$ matrix*/

$mat\_S \leftarrow \begin{pmatrix} 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{pmatrix}$ /*$n \times n$ matrix*/

For $i \leftarrow 0$ to $(\mathrm{n}-1)$ **Do**

For $j \leftarrow 0$ to $(\mathrm{n}-1)$ **Do**

$$mat\_H[i][j] \leftarrow \frac{-e^{-(list1[i]-list1[j])^2}\left(16\left(2(list1[i]-list1[j])^2-1\right)\right)+1+2(list1[i]+list[j])^2}{8\sqrt{\pi}}$$

$$mat\_S[i][j] \leftarrow \frac{e^{-(list1[i]-list1[j])^2}}{\sqrt{\pi}}$$

$matrix \leftarrow (mat\_S)^{-\frac{1}{2}}(mat\_H)(mat\_S)^{-\frac{1}{2}}$

Return $matrix$

main()函数就是由 generator()函数生成矩阵H'，再求本征值、本征向量即可。