In science and engineering an important goal is to become a skilled practitioner by doing it yourself. The computer labs provide computational experience related to the analytic theory presented in the lectures.

The Eigenvalue Algorithm of Jacobi

This lab is about using an iterative technique involving plane rotations to find the eigenvalues of a symmetric matrix.

When $A \in \mathbf{R}^{n\times n}$ with $A = A^T$ the spectral theorem states the eigenvalues λ_i of A are real and the corresponding eigenvectors ξ_i may be chosen to form an orthonormal basis of $\mathbf{R}^n$. Thus $A = QDQ^T$ where D is the diagonal matrix and Q is the orthogonal matrix given by

$$D = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \qquad \text{and} \qquad Q = \left[\begin{array}{c|c|c|c} & & & \\ \xi_1 & \xi_2 & \cdots & \xi_n \\ & & & \end{array}\right]$$

Jacobi's method for solving the eigenvalue-eigenvector problem constructs a sequence of plane rotations which in the limit reduce the symmetric matrix A to a diagonal matrix. To understand how the method works consider the symmetric matrix $A \in \mathbf{R}^2$ and the rotation R given by

$$A = \begin{bmatrix} \alpha & \beta \\ \beta & \delta \end{bmatrix} \qquad \text{and} \qquad R = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}.$$

For convenience denote $c = \cos\varphi$ and $s = \sin\varphi$. Now compute

$$\begin{aligned} R^TAR &= \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} \alpha & \beta \\ \beta & \delta \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \\ &= \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} \alpha c - \beta s & \alpha s + \beta c \\ \beta c - \delta s & \beta s + \delta c \end{bmatrix} \\ &= \begin{bmatrix} \alpha c^2 - \beta cs - \beta cs + \delta s^2 & \alpha cs + \beta c^2 - \beta s^2 - \delta cs \\ \alpha cs - \beta s^2 + \beta c^2 - \delta cs & \alpha s^2 + \beta cs + \beta cs + \delta c^2 \end{bmatrix} \\ &= \begin{bmatrix} \alpha c^2 - 2\beta cs + \delta s^2 & (\alpha - \delta)cs + \beta(c^2 - s^2) \\ (\alpha - \delta)cs + \beta(c^2 - s^2) & \alpha s^2 + 2\beta cs + \delta c^2 \end{bmatrix}. \end{aligned}$$

The above matrix will be diagonal provided c and s are chosen such that

$$(\alpha - \delta)cs + \beta(c^2 - s^2) = 0.$$

To solve for c and s substitute $t = s/c$ so that

$$cs = c^2 t \quad \text{and} \quad c^2 - s^2 = c^2(1 - t^2).$$

Upon canceling c^2 it follows that t satisfies

$$(\alpha - \delta)t + \beta(1 - t^2) = 0.$$

To finish we need to show this equation has real solutions in t and then express c and s in terms of t.

If $\beta = 0$ then the original matrix was diagonal and there is nothing to do. Otherwise, $\beta \neq 0$ and the equation is quadratic. In standard form we have

$$-\beta t^2 + (\alpha - \delta)t + \beta = 0$$

with discriminant

$$(\alpha - \delta)^2 + 4\beta^2 \geq 0.$$

Since the discriminant is non-negative, the solutions t are real.

Now solve for c and s as follows. By the Pythagorean theorem

$$1 = c^2 + s^2 = c^2(1 + t^2).$$

Therefore

$$c = \frac{1}{\sqrt{1 + t^2}} \qquad \text{and} \qquad s = \frac{t}{\sqrt{1 + t^2}}.$$

We remark that $c \in [0, 1]$ and $s \in [-1, 1]$ implies $\varphi \in [-\pi/4, \pi/4]$.

To generalize the above idea to larger matrices $A \in R^{n \times n}$, consider the rotation in the plane spanned by the basis vectors e_p and e_q holding the other directions constant. Namely, let $R^{(pq)}(\varphi)$ be the matrix whose elements r_{ij} are the same as the identity except for the four elements

$$\begin{aligned} r_{pp} &= \cos\varphi, & r_{pq} &= \sin\varphi \\ r_{qp} &= -\sin\varphi, & r_{qq} &= \cos\varphi. \end{aligned}$$

Now construct a sequence of matrices $A^{(k)}$ such that $A^{(k)} \to D$ where D is the diagonal matrix of eigenvalues of A. Let $A^{(0)} = A$ and define

$$A^{(k+1)} = R^{(pq)}(\varphi_k)^T A^{(k)} R^{(pq)}(\varphi_k)$$

where p and q are such that

$$|A_{pq}^{(k)}| = \max\{\, |A_{ij}^{(k)}| : i \neq j \,\}$$

and φ_k is chosen as in the $n = 2$ case so $A_{pq}^{(k+1)} = 0$ with the additional requirement to choose t to be smallest in absolute value from among the two possible choices given by the quadratic equation.

The goal today is to write a program that uses the Jacobi method to find the eigenvalues of an individualized matrix. Your matrix may be obtained by clicking on the following link:

`https://fractal.math.unr.edu/~ejolson/466-22/eigvals/matrix.cgi`

Please do not use anyone else's matrix for this lab.

Solving Quadratic Equations

As already seen, finding the values of $\cos\varphi_k$ and $\sin\varphi_k$ that appear in the rotation $R^{(pq)}(\varphi_k)$ involve solving a quadratic equation. In the general case that equation is

$$-a_{pq}t^2 + (a_{pp} - a_{qq})t + a_{pq} = 0$$

where a_{ij} are the entries of the matrix $A^{(k)}$.

We discussed solving quadratic equations in the introductory lab; however, in this case extra care needs to be taken to reduce rounding errors because the size of the off-diagonal entries of $A^{(k)}$ becomes small in comparison to the diagonal entries as $A^{(k)} \to D$.

The usual quadratic formula gives

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

where $a = -a_{pq}$, $b = a_{pp} - a_{qq}$ and $c = a_{pq}$. The off-diagonal terms being small means a and c are small. Thus,

$$\sqrt{b^2 - 4ac} \approx |b| \quad \text{and so} \quad -b + \sqrt{b^2 - 4ac} \approx 0 \quad \text{if} \quad b > 0.$$

To avoid the loss of precision which comes from the near cancellation of two nearly equal numbers, rewrite the formula as

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a} \cdot \frac{b + \sqrt{b^2 - 4ac}}{b + \sqrt{b^2 - 4ac}} = \frac{-2c}{b + \sqrt{b^2 - 4ac}}.$$

Now $b > 0$ with a and c small implies

$$b + \sqrt{b^2 - 4ac} \approx b + |b| \approx 2b$$

and there is no loss of precision. If $b < 0$ a similar technique must be used to find the solution corresponding to the negative square root.

Following is a Julia function that employs the above ideas to find the solutions to the quadratic equation $at^2 + bt + c = 0$.

```
function quadsolv(a,b,c)
    sq=sqrt(b^2-4*a*c)
    np=-b+sq
    nm=-b-sq
    if abs(nm)<abs(np)
        t1=np/(2*a)
        t2=2*c/np
    else
        t1=2*c/nm
        t2=nm/(2*a)
    end
    return t1,t2
end
```

To test how it is working try solving

$$t^2 + 10000t + 1 = 0$$

and compare the result to the solution computed using the usual formula.

```
julia> a=1; b=10000; c=1
1

julia> p(t)=a*t^2+b*t+c
p (generic function with 1 method)

julia> t1,t2=quadsolv(a,b,c)
(-0.00010000000100000001, -9999.999899999999)
```

```
julia> tbad=(-b+sqrt(b^2-4*a*c))/(2*c)
-0.00010000000111176632

julia> p(t1)
0.0

julia> p(tbad)
-1.1176630732023796e-9
```

Note 8 digits are different between `t1` computed using the transformed expression versus `tbad` obtained from the usual quadratic formula.

Coding the Jacobi Eigenvalue Solver

The following discussion concerns the matrix which appears when I click the above link. That matrix is different than what you will obtain when you click the same link. To finish this lab please repeat these same steps but for own individualized matrix.

Upon clicking on the link, I obtained

Your matrix in Julia-compatible code is

```
A=[  8.52 -1.80  1.08 -3.27 -3.25;
    -1.80  1.76 -6.67 -2.67  4.05;
     1.08 -6.67  1.16 -8.34  2.78;
    -3.27 -2.67 -8.34  7.18  6.27;
    -3.25  4.05  2.78  6.27 -3.28 ]
```

Start editing a new file called `lab06.jl` and insert `using LinearAlgebra` at the top. Then copy the matrix with your mouse and paste it below. The beginning of your program should now look similar to

```
1 using LinearAlgebra
2
3  A=[  8.52 -1.80  1.08 -3.27 -3.25;
4      -1.80  1.76 -6.67 -2.67  4.05;
5       1.08 -6.67  1.16 -8.34  2.78;
6      -3.27 -2.67 -8.34  7.18  6.27;
7      -3.25  4.05  2.78  6.27 -3.28 ]
```

After adding the quadratic solver developed in the previous section write a new function `getpq` that find p and q such that

$$|A_{pq}| = \max\{\, |A_{ij}| : i \neq j \,\}.$$

One way to find p and q is with the built-in `maximum` operator followed by `findfirst` applied over the Cartesian indices such that $i \neq j$. I found the functional approach too confusing and decided to use a nested loop instead.

```
23 function getpq(A)
24     N,_=size(A)
25     m=A[1,2]; p=1; q=2
26     for i=1:N-1
27         for j=i+1:N
28             if abs(A[i,j])>m
29                 m=abs(A[i,j]); p=i; q=j
30             end
31         end
32     end
33     return p,q
34 end
```

One advantage of the Julia just-in-time compiler over interpreted languages is that loops such as written above perform well.

What remains is to write a `jacobi` function that computes the elements b_{ij} of $R^{(pq)}(\varphi_k)^T A^{(k)} R^{(pq)}(\varphi_k)$. As computed for the case $n = 2$ we have

$$\begin{aligned} b_{pp} &= a_{pp}c^2 - 2a_{pq}sc + a_{qq}s^2 \\ b_{qq} &= a_{pp}s^2 + 2a_{pq}sc + a_{qq}c^2 \\ b_{pq} &= 0 \\ b_{qp} &= 0. \end{aligned}$$

A similar computation—not shown—provides the off diagonal entries along the p and q columns and rows as

$$\begin{aligned} b_{pi} &= b_{ip} = a_{ip}c - a_{iq}s \\ b_{qi} &= b_{iq} = a_{iq}s + a_{iq}c \end{aligned} \qquad \text{for} \qquad i \neq p \text{ and } j \neq q.$$

Since the rest of the matrix R is given by the identity, the rest of the elements are unchanged. Here is a possible implementation.

```
36 function jacobi(A)
37     p,q=getpq(A)
38     t1,t2=quadsolv(-A[p,q],A[p,p]-A[q,q],A[p,q])
39     t=t1
40     if abs(t2)<abs(t1)
41         t=t2
42     end
43     c=1/sqrt(1+t^2)
44     s=c*t
45     B=copy(A)
46     B[p,p]=A[p,p]*c^2-2*A[p,q]*s*c+A[q,q]*s^2
47     B[q,q]=A[p,p]*s^2+2*A[p,q]*s*c+A[q,q]*c^2
48     B[p,q]=0
49     B[q,p]=0
50     N,_=size(A)
51     for i=1:N
52         if i==p||i==q
53             continue
54         end
55         B[i,p]=A[i,p]*c-A[i,q]*s
56         B[p,i]=B[i,p]
57         B[i,q]=A[i,p]*s+A[i,q]*c
58         B[q,i]=B[i,q]
59     end
60     return B
61 end
```

Note that line 37 calls `getpq` to determine which off-diagonal element of A has the largest absolute magnitude so it can be rotated to zero. Lines 39 through 42 choose the smallest t from among the two roots which solve the quadratic equation. Finally, lines 46 through 59 compute $B = R^{(pq)}(\varphi_k)^T A^{(k)} R^{(pq)}(\varphi_k)$ using the relations just discussed.

To complete this assignment write a loop that performs 30 iterations using a statement like

```
Ak=jacobi(Ak)
```

inside a loop. Then print out the approximately-diagonal matrix $A^{(30)}$. The output should like like

```
A(30)=
5×5 Matrix{Float64}:
  7.12263       7.22391e-23   -1.809e-16     0.0          -1.56938e-33
  7.22391e-23   5.98028        0.0          -4.57679e-17  -8.3952e-13
 -1.809e-16     0.0          -14.0027       -4.90817e-28   2.59551e-14
  0.0          -4.57679e-17   -4.90817e-28  16.6483        1.04593e-21
 -1.56938e-33  -8.3952e-13     2.59551e-14   1.04593e-21  -0.408503
```

For extra credit extract the eigenvalue approximations from the diagonal and sort them with `sort(diag(Ak))`. Then compare your approximation to the output of `eigvals(A)` which is part of the `LinearAlgebra` library. Determine which approximation is more accurate by plugging the eigenvalues into the characteristic polynomial.

Submitting Your Work

One PDF file should be submitted for grading that contains two parts:

- A program that performs 30 iterations of the Jacobi algorithm for finding eigenvalues and produces output similar to above for your individualized matrix.
- The output from running that program.

After debugging and making sure your program runs correctly, please prepare your submission by typing

```
$ julia lab06.jl >lab06.out
$ j2pdf -o lab06.pdf lab06.jl lab06.out
```

Before uploading, check `lab06.pdf` with

```
$ evince lab06.pdf &
```

to make sure the program and output looks correct. Please reboot into Microsoft Windows before leaving the lab.