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 Francis

This lab is about using an iterative technique involving QR factorization 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 of A are real and the corresponding eigenvectors may be chosen to form an orthonormal basis of $\mathbf{R}^n$. In this lab we focus on only the eigenvalues.

Recall the Hessenberg factorization of a matrix is given by

$$A = QHQ^T$$

where Q is orthogonal and H is a matrix with zeros below the first subdiagonal. That is $H_{ij} = 0$ for $i + 2 \le j$. Note when A is symmetric that

$$H = Q^T AQ = Q^T A^T Q = (Q^T AQ)^T = H^T.$$

Consequently $A = A^T$ implies H is tridiagonal.

This may be illustrated in Julia as follows:

```
julia> using LinearAlgebra

julia> A=Symmetric(rand(4,4))
4×4 Symmetric{Float64, Matrix{Float64}}:
 0.623568  0.229771   0.64867    0.769046
 0.229771  0.339733   0.0348077  0.362375
 0.64867   0.0348077  0.428924   0.540155
 0.769046  0.362375   0.540155   0.0574243

julia> H=hessenberg(A).H
4×4 SymTridiagonal{Float64, Vector{Float64}}:
 -0.118629   0.140387    ⋅           ⋅
  0.140387   0.309128  -0.149319    ⋅
   ⋅        -0.149319   1.20173   -1.00723
   ⋅          ⋅        -1.00723    0.0574243
```

The eigenvalues of A are the same as H since they are related through a similarity transform. At the same time, given the simpler structure of H, finding the eigenvalues of H is easier. Thus, we shall apply the QR iteration to H rather than A. This can be done as follows:

```
julia> Hk=Matrix(H)
       for k=1:100
           Q,R=qr(Hk)
           global Hk=R*Q
       end

julia> Hk
4×4 Matrix{Float64}:
 1.79924      9.30456e-17   3.27303e-17  -9.08421e-17
 2.9531e-52  -0.535952      5.23572e-16  -3.58532e-17
 0.0         -5.50255e-19   0.346403      2.23225e-16
 0.0          0.0          -2.83161e-34  -0.160041
```

After 100 iterations the off-diagonal terms of `Hk` are approximately zero and an approximation of the eigenvalues appear on the diagonal. These eigenvalue approximations may be compared to the ones for A found using the built-in eigenvalue solver in Julia.

```
julia> sort(diag(Hk))
4-element Vector{Float64}:
 -0.5359520089927022
 -0.16004088758880344
  0.3464027616902269
  1.799240001852394

julia> sort(eigvals(A))
4-element Vector{Float64}:
 -0.5359520089927006
 -0.16004088758880344
  0.3464027616902272
  1.7992400018523949
```

The slight differences result because the built-in function for Julia actually uses shifted QR iterations behind the scenes which converge faster than the basic algorithm we used. While taking more iterations would reduce some of the differences, using the improved algorithm with shifts would result in more accurate results with fewer iterations.

In this lab you will find the eigenvalues of an individualized symmetric matrix using 200 iterations of the QR algorithm and compare your results to the eigenvalues obtained using the built-in Julia function. 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.

This 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 `lab07.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

```julia
using LinearAlgebra

 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 ]
```

The rest of your program should be based on the example presented earlier in this section. Please see below for more details.

In general it is a good idea for a numerical program to display the problem as well the answer in the output. To this end add a couple lines to display the matrix to your code as well as some lines to display H, the diagonal of H_k for $k = 200$ as well as the eigenvalues for A found using the built-in `eigvals` function. The output of your program should look like

---

```
The matrix A is
5×5 Matrix{Float64}:
  8.52  -1.8    1.08  -3.27  -3.25
 -1.8    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

A=QHQ' where H is given by
5×5 SymTridiagonal{Float64, Vector{Float64}}:
  6.25666   -0.507373    ⋅           ⋅            ⋅
 -0.507373   6.8288    -2.41658     ⋅            ⋅
   ⋅        -2.41658    5.60627  -11.656         ⋅
   ⋅          ⋅        -11.656    -0.0717242  -8.60269
   ⋅          ⋅           ⋅       -8.60269    -3.28
The diagonal entries of H_200 are
5-element Vector{Float64}:
 -14.002678215914495
  -0.40850297049361384
   5.980284643865611
   7.122625449410926
  16.648271093131523

The eigenvalues of A are
5-element Vector{Float64}:
 -14.002678215914505
  -0.40850297049361134
   5.9802846438656205
   7.122625449410943
  16.648271093131555
```

Submitting Your Work

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

- A program that performs 200 iterations of the QR algorithm 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 lab07.jl >lab07.out
$ j2pdf -o lab07.pdf lab07.jl lab07.out
```

Before uploading, check `lab07.pdf` with

```
$ evince lab07.pdf &
```

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