Mario L. Gutierrez Abed
364009832
mlg3843@rit.edu

Final Problem Set
Numerical Analysis I

06-10-2021

Problem 1. *Put the matrix*

$$A = \begin{bmatrix} 1 & 0 & 2 & 3 \\ -1 & 0 & 5 & 2 \\ 2 & -2 & 0 & 0 \\ 2 & -1 & 2 & 0 \end{bmatrix} \tag{1}$$

in upper-Hessenberg form.

Solution. A square matrix in ***Hessenberg*** form is a matrix that is "almost diagonal;" i.e., it is a matrix in the form:

$$\underbrace{\begin{bmatrix} \times & \times & & 0 \\ \vdots & & \ddots & \\ \times & & & \times \\ \times & \times & \cdots & \times \end{bmatrix}}_{\text{Lower-Hessenberg}} \quad \text{or} \quad \underbrace{\begin{bmatrix} \times & \cdots & \times & \times \\ \times & \cdots & \times & \times \\ & \ddots & \vdots & \vdots \\ 0 & & \times & \times \end{bmatrix}}_{\text{Upper-Hessenberg}}.$$

In other words, a square matrix A is ***lower-Hessenberg*** if $a_{ij} = 0$ for $j > i+1$, or it is ***upper-Hessenberg*** if $a_{ij} = 0$ for $i > j+1$.

In practice, given a matrix A for which we want to find its *Schur decomposition* $A = QTQ^*$ (where T is a triangular matrix and Q is unitary), it is computationally convenient to proceed in two phases: the first phase is to reduce A to Hessenberg form, and the second phase is to continue to iterate to generate a sequence of Hessenberg matrices that converge to a triangular form. The procedure follows as this:

$$\underset{A \neq A^*}{\begin{bmatrix} \times&\times&\times&\times&\times \\ \times&\times&\times&\times&\times \\ \times&\times&\times&\times&\times \\ \times&\times&\times&\times&\times \\ \times&\times&\times&\times&\times \end{bmatrix}} \xrightarrow{\text{Phase 1}} \underset{H}{\begin{bmatrix} \times&\times&\times&\times&\times \\ \times&\times&\times&\times&\times \\ &\times&\times&\times&\times \\ &&\times&\times&\times \\ &&&\times&\times \end{bmatrix}} \xrightarrow{\text{Phase 2}} \underset{T}{\begin{bmatrix} \times&\times&\times&\times&\times \\ &\times&\times&\times&\times \\ &&\times&\times&\times \\ &&&\times&\times \\ &&&&\times \end{bmatrix}}$$

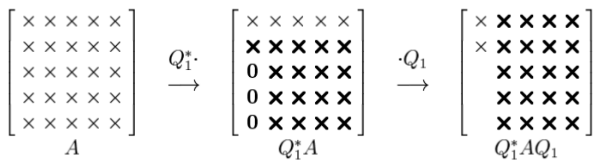
(Note that if $A = A^*$ above (i.e., A is Hermitian), then the resulting matrix from phase 1 is tridiagonal and the resulting matrix from phase 2 is diagonal.) Our focus for this problem is on the first phase; i.e. we want to reduce A to (upper) Hessenberg form. Naïvely, we may think of using Householder reflectors to play the role of the unitary matrices Q_i's; after all we have used them before and we know that they are very efficient at introducing zeroes below the diagonal, as in

$$\underset{A}{\begin{bmatrix} \times&\times&\times&\times&\times \\ \times&\times&\times&\times&\times \\ \times&\times&\times&\times&\times \\ \times&\times&\times&\times&\times \\ \times&\times&\times&\times&\times \end{bmatrix}} \xrightarrow{Q_1^* \cdot} \underset{Q_1^* A}{\begin{bmatrix} \boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times} \\ \mathbf{0}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times} \\ \mathbf{0}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times} \\ \mathbf{0}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times} \\ \mathbf{0}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times} \end{bmatrix}}$$

This, however, is a terrible idea in this case, since we now have to multiply also on the right of A and this results in all the work from the previous step being undone:

$$\underset{Q_1^* A}{\begin{bmatrix} \times&\times&\times&\times&\times \\ &\times&\times&\times&\times \\ &\times&\times&\times&\times \\ &\times&\times&\times&\times \\ &\times&\times&\times&\times \end{bmatrix}} \xrightarrow{\cdot Q_1} \underset{Q_1^* A Q_1}{\begin{bmatrix} \boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times} \\ \boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times} \\ \boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times} \\ \boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times} \\ \boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times}&\boldsymbol{\times} \end{bmatrix}}$$

What we must do instead is to select, on the first step, a Householder reflector that leaves the first row unchanged and introduces zeroes below the second row on the first column, so that when we multiply on the right the zeroes formed on the previous step remain intact:



The process is then repeated to introduce zeroes below the subdiagonal in the remaining columns… Without further ado, let us apply this technique to our matrix A from Eq. (1):

Take the first column of A, except its first entry,

$$\hat{\mathbf{a}}_{(1)} = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}.$$

Then the corresponding Householder vector is given

$$\begin{aligned}
\hat{\mathbf{u}} &= \hat{\mathbf{a}}_{(1)} - \operatorname{sgn}(\hat{a}_1)\|\hat{\mathbf{a}}_{(1)}\|_2\hat{\mathbf{e}}_{(1)} \\
&= \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix} - \operatorname{sgn}(-1)\cdot\left(\sqrt{|-1|^2+|2|^2+|2|^2}\right)\cdot\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix} + \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}.
\end{aligned}$$

This Householder vector $\hat{\mathbf{u}}$ yields the Householder matrix

$$\begin{aligned}
\hat{H}_1 &= I - 2\frac{\hat{\mathbf{u}}\hat{\mathbf{u}}^\top}{\hat{\mathbf{u}}^\top\hat{\mathbf{u}}} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - 2\left(\begin{bmatrix} 2 & 2 & 2 \end{bmatrix}\begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}\right)^{-1}\begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}\begin{bmatrix} 2 & 2 & 2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{6}\begin{bmatrix} 4 & 4 & 4 \\ 4 & 4 & 4 \\ 4 & 4 & 4 \end{bmatrix} \\
&= \frac{1}{3}\begin{bmatrix} 1 & -2 & -2 \\ -2 & 1 & -2 \\ -2 & -2 & 1 \end{bmatrix}.
\end{aligned}$$

Then, letting

$$H_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & & & \\ 0 & & \hat{H}_1 & \\ 0 & & & \end{bmatrix},$$

we have

$$H_1A = \begin{bmatrix} 1 & 0 & 2 & 3 \\ -3 & 2 & 1/3 & 2/3 \\ 0 & 0 & -14/3 & -4/3 \\ 0 & 1 & -8/3 & -4/3 \end{bmatrix},$$

and, multiplying on the right,

$$H_1AH_1 = \begin{bmatrix} 1 & -10/3 & -4/3 & -1/3 \\ -3 & 0 & -5/3 & -4/3 \\ 0 & 4 & -2/3 & 8/3 \\ 0 & 3 & -2/3 & 2/3 \end{bmatrix}.$$

Note how multiplying on the right now did not affect the zeroes previously obtained on the first column. We haven't finished yet though; next we take the (truncated) second column of H_1AH_1 (denote it by $\hat{\mathbf{a}}_{(2)}$):

$$\hat{\mathbf{a}}_{(2)} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}.$$

Then the corresponding Householder vector is given

$$\begin{aligned}\hat{\mathbf{u}} &= \hat{\mathbf{a}}_{(2)} - \operatorname{sgn}(\hat{a}_2)\|\hat{\mathbf{a}}_{(2)}\|_2\hat{\mathbf{e}}_{(1)} \\ &= \begin{bmatrix}4\\3\end{bmatrix} - \operatorname{sgn}(4)\cdot\left(\sqrt{|4|^2+|3|^2}\right)\cdot\begin{bmatrix}1\\0\end{bmatrix} \\ &= \begin{bmatrix}4\\3\end{bmatrix} - \begin{bmatrix}5\\0\end{bmatrix} \\ &= \begin{bmatrix}-1\\3\end{bmatrix}.\end{aligned}$$

This Householder vector $\hat{\mathbf{u}}$ yields the Householder matrix

$$\begin{aligned}\hat{H}_2 &= I - 2\frac{\hat{\mathbf{u}}\hat{\mathbf{u}}^\top}{\hat{\mathbf{u}}^\top\hat{\mathbf{u}}} \\ &= \begin{bmatrix}1&0\\0&1\end{bmatrix} - 2\left(\begin{bmatrix}-1&3\end{bmatrix}\begin{bmatrix}-1\\3\end{bmatrix}\right)^{-1}\begin{bmatrix}-1\\3\end{bmatrix}\begin{bmatrix}-1&3\end{bmatrix} \\ &= \begin{bmatrix}1&0\\0&1\end{bmatrix} - \frac{1}{5}\begin{bmatrix}1&-3\\-3&9\end{bmatrix} \\ &= \frac{1}{5}\begin{bmatrix}4&3\\3&-4\end{bmatrix}.\end{aligned}$$

Then, letting

$$H_2 = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&&\\0&0&&\hat{H}_2\end{bmatrix},$$

we have

$$H_2(H_1AH_1) = \begin{bmatrix}1&-10/3&-4/3&-1/3\\-3&0&-5/3&-4/3\\0&5&-14/15&38/15\\0&0&2/15&16/15\end{bmatrix},$$

and, multiplying on the right,

$$H_2(H_1AH_1)H_2 = \begin{bmatrix}1&-10/3&-19/15&-8/15\\-3&0&-32/15&1/15\\0&5&58/75&-194/75\\0&0&56/75&-58/75\end{bmatrix}.$$

The resulting matrix is in upper-Hessenberg form, as desired. ♠

Problem 2. *Use the Power Iteration Method to find the dominant eigenvector of*

$$A = \begin{bmatrix}10&-12&-6\\5&-5&-4\\-1&0&3\end{bmatrix}, \tag{2}$$

and the dominant eigenvalue by calculating a Rayleight quotient.

Solution. The idea behind the ***Power Iteration Method*** is that the sequence

$$\frac{\mathbf{x}}{\|\mathbf{x}\|}, \frac{A\mathbf{x}}{\|A\mathbf{x}\|}, \frac{A^2\mathbf{x}}{\|A^2\mathbf{x}\|}, \frac{A^3\mathbf{x}}{\|A^3\mathbf{x}\|}, \dots$$

converges, under certain assumptions, to the *dominant eigenvector* of A, i.e., to the eigenvector that corresponds to the eigenvalue of A whose absolute value is the largest (such eigenvalue is known as the *dominant eigenvalue*; if it is postive it coincides with the *spectral radius* of A).

Let us apply the Power Iteration Method to A to find its dominant eigenvector; starting with initial guess $\mathbf{x}^{(0)} = [1\ 1\ 1]^\top$,[1]

$$A\mathbf{x}^{(0)} = \begin{bmatrix} 10 & -12 & -6 \\ 5 & -5 & -4 \\ -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -8 \\ -4 \\ 2 \end{bmatrix}; \qquad \mathbf{x}^{(1)} = \frac{A\mathbf{x}^{(0)}}{\|A\mathbf{x}^{(0)}\|} = \begin{bmatrix} -1 \\ -0.5 \\ 0.25 \end{bmatrix}.$$

$$A\mathbf{x}^{(1)} = \begin{bmatrix} 10 & -12 & -6 \\ 5 & -5 & -4 \\ -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1 \\ -0.5 \\ 0.25 \end{bmatrix} = \begin{bmatrix} -5.5 \\ -3.5 \\ 1.75 \end{bmatrix}; \qquad \mathbf{x}^{(2)} = \frac{A\mathbf{x}^{(1)}}{\|A\mathbf{x}^{(1)}\|} = \begin{bmatrix} -1 \\ -0.636364 \\ 0.318182 \end{bmatrix}.$$

$$A\mathbf{x}^{(2)} = \begin{bmatrix} 10 & -12 & -6 \\ 5 & -5 & -4 \\ -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1 \\ -0.636364 \\ 0.318182 \end{bmatrix} = \begin{bmatrix} -4.27273 \\ -3.09091 \\ 1.95455 \end{bmatrix}; \qquad \mathbf{x}^{(3)} = \frac{A\mathbf{x}^{(2)}}{\|A\mathbf{x}^{(2)}\|} = \begin{bmatrix} -1 \\ -0.723404 \\ 0.457447 \end{bmatrix}.$$

$$A\mathbf{x}^{(3)} = \begin{bmatrix} 10 & -12 & -6 \\ 5 & -5 & -4 \\ -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1 \\ -0.723404 \\ 0.457447 \end{bmatrix} = \begin{bmatrix} -4.06383 \\ -3.21277 \\ 2.37234 \end{bmatrix}; \qquad \mathbf{x}^{(4)} = \frac{A\mathbf{x}^{(3)}}{\|A\mathbf{x}^{(3)}\|} = \begin{bmatrix} -1 \\ -0.790576 \\ 0.58377 \end{bmatrix}.$$

$$A\mathbf{x}^{(4)} = \begin{bmatrix} 10 & -12 & -6 \\ 5 & -5 & -4 \\ -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1 \\ -0.790576 \\ 0.58377 \end{bmatrix} = \begin{bmatrix} -4.01571 \\ -3.3822 \\ 2.75131 \end{bmatrix}; \qquad \mathbf{x}^{(5)} = \frac{A\mathbf{x}^{(4)}}{\|A\mathbf{x}^{(4)}\|} = \begin{bmatrix} -1 \\ -0.842243 \\ 0.685137 \end{bmatrix}.$$

$$A\mathbf{x}^{(5)} = \begin{bmatrix} 10 & -12 & -6 \\ 5 & -5 & -4 \\ -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1 \\ -0.842243 \\ 0.685137 \end{bmatrix} = \begin{bmatrix} -4.00391 \\ -3.52934 \\ 3.05541 \end{bmatrix}; \qquad \mathbf{x}^{(6)} = \frac{A\mathbf{x}^{(5)}}{\|A\mathbf{x}^{(5)}\|} = \begin{bmatrix} -1 \\ -0.881472 \\ 0.763106 \end{bmatrix}.$$

$$A\mathbf{x}^{(6)} = \begin{bmatrix} 10 & -12 & -6 \\ 5 & -5 & -4 \\ -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1 \\ -0.881472 \\ 0.763106 \end{bmatrix} = \begin{bmatrix} -4.00098 \\ -3.64507 \\ 3.28932 \end{bmatrix}; \qquad \mathbf{x}^{(7)} = \frac{A\mathbf{x}^{(6)}}{\|A\mathbf{x}^{(6)}\|} = \begin{bmatrix} -1 \\ -0.911044 \\ 0.822129 \end{bmatrix}.$$

$$A\mathbf{x}^{(7)} = \begin{bmatrix} 10 & -12 & -6 \\ 5 & -5 & -4 \\ -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1 \\ -0.911044 \\ 0.822129 \end{bmatrix} = \begin{bmatrix} -4.00024 \\ -3.7333 \\ 3.46639 \end{bmatrix}; \qquad \mathbf{x}^{(8)} = \frac{A\mathbf{x}^{(7)}}{\|A\mathbf{x}^{(7)}\|} = \begin{bmatrix} -1 \\ -0.933267 \\ 0.866544 \end{bmatrix}.$$

$$A\mathbf{x}^{(8)} = \begin{bmatrix} 10 & -12 & -6 \\ 5 & -5 & -4 \\ -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1 \\ -0.933267 \\ 0.866544 \end{bmatrix} = \begin{bmatrix} -4.00006 \\ -3.79984 \\ 3.59963 \end{bmatrix}; \qquad \mathbf{x}^{(9)} = \frac{A\mathbf{x}^{(8)}}{\|A\mathbf{x}^{(8)}\|} = \begin{bmatrix} -1 \\ -0.949946 \\ 0.899894 \end{bmatrix} \ldots$$

The convergence may be slow but the pattern is clear: the Power Method is converging to the dominant eigenvector $[-1\ \ -1\ \ 1]^\top$. Once we have an eigenvector $\mathbf{x}$ we can get its corresponding eigenvalue by using the ***Rayleight quotient***

$$\boxed{\lambda = \frac{\mathbf{x}^\top A\mathbf{x}}{\mathbf{x}^\top \mathbf{x}}.} \tag{3}$$

Hence, applying this expression to the (dominant) eigenvector we got from using the Power Method, we get the corresponding (dominant) eigenvalue:

$$\begin{aligned} \lambda_{\max} &= \frac{\mathbf{x}^\top A\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} \\ &= \left(\begin{bmatrix} -1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 10 & -12 & -6 \\ 5 & -5 & -4 \\ -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \\ &= \frac{1}{3} \cdot 12 = 4. \end{aligned}$$

♠

[1]We are using the ∞-norm throughout; i.e., $\|\cdot\| = \|\cdot\|_\infty$.

Problem 3. *Apply Rayleigh Quotient Iteration (RQI) to the matrix (2). Try different starting vectors until all three eigenvalues are found.*

Solution. The pseudocode for RQI is as follows:

```
Start with some initial eigenvector guess x^(0), and normalize x^(0)/|| x^(0)|| ;
Compute corresponding Rayleigh Quotient:
    lambda^(0) =   (x^(0))^T A x^(0);

For k = 1, 2, ...
    Solve (A - lambda^(k-1)I) x(k) = x^(k-1) for x(k);
    Normalize x(k) = x(k)/||x(k)||;
    Apply Rayleigh Quotient using newly updated x^(k);
```

In order to avoid more tedious algebra (as in the previous problem!) we shall automate the algorithm by turning the above pseudocode into Matlab code:

```
function [lam,u] = RQI(A,x,k)
%Inputs:
%A = given matrix
%x = initial guess x^(0)
%k = number of steps

    for j = 1:k
      u = x/norm(x);                      %normalize
      lam = u'* A * u;                    %Rayleigh Quotient
      x = (A - lam * eye(size(A))) \ u;   %Inverse power iteration
    end

    u = x/norm(x);
    lam = u' * A * u;                     %Rayleigh quotient

end
```

We already know from the previous problem that one eigenvector (the dominant one) is $\mathbf{x}_1 = [-1 \;\; -1 \;\; 1]^\top$ (or, normalized, $\mathbf{x}_1 = [-0.5774 \;\; -0.5774 \;\; 0.5774]^\top$), and its corresponding eigenvalue $\lambda_1 = \lambda_{\max} = 4$ was also computed. We can now find the remaining two eigenpairs using the above Matlab routine. Taking as initial guesses

$$\mathbf{x}_2^{(0)} = \begin{bmatrix} 0.5 \\ -0.5 \\ 3 \end{bmatrix} \qquad \text{and} \qquad \mathbf{x}_3^{(0)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$

from `[lam,u] = RQI(A,x_2,6)` and `[lam,u] = RQI(A,x_3,5)`, we get the eigenpairs

$$\lambda_2 = 3 \qquad \text{and} \qquad \mathbf{x}_2 = \begin{bmatrix} 0 \\ 0.4472 \\ -0.8944 \end{bmatrix};$$

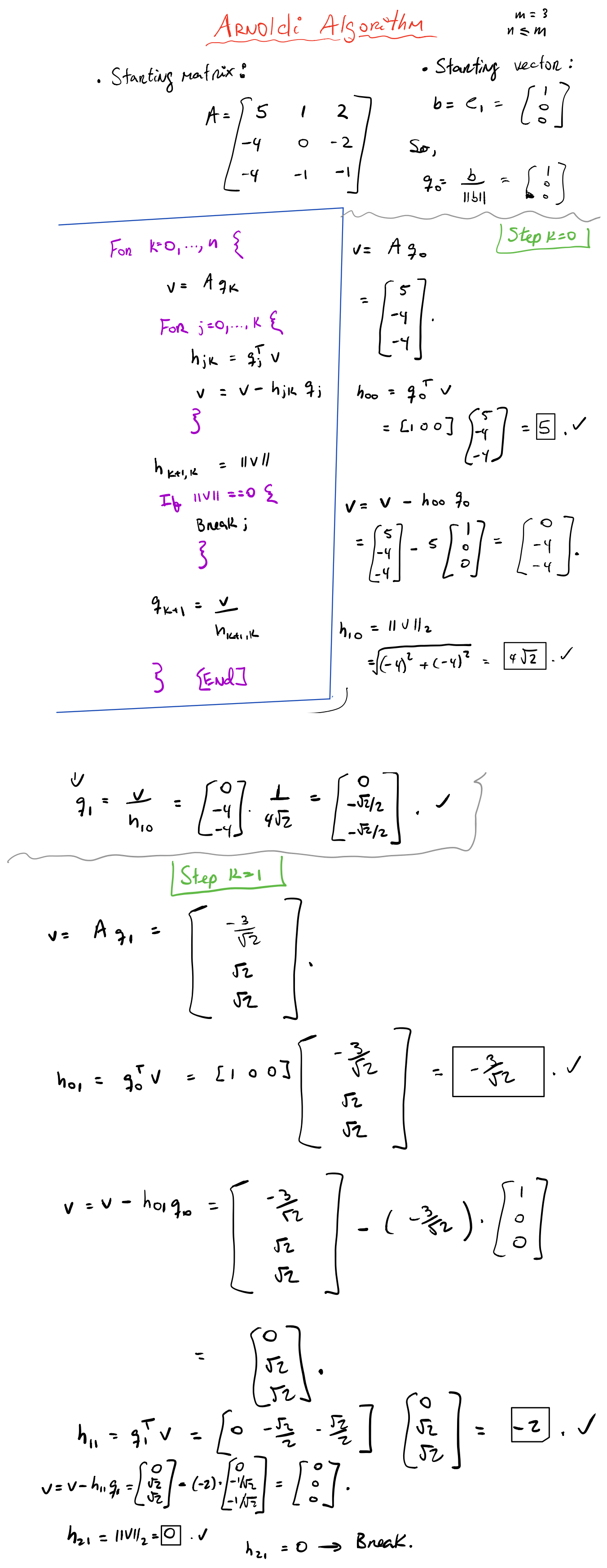
$$\lambda_3 = 1 \qquad \text{and} \qquad \mathbf{x}_3 = \begin{bmatrix} 0.8165 \\ 0.4082 \\ 0.4082 \end{bmatrix}. \qquad \spadesuit$$

Problem 4. *Use Arnoldi's algorithm to find an orthogonal matrix Q such that $Q^\top AQ = H$ is an upper-Hessenberg matrix, where*

$$A = \begin{bmatrix} 5 & 1 & 2 \\ -4 & 0 & -2 \\ -4 & -1 & -1 \end{bmatrix}. \tag{4}$$

(Related: See also Problem 8.)

Solution. The Arnoldi algorithm is applied as follows (for the sake of saving some time, I'm just pasting here what I wrote instead of typing the whole thing):



Arnoldi Algorithm

$m = 3$
$n \leq m$

• Starting matrix:

$$A = \begin{bmatrix} 5 & 1 & 2 \\ -4 & 0 & -2 \\ -4 & -1 & -1 \end{bmatrix}$$

• Starting vector:

$$b = e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

So,

$$q_0 = \frac{b}{\|b\|} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

```
For $k=0,\dots,n$ {
    $v = A q_k$
    For $j=0,\dots,k$ {
        $h_{jk} = q_j^T v$
        $v = v - h_{jk} q_j$
    }
    $h_{k+1,k} = \|v\|$
    If $\|v\| == 0$ {
        Break;
    }
    $q_{k+1} = \frac{v}{h_{k+1,k}}$
} [End]
```

Step $k=0$

$$v = A q_0 = \begin{bmatrix} 5 \\ -4 \\ -4 \end{bmatrix}.$$

$$h_{00} = q_0^T v = [1\ 0\ 0] \begin{bmatrix} 5 \\ -4 \\ -4 \end{bmatrix} = \boxed{5}. \checkmark$$

$$v = v - h_{00} q_0 = \begin{bmatrix} 5 \\ -4 \\ -4 \end{bmatrix} - 5 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -4 \\ -4 \end{bmatrix}.$$

$$h_{10} = \|v\|_2 = \sqrt{(-4)^2 + (-4)^2} = \boxed{4\sqrt{2}}. \checkmark$$

$$q_1 = \frac{v}{h_{10}} = \begin{bmatrix} 0 \\ -4 \\ -4 \end{bmatrix} \cdot \frac{1}{4\sqrt{2}} = \begin{bmatrix} 0 \\ -\sqrt{2}/2 \\ -\sqrt{2}/2 \end{bmatrix}. \checkmark$$

Step $k=1$

$$v = A q_1 = \begin{bmatrix} -\frac{3}{\sqrt{2}} \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix}.$$

$$h_{01} = q_0^T v = [1\ 0\ 0] \begin{bmatrix} -\frac{3}{\sqrt{2}} \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix} = \boxed{-\frac{3}{\sqrt{2}}}. \checkmark$$

$$v = v - h_{01} q_0 = \begin{bmatrix} -\frac{3}{\sqrt{2}} \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix} - \left(-\frac{3}{\sqrt{2}}\right) \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix}.$$

$$h_{11} = q_1^T v = \left[0\ \ -\frac{\sqrt{2}}{2}\ \ -\frac{\sqrt{2}}{2}\right] \begin{bmatrix} 0 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix} = \boxed{-2}. \checkmark$$

$$v = v - h_{11} q_1 = \begin{bmatrix} 0 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix} - (-2) \cdot \begin{bmatrix} 0 \\ -1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

$h_{21} = \|v\|_2 = \boxed{0}. \checkmark$ $\quad h_{21} = 0 \rightarrow$ Break.

At this point the algorithm breaks because we found a zero entry in the H matrix. This, however, is a benign breakdown of the algorithm, since it means we can now work with a factorization in a smaller dimension than we started with, which will be less computationally expensive. Thus, instead of working with 3×3 matrices, we ended up with the factorization

$$H_k = \begin{bmatrix} 5 & -3/\sqrt{2} \\ 4\sqrt{2} & -2 \end{bmatrix}; \qquad Q_k = \begin{bmatrix} 1 & 0 \\ 0 & -\sqrt{2}/2 \\ 0 & -\sqrt{2}/2 \end{bmatrix}. \tag{5}$$

Indeed, note that $H_k = Q_k^\top A Q_k$, as desired.

The power of the Arnoldi algorithm (and Krylov methods in general) lies in the ability to reduce systems of dimension m to factorizations of dimension n, with $n \ll m$. We shall see this in Problem 8, in which we are given an $m = 100$ system. For the sake of this exercise, however, since we have a much smaller system, we could continue to produce a full factorization by introducing some random vector that is orthogonal/orthonormal to both $\mathbf{q}_0$ and $\mathbf{q}_1$. For instance, we may take

$$\mathbf{q}_2 = \begin{bmatrix} 0 \\ \sqrt{2}/2 \\ -\sqrt{2}/2 \end{bmatrix}.$$

Then we have the full Hessenberg factorization

$$\begin{aligned} H = Q^\top A Q &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\sqrt{2}/2 & -\sqrt{2}/2 \\ 0 & \sqrt{2}/2 & -\sqrt{2}/2 \end{bmatrix} \begin{bmatrix} 5 & 1 & 2 \\ -4 & 0 & -2 \\ -4 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\sqrt{2}/2 & \sqrt{2}/2 \\ 0 & -\sqrt{2}/2 & -\sqrt{2}/2 \end{bmatrix} \\ &= \begin{bmatrix} 5 & -3/\sqrt{2} & -\sqrt{2}/2 \\ 4\sqrt{2} & -2 & -1 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

Problem 5. *Let A be the matrix (4), and $\mathbf{b} = [1\ 2\ 1]^\top$. Then,*

a) Use Galerkin method to solve $A\mathbf{x} = \mathbf{b}$.

b) Use GMRES method to solve $A\mathbf{x} = \mathbf{b}$.

(Related: See also Problem 8.)

Solution to a). The Galerkin method requires an initial guess, which is typically taken to be $\mathbf{x}_0 = \mathbf{b}$. From this we get the initial residual $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, which is then used as the initial vector $\mathbf{q}_0 = \mathbf{r}_0/\|\mathbf{r}_0\|$ in the Arnoldi algorithm. However, we already factorized the matrix A in the previous problem via the Arnoldi scheme, so using a new initial vector and applying Arnoldi all over again would be a colossal waste of time. Let us instead use the initial $\mathbf{x}_0$ that corresponds to the $\mathbf{q}_0$ we used in the previous exercise. Thus we use the $\mathbf{x}_0$ that satisfies

$$A\mathbf{x}_0 = \mathbf{b} - \mathbf{r}_0 = \mathbf{b} - \frac{\mathbf{q}_0}{\|\mathbf{q}_0\|} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} \qquad \Longrightarrow \qquad \mathbf{x}_0 = \begin{bmatrix} -2 \\ 4 \\ 3 \end{bmatrix}. \tag{6}$$

- The next step is to solve the system $H\mathbf{y} = \|r_0\|\mathbf{e}_0$ for $\mathbf{y}$:

$$\begin{bmatrix} 5 & -3/\sqrt{2} & -\sqrt{2}/2 \\ 4\sqrt{2} & -2 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = 1 \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \Longrightarrow \qquad \mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} -1 \\ -2\sqrt{2} \\ 0 \end{bmatrix}.$$

- Now compute the correction vector $\mathbf{z}$ from

$$\mathbf{z} = Q\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\sqrt{2}/2 & \sqrt{2}/2 \\ 0 & -\sqrt{2}/2 & -\sqrt{2}/2 \end{bmatrix} \begin{bmatrix} -1 \\ -2\sqrt{2} \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}.$$

- Compute the new solution vector $\mathbf{x}$ from

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{z} = \begin{bmatrix} -2 \\ 4 \\ 3 \end{bmatrix} + \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 6 \\ 5 \end{bmatrix}.$$

- Compute the new residual $\mathbf{r}$ from $A\mathbf{x} - b$, using the newly obtained $\mathbf{x}$:

$$\mathbf{r} = \begin{bmatrix} 5 & 1 & 2 \\ -4 & 0 & -2 \\ -4 & -1 & -1 \end{bmatrix} \begin{bmatrix} -3 \\ 6 \\ 5 \end{bmatrix} - \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

  If $\|\mathbf{r}\| \leq \epsilon$, where ϵ is some user-defined tolerance, then we accept $\mathbf{x}$ as the final solution to the system and stop the algorithm; otherwise we use the newly obtained residual to restart from the very beginning, setting $\mathbf{r}_0 \equiv \mathbf{r}$. In this particular case we have found exact convergence, since the residual is zero. Thus our final solution to the system $A\mathbf{x} = b$ is

$$\mathbf{x} = \begin{bmatrix} -3 \\ 6 \\ 5 \end{bmatrix}.$$

It is important to stress that we did not have to use the full Hessenberg factorization to solve this system. Using H_k instead of H would have also yielded the same exact result. I used the full factorization on this problem because $m = 3$ is quite managable. On Problem 8 when we deal with a much larger system, we will be using the partial factorization to solve the system. ♠

Solution to b). The show the validity of the remark made at the end of part a), we shall now use the partial Hessenberg factorization ($n < m$); we will indeed obtain the same result. Starting with the same initial approximation $\mathbf{x}_0$ from Eq. (6), we proceed as follows:

- Apply n Arnoldi steps to find Q_n and H_n (already done on Problem 4).

- Find the vector $\mathbf{y}$ that minimizes

$$J(\mathbf{y}) = \|H_n\mathbf{y} - \|r_0\|\mathbf{e}_0\|.$$

  We can solve this, e.g., using QR least squares. Doing this we find

$$\mathbf{y} = \begin{bmatrix} -1 \\ -2\sqrt{2} \end{bmatrix}.$$

- Now compute the correction vector $\mathbf{z}$ from

$$\mathbf{z} = Q\mathbf{y} = \begin{bmatrix} 1 & 0 \\ 0 & -\sqrt{2}/2 \\ 0 & -\sqrt{2}/2 \end{bmatrix} \begin{bmatrix} -1 \\ -2\sqrt{2} \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}.$$

- Compute the new solution vector $\mathbf{x}$ from

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{z} = \begin{bmatrix} -2 \\ 4 \\ 3 \end{bmatrix} + \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 6 \\ 5 \end{bmatrix}.$$

- Compute the new residual $\mathbf{r}$ from $A\mathbf{x} - b$, using the newly obtained $\mathbf{x}$:

$$\mathbf{r} = \begin{bmatrix} 5 & 1 & 2 \\ -4 & 0 & -2 \\ -4 & -1 & -1 \end{bmatrix} \begin{bmatrix} -3 \\ 6 \\ 5 \end{bmatrix} - \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

If $\|\mathbf{r}\| \leq \epsilon$, where ϵ is some user-defined tolerance, then we accept $\mathbf{x}$ as the final solution to the system and stop the algorithm; otherwise we use the newly obtained residual to restart from the very beginning, setting $\mathbf{r}_0 \equiv \mathbf{r}$. In this particular case, as in part a), we have found exact convergence, since the residual is zero. Thus our final solution to the system $A\mathbf{x} = b$ is

$$\mathbf{x} = \begin{bmatrix} -3 \\ 6 \\ 5 \end{bmatrix}. \qquad \spadesuit$$

Problem 6. *Let $A \in \mathbb{R}^{n \times n}$ and let $\mathbf{x}, \mathbf{y}$, and $\mathbf{z}$ be n-vectors such that $A\mathbf{x} = \mathbf{b}$ and $A\mathbf{y} = \mathbf{b} + \mathbf{z}$. Then show that*

$$\frac{\|\mathbf{z}\|_2}{\|A\|_2} \leq \|\mathbf{x} - \mathbf{y}\|_2 \leq \|A^{-1}\|_2 \|\mathbf{z}\|_2. \tag{7}$$

Proof. For the first inequality, note that

$$\begin{aligned} \|\mathbf{z}\|_2 &= \|-\mathbf{z}\|_2 \\ &= \|-(A\mathbf{y} - b)\|_2 \\ &= \|-(A\mathbf{y} - A\mathbf{x})\|_2 \\ &= \|A(\mathbf{x} - \mathbf{y})\|_2 \\ &\leq \|A\|_2 \, \|\mathbf{x} - \mathbf{y}\|_2, \end{aligned}$$

from which

$$\frac{\|\mathbf{z}\|_2}{\|A\|_2} \leq \|\mathbf{x} - \mathbf{y}\|_2$$

follows. Now, for the second inequality, we have

$$\begin{aligned} \|\mathbf{x} - \mathbf{y}\|_2 &= \|A^{-1}\mathbf{b} - A^{-1}(\mathbf{b} + \mathbf{z})\|_2 \\ &= \|-A^{-1}\mathbf{z}\|_2 \\ &= \|A^{-1}\mathbf{z}\|_2 \\ &\leq \|A^{-1}\|_2 \, \|\mathbf{z}\|_2. \end{aligned} \qquad \spadesuit$$

Problem 7. *For $A\mathbf{x} = \mathbf{b}$ and $(A + \delta A)(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b}$:*

a) Show that

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x} + \delta\mathbf{x}\|} \leq \operatorname{cond}(A) \frac{\|\delta A\|}{\|A\|}. \tag{8}$$

b) Verify Inequality (8) for the system

$$\begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \textit{with} \quad \delta A = \begin{bmatrix} 0 & 0 & 0.00003 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Solution to a). Expanding,

$$\begin{aligned}
\mathbf{b} &= (A+\delta A)(\mathbf{x}+\delta\mathbf{x}) \\
&= A\mathbf{x} + A\delta\mathbf{x} + \delta A\mathbf{x} + \delta A\delta\mathbf{x} \\
\implies 0 &= A\delta\mathbf{x} + \delta A\,(\mathbf{x}+\delta\mathbf{x}) \\
\implies \delta\mathbf{x} &= -A^{-1}\left[\delta A\,(\mathbf{x}+\delta\mathbf{x})\right].
\end{aligned}$$

Here from the second to third line we used $A\mathbf{x} = \mathbf{b}$. [2] Now, taking norms on this resulting expression and using the the norm's submultiplicative property,

$$\|\delta\mathbf{x}\| = \|A^{-1}\left[\delta A\,(\mathbf{x}+\delta\mathbf{x})\right]\| \le \|A^{-1}\|\,\|\delta A\|\,\|\mathbf{x}+\delta\mathbf{x}\|.$$

Finally, diving both sides by $\|\mathbf{x}+\delta\mathbf{x}\|$,

$$\begin{aligned}
\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}+\delta\mathbf{x}\|}\| &\le \|A^{-1}\|\,\|\delta A\| \times \frac{\|A\|}{\|A\|} \\
&= \mathrm{cond}(A)\frac{\|\delta A\|}{\|A\|}.
\end{aligned}$$

♠

Solution to b). The solution to the systems $A\mathbf{x} = \mathbf{b}$ and $(A+\delta A)(\mathbf{x}+\delta\mathbf{x}) = \mathbf{b}$ are

$$\mathbf{x} = \begin{bmatrix} 3 \\ -24 \\ 30 \end{bmatrix} \qquad \text{and} \qquad \mathbf{x}+\delta\mathbf{x} = \begin{bmatrix} 2.99191 \\ -23.9676 \\ 29.973 \end{bmatrix},$$

respectively. The inequality (8) must hold for any norm, so let's just pick, say, the 1-norm. Then,

$$\|\delta\mathbf{x}\| = 0.0674393, \qquad \|\mathbf{x}+\delta\mathbf{x}\| = 56.9326, \qquad \|A\| = \frac{11}{6}, \qquad \|A^{-1}\| = 408, \qquad \|\delta A\| = 0.00003.$$

Plugging these numbers into the inequality yields

$$0.00118455 \le 0.01224 \quad \surd$$

which validates (8).

♠

Problem 8. $\boxed{\text{C}}$ *Construct a sparse matrix of size 100×100 with 2 on the main diagonal and -1 on the sub- and super diagonals, and zeros everywhere else. Choose a suitable vector $\mathbf{b}$ such that the solution of the system $A\mathbf{x} = \mathbf{b}$ is $\mathbf{x} = [1,\dots,1]^\top$. Take the initial vector $\mathbf{x}^{(0)} = \mathbf{0}$ and tolerance of $\epsilon = 10^{-10}$.*

a) Solve the system by implementing GMRES and Galerkin methods with Arnoldi algorithm.

b) Plot the history of convergence by taking the residual as a function of the number of iterations.

Solution. All codes are written in C++. Here is the content of the main file:

```cpp
#include <istream>
#include <fstream>
#include <iostream>
#include <cmath>
#include <Eigen/Dense>
#include <Eigen/Sparse>

using namespace std;
using namespace Eigen;

//max number of rows/columns set to 100; avoids dynamic allocation
typedef Matrix<double, Dynamic, Dynamic, 0, 100, 100> MatrixXd100;
typedef Matrix<double, Dynamic, 1, 0, 100> VectorXd100;

```

[2] Also, note that if we were performing a linear stability analysis, the factor $\delta A\delta\mathbf{x}$ would be ignored since it is a quadratic term of very small perturbations (and therefore negligible if we are considering only a *linear* perturbation). For the purposes of this exercise, however, we are not ignoring higher-order terms.

```cpp
//Function prototypes
void  Arnoldi(const MatrixXd100 &A, const VectorXd100 &b, const size_t &n,  MatrixXd100 &Q,
              MatrixXd100 &H, const double tol = 1.0e-10);
void Galerkin(const MatrixXd100 &A, const VectorXd100 &b, VectorXd100 &x0, const size_t &n,
              MatrixXd100 &Q, MatrixXd100 &H, const double tol = 1.0e-10);
void GMRES(const MatrixXd100 &A, const VectorXd100 &b, VectorXd100 &x0, const size_t &n,
           MatrixXd100 &Q, MatrixXd100 &H, const double tol = 1.0e-10);
void Gaussian(MatrixXd100 &A,  VectorXd100 &b, char pivot_switch = 'n');


int main(int argc, const char * argv[]) {

    const size_t m {100};
    const size_t n {25};

    //Initialize matrices and vectors
    MatrixXd100 A = MatrixXd100::Zero(m,m);
    for (size_t i {0}; i < m; ++i){
        A(i,i) = 2.;
        if (i != m-1){
            A(i+1,i) = -1.;
            A(i,i+1) = A(i+1,i);
        }
    }

    VectorXd100 x = VectorXd100::Ones(m);
    VectorXd100 b = A*x;
    VectorXd100 x0  = VectorXd100::Zero(m);
    MatrixXd100 Q = MatrixXd100::Zero(m,n+1);
    MatrixXd100 H = MatrixXd100::Zero(n+1,n);

    Galerkin(A, b, x0, n, Q,  H);
    GMRES(A, b, x0, n, Q, H);

    return 0;
}
```

We now show the codes for all algorithms, starting with the Arnoldi iteration, which is present in both Galerkin and GMRES methods:

```cpp
void  Arnoldi(const MatrixXd100 &A, const VectorXd100 &b,  const size_t &n,  MatrixXd100 &Q,
    MatrixXd100 &H, const double tol){
    /*Inputs:
     A = mxm matrix
     b = mx1 initial vector
     n = number of Arnoldi iterations (typically, n << m)
     Q = m x (n+1) matrix (initialized to zero)
     H = (n+1) x n matrix (initialized to zero)
     tol = user-defined tolerance. We need this to avoid dividing by zero (see below)
  */
    /* Purpose:
     - Orthogonalize Q via Modified Gram-Schmidt
     - Put H in upper-Hessenberg form
  */

    VectorXd100 v (b.size());
    Q.col(0) = b/b.norm();

    for (size_t k {0}; k < n; ++k){

        v = A*Q.col(k);

        for (size_t j {0}; j <= k; ++j){
            H(j,k) = Q.col(j).transpose() * v;
            v = v - H(j,k)* Q.col(j);
        }

        if (v.norm() <= tol){               //Happy breakdown :) No need to continue any further
        /* Need to resize H and Q to get rid of unused zero entries, using "conservative"
           to preserve nonzero entries already computed */
            H.conservativeResize(k+1,k+1);
            Q.conservativeResize(A.rows(),k+1);
            break;
        }

        H(k+1,k) = v.norm();
        Q.col(k+1) = v/H(k+1,k);

        if (k == n-1){
            H.conservativeResize(k+1,k+1);
            Q.conservativeResize(A.rows(),k+1);
        }
    }
    //End of Arnoldi. H is now Hesssenberg and Q is orthonormal
};
```

Inside the Galerkin method we also need to tackle a (relatively) small $n \times n$ system, which we solve using our user-defined Gaussian elimination (the code is written so that either LU or LU with pivoting can be used):

```
void Gaussian(MatrixXd100 &A,  VectorXd100 &b, char pivot_switch){
    /* Input A and b to solve system Ax=b for x.
     pivot_switch set to 'y' (yes) applies LU decomposition of A with pivoting;
     pivot_switch set to 'n' (no) or any other char (default case) applies LU decomposition of A
    without pivoting.
    */

    /* In this implementation both A and b are altered, to avoid extra memory allocation.
     An alternate code that leaves A and b untouched would need to allocate memory for
     explicit matrices L, U, P...
    */

    double pivot {};
    long pivot_index {};
    long m {A.rows()};  //using long for compatibility with Eigen's Matrix.rows() /.columns() type
    double l_ik {};

    switch (pivot_switch) {

        case 'Y':
        case 'y':     //yes...pivoting on

            for (long k {0}; k < m-1; ++k){
                //set pivot as the max coeff (in abs value) of (the truncated) column k
                pivot = A.col(k).tail(m-k).cwiseAbs().maxCoeff();
                for (long i {k}; i < m; ++i){
                    if(  (A.col(k)(i) == pivot) || (A.col(k)(i) == -pivot) )
                        pivot_index = i;
                }

                A.row(pivot_index).swap(A.row(k));    //swap rows in both A and b
                b.row(pivot_index).swap(b.row(k));

                for(long i {k+1}; i < m; ++i){

                    l_ik = A(i,k)/A(k,k);

                    for(long j {k}; j < m; ++j)
                        A(i,j) = A(i,j) - l_ik * A(k,j);

                    b(i) = b(i) - l_ik * b(k);
                }
            }
            break;

        case 'n':
        case 'N':
        default:    //no...pivoting off
            for (long k {0}; k < m-1; ++k){
                for(long i {k+1}; i < m; ++i){

                    l_ik = A(i,k)/A(k,k);

                    for(long j {k}; j < m; ++j)
                        A(i,j) = A(i,j) - l_ik * A(k,j);

                    b(i) = b(i) - l_ik * b(k);
                }
            }
            break;
    }       //end of switch statement...moving on to next phase...



    /* Final step (backsolving to solve Ux=y)
     Note, again, we are not allocating new memory for y or x;
     Instead, all calculations are done overwriting b
    */
    b(m-1) = b(m-1)/A(m-1,m-1);
    for(long k {m-2}; k >=0; --k){
        for(long j {k+1}; j < m; ++j)
             b(k) = b(k) - A(k,j) * b(j);
        b(k) = b(k)/A(k,k);
    }
    //Gaussian elimination process done, we now have b = x = solution.
};
```

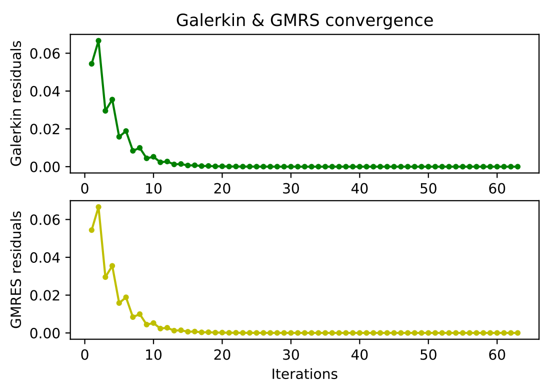
Lastly, we show the GMRES code:

```
void GMRES(const MatrixXd100 &A, const VectorXd100 &b,   VectorXd100 &x0,  const size_t &n,
           MatrixXd100 &Q, MatrixXd100 &H, const double tol){

    cout << "  Applying GMRES to solve Ax=b ... \n "
            << "--------------------------------- \n "<< endl;

    vector<size_t> it_vec {};
    vector<double> res_vec {};
    size_t it {0};
    const size_t it_max {100};   //max number of iterations allowed
    double residual {1.0};
    VectorXd100 x;
    VectorXd100 r0 = b - A*x0;

    do {

        it += 1;
        it_vec.push_back(it);

        Arnoldi(A, r0, n,  Q, H);

        VectorXd100 e0 = VectorXd100::Zero(H.rows());
        e0(0) = 1.0;

        /*Find the vector y that minimizes
            J(y) = || Hy - ||r_0|| * e_0 ||
         using QR least squares
        */
        VectorXd100 temp = r0.norm()*e0;
        VectorXd100 y =  H.colPivHouseholderQr().solve(temp);

        //Compute the correction vector z
        VectorXd100 z = Q*y;

        //Add correction vector to original guess to get the new solution
        x = x0 + z;

        residual = (b - A*x).norm();
        res_vec.push_back(residual);
        cout << "The residual at iteration " << it << " is " << residual <<  ". \n " << endl;

        //update values for next iteration
        x0 = x;
        r0 = b - A*x;

        if (it == it_max) {
            cout << "Max number of iterations reached. \n " << endl;
            break;
        }

        //resize H and Q for next Arnoldi iteration
        H.conservativeResize(n+1,n);
        Q.conservativeResize(A.rows(),n+1);

    } while (residual > tol);

    cout << "Convergence obtained at iteration " << it
         << ". The solution is \n x = \n " << x << " \n \n " << endl;



    //Save residuals and iterations to files
    ofstream resfile ("GMRES_residuals.csv");
    ofstream itfile ("GMRES_iterations.csv");
    for (int i{0}; i < res_vec.size(); ++i) {
        resfile << res_vec.at(i) << endl;
        itfile << it_vec.at(i) << endl;
    }
    resfile.close();
    itfile.close();
};
```

Our last order of business is to plot the convergence of both the Galerkin and GMRES algorithms:

```
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd

font = {'family' : 'serif',
        'weight' : 'normal',
        'size'   : 14}

it = pd.read_csv("Galerkin_iterations.csv", header = None)
Gal_res = pd.read_csv("Galekin_residuals.csv", header = None)
GMRES_res = pd.read_csv("GMRES_residuals.csv", header = None)

# Create two subplots sharing y axis
fig, (ax1, ax2) = plt.subplots(2)   #sharey=True

ax1.plot(it, Gal_res[0], 'g.-')
ax1.set(title='Galerkin & GMRS convergence', ylabel='Galerkin residuals')

ax2.plot(it, GMRES_res[0], 'y.-')
ax2.set(xlabel='Iterations', ylabel=r"GMRES residuals")

# plt.show()
# Save the figure
plt.savefig('Krylov_Convergence.pdf')
plt.close()
```

The output is shown in the following figure:



The convergence is quite fast with either method. Both algorithms allow us to solve a big system such as $m = 100$ using $n \ll m$. ♠

Problem 9. *Solve the following system by using the Cholesky factorization:*

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1.001 & 1.001 \\ 1 & 1.001 & 2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 3 \\ 3.0020 \\ 4.0010 \end{bmatrix}}_{\mathbf{b}}. \tag{9}$$

Solution. We recall that a ***positive-definite matrix*** A is one that, for every nonzero $\mathbf{x}$, satisfies $\mathbf{x}^\top A\mathbf{x} > 0$. Given such symmetric positive-definite matrix A, there exists a lower-triangular matrix L with positive diagonal entries such that

$$\boxed{A = LL^\top} \tag{10}$$

This factorization is called the ***Cholesky factorization***, and L is called the ***Cholesky factor***. Now, in order to solve the system (9), we first find the Cholesky factorization of the matrix A, i.e., a factorization of the form

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}}_{L^\top}. \tag{11}$$

From this expression we can see that

$$a_{11} = l_{11}^2 \implies l_{11} = \sqrt{a_{11}}, \tag{12a}$$

$$a_{21} = l_{11}l_{21} \implies l_{21} = \frac{a_{21}}{l_{11}}, \tag{12b}$$

$$a_{31} = l_{11}l_{31} \implies l_{31} = \frac{a_{31}}{l_{11}}. \tag{12c}$$

That takes care of the first column of L. Similarly, for the second column,

$$a_{22} = l_{21}^2 + l_{22}^2 \implies l_{22} = \sqrt{a_{22} - l_{21}^2}, \tag{12d}$$

$$a_{32} = l_{31}l_{21} + l_{32}l_{22} \implies l_{32} = \frac{a_{32} - l_{31}l_{21}}{l_{22}}. \tag{12e}$$

And lastly, the third column,

$$a_{33} = l_{31}^2 + l_{32}^2 + l_{33}^2 \implies l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2}. \tag{12f}$$

Taking A as in Eq. (9) and using Eqs. (12), we get

- Column 1: $l_{11} = l_{21} = l_{31} = 1$;
- Column 2: $l_{22} = \sqrt{1.001 - 1} = \sqrt{0.001}$, $\quad l_{32} = \frac{1.001-1}{\sqrt{0.001}} = \frac{0.001}{\sqrt{0.001}} = \sqrt{0.001}$;
- Column 3: $l_{33} = \sqrt{2 - 1 - 0.001} = \sqrt{0.999}$.

Thus we have

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & \sqrt{0.001} & 0 \\ 1 & \sqrt{0.001} & \sqrt{0.999} \end{bmatrix}, \qquad L^\top = \begin{bmatrix} 1 & 1 & 1 \\ 0 & \sqrt{0.001} & \sqrt{0.001} \\ 0 & 0 & \sqrt{0.999} \end{bmatrix}.$$

Now, from

$$A\mathbf{x} = \mathbf{b} \implies L\underbrace{L^\top\mathbf{x}}_{:=\mathbf{y}} = \mathbf{b},$$

we see that instead of tackling the system $A\mathbf{x} = \mathbf{b}$ directly, we can solve $L\mathbf{y} = \mathbf{b}$ and subsequently $L^\top\mathbf{x} = \mathbf{y}$. These are triangular systems that are much easier to compute than $A\mathbf{x} = \mathbf{b}$.

Hence, we have

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 1 & \sqrt{0.001} & 0 \\ 1 & \sqrt{0.001} & \sqrt{0.999} \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} 3 \\ 3.0020 \\ 4.0010 \end{bmatrix}}_{\mathbf{b}},$$

from which we get

$$\begin{aligned} & y_1 = 3, \\ 3 + \sqrt{0.001}y_2 = 3.0020 \implies & y_2 = \frac{0.0020}{\sqrt{0.001}}, \\ 3 + 0.0020 + \sqrt{0.999}y_3 = 4.0010 \implies & y_3 = \sqrt{0.999}. \end{aligned}$$

From this we solve the following system to get our solution vector $\mathbf{x}$:

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 0 & \sqrt{0.001} & \sqrt{0.001} \\ 0 & 0 & \sqrt{0.999} \end{bmatrix}}_{L^\top} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 3 \\ 0.0020/\sqrt{0.001} \\ \sqrt{0.999} \end{bmatrix}}_{\mathbf{y}},$$

from which we get

$$\begin{aligned} x_3 &= 1, \\ \sqrt{0.001}x_2 + \sqrt{0.001} = \frac{0.0020}{\sqrt{0.001}} \implies x_2 &= \frac{\sqrt{0.001}}{\sqrt{0.001}} = 1, \\ x_1 + 1 + 1 = 3 \implies x_1 &= 1. \end{aligned}$$

Thus, our solution to the system (9) is the vector $\mathbf{x} = (1 \quad 1 \quad 1)^\top$. ♠

Problem 10. *Show that the following matrix is positive-definite with and without finding the Cholesky factorization:*

$$A = \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \tag{13}$$

Solution. Without using the Cholesky factorization, we may conclude that a symmetric matrix (which is indeed the case for the matrix in (13)) is positive-definitive if and only if its eigenvalues are all positive. [3] The eigenvalues of A are $\{6, 4, 4, 2\}$, which are all positive, so we may conculde that A is indeed positive-definite. $\surd$

Now, using the Cholesky factorization, if we manage to write A in the form (10) with L having positive diagonal entries, then that would prove that A is indeed positive-definite. To see this, note that

$$A = LL^\top \quad \implies \quad \mathbf{x}^\top A\mathbf{x} = \mathbf{x}^\top LL^\top\mathbf{x} = \left(L^\top\mathbf{x}\right)^\top \left(L^\top\mathbf{x}\right) \geq 0,$$

but, since L is invertible, $L^\top\mathbf{x} \neq 0$ for all nonzero $\mathbf{x}$. Thus the above inequality becomes a strict inequality, i.e., $\mathbf{x}^\top A\mathbf{x} > 0$ for all $\mathbf{x} \neq 0$. We now have

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} l_{11} & l_{21} & l_{31} & l_{41} \\ 0 & l_{22} & l_{32} & l_{42} \\ 0 & 0 & l_{33} & l_{43} \\ 0 & 0 & 0 & l_{44} \end{bmatrix}}_{L^\top}. \tag{14}$$

From this expression we can see that

$$a_{11} = l_{11}^2 \implies l_{11} = \sqrt{a_{11}}, \tag{15a}$$

$$a_{21} = l_{11}l_{21} \implies l_{21} = \frac{a_{21}}{l_{11}}, \tag{15b}$$

$$a_{31} = l_{11}l_{31} \implies l_{31} = \frac{a_{31}}{l_{11}}, \tag{15c}$$

$$a_{41} = l_{11}l_{41} \implies l_{41} = \frac{a_{41}}{l_{11}}. \tag{15d}$$

That takes care of the first column of L. Similarly, for the second column,

$$a_{22} = l_{21}^2 + l_{22}^2 \implies l_{22} = \sqrt{a_{22} - l_{21}^2}, \tag{15e}$$

$$a_{32} = l_{31}l_{21} + l_{32}l_{22} \implies l_{32} = \frac{a_{32} - l_{31}l_{21}}{l_{22}}, \tag{15f}$$

$$a_{42} = l_{41}l_{21} + l_{42}l_{22} \implies l_{42} = \frac{a_{42} - l_{41}l_{21}}{l_{22}}. \tag{15g}$$

[3] We also have the fact that if the matrix A is positive-definite, then the diagonal entries a_{ii} must all be positive. However, this is a necessary condition, not a sufficient one (i.e., the converse does not hold in general).

Third column,

$$a_{33} = l_{31}^2 + l_{32}^2 + l_{33}^2 \quad\Longrightarrow\quad l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2}, \tag{15h}$$

$$a_{43} = l_{41}l_{31} + l_{42}l_{32} + l_{43}l_{33} \quad\Longrightarrow\quad l_{43} = \frac{a_{43} - l_{41}l_{31} - l_{42}l_{32}}{l_{33}}. \tag{15i}$$

And lastly, the fourth column,

$$a_{44} = l_{41}^2 + l_{42}^2 + l_{43}^2 + l_{44}^2 \quad\Longrightarrow\quad l_{44} = \sqrt{a_{44} - l_{41}^2 - l_{42}^2 - l_{43}^2}. \tag{15j}$$

Taking A as in Eq. (13) and using Eqs. (15), we get

- Column 1: $l_{11} = 2, \qquad l_{21} = l_{31} = -\frac{1}{2}, \qquad l_{41} = 0;$
- Column 2: $l_{22} = \sqrt{4 - \frac{1}{4}} = \frac{\sqrt{15}}{2}, \qquad l_{32} = \frac{0 - (-\frac{1}{2})(-\frac{1}{2})}{\frac{\sqrt{15}}{2}} = -\frac{1}{2\sqrt{15}}, \qquad l_{42} = \frac{-1 - 0\cdot(-\frac{1}{2})}{\frac{\sqrt{15}}{2}} = -\frac{2}{\sqrt{15}};$
- Column 3: $l_{33} = \sqrt{4 - \frac{1}{4} - \frac{1}{60}} = 2\sqrt{\frac{14}{15}}, \qquad l_{43} = \frac{-1 - 0\cdot(-\frac{1}{2}) - (-\frac{1}{2\sqrt{15}})(-\frac{2}{\sqrt{15}})}{2\sqrt{\frac{14}{15}}} = -4\sqrt{\frac{2}{105}};$
- Column 4: $l_{44} = \sqrt{4 - 0 - \frac{4}{15} - 16\cdot\frac{2}{105}} = 2\sqrt{\frac{6}{7}}.$

Hence we have formed

$$\underbrace{\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 2 & 0 & 0 & 0 \\ -1/2 & \sqrt{15}/2 & 0 & 0 \\ -1/2 & -1/(2\sqrt{15}) & 2\sqrt{14/15} & 0 \\ 0 & -2/\sqrt{15} & -4\sqrt{2/105} & 2\sqrt{6/7} \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & -1/2 & -1/2 & 0 \\ 0 & \sqrt{15}/2 & -1/(2\sqrt{15}) & -2/\sqrt{15} \\ 0 & 0 & 2\sqrt{14/15} & -4\sqrt{2/105} \\ 0 & 0 & 0 & 2\sqrt{6/7} \end{bmatrix}}_{L^\top}.$$

Since we have found a decomposition for A of the form $A = LL^\top$, with all the diagonal entries of L being positive, we may conclude that the matrix A is positive-definite. ♠