28.1) Firstly, the QR algorithm without shifts is given by,

$$\left[\begin{array}{ll} A_0 = A & \\ \text{for } K = 1, 2, \ldots & \\ \quad Q_K R_K = A_{K-1} & \text{(QR factorization)} \\ \quad A_K = R_K Q_K & \end{array}\right.$$

If A is orthogonal, the QR factorization

$A = A_0 = Q_1 R_1 = A \cdot I$ (If A is already orthogonal $QR(A) = [A, I]$)

So $A_1 = R_1 Q_1 = I \cdot A = A$.

It follows that after every iteration, we will have $Q_K = A$, $R_K = I$, and $A_K = A$, so the algorithm really doesn't do much at all.

Comparing with Theorem 28.4, A_K no longer (always)* converges to $\text{diag}(\vec{\lambda})$ as $K \to \infty$, $A_K = A$ $\forall K \geq 0$, but Q_K still converges to $Q = A$.

In general, an orthogonal matrix need not have eigenvalues of distinct modulus, and need not be real and symmetric.

* If $A = \text{diag}(\vec{\lambda})$ with $\vec{\lambda}$ a vector of distinct real eigenvalues, it is technically orthogonal, real, and symmetric, so Theorem 28.4 trivially applies, but this is a special case, and is already diagonalized.

30.1) We are looking for a 2×2 rotation matrix $J = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$ such that

$(\lambda_i \neq 0)$

$$J^T A J = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}\begin{bmatrix} a & d \\ d & b \end{bmatrix}\begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = D$$

We have, by multiplying everything out, the following system of equations

$$\begin{bmatrix} c(ac-ds)-s(cd-bs) & \overset{2}{c(cd-bs)+s(ac-ds)} \\ \underset{}{\overset{1}{c(as+cd)-s(bc+ds)}} & c(bc+ds)+s(as+cd) \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Focusing on equation 1 and 2,

1 $\rightarrow$ $c(as+cd) = s(bc+ds)$

2 $\rightarrow$ $c(cd-bs) = s(ds-ac)$

Dividing 1 with 2,

$$\frac{(as+cd)}{(cd-bs)} = \frac{(bc+ds)}{(ds-ac)}$$

$\rightarrow$ $(as+cd)(ds-ac) = (cd-bs)(bc+ds)$

$\underline{ads^2} - a^2sc + \cancel{sc}d^2 \underline{-adc^2} = \underline{bdc^2} + d^2\cancel{s}c - b^2sc \underline{-bds^2}$

$\rightarrow$ $bd(c^2-s^2) + ad(c^2-s^2) = sc(b^2-a^2)$

$\rightarrow$ $\cancel{(b+a)}d\cos(2\theta) = \frac{1}{2}\sin(2\theta)(b-a)\cancel{(b+a)}$

(since $\cos^2\theta - \sin^2\theta = \cos(2\theta)$ and $2\sin\theta\cos\theta = \sin(2\theta)$)

Dividing both sides by $\cos(2\theta)$ we see

$$d = \frac{1}{2}\tan(2\theta)\,[b-a], \quad \text{or}$$

$$\tan(2\theta) = \frac{2d}{b-a}, \quad \text{as desired.}$$

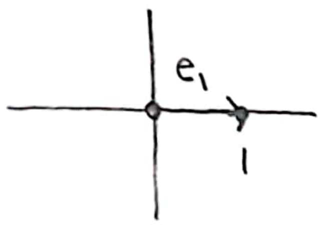
We have chosen θ such that D is diagonal.
Geometrically speaking, we have

$$J^T A J = D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

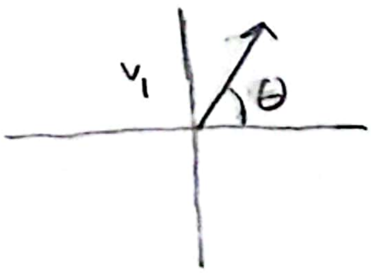
or

$$J^T A J = J^T A J I = J^T A J \left[\vec{e}_1 \,|\, \vec{e}_2 \right] = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Consider the vector e_1

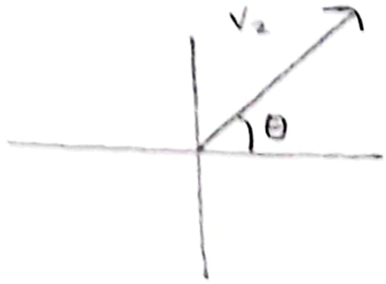


$$\Big\downarrow \; Je_1 = v_1$$



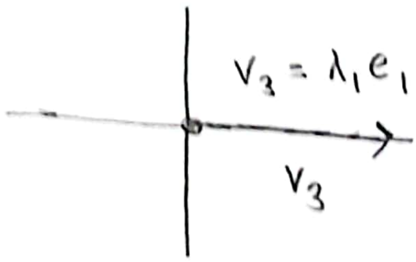
(still unit length)
J rotates e_1 by the θ we found

$\Big\downarrow \quad v_2 = \Lambda J e_1$



Now Λ stretches v_1 so that its signed length is λ_1

$\Big\downarrow \quad v_3 = J^T \Lambda J e_1$



Now J^T rotates v_2 back to be along the e_1 axis but now has signed length λ_1

The same happens for e_2, but it will get mapped to $\lambda_2 e_2$ instead. The significance of choosing θ such that $\tan(2\theta) = \dfrac{2d}{b-a}$, is that if we chose any other rotation angle, we wouldn't have this nice property that the e_i's get mapped to $\lambda_i e_i$, meaning if this angle was not chosen, at the end of our rotating/stretching, we wouldn't land back onto the principle axes.

30.3 Put $S_0 = \sum_{i \neq j} (a_{ij})^2$, the sum of the off diagonal entries, then after 1 step we have

$$S_1 = S_0 - 2a^2 \quad \left(\text{where } a = \max_{i \neq j} (a_{ij})^2\right)$$

So,

$$\overset{(*)}{\frac{S_1}{S_0}} = 1 - \frac{2a^2}{S_0}$$

] since A is symmetric we will 0 out 2 terms

However, we can say

$$\sum_{i \neq j} a_{ij}^2 \leq \overbrace{(m^2 - m)}^{\#\text{ of off-diags}} a^2$$

↖ max off-diag square

$$\rightarrow \quad \frac{\sum_{i \neq j} a_{ij}^2}{m^2 - m} \leq a^2$$

$$\rightarrow \quad -a^2 \leq -\frac{S_0}{m^2 - m}$$

So

$$(*) = 1 - \frac{2a^2}{S_0} \leq 1 - \frac{2S_0}{S_0(m^2 - m)} = \overset{(A)}{1 - \frac{2}{(m^2 - m)}}$$

So the ratio of the sum decreases by a factor of at least (A) in going from the current step to the next, so we are done.

Problem A1:

a) Gerschgorin's here tells us that eigenvalues of A must either satisfy |lamda – 2| <= 1 (the sum of the absolute-values of the off-diagonals in the first/last row is 1 and the diagonal is 2), |lambda – 2| <= 2 (the sum of the absolute-values of the off-diagonals in the rest of the rows is 2). So the eigenvalues are either in the interval [1, 3] or [0, 4], as our matrix is real and symmetric it must have only real eigenvalues.

b)

Editor - C:\Users\Nwhybra\Desktop\UW AMATH Masters\AMATH 584\HW\HW8\power_iteration.m

pure_qr_no_shifts.m | test.m | RandOrthMat.m | power_iteration.m | inverse_iteration.m

```
function max_eig = power_iteration(A, stop)
    m = size(A, 1);
    v = rand(m, 1);
    v = v / norm(v, 2);

    error = inf;
    while error > stop
       w = A * v;
       v = w / norm(w, 2);
       max_eig = v.' * A * v;
       error = norm(A*v - max_eig*v, 2);
    end
end
```

1x1 double

	1
1	3.9190

c)

Editor - C:\Users\Nwhybra\Desktop\UW AMATH Masters\AMATH 584\HW\HW8\pure_qr_no_shifts.m

pure_qr_no_shifts.m | test.m | RandOrthMat.m | power_iteration.m | inverse_iteration.m

```matlab
function [B, Q] = pure_qr_no_shifts(A, stop)
    B = A;
    m = size(A, 1);
    % A matrix with 1's on the off-diagonals and 0's on the diagonals.
    % Used to conveniently grab the matrix entries I want to compute the
    % error.
    off_diagonal_mask = logical(ones(m, m) - eye(m));

    error = inf;
    while error > stop
        [Q, R] = qr(B);
        B = R * Q;
        % Here the error is the maximum of the off-diagonal entries.
        error = sum(max(abs(B(off_diagonal_mask))));
    end
end
```

10x10 double

	1	2	3	4	5	6	7	8	9	10
1	3.9190	-9.5441e-13	2.5982e-16	-1.8796e-16	-2.8892e-16	-1.3452e-16	-2.5568e-16	-2.5253e-17	5.7313e-17	1.8443e-17
2	-9.5467e-13	3.6825	1.2341e-16	1.5202e-16	-1.4776e-16	-2.1867e-16	7.2846e-17	-1.3462e-16	-8.3503e-17	1.9793e-16
3	0	-1.2700e-20	3.3097	1.3249e-16	1.5238e-16	1.5350e-16	5.3961e-18	1.5278e-16	-2.8109e-17	-4.7965e-16
4	0	0	-1.1176e-29	2.8308	2.7220e-16	2.9658e-16	1.3209e-16	-1.7298e-16	-1.7246e-16	1.9614e-16
5	0	0	0	-1.8971e-40	2.2846	5.9651e-17	2.5885e-16	3.2062e-16	3.1835e-16	1.4064e-16
6	0	0	0	0	-5.8826e-54	1.7154	-5.9967e-17	2.4485e-16	1.4407e-16	1.3595e-17
7	0	0	0	0	0	-3.7753e-72	1.1692	1.5039e-17	8.4684e-17	1.7691e-16
8	0	0	0	0	0	0	-3.2122e-99	0.6903	1.4902e-17	-2.3424e-16
9	0	0	0	0	0	0	0	-2.6214e-1...	0.3175	2.0518e-16
10	0	0	0	0	0	0	0	0	-2.5458e-2...	0.0810

d)

Editor - C:\Users\Nwhybra\Desktop\UW AMATH Masters\AMATH 584\HW\HW8\inverse_iteration.m

pure_qr_no_shifts.m | test.m | RandOrthMat.m | power_iteration.m | inverse_iteration.m

```
function eigen_vector = inverse_iteration(A, lambda, stop)
    m = size(A, 1);
    v = rand(m, 1);
    v = v / norm(v, 2);

    error = inf;
    I = eye(m);
    while error > stop
        w = (A - lambda * I) \ v;
        v = w / norm(w, 2);
        error = norm(A * v - lambda * v, 2);
    end

    eigen_vector = v;
end
```

10x1 double

	1
1	-0.4221
2	0.1201
3	0.3879
4	-0.2305
5	-0.3223
6	0.3223
7	0.2305
8	-0.3879
9	-0.1201
10	0.4221

Full test script:

Editor - C:\Users\Nwhybra\Desktop\UW AMATH Masters\AMATH 584\HW\HW8\test.m

pure_qr_no_shifts.m | test.m | RandOrthMat.m | power_iteration.m | inverse_iteration.m | +

```
% Construct the matrix A.
m = 10;
v = ones(10, 1);
A = diag(2 * v, 0) + diag(-1 * v(1:m-1), -1) + diag(-1 * v(1:m-1), 1);

% Computing eigen-values to check my work.
lambs = eigs(A);

% Part (b).
max_eig = power_iteration(A, 1e-6);

% Part (c).
D = pure_qr_no_shifts(A, 1e-12);

% Part(d).
fifth_eig = D(5, 5);
eigen_vector_5 = inverse_iteration(A, fifth_eig, 1e-8);
error = norm(A * eigen_vector_5 - fifth_eig * eigen_vector_5, 2);
```