1. **The Power Method**

   (a) Pseudocode:

   POWER ITERATION
   Given an initial vector u_0,
   $i = 0$
   repeat
   $t_{i+1} = Au_i$
   $u_{i+1} = t_{i+1}/\|t_{i+1}\|_2$ (approximate eigenvector)
   $\theta_{i+1} = u_{i+1}^H A u_{i+1}$ (approximate eigenvalue)
   $i = i + 1$
   until convergence

   (b) Practical stopping criterion: $|\theta_{i+1} - \theta_i| \leq tol \cdot |\theta_i|$.

   (c) Example: Let

   $$A = \begin{bmatrix} -261 & 209 & -49 \\ -530 & 422 & -98 \\ -800 & 631 & -144 \end{bmatrix}.$$

   and $\lambda(A) = \{10, 4, 3\}$. Let $u_0 = (1, 0, 0)^T$, then

i	1	2	3	$\cdots$	10
θ_i	994.49	13.0606	10.07191	$\cdots$	10.0002

   (d) Convergence analysis: Assume

   $$A = X\Lambda X^{-1}$$

   with $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ and $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$. Then, we can show that

   - $u_i = \frac{A^i u_0}{\|A^i u_0\|} \to x_1/\|x_1\|$, where $x_1 = Xe_1$ as $i \to \infty$.
   - $\theta_i \to \lambda_1$ as $i \to \infty$.

   (e) The convergence rate depends on $\frac{|\lambda_2|}{|\lambda_1|}$. Therefore, if $\frac{|\lambda_2|}{|\lambda_1|}$ is close to 1, then the power method could be very slow convergent or doesn't converge at all.

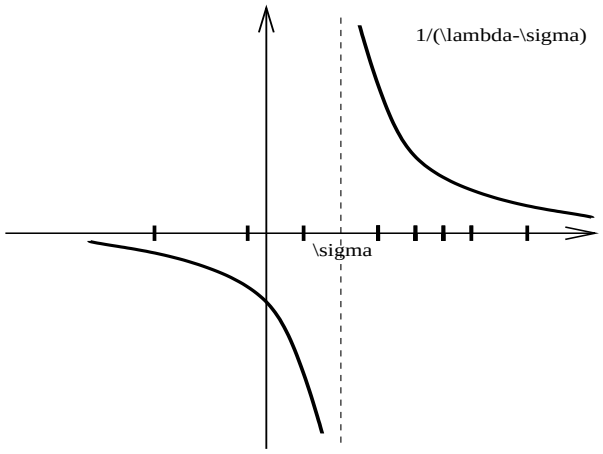
2. **Inverse Iteration**

(a) Purposes:

- Overcome the drawbacks of the power method (slow convergence)
- find an eigenvalue closest to a particular given number (called *shift*): σ

(b) Observation: if λ is an eigenvalue of A, then

- $\lambda - \sigma$ is an eigenvalue of $A - \sigma I$,
- $\frac{1}{\lambda - \sigma}$ is an eigenvalue of $(A - \sigma I)^{-1}$.



(c) Pseudocode

INVERSE ITERATION
Given an initial vector u_0 and a shift σ
$i = 0$
repeat
  solve $(A - \sigma I)t_{i+1} = u_i$ for t_{i+1}
  $u_{i+1} = t_{i+1}/\|t_{i+1}\|_2$ (approximate eigenvector)
  $\theta_{i+1} = u_{i+1}^H A u_{i+1}$ (approximate eigenvalue)
  $i = i + 1$
until convergence

(d) Convergence analysis: Assume $A = X\Lambda X^{-1}$ with $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ and λ_k is the eigenvalue cloest to the shift σ. It can be shown that

- $u_i \to x_k/\|x_k\|$ as $i \to \infty$, where $x_k = Xe_k$
- θ_i converges to λ_k $i \to \infty$.
- Convergence rate depends on $\max_{j \neq k} \frac{|\lambda_k - \sigma|}{|\lambda_j - \sigma|}$.

(e) Advantages: the ability to converge to any desired eigenvalue (the one nearest to the shift σ). By choosing σ very close to a desired eigenvalue, the method converges very quickly and thus not be as limited by the proximity of nearby eigenvalues as is the original power method. The method is particularly effective when we have a good approximation to an eigenvalue and want only its corresponding eigenvector.

(f) Drawbacks: (a) expensive in general: solving $(A - \sigma I)t_{i+1} = u_i$ for u_{i+1}. One LU factorization of $A - \sigma I$ is required, which could be very expensive for large matrices, (b) Only compute one eigenpair.

3. **Orthogonal iteration (subspace iteration, simultaneous iteration)**

(a) Purpose: compute a p-dimensional invaraint subspace, $p > 1$, rather than one eigenvector at a time.

(b) Pseudocode:

ORTHOGONAL ITERATION
Given an initial $n \times p$ orthogonal matrix Z_0
$i = 0$
repeat
  $Y_{i+1} = AZ_i$
  $Y_{i+1} = Z_{i+1}R_{i+1}$ (QR decomposition)
  $i = i + 1$
until convergence

The use of QR decomposition keeps the vectors spanning span$\{A^i Z_0\}$ of full rank.

(c) Convergence: under mild conditions, Z_i converges to the invariant subspace spanned by the first p eigenvectors corresponding to the p dominant eigenvalues, where

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_p| > |\lambda_{p+1}| \geq \cdots \geq |\lambda_n|.$$

If we let $B_i = Z_i^T A Z_i$, then

$$\|AZ_i - Z_i B_i\| \to 0 \quad \text{as} \quad i \to \infty$$

and eigenvalues of B_i approximate the dominant eigenvalues of A.

Convergence rate depends on $|\lambda_{p+1}|/|\lambda_p|$.

(d) Example: Let $Z_0 = [e_1, e_2, e_3]$ and

```
A =
   -0.4326    1.1892   -0.5883   -0.0956   -0.6918   -0.3999
   -1.6656   -0.0376    2.1832   -0.8323    0.8580    0.6900
    0.1253    0.3273   -0.1364    0.2944    1.2540    0.8156
    0.2877    0.1746    0.1139   -1.3362   -1.5937    0.7119
   -1.1465   -0.1867    1.0668    0.7143   -1.4410    1.2902
    1.1909    0.7258    0.0593    1.6236    0.5711    0.6686

Eigenvalues of A = -2.1659 +- 0.5560i, 2.1493, 0.2111 +- 1.9014i, -0.9548

i = 10: Eigenvalues of Z'_10*A*Z_10: -1.4383 +- 0.3479i,  2.1500

i = 30: Eigenvalues of Z'_30*A*Z_30: -2.1592 +- 0.5494i,  2.1118

i = 70: Eigenvalues of Z'_70*A*Z_70: -2.1659 +- 0.5560i,  2.1493
```

(e) An important special case: Let $p = n$ and $Z_0 = I$, then $A_i = Z_i^T A Z_i$ converges to the *Schur form* of A provided that

(1) all eigenvalues of A have distinct absolute values and

(2) all the principal submatrices of S have full rank, where we assume $A = S\Lambda S^{-1}$.

(f) Example: the same test matrix, numerical results of orthogonal iteration with $Z_0 = I$:

```
A_10 =
   -1.6994   -0.2201   -0.8787    1.4292    0.3847    0.0112
    0.0007    1.1325   -1.2186    1.2245   -0.0867   -0.0648
    0.2637   -1.9636   -0.1598    2.3959   -0.8136   -0.4311
   -0.0364   -0.2346    0.5527   -0.4393   -1.9263   -1.2496
   -0.4290    1.3482    1.1484    0.6121   -0.5937   -0.2416
    0.0003   -0.0013   -0.0003    0.0011   -0.0014   -0.9554


A_30 =
   -2.4055   -1.0586   -1.3420    0.0991   -1.1210    0.1720
    0.0517    0.9645   -1.6519    0.8512    0.7215    0.7654
    0.2248   -1.9947   -0.7656   -1.1876   -0.2736    0.1552
    0.0029    0.0263   -0.0682    0.1381   -2.3094   -0.6765
    0.0147   -0.0808   -0.0569    1.5462    0.3082    0.8476
    0.0000    0.0000    0.0000    0.0000    0.0000   -0.9548


A_70 =
   -2.0800    1.6092   -0.6426    1.1025   -0.1553    0.0734
   -0.1690   -1.6820    1.7425   -0.0311    0.9229   -0.3087
    0.0677    1.2521    1.5795   -0.1458    1.2092    0.7088
    0.0000    0.0000   -0.0005    0.5575   -1.7386   -1.0520
    0.0000    0.0004    0.0003    2.1489   -0.1353   -0.3250
    0.0000    0.0000    0.0000    0.0000    0.0000   -0.9548

....
```

As we see A_k is converging to a quasi-upper triangular matrix as k increasing.

4. **QR iteration**

(a) Our goal is to reorganize orthogonal iteration to incorporate shifting and inverting as in the inverse iteration. This will make it more efficient and eliminate the assumption that eigenvalues differ in magnitude.

(b) Pseudocode

QR Iteration, without shift
$A_0 = A$
$i = 0$
repeat
  $A_i = Q_i R_i$ (QR decomposition)
  $A_{i+1} = R_i Q_i$
  $i = i + 1$
until convergence

(c) Properties

- Observe that
$$A_{i+1} = R_i Q_i = Q_i^T Q_i R_i Q_i = Q_i^T A_i Q_i$$
- A_{i+1} is orthogonally similar to $A_0 = A$. Therefore A_{i+1} and A have same eigenvalues:
$$A_{i+1} = (Q_0 Q_1 \cdots Q_{i-1} Q_i)^T A (Q_0 Q_1 \cdots Q_{i-1} Q_i).$$
Note that $Q_0 \cdots Q_{i-1} Q_i$ is an orthogonal matrix since all Q_j are.
- A_i computed by QR iteration is identical to the matrix $Z_i^T A Z_i$ implicitly computed by orthogonal iteration. In fact, we have

  Proposition. Let A_i be the matrix computed by QR iteration. Then $A_i = Z_i^T A Z_i$, where Z_i is the matrix computed from orthogonal iteration starting with $Z_0 = I$.

  Therefore, A_i converges to Schur form if all the eigenvalues have different absolute values.

(d) Example. The same test matrix, numerical results of QR iteration

```
A_10 =
   -1.6994    0.2201   -0.8787   -1.4292   -0.3847    0.0112
   -0.0007    1.1325    1.2186    1.2245   -0.0867    0.0648
    0.2637    1.9636   -0.1598   -2.3959    0.8136   -0.4311
    0.0364   -0.2346   -0.5527   -0.4393   -1.9263    1.2496
    0.4290    1.3482   -1.1484    0.6121   -0.5937    0.2416
    0.0003    0.0013   -0.0003   -0.0011    0.0014   -0.9554


A_30 =
   -2.4055   -1.0586    1.3420   -0.0991    1.1210    0.1720
    0.0517    0.9645    1.6519   -0.8512   -0.7215    0.7654
   -0.2248    1.9947   -0.7656   -1.1876   -0.2736   -0.1552
   -0.0029   -0.0263   -0.0682    0.1381   -2.3094    0.6765
   -0.0147    0.0808   -0.0569    1.5462    0.3082   -0.8476
    0.0000    0.0000    0.0000    0.0000    0.0000   -0.9548
```

```
A_70 =
   -2.0800    1.6092   -0.6426    1.1025   -0.1553   -0.0734
   -0.1690   -1.6820    1.7425   -0.0311    0.9229    0.3087
    0.0677    1.2521    1.5795   -0.1458    1.2092   -0.7088
    0.0000    0.0000   -0.0005    0.5575   -1.7386    1.0520
    0.0000    0.0004    0.0003    2.1489   -0.1353    0.3250
    0.0000    0.0000    0.0000    0.0000    0.0000   -0.9548
```

Note that the results are identical to the orthogonal iteration.

5. **QR iteration with shifts ⇒ QR Algorithm**

(a) Purpose: accelerate the convergence of QR iteration by using shifts

(b) Pseudo-code

QR Iteration with Shifts
$A_0 = A$
$i = 0$
repeat
  Choose a shift σ_i
  $A_i - \sigma_i I = Q_i R_i$ (QR decomposition)
  $A_{i+1} = R_i Q_i + \sigma_i I$
  $i = i + 1$
until convergence

(c) Property: A_i and A_{i+1} are orthogonally similar:

$$A_{i+1} = Q_i^T A_i Q_i.$$

Therefore, A_{i+1} and A are orthogonally similar, and A_{i+1} and A have the same eigenvalues.

(d) How to choose the shifts σ_i?

- If σ_i is an exact eigenvalue of A, then it can be shown that

$$A_{i+1} = R_i Q_i + \sigma_i I = \begin{bmatrix} A' & a \\ 0 & \sigma_i \end{bmatrix}.$$

  This means that the algorithm converges in one iteration. If more eigenvalues are wanted, we can apply the algorithm again to the $n-1$ by $n-1$ matrix A'.

- In practice, a common choice of the σ_i is

$$\sigma_i = A_i(n, n).$$

  A motivation of this choice is by observing that the convergence of the QR iteration (without a shift), the (n, n) entry of A_i usually converges to an eigenvalue of A first.

(e) Example. The same test matrix as before. The following is the numerical result of QR iteration with a shift.

With the shift σ_0 as an exact eigenvalue $\sigma_0 = 2.1493$, then

```
A_1 =
   -1.4127    1.4420    1.0845   -0.6866   -0.1013   -0.2042
   -1.2949   -0.2334    1.4047   -1.3695    1.5274   -0.7062
    0.5473    0.1343   -0.7991   -0.6716    1.1585    0.0736
   -0.2630    0.0284    0.5440   -1.4616   -1.5892    0.9205
   -1.6063   -0.3898    0.3410    0.1623   -0.9576   -0.5795
    0.0000    0.0000    0.0000    0.0000    0.0000    2.1493
```

With the shifts $\sigma_i = A_i(n,n)$.

```
A_7 =
   -2.4302    2.0264   -0.2799   -0.2384    0.3210   -0.0526
   -0.1865   -1.4295   -1.3515    0.0812    0.8577   -0.0388
   -0.1087   -0.8991    0.4491    0.4890   -1.8463   -1.2034
   -0.0008    0.0511   -0.5997   -0.7839   -0.8088   -0.5188
   -0.0916   -0.8273    1.6940    0.0645   -0.6698   -0.0854
    0.0000    0.0000    0.0000    0.0000    0.0000    2.1493
```

We observe that by 7th iteration, we have found an eigenvalue of A.

(f) Note that the QR decomposition in the algorithm takes $\mathcal{O}(n^3)$ flops. Even if the algorithm took n iterations to converge, the overall cost of the algorithm will be $\mathcal{O}(n^4)$. This is too expensive (today, the complexity of algorithms for all standard matrix computation problems is at $\mathcal{O}(n^3)$.) However, if the matrix is initially reduced to *upper Hessenberg form*, then the QR decomposition of a Hessenberg form costs $\mathcal{O}(n^2)$ flops. As a result, the overall cost of the algorithm is reduced to $\mathcal{O}(n^3)$. This is referred to as the *Hessenberg QR algorithm*, the method of choice for dense eigenvalue problem today, say Matlab's eigensolver `eig` use LAPACK's implementation of the **QR algorithm**.

(g) QR algorithm is one of the top 10 algorithms in the 20th century.