Applied Numerical Linear Algebra. Lecture 11

Algorithms for the Nonsymmetric Eigenvalue Problem

We assume that A is real.

- Power method

  This method can find only the largest eigenvalue for A and the corresponding eigenvector.

- Inverse iteration

  We find all other eigenvalues and eigenvectors applying method for $(A - \sigma I)^{-1}$ for some shift σ.

- Orthogonal iteration

  Lets compute entire invariant subspace.

- QR iteration

  reorganized orthogonal iteration, ultimate algorithm.

- Hessenberg reduction

- Tridiagonal and bidiagonal reduction

Power Method

ALGORITHM. Power method: Given x_0, we iterate

```
$i = 0$
$repeat$
    $y_{i+1} = Ax_i$
    $x_{i+1} = y_{i+1}/||y_{i+1}||_2$    $(approximate\ eigenvector)$
    $\tilde{\lambda}_{i+1} = x_{i+1}^T Ax_{i+1}$    $(approximate\ eigenvalue)$
    $i = i + 1$
$until\ convergence$
```

Inverse Iteration

We will overcome the drawbacks of the power method just described by applying the power method to $(A - \sigma I)^{-1}$ instead of A, where σ is called a *shift*. This will let us converge to the eigenvalue closest to σ, rather than just λ_1. This method is called *inverse iteration* or the *inverse power method*.

ALGORITHM. Inverse iteration: Given x_0, we iterate

```
$i = 0$
repeat
    $y_{i+1} = (A - \sigma I)^{-1} x_i$
    $x_{i+1} = y_{i+1} / ||y_{i+1}||_2$    (approximate eigenvector)
    $\tilde{\lambda}_{i+1} = x_{i+1}^T A x_{i+1}$    (approximate eigenvalue)
    $i = i + 1$
until convergence
```

To analyze the convergence, note that $A = S\Lambda S^{-1}$ implies $A - \sigma I = S(\Lambda - \sigma I)S^{-1}$ and so $(A - \sigma I)^{-1} = S(\Lambda - \sigma I)^{-1}S^{-1}$. Thus $(A - \sigma I)^{-1}$ has the same eigenvectors s_i as A with corresponding eigenvalues $((\Lambda - \sigma I)^{-1})_{jj} = (\lambda_j - \sigma)^{-1}$. The same analysis as before tells us to expect x_i to converge to the eigenvector corresponding to the largest eigenvalue in absolute value.

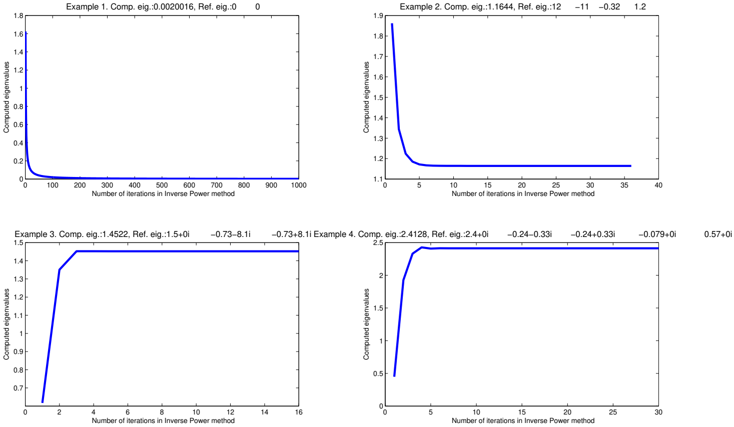
Assume that $|\lambda_k - \sigma|$ **is smaller than all the other** $|\lambda_i - \sigma|$ **so that** $(\lambda_k - \sigma)^{-1}$ **is the largest eigenvalue in absolute value**. Also, write $x_0 = S([\xi_1, \ldots, \xi_n]^T)$ as before, and assume $\xi_k \neq 0$. Then

$$\begin{aligned}
(A - \sigma I)^{-i} x_0 &= (S(\Lambda - \sigma I)^{-i} S^{-1}) S \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} = S \begin{bmatrix} \xi_1(\lambda_1 - \sigma)^{-i} \\ \vdots \\ \xi_n(\lambda_n - \sigma)^{-i} \end{bmatrix} \\
&= \xi_k(\lambda_k - \sigma)^{-i} S \begin{bmatrix} \frac{\xi_1}{\xi_k}(\frac{\lambda_k - \sigma}{\lambda_1 - \sigma})^i \\ \vdots \\ 1 \\ \vdots \\ \frac{\xi_n}{\xi_k}(\frac{\lambda_k - \sigma}{\lambda_n - \sigma})^i \end{bmatrix},
\end{aligned}$$

where the 1 is in entry k. Since all the fractions $(\lambda_k - \sigma)/(\lambda_i - \sigma)$ are less than one in absolute value, the vector in brackets approaches e_k, so $(A - \sigma I)^{-i} x_0$ gets closer and closer to a multiple of $Se_k = s_k$, the eigenvector corresponding to λ_k. As before, $\tilde{\lambda}_i = x_i^T A x_i$ also converges to λ_k.

- The advantage of inverse iteration over the power method is the ability to converge to any desired eigenvalue (the one nearest the shift σ).
- By choosing σ a very close to a desired eigenvalue, we can converge 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.

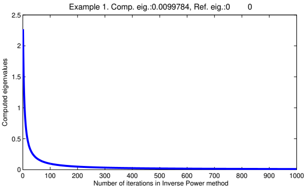
Examples of running of Inverse iteration method in Matlab



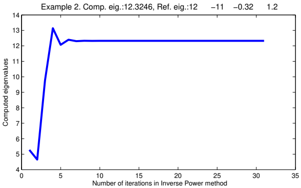
$$\sigma = 2$$

Examples of running of Inverse iteration method in Matlab

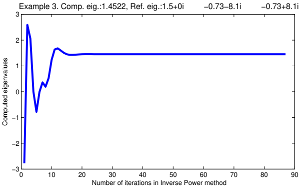
Example 1. Comp. eig.:0.0099784, Ref. eig.:0 0



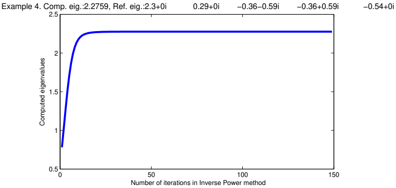
Example 2. Comp. eig.:12.3246, Ref. eig.:12 −11 −0.32 1.2



Example 3. Comp. eig.:1.4522, Ref. eig.:1.5+0i −0.73−8.1i −0.73+8.1i



Example 4. Comp. eig.:2.2759, Ref. eig.:2.3+0i 0.29+0i −0.36−0.59i −0.36+0.59i −0.54+0i



$$\sigma = 10$$

- Example 1. In this example we tested the matrix

  $$A = \begin{bmatrix} 0 & 10 \\ 0 & 0 \end{bmatrix}$$

  which has exact eigenvalues $\lambda = (0, 0)$ with multiplicity $m = 2$. From Figure we observe that Inverse Iteration method could converge to the reference eigenvalues for both shifts $\sigma = 2$ and $\sigma = 10$. We note that by applying the Power method to this matrix as output eigenvalue we could get only *NaN*.
- Example 2. We recall that reference eigenvalues in this case are
  $\lambda = (12.3246, -11.1644, -0.3246, 1.1644)$.
  In this example we observe nice convergence too, see Figure. For the shift $\sigma = 2$ we could get eigenvalue 1.1644 which is the same as the last reference eigenvalue. This is because shift $\sigma = 2$ is closer to this eigenvalue than to all others. For the shift $\sigma = 10$ algorithm converged to the first reference eigenvalue 12.3246, as expected.
  This test confirms that the Inverse iteration method converges to the eigenvalue which is closest to the shift σ.
- Example 3. Figure shows nice convergence in this case too for both shifts σ. Recall, that Power method does not converged at all, compare results on Figures.
- Example 4.
  From Figure we observe nice convergence to the first eigenvalue of the matrix A for both shifts $\sigma = 2, 10$.

Orthogonal Iteration

Our next improvement will permit us to converge to a ($p > 1$)-dimensional invariant subspace, rather than one eigenvector at a time. It is called *orthogonal iteration* (and sometimes *subspace iteration* or *simultaneous iteration*).

ALGORITHM. Orthogonal iteration: Let Z_0 be an $n \times p$ orthogonal matrix. Then we iterate

```
$i = 0$
*repeat*
   $Y_{i+1} = AZ_i$
   *Factor* $Y_{i+1} = Z_{i+1}R_{i+1}$   (*using Algorithm  QR decomposition*)
                                   ($Z_{i+1}$ *spans an approximate*
                                   *invariant subspace*)
   $i = i + 1$
*until convergence*
```

An informal analysis of the method of Orthogonal iteration

- Assume $|\lambda_p| > |\lambda_{p+1}|$. If $p = 1$, this method and its analysis are identical to the power method.
- When $p > 1$, we write $\mathrm{span}\{Z_{i+1}\} = \mathrm{span}\{Y_{i+1}\} = \mathrm{span}\{AZ_i\}$, so $\mathrm{span}\{Z_i\} = \mathrm{span}\{A^i Z_0\} = \mathrm{span}\{S\Lambda^i S^{-1} Z_0\}$. Note that

$$
\begin{aligned}
S\Lambda^i S^{-1} Z_0 &= S \,\mathrm{diag}(\lambda_1^i, \dots, \lambda_n^i) S^{-1} Z_0 \\
&= \lambda_p^i S \begin{bmatrix} (\lambda_1/\lambda_p)^i & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & (\lambda_n/\lambda_p)^i \end{bmatrix} S^{-1} Z_0 .
\end{aligned}
$$

Since $|\frac{\lambda_j}{\lambda_p}| \geq 1$ for $j \leq p$ and $|\frac{\lambda_j}{\lambda_p}| < 1$ if $j > p$, we get

$$\begin{bmatrix} (\lambda_1/\lambda_p)^i & & \\ & \ddots & \\ & & (\lambda_n/\lambda_p)^i \end{bmatrix} S^{-1} Z_0 = \begin{bmatrix} V_i^{p \times p} \\ W_i^{(n-p)\times p} \end{bmatrix} = X_i,$$

where W_i approaches zero like $(\lambda_{p+1}/\lambda_p)^i$, and V_i does not approach zero. Indeed, if V_0 has full rank (a generalization of the assumption that $\xi_1 \neq 0$), then V_i will have full rank too. Write the matrix of eigenvectors . Then $S = [s_1, \ldots, s_n] \equiv [S_p^{n\times p}, \hat{S}_p^{n\times(n-p)}]$, i.e.$S_p = [s_1, \ldots, s_p]$. Then $S\Lambda^i S^{-1} Z_0 = \lambda_p^i S[\begin{array}{c} V_i \\ W_i \end{array}] = \lambda_p^i (S_p V_i + \hat{S}_p W_i)$. Thus

$$\mathrm{span}(Z_i) = \mathrm{span}(S\Lambda^i S^{-1} Z_0) = \mathrm{span}(S_p V_i + \hat{S}_p W_i) = \mathrm{span}(S_p X_i)$$

converges to $\mathrm{span}(S_p V_i) = \mathrm{span}(S_p)$, the invariant subspace spanned by the first p eigenvectors, as desired. □

- The use of the QR decomposition keeps the vectors spanning $\mathrm{span}\{A^i Z_0\}$ of full rank despite roundoff.
- Note that if we follow only the first $\tilde{p} < p$ columns of Z_i through the iterations of the algorithm, they are *identical* to the columns that we would compute if we had started with only the first $\tilde{p}$ columns of Z_0 instead of p columns. In other words, orthogonal iteration is effectively running the algorithm for $\tilde{p} = 1, 2, \ldots, p$ all at the same time. So if all the eigenvalues have distinct absolute values, the same convergence analysis as before implies that the first $\tilde{p} \leq p$ columns of Z_i converge to $\mathrm{span}\{s_1, \ldots, s_{\tilde{p}}\}$ for any $\tilde{p} \leq p$.
- Thus, we can let $p = n$ and $Z_0 = I$ in the orthogonal iteration algorithm. The next theorem shows that under certain assumptions, we can use orthogonal iteration to compute the Schur form of A.

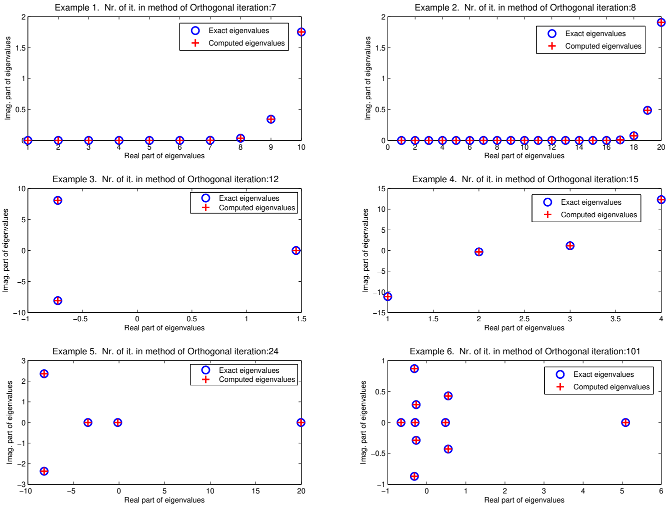
THEOREM. Consider running orthogonal iteration on matrix A with $p = n$ and $Z_0 = I$. If all the eigenvalues of A have distinct absolute values and if all the principal submatrices $S(1:j, 1:j)$ have full rank, then $A_i \equiv Z_i^T A Z_i$ converges to the Schur form of A, i.e., an upper triangular matrix with the eigenvalues on the diagonal. The eigenvalues will appear in decreasing order of absolute value.

Method of Orthogonal Iteration

```
lambda0= inf(n,1);
iter =1;
// here, dim(A)= $n \times n$
Q = eye(n);
// we choose number of iterations here
for k = 1:100
Y = A*Q;
[Q,R] = qr(Y);
// Compute Shur form of A
T=Q'*A*Q;
 //Find eigenvalues from Real Schur block
```

```
computed_lambda = sort(eigs);
computed_lambda = computed_lambda';
if(norm(abs(computed_lambda - lambda0 )) <eps )
break ;
end
lambda0 = computed_lambda ;
iter = iter + 1;
end
```

Performance of Method of Orthogonal Iteration



- Example 1. In this example we tested such called Hilbert matrix of the size 10 × 10 which is a well known example of a badly conditioned matrix. Elements of this matrix are given by $1/(i + j - 1)$, where i, j are indices for the nodes in x and y directions, correspondingly. From Figure we observe that we have obtained all computed eigenvalues of this matrix which coincides with reference eigenvalues already at 7-th iteration.
- Example 2. Here we tested Hilbert matrix of the size 20 × 20. Again, we have computed almost exact eigenvalues of this matrix at 8-th iteration, see Figure.
- Example 3. This is the same as example 3 in the Power method. Figure shows nice convergence to the one real and two complex eigenvalues of the matrix A at 12-th iteration.
- Example 4. This is the same as example 2 in the Power method. Figure shows convergence to the four real eigenvalues of the matrix A at 15-th iteration.
- Example 5. Here we tested the matrix

$$A = \begin{bmatrix} 3 & 7 & 8 & 9 & 12 \\ 5 & -7 & 4 & -7 & 8 \\ 1 & 1 & -1 & 1 & -1 \\ 4 & 3 & 2 & 1 & 7 \\ 9 & 3 & 2 & 5 & 4 \end{bmatrix}$$

  which has three real and two complex reference eigenvalues
  $\lambda = (19.9655, -8.2137 + 2.3623i, -8.2137 - 2.3623i, -3.4043, -0.1337)$.
  From Figure we observe convergence of the all computed eigenvalues to reference eigenvalues at 24-th iteration.
- Example 6. Here we choose the size of the matrix $dim(A) = 10 \times 10$. Elements of this matrix are uniformly distributed pseudorandom numbers on the open interval $(0, 1)$.

QR Iteration

ALGORITHM. QR iteration: Given A_0, we iterate

```
$i = 0$
repeat
    Factor $A_i = Q_i R_i$   (the QR decomposition)
    $A_{i+1} = R_i Q_i$
    $i = i + 1$
until convergence
```

Since $A_{i+1} = R_i Q_i = Q_i^T (Q_i R_i) Q_i = Q_i^T A_i Q_i$, A_{i+1} and A_i are orthogonally similar.
We claim that the A_i computed by QR iteration is identical to the matrix $Z_i^T A Z_i$ implicitly computed by orthogonal iteration.

LEMMA. Let A_i be the matrix computed by Algorithm of QR iteration. Then $A_i = Z_i^T A Z_i$, where Z_i is the matrix computed from orthogonal iteration (Algorithm of Orthogonal iteration) starting with $Z_0 = I$. Thus A_i converges to Schur form if all the eigenvalues have different absolute values.

Proof. We use induction.

- Assume $A_i = Z_i^T A Z_i$. From Algorithm of Orthogonal iteration, we can write $AZ_i = Z_{i+1} R_{i+1}$, where Z_{i+1} is orthogonal and R_{i+1} is upper triangular. Thus, $A = Z_i^T Z_{i+1} R_{i+1}$.
- Then $Z_i^T A Z_i = \underbrace{Z_i^T (Z_{i+1}}_{Q} R_{i+1})$ is the product of an orthogonal matrix $Q = Z_i^T Z_{i+1}$ and an upper triangular matrix $R = R_{i+1} = Z_{i+1}^T A Z_i$ since $AZ_i = Z_{i+1} R_{i+1}$;

- This must be the QR decomposition $A_i = QR$, since the QR decomposition is unique (except for possibly multiplying each column of Q and row of R by -1).
- Then

$$A_{i+1} = Z_{i+1}^T A Z_{i+1} = \underbrace{(Z_{i+1}^T A \mathbf{Z_i})}_{R_{i+1}} (Z_i^T Z_{i+1}) = R_{i+1} \underbrace{(Z_i^T Z_{i+1})}_{Q} = RQ.$$

- This is precisely how the QR iteration maps A_i to A_{i+1}.

ALGORITHM. QR iteration with a shift: Given A_0, we iterate

```
$i = 0$
repeat
    Choose a shift $\sigma_i$ near an eigenvalue of $A$
    Factor $A_i - \sigma_i I = Q_i R_i$ (QR decomposition)
    $A_{i+1} = R_i Q_i + \sigma_i I$
    $i = i + 1$
until convergence
```

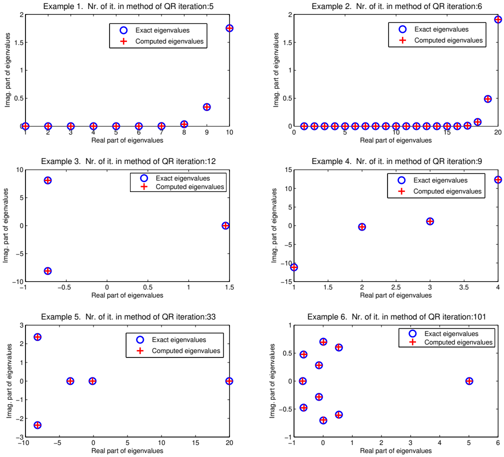
LEMMA. A_i and A_{i+1} are orthogonally similar.

Proof. $A_{i+1} = R_i Q_i + \sigma_i I = Q_i^T Q_i R_i Q_i + \sigma_i Q_i^T Q_i = Q_i^T (Q_i R_i + \sigma_i I) Q_i = Q_i^T A_i Q_i$. $\square$
If R_i is nonsingular, we may also write

$$\begin{aligned} A_{i+1} &= R_i Q_i + \sigma_i I = R_i Q_i R_i R_i^{-1} + \sigma_i R_i R_i^{-1} \\ &= R_i (Q_i R_i + \sigma_i I) R_i^{-1} = R_i A_i R_i^{-1}. \end{aligned}$$

- If σ_i is an exact eigenvalue of A_i then we claim that QR iteration converges in one step: since σ_i is an eigenvalue, $A_i - \sigma_i I$ is singular, so R_i is singular, and so some diagonal entry of R_i must be zero. Suppose $R_i(n,n) = 0$. This implies that the last row of $R_i Q_i$ is 0, so the last row of $A_{i+1} = R_i Q_i + \sigma_i I$ equals $\sigma_i e_n^T$, where e_n is the nth column of the n-by-n identity matrix. In other words, the last row of A_{i+1} is zero except for the eigenvalue σ_i appearing in the (n,n) entry. This means that the algorithm has converged, because A_{i+1} is block upper triangular, with a trailing 1-by-1 block σ_i; the leading $(n-1)$-by-$(n-1)$ block A' is a new, smaller eigenproblem to which QR iteration can be solved without ever modifying σ_i again:
$$A_{i+1} = \begin{bmatrix} A' & a \\ 0 & \sigma_i \end{bmatrix}$$
- When σ_i is not an exact eigenvalue, then we will accept $A_{i+1}(n,n)$ as having converged when the lower left block $A_{i+1}(n, 1:n-1)$ is small enough. Recall from our earlier analysis that we expect $A_{i+1}(n, 1:n-1)$ to shrink by a factor $|\lambda_k - \sigma_i| / \min_{j \neq k} |\lambda_j - \sigma_i|$, where $|\lambda_k - \sigma_i| = \min_j |\lambda_j - \sigma_i|$.
- If σ_i is a very good approximation to eigenvalue λ_k - fast conv.

Examples of Performance of Method of QR Iteration.



Example 1. Nr. of it. in method of QR iteration:5

Example 2. Nr. of it. in method of QR iteration:6

Example 3. Nr. of it. in method of QR iteration:12

Example 4. Nr. of it. in method of QR iteration:9

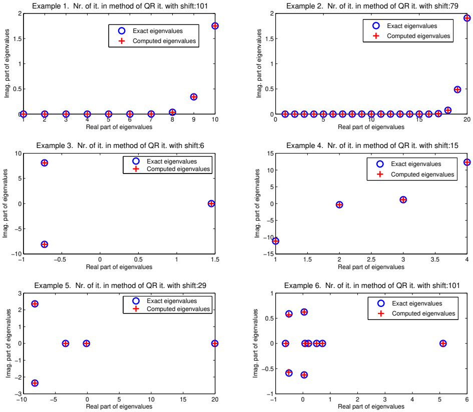
Example 5. Nr. of it. in method of QR iteration:33

Example 6. Nr. of it. in method of QR iteration:101

Tested on matrices

```
A =[5 1, 10, 10; 0, 5, 1, 1; 0,0,5,1; 0,0,0,5];
A=[3,7,8,9;5,-7,4,-7;1,1,-1,-1;4,3,-2,-1];
A =[0 -5 2; 6 0 -12; 1 3 0];
A=[3,7,8,9;5,-7,4,-7;1,-1,1,-1;9,3,2,5];
A=[3,7,8,9,12;5,-7,4,-7,8;1,1,-1,1,-1;4,3,2,1,7;9,3,2,5,4];
A =[1 20; 0 1];
```

QR iteration with shift



Example 1. Nr. of it. in method of QR it. with shift:101

Example 2. Nr. of it. in method of QR it. with shift:79

Example 3. Nr. of it. in method of QR it. with shift:6

Example 4. Nr. of it. in method of QR it. with shift:15

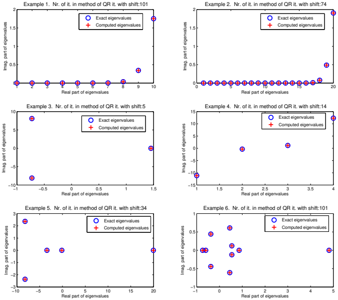
Example 5. Nr. of it. in method of QR it. with shift:29

Example 6. Nr. of it. in method of QR it. with shift:101

$$\sigma = A_{nn}$$

We again test the same matrices as in the method of Orthogonal iteration. Results of the convergence of the algorithm when the shift is chosen as $\sigma = A_{nn}$ at every iteration of this algorithm are presneted in figure above.

QR iteration with shift



Wilkinson's shift

Wilkonson's shift: let shift σ_i is chosen as an eigenvalue of the matrix

$$\begin{bmatrix} a_{n-1,n-1} & a_{n-1,n} \\ a_{n,n-1} & a_{n,n} \end{bmatrix}$$

which is closest to the value $a_{n,n}$ of the matrix A_i.

Hessenberg matrix

A Hessenberg matrix is a special kind of square matrix, one that is "almost" triangular. To be exact, an upper Hessenberg matrix has zero entries below the first subdiagonal, and a lower Hessenberg matrix has zero entries above the first superdiagonal. They are named after Karl Hessenberg. For example:

$$\begin{bmatrix} 1 & 4 & 2 & 3 \\ 3 & 4 & 1 & 7 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

is upper Hessenberg and

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 5 & 2 & 3 & 0 \\ 3 & 4 & 3 & 7 \\ 5 & 6 & 1 & 1 \end{bmatrix}$$

is lower Hessenberg.

Hessenberg Reduction

Given a real matrix A, we seek an orthogonal Q so that QAQ^T is upper Hessenberg. The algorithm is a simple variation on the idea used for the QR decomposition.

EXAMPLE. We illustrate the general pattern of Hessenberg reduction with a 5-by-5 example. Each Q_i below is a 5-by-5 Householder reflection, chosen to zero out entries $i+2$ through n in column i and leaving entries 1 through i unchanged.

1. Choose Q_1 so

$$Q_1 A = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \end{bmatrix} \text{ and } A_1 \equiv Q_1 A Q_1^T = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \end{bmatrix}.$$

Q_1 leaves the first row of $Q_1 A$ unchanged, and Q_1^T leaves the first column of $Q_1 A Q_1^T$ unchanged, including the zeros.

2. Choose Q_2 so

$$Q_2 A_1 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \end{bmatrix} \text{ and } A_2 \equiv Q_2 A_1 Q_2^T = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \end{bmatrix}.$$

Q_2 changes only the last three rows of A_1, and Q_2^T leaves the first two columns of $Q_2 A_1 Q_2^T$ unchanged, including the zeros.

3. Choose Q_3 so

$$Q_3A_2 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{bmatrix} \text{ and } A_3 = Q_3A_2Q_3^T = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{bmatrix}.$$

which is upper Hessenberg. Altogether
$A_3 = Q_3A_2Q_3^T = Q_3Q_2A_1Q_2^TQ_3^T = (Q_3Q_2Q_1)A(Q_1^TQ_2^TQ_3^T) \equiv QAQ^T$.
$\diamond$

The general algorithm for Hessenberg reduction is as follows.
ALGORITHM. Reduction to upper Hessenberg form:

```
$if\ Q\ is\ desired,\ set\ Q = I$
$for\ i = 1 : n-2$
   $u_i = House(A(i+1:n, i))$
   $P_i = I - 2u_iu_i^T \quad /^*\ Q_i = \mathrm{diag}(I^{i\times i}, P_i)^*/$
   $A(i+1:n, i:n) = P_i \cdot A(i+1:n, i:n)$
   $A(1:n, i+1:n) = A(1:n, i+1:n) \cdot P_i$
   $if\ Q\ is\ desired$
      $Q(i+1:n, i:n) = P_i \cdot Q(i+1:n, i:n) \quad /^*\ Q = Q_i \cdot Q\,^*/$
   $end\ if$
$end\ \ for$
```

Example

Part I: computation of upper Hessenberg form when we use rounding.
We want zero out value of entry (3,1) in the following matrix:

$$A = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}.$$

First, we need to find a Hauseholder reflection that transforms the first column of matrix A, vector $\mathbf{x} = (6, -4)^T$, to
$\|\mathbf{x}\| \; \mathrm{e}_1 = (\sqrt{6^2 + (-4)^2}, 0)^T = (2\sqrt{13}, 0)^T$.
Recall algorithm of using Householder reflection for QR decomposition (see lecture 8):

$$\mathbf{u} = \mathbf{x} + \alpha \mathbf{e}_1,$$
$$\alpha = -\mathtt{sign}(x_1)\|x\|,$$
$$\mathbf{v} = \frac{\mathbf{u}}{\|\mathbf{u}\|}.$$

to construct the Householder matrix $Q = I - 2v \; v^T$.

Here,

$$\alpha = -2\sqrt{13} \text{ and } \mathbf{x} = (6, -4)^T$$

Therefore

$$\mathbf{u} = (6 - 2\sqrt{13}, -4)^T \approx (-1.21, -4)^T$$

and $\mathbf{v} = \frac{\mathbf{u}}{\|\mathbf{u}\|} \approx (-0.29, -0.96)^T$, and then

$$\begin{aligned} Q_1 &= I - 2\begin{pmatrix} -0.29 \\ -0.96 \end{pmatrix}\begin{pmatrix} -0.29 & -0.96 \end{pmatrix} \\ &= I - \begin{pmatrix} 0.1682 & 0.5568 \\ 0.5568 & 1.84 \end{pmatrix} \\ &= \begin{pmatrix} 0.8318 & -0.5568 \\ -0.5568 & -0.84 \end{pmatrix}. \end{aligned}$$

Now observe that $Q_1 A$ leaves the first row unchanged:

$$Q_1 A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.8318 & -0.5568 \\ 0 & -0.5568 & -0.84 \end{pmatrix} \cdot \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}$$

$$= \begin{pmatrix} \mathbf{12} & \mathbf{\text{-}51} & \mathbf{4} \\ 7.2180 & 125.5474 & -33.7336 \\ 0.0192 & -113.1456 & 72.3024 \end{pmatrix},$$

and $Q_1 A Q_1^T$ leaves the first column of $Q_1 A$ unchanged:

$$A_1 = Q_1 A Q_1^T = \begin{pmatrix} \mathbf{12} & -44.6490 & 25.0368 \\ \mathbf{7.2180} & 123.2132 & -41.5686 \\ \mathbf{0.0192} & -134.3725 & 2.2655 \end{pmatrix}.$$

Example

Part II: computation of upper Hessenberg form without rounding. We want zero out value of entry (3,1) in the following matrix:

$$A = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}.$$

First, we need to find a Hauseholder reflection that transforms the first column of matrix A, vector $\mathbf{x} = (6, -4)^T$, to
$\|\mathbf{x}\| \, \mathrm{e}_1 = (\sqrt{6^2 + (-4)^2}, 0)^T = (2\sqrt{13}, 0)^T$.
Recall algorithm of using Householder reflection for QR decomposition (see lecture 8):

$$\begin{aligned} \mathbf{u} &= \mathbf{x} + \alpha \mathbf{e}_1, \\ \alpha &= -\mathtt{sign}(x_1)\|x\|, \\ \mathbf{v} &= \frac{\mathbf{u}}{\|\mathbf{u}\|}. \end{aligned}$$

to construct the Householder matrix $Q = I - 2v \; v^T$.

Here,

$$\alpha = -2\sqrt{13} \text{ and } \mathbf{x} = (6, -4)^T$$

Therefore

$$\mathbf{u} = (6 - 2\sqrt{13}, -4)^T \approx (-1.2111, -4)^T$$

and $\mathbf{v} = \frac{\mathbf{u}}{\|\mathbf{u}\|} \approx (-0.2898, -0.9571)^T$, and then

$$\begin{aligned} Q_1 &= I - 2\begin{pmatrix} -0.2898 \\ -0.9571 \end{pmatrix}\begin{pmatrix} -0.2898 & -0.9571 \end{pmatrix} \\ &= \begin{pmatrix} 0.8321 & -0.5547 \\ -0.5547 & -0.8321 \end{pmatrix}. \end{aligned}$$

Now observe that $Q_1 A$ leaves the first row unchanged:

$$Q_1 A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.8321 & -0.5547 \\ 0 & -0.5547 & -0.8321 \end{pmatrix} \cdot \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}$$

$$= \begin{pmatrix} \mathbf{12} & \mathbf{\text{-}51} & \mathbf{4} \\ 7.2111 & 125.6396 & -33.8367 \\ 0.0 & -112.6041 & 71.8337 \end{pmatrix},$$

and $Q_1 A Q_1^T$ leaves the first column of $Q_1 A$ unchanged:

$$A_1 = Q_1 A Q_1^T = \begin{pmatrix} \mathbf{12} & -44.6534 & 24.9615 \\ \mathbf{7.2111} & 123.3077 & -41.5385 \\ \mathbf{0.0} & -133.5385 & 2.6923 \end{pmatrix}.$$

PROPOSITION. Hessenberg form is preserved by QR iteration.

Proof. It is easy to confirm that the QR decomposition of an upper Hessenberg matrix like $A_i - \sigma I$ yields an upper Hessenberg Q (since the jth column of Q is a linear combination of the leading j columns of $A_i - \sigma I$). Then it is easy to confirm that RQ remains upper Hessenberg and adding does not change this. $\square$

DEFINITION. An upper Hessenberg matrix H is unreduced if all subdiagonals are nonzero.

Reduction to a lower Hessenberg form

Consider example in Matlab: program Hessenberg.m:

```
n =4;
A=[3,7,8,9;5,-7,4,-7;1,-1,1,-1;9,3,2,5];
A=A’;
Q=eye(n);
for i=1:n-2
x= A(i+1:n,i)
u=x;
u(1) = u(1)+ sign(x(1))*norm(x);
u=u/norm(u);
P= eye(n-i) - 2*(u*u’) ;
A(i +1:n ,i:n) =P*A(i +1:n,i:n) ;
A(1:n,i +1:n)=A(1:n,i +1: n)*P;
Q(i+1:n,i:n) = P*Q(i+1:n,i:n);
end
Q’*A’*Q
```

Tridiagonal and Bidiagonal Reduction

If A is symmetric, the Hessenberg reduction process leaves A symmetric at each step, so zeros are created in symmetric positions. This means we need work on only half the matrix, reducing the operation count to $\frac{4}{3}n^3 + O(n^2)$ or $\frac{8}{3}n^3 + O(n^2)$ to form $Q_{n-1}, \ldots, Q_1$ as well. We call this algorithm *tridiagonal reduction*.

We recall that the eigenvalues of the symmetric matrix $A^T A$ are the squares of the singular values of A. Our eventual SVD algorithm will use this fact, so we would like to find a form for A which implies that $A^T A$ is tridiagonal. We will choose A to be *upper bidiagonal*, or nonzero only on the diagonal and first superdiagonal. Thus, we want to compute orthogonal matrices Q and V such that QAV is bidiagonal. The algorithm, called *bidiagonal reduction*, is very similar to Hessenberg and tridiagonal reduction.

EXAMPLE. Here is a 4-by-4 example of bidiagonal reduction, which illustrates the general pattern:

1. Choose Q_1 so

$$Q_1A = \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{bmatrix} \text{ and } V_1 \; so \, A_1 \equiv Q_1AV_1 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{bmatrix}.$$

Q_1 is a Householder reflection, and V_1 is a Householder reflection that leaves the first column of Q_1A unchanged.

2. Choose Q_2 so

$$Q_2 A_1 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{bmatrix} \text{ and } V_2 \ so \ A_2 \equiv Q_2 A_1 V_2 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{bmatrix}.$$

Q_2 is a Householder reflection that leaves the first row of A_1 unchanged. V_2 is a Householder reflection that leaves the first two columns of $Q_2 A_1$ unchanged.

3. Choose Q_3 so

$$Q_3 A_2 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{bmatrix} \text{ and } V_3 = I \ so \ A_3 = Q_3 A_2.$$

Q_3 is a Householder reflection that leaves the first two rows of A_2 unchanged. We have obtained: $A_3 = Q_3 A_2 V_3 = Q_3 Q_2 A_1 V_2 V_3 = (Q_3 Q_2 Q_1) A (V_1 V_2 V_3) = QAV$ is upper diagonal matrix. $\diamond$

In general, if A is n-by-n, then we get orthogonal matrices $Q = Q_{n-1} \cdots Q_1$ and $V = V_1 \cdots V_{n-2}$ such that

$$QAV = A'$$

is upper bidiagonal.
Multiply both sides of $A' = QAV$ by $A'^T = (QAV)^T = V^T A^T Q^T$ to get:

$$A'^T A' = V^T A^T Q^T QAV = V^T A^T AV,$$

so $A'^T A'$ has the same eigenvalues as $A^T A$; i.e., A' has the same singular values as A.

Example of tridiagonal reduction using Householder transformation

$A = \begin{bmatrix} 5 & 4 & 3 \\ 4 & 6 & 1 \\ 3 & 1 & 7 \end{bmatrix}$ using Householder transformation (see alg. in Lecture 8) we make following steps:

- Step1 . First compute α as

$$\alpha = -\operatorname{sgn}(a_{21})\sqrt{\sum_{j=2}^{n} a_{j1}^2} = -\sqrt{(a_{21}^2 + a_{31}^2)} = -\sqrt{4^2 + 3^2} = -5.$$

- Step 2. Using α we find r as

$$r = \sqrt{\frac{1}{2}(\alpha^2 - a_{21}\alpha)} = \sqrt{\frac{1}{2}((-5)^2 - 4 \cdot (-5))} = \frac{3\sqrt{5}}{\sqrt{2}}.$$

- Step 3. Then we compute components of vector v:

$$\begin{aligned} v_1 &= 0, \\ v_2 &= \frac{a_{21} - \alpha}{2r} = \frac{3\sqrt{2}}{2\sqrt{5}}, \\ v_3 &= \frac{a_{31}}{2r} = \frac{\sqrt{2}}{2\sqrt{5}}. \end{aligned}$$

  and we have

$$v^{(1)} = \begin{bmatrix} 0 \\ \frac{3\sqrt{2}}{2\sqrt{5}} \\ \frac{\sqrt{2}}{2\sqrt{5}} \end{bmatrix},$$

- Step 4 . Then compute matrix P^1

$$P^1 = I - 2v^{(1)}(v^{(1)})^T$$

to get $\mathrm{P}^1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -4/5 & -3/5 \\ 0 & -3/5 & 4/5 \end{bmatrix}$

- Step 5.

  After that we can obtain tridiagonal matrix $A^{(1)}$ as

$$A^{(1)} = P^1 A P^1 = \begin{bmatrix} 5 & -5 & 0 \\ -5 & 7.32 & -0.76 \\ 0 & -0.76 & 5.68. \end{bmatrix}$$

Example

We will construct tridiagonal matrix from the matrix

$$A = \begin{bmatrix} 5 & 4 & 3 \\ 4 & 6 & 1 \\ 3 & 1 & 7 \end{bmatrix}.$$

using Hauseholder transformations.

Example

To perform tridiagonal reduction for the matrix A we use Householder transformation in following steps:

- Choose $x = (4,3)^T$ and compute

$$u = x + \alpha e_1,$$

where $\alpha = -sign(4) \cdot ||x||, \|x\| = \sqrt{25} = 5$, and thus $\alpha = -5$.

- Construct $u = x + \alpha e_1 = (4,3)^T - (5,0)^T = (-1,3)^T$.
- Construct

$$v = \frac{u}{\|u\|}$$

with $\|u\| = \sqrt{10}$.
Therefore $v = (-1/\sqrt{10}, 3/\sqrt{10})^T$.

- Compute

$$Q' = I - 2vv^T = \begin{pmatrix} 0.8 & 0.6 \\ 0.6 & -0.8 \end{pmatrix}.$$

- Construct the matrix of the Householder transformation as:

$$Q_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.8 & 0.6 \\ 0 & 0.6 & -0.8 \end{pmatrix}$$

- Then compute

$$A_1 = Q_1 A = \begin{pmatrix} 5 & 4 & 3 \\ 5 & 5.4 & 5 \\ 0 & 2.8 & -5 \end{pmatrix}.$$

  such that Q_1 leaves the first row of $Q_1 A$ unchanged.

- Choose new vector $x = (4,3)^T$ for A_1^T and compute

$$u = x + \alpha e_1,$$

  where $\alpha = -sign(4) \cdot ||x||, \|x\| = \sqrt{25} = 5$, and thus $\alpha = -5$.

- Construct $u = x + \alpha e_1 = (4,3)^T - (5,0)^T = (-1,3)^T$.
- Construct
$$v = \frac{u}{\|u\|}$$
with $\|u\| = \sqrt{10}$.
Therefore $v = (-1/\sqrt{10}, 3/\sqrt{10})^T$.
- Compute
$$V' = I - 2vv^T = \begin{pmatrix} 0.8 & 0.6 \\ 0.6 & -0.8 \end{pmatrix}.$$

- Construct the second matrix of the Householder transformation V_1 as:

$$V_1 = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & V' \end{array}\right]$$

to get

$$V_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.8 & 0.6 \\ 0 & 0.6 & -0.8 \end{pmatrix}$$

and then compute

$$Q_1 A V_1 = \begin{pmatrix} 5 & 5 & 0 \\ 5 & 7.32 & -0.76 \\ 0 & -0.76 & 5.68 \end{pmatrix}.$$

such that V_1 leaves the first column of A_1 unchanged.

Example of tridiagonalization using Given's rotation

To make tridiagonal matrix from the matrix $A = \begin{bmatrix} -10 & 3 & 4 \\ 3 & 5 & 1 \\ 4 & 1 & 9 \end{bmatrix}$ using Given's rotation we have to zero out $(3,1)$ and $(1,3)$ elements of the matrix A. Thus we use the Given's rotation $R(2,3,\theta)$ such that

$$\mathbf{G_1} = R(2,3,\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix}$$

We compute

$$\mathbf{G_1} \cdot A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix} \cdot \begin{bmatrix} -10 & 3 & 4 \\ 3 & 5 & 1 \\ 4 & 1 & 9 \end{bmatrix} = \begin{bmatrix} -10 & 3 & 4 \\ 3c - 4s & 5c - s & c - 9s \\ 3s + 4c & 5s + c & s + 9c \end{bmatrix}$$

Element $(3,1)$ of the matrix will be zero if $3s + 4c = 0$. This is true when $c = 4/5$ and $s = -3/5$. To compute c, s we have used formulas:

$$\begin{aligned} r &= \sqrt{a^2 + b^2} = \sqrt{3^2 + 4^2} = 5, \\ c &= \frac{a}{r} = 3/5, \\ s &= \frac{-b}{r} = -4/5. \end{aligned}$$

Next, to get tridiagonal matrix we have to do :

$$A_1 = \mathbf{G_1} A \mathbf{G_1}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & 4/5 \\ 0 & -4/5 & 3/5 \end{bmatrix} \cdot \begin{bmatrix} -10 & 3 & 4 \\ 3 & 5 & 1 \\ 4 & 1 & 9 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & -4/5 \\ 0 & 4/5 & 3/5 \end{bmatrix} = \begin{bmatrix} -10 & 5 & 0 \\ 5 & 8.52 & 1.64 \\ 0 & 1.64 & 5.48 \end{bmatrix} \tag{1}$$

Example to make upper triangular matrix using Given's rotation

Consider previous example and obtained matrix A_1:

$$\mathrm{A}_1 = \begin{bmatrix} -10 & 5 & 0 \\ 5 & 8.52 & 1.64 \\ 0 & 1.64 & 5.48 \end{bmatrix}$$

Now we want to zero out elements (2,1) and (3,2) in order to get upper triangular matrix. To zero out element (2,1) we compute c, s from the known $a = -10$ and $b = 5$ as

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

to get:

$$\begin{aligned} r &= \sqrt{a^2 + b^2} = \sqrt{-(10)^2 + 5^2} \approx 11.18, \\ c &= \frac{a}{r} \approx -0.894 \\ s &= \frac{-b}{r} \approx -0.4472. \end{aligned}$$

Then the Given's matrix will be

$$\mathbf{G} = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

or

$$\mathbf{G} = \begin{bmatrix} -0.89445 & 0.44722 & 0 \\ -0.44722 & -0.89445 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Finally, we obtain the matrix:

$$\begin{aligned} A_2 = \mathbf{G}A_1 &= \begin{bmatrix} -0.89445 & 0.44722 & 0 \\ -0.44722 & -0.89445 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -10 & 5 & 0 \\ 5 & 8.52 & 1.64 \\ 0 & 1.64 & 5.48 \end{bmatrix} \\ &= \begin{bmatrix} 11.1806 & -0.6619356 & 0.7334408 \\ -0.00005 & -9.856814 & -1.466898 \\ 0 & 1.64 & 5.4 \end{bmatrix} \end{aligned} \tag{2}$$

Now to zero out element (3,2) we compute c, s from the known $a = -9.856814$ and $b = 1.64$ to get:

$$\begin{aligned} r &= \sqrt{a^2 + b^2} = \sqrt{(-9.856814)^2 + (1.64)^2} \approx 9.9923, \\ c &= \frac{a}{r} \approx -0.98644, \\ s &= \frac{-b}{r} \approx -0.16413. \end{aligned}$$

The next Given's matrix will be

$$\mathbf{G}' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix}$$

or

$$\mathbf{G}' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.98644 & 0.16413 \\ 0 & -0.16413 & -0.98644 \end{bmatrix}$$

Finally, we obtain the upper triangular matrix as $A_3 = G' \cdot A_2$:

$$A_3 = \mathbf{G}' A_2 = \begin{bmatrix} 11.1806 & -0.6619 & 0.7334 \\ 0.0000 & 9.9923 & 2.3333 \\ 0.0000 & 0.0000 & -5.0860 \end{bmatrix} \tag{3}$$

Regular Matrix Pencils and Weierstrass Canonical Form

The standard eigenvalue problem asks for which scalars z the matrix $A - zI$ is singular; these scalars are the eigenvalues. This notion generalizes in several important ways.

DEFINITION. $A - \lambda B$, where A and B are m-by-n matrices, is called a *matrix pencil*, or just a *pencil*. Here is λ an indeterminate, not a particular, numerical value.

DEFINITION. If A and B are square and $\det(A - \lambda B)$ is not identically zero (or when there exists at least one $\lambda : \det(A - \lambda B) \neq 0$), the pencil $A - \lambda B$ is called *regular*. Otherwise it is called *singular*. When $A - \lambda B$ is regular, $p(\lambda) \equiv \det(A - \lambda B)$ is called the *characteristic polynomial* of $A - \lambda B$ and the eigenvalues of $A - \lambda B$ are defined to be
(1) the roots of $p(\lambda)$,
(2) ∞ (with multiplicity $n - \deg(p)$) if $\deg(p) < n$.

EXAMPLE. Let

$$A - \lambda B = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} - \lambda \begin{bmatrix} 2 & & \\ & 0 & \\ & & 1 \end{bmatrix}.$$

Then $p(\lambda) = \det(A - \lambda B) = (1 - 2\lambda) \cdot (1 - 0\lambda) \cdot (0 - \lambda) = (1 - 2\lambda)(-\lambda)$, so the eigenvalues are $\lambda = \frac{1}{2}$, 0 and ∞. $\diamond$

PROPOSITION. Let $A - \lambda B$ be regular (when there exists at least one $\lambda : \det(A - \lambda B) \neq 0$).

- If B is nonsingular, all eigenvalues of $A - \lambda B$ are finite and the same as the eigenvalues of AB^{-1} or $B^{-1}A$.
- If B is singular, $A - \lambda B$ has eigenvalue ∞ with multiplicity $n - \mathrm{rank}(B)$.
- If A is nonsingular, the eigenvalues of $A - \lambda B$ are the same as the reciprocals of the eigenvalues (or $\frac{1}{\lambda_i}$) of $A^{-1}B$ or BA^{-1}, where a zero eigenvalue of $A^{-1}B$ corresponds to an infinite eigenvalue of $A - \lambda B$.

Proof.

- If B is nonsingular and λ' is an eigenvalue, then $0 = \det(A - \lambda' B) = \det(AB^{-1} - \lambda' I) = \det(B^{-1}A - \lambda' I)$ so λ' is also an eigenvalue of AB^{-1} and $B^{-1}A$.
- If B is singular, then take $p(\lambda) = \det(A - \lambda B)$, write the SVD of B as $B = U\Sigma V^T$, and substitute to get

$$p(\lambda) = \det(A - \lambda U\Sigma V^T) = \det(U(U^T AV - \lambda\Sigma)V^T)$$

$$= \pm \det(U^T AV - \lambda\Sigma).$$

Since $\mathrm{rank}(B) = \mathrm{rank}(\Sigma)$, only $\mathrm{rank}(B)$ λ's appear in $U^T AV - \lambda\Sigma$, so the degree of the polynomial $\det(U^T AV - \lambda\Sigma)$ is $\mathrm{rank}(B)$.

- If A is nonsingular, $\det(A - \lambda B) = 0$ and $\det(A(I - \lambda A^{-1}B)) = 0$ if and only if $\det(I - \lambda A^{-1}B) = 0$ or $\det(I - \lambda BA^{-1}) = 0$.

$$\det(I - \lambda A^{-1}B) = 0 \rightarrow \det(\frac{1}{\lambda} I - A^{-1}B) = 0$$

This equality can hold only if $\lambda \neq 0$ and $1/\lambda$ is an eigenvalue of $A^{-1}B$ and BA^{-1}. □

DEFINITION. Let P_L and P_R be nonsingular matrices. Then pencils $A - \lambda B$ and $P_L A P_R - \lambda P_L B P_R$ are called *equivalent*.

PROPOSITION. The equivalent regular pencils $A - \lambda B$ and $P_L A P_R - \lambda P_L B P_R$ have the same eigenvalues. The vector x is a right eigenvector of $A - \lambda B$ if and only if $P_R^{-1} x$ is a right eigenvector of $P_L A P_R - \lambda P_L B P_R$. The vector y is a left eigenvector of $A - \lambda B$ if and only if $(P_L^*)^{-1} y$ is a left eigenvector of $P_L A P_R - \lambda P_L B P_R$.

Proof.

- $\det(A - \lambda B) = 0$ if and only if $\det(P_L (A - \lambda B) P_R) = 0$.
- $(A - \lambda B)x = 0$ if and only if $P_L (A - \lambda B) P_R P_R^{-1} x = 0$.
- $(A - \lambda B)^* y = 0$ if and only if $P_R^* (A - \lambda B)^* P_L^* (P_L^*)^{-1} y = 0$. $\square$

THEOREM. *Weierstrass canonical form*. Let $A - \lambda B$ be regular. Then there are nonsingular P_L and P_R such that

$$P_L(A - \lambda B)P_R = \mathrm{diag}(J_{n_1}(\lambda_1) - \lambda I_{n_1}, \ldots, J_{n_k}(\lambda_{n_k}) - \lambda I_{n_k}, N_{m_1}, \ldots, N_{m_r}),$$

where $J_{n_i}(\lambda_i)$ is an n_i-by-n_i Jordan block with eigenvalue λ_i,

$$J_{n_i}(\lambda_i) = \begin{bmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix},$$

and N_{m_i} is a "Jordan block for $\lambda = \infty$ with multiplicity m_i,"

$$N_{m_i} = \begin{bmatrix} 1 & \lambda & & \\ & 1 & \ddots & \\ & & \ddots & \lambda \\ & & & 1 \end{bmatrix} = I_{m_i} - \lambda J_{m_i}(0).$$

For a proof, see [F. Gantmacher. The Theory of Matrices, vol. II (translation). Chelsea, New York, 1959].