Section 6.10.3

Finding Eigenvalues

$$Ax = \lambda x$$

$$A\underline{x} - \lambda\underline{x} = 0$$

$$\underline{A}x - \lambda\underline{I}x = 0$$

$$(A - \lambda\underline{I})x = 0$$

λ shift of $A \rightarrow$ singular C (diag)

$x \neq 0$

$C \neq 0$

zero as a linear comb of C & X

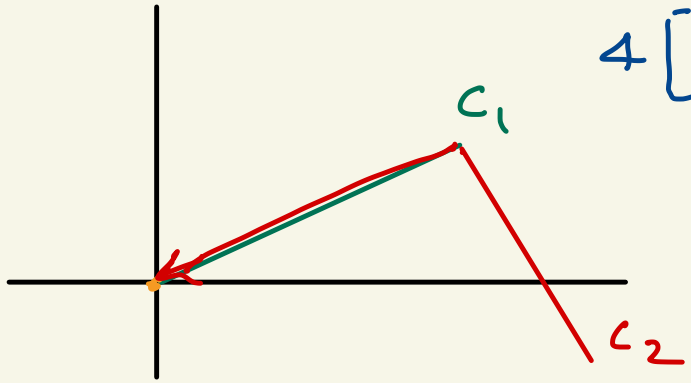
$$\begin{bmatrix} \lambda x_1 \\ \lambda x_2 \\ \vdots \end{bmatrix}$$

C must be Singular

Column view - section 6.4.5

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad Cx$$

$$x_1\begin{bmatrix} c_1 \end{bmatrix} + x_2\begin{bmatrix} c_2 \end{bmatrix} = 0$$



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

$\det(C) = 0$, other QR

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \Rightarrow C = \begin{bmatrix} a-\lambda & b \\ c & d-\lambda \end{bmatrix}$$

- λ - real
- λ - complex
- λ - repeati... values
  degenerate case

Example

$$A = \begin{bmatrix} -9 & -5 \\ 10 & 6 \end{bmatrix} \qquad C = \begin{bmatrix} -9-\lambda & -5 \\ 10 & 6-\lambda \end{bmatrix}$$

$tr = -3$ $\qquad \det(A)$

$|C| = 0$

$(-9-\lambda)(6-\lambda) + 50 = 0$

$\lambda_1 = -4$

$\lambda_2 = 1$

$\lambda^2 + 3\lambda - 4 = 0$

$(\lambda+4)(\lambda-1) = 0$

Eigenvalues by QR method

$A1 = A$

loop:
- $A1 \to$ orth Q $\quad Q^{-1} = Q^T$
- $R = Q^T A1 \to A1 = QR$
- $A1 = RQ$ $\quad RQ$ similar to A

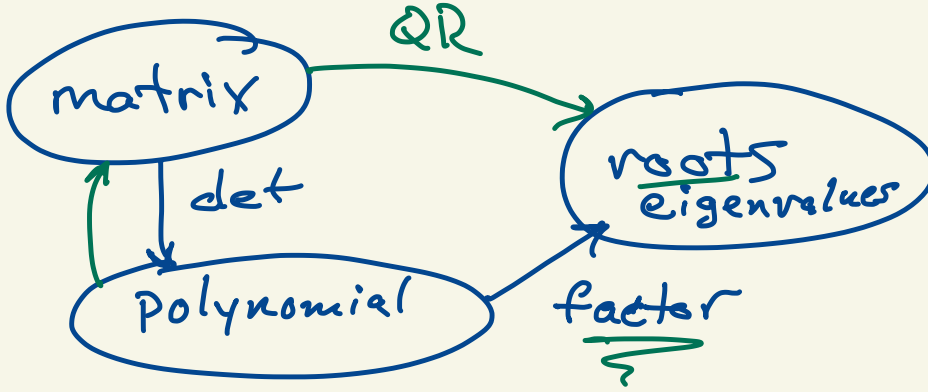
$Q \to I$

$A1 = R$ - upper triangular

$A1$ stable

eigenvalue on diagonal

Roots of a polynomial



Section 6.10.3

Finding Eigenvectors

$$A = \begin{bmatrix} -9 & -5 \\ 10 & 6 \end{bmatrix}$$

$\lambda_1 = -4$ $\qquad C = A - I\lambda$ $\qquad \lambda_2 = 1$

$CX = 0$

$$C_1 = \begin{bmatrix} -5 & -5 \\ 10 & 10 \end{bmatrix} \qquad C_2 = \begin{bmatrix} -10 & -5 \\ 10 & 5 \end{bmatrix}$$

$$\left[\begin{array}{cc|c} -5 & -5 & 0 \\ 10 & 10 & 0 \end{array}\right]$$

$$\left[\begin{array}{cc|c} -5 & -5 & 0 \\ 0 & 0 & 0 \end{array}\right]$$

$$-5x_a - 5x_b = 0$$

$$x_a + x_b = 0$$

$$x_a = -x_b$$

$$x_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$-10x_a - 5x_b = 0$$

$$2x_a + x_b = 0$$

$$x_a = 1$$

$$x_b = -2$$

$$x_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

6.10.3 Finding Eigenvectors and Eigenvalues

We will begin with the equation for eigenvectors and eigenvalues and insert an identity matrix so that we have a matrix multiplied by a vector on both sides of the equality.

$$\boldsymbol{A}\,\boldsymbol{x} = \lambda\,\boldsymbol{x}$$
$$\boldsymbol{A}\,\boldsymbol{x} = \lambda\,\boldsymbol{I}\,\boldsymbol{x}$$

Then we rearrange the equation to find what is called the characteristic eigenvalue equation.

$$\boldsymbol{A}\,\boldsymbol{x} - \lambda\boldsymbol{I}\,\boldsymbol{x} = \boldsymbol{0}$$

$$\boxed{(\boldsymbol{A} - \lambda\boldsymbol{I})\,\boldsymbol{x} = \boldsymbol{0}}$$

The case where $\boldsymbol{x} = \boldsymbol{0}$ is a trivial solution that is not of general interest to us. Eigenvectors are defined to be nonzero vectors. Thus, the only solution exists when the columns of matrix $\boldsymbol{A} - \lambda\,\boldsymbol{I}$ form a linear combination with $\boldsymbol{x}$ yielding zero. This linear dependence of the columns of the characteristic equation means that it is singular – having a zero determinant.

Finding Eigenvalues

Note: We will use the determinant here on small matrices because it keeps things simple. But as noted in *Determinant*, calculating determinants is computationally slow. So it is not used for large matrices. See *Eigenvalue Computation in MATLAB* for more about other ways to find the eigenvalues of a matrix.

The n scalar eigenvalues, $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, can be viewed as the shift of the matrix's main diagonal that will make the matrix singular. Eigenvalues are found by subtracting λ along the main diagonal and finding the set of λ for which the determinant is zero. The following equation is referred to as the characteristic equation for the matrix $\boldsymbol{A}$.

$$det(\boldsymbol{A} - \lambda\,\boldsymbol{I}) = 0$$

$$\begin{vmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{vmatrix} = 0$$

Why singular requirement?

You may be wondering why it is required that $(\boldsymbol{A} - \lambda\,\boldsymbol{I})$ be a singular matrix. Let us call this matrix $\boldsymbol{C}$, and the columns of $\boldsymbol{C}$ will be labeled as $\boldsymbol{c_i}$. For $\boldsymbol{C}\,\boldsymbol{x} = \boldsymbol{0}$, it must be that either:

1. $\boldsymbol{c_1} = \boldsymbol{c_2} = \ldots = \boldsymbol{c_n} = \mathbf{0}$, and hence $\boldsymbol{C} = \mathbf{0}$, which is not the case we are looking for.
2. $\boldsymbol{x} = \mathbf{0}$, which is also not what we are looking for.
3. The columns of $\boldsymbol{C}$ are linearly dependent such that such that $\boldsymbol{C}\,\boldsymbol{x} = \mathbf{0}$. It is the linear dependent property of the columns of $\boldsymbol{C}$ that yields $\boldsymbol{C}$ to be a singular matrix.

The determinant yields a degree-n polynomial, which can be factored to find the eigenvalue roots.

$$\boldsymbol{A} = \begin{bmatrix} 2 & 2 \\ 2 & -1 \end{bmatrix}$$

$$\begin{vmatrix} (2-\lambda) & 2 \\ 2 & (-1-\lambda) \end{vmatrix} = 0$$

$$\begin{aligned} (2-\lambda)(-1-\lambda) - 4 &= 0 \\ \lambda^2 - \lambda - 6 &= 0 \\ (\lambda - 3)(\lambda + 2) &= 0 \end{aligned}$$

$$\lambda_1 = -2, \qquad \lambda_2 = 3$$

For the generalized 2-by-2 matrix, the coefficient of the λ term in the quadratic equation is the negative of the sum of the matrix diagonal (the trace), while the constant term is the determinant of the matrix.

$$\begin{vmatrix} (a-\lambda) & b \\ c & (d-\lambda) \end{vmatrix} = 0$$

$$\begin{aligned} (a-\lambda)(d-\lambda) - bc &= 0 \\ \lambda^2 - (a+d)\lambda + (ad - bc) &= 0 \end{aligned}$$

Note: Two other properties of eigenvalues may be useful for finding and verifying them.

1. The *trace* of $\boldsymbol{A}$ is equal to the sum of the eigenvalues. The trace of a matrix is the sum of the values along the forward diagonal. For a 2-by-2 matrix: $a_{1,1} + a_{2,2} = \lambda_1 + \lambda_2$.
2. The determinant of $\boldsymbol{A}$ is equal to the product of all of the eigenvalues.

Note: Eigenvalues, and hence eigenvectors, often have complex numbers. In some cases, algorithms will force real eigenvalues by using symmetric matrices, which have only real eigenvalues. In some applications, when taking products and sums of eigenvalues and eigenvectors the imaginary parts will cancel leaving only real numbers. In other cases, the presence of complex eigenvalues implys oscillation in a system.

Eigenvalues are not always unique – the same number may be repeated in the set of eigenvalues. Such cases are called *degenerate* because the matrix does not have *Linearly Independent Eigenvectors* and thus can not factored using the *Diagonalization* procedure, which is required for some application algorithms.

Roots of a Polynomial by Eigenvalues

See also:

If analytic roots are needed, see *Solve*.

To find numeric the roots of a non-polynomial function, see the *Fzero* function.

As we saw above, finding the eigenvalues of a matrix is equivalent to finding the roots of the determinant of the characteristic equation. But as noted above, the algorithm that MATLAB uses to find eigenvalues neither calculates a determinate nor finds the roots of a polynomial. Instead it uses the faster algorithm described in *Eigenvalue Computation in MATLAB*.

Rather than finding polynomial roots to calculate eigenvalues, finding the roots of a polynomial is instead an application of the eigenvalue algorithm. This is how the MATLAB function `roots()` finds the roots of a polynomial. To take advantage of the eigenvalue algorithm, a matrix is cleverly found that has eigenvalues equivalent to the roots of the polynomial.

An example should illustrate how this works. Consider the following polynomial equation.

$$f(x) = x^3 - 4x^2 + x + 6$$

The argument passed to the `roots()` function is a row vector containing the coefficients of the polynomial.

```
>> r = roots([1 -4 1 6])
r =
    3.0000
    2.0000
   -1.0000
```

The `poly()` function is the inverse of `roots()` :

```
>> poly(r)
ans =
    1.0000   -4.0000    1.0000    6.0000
```

The algorithm that `roots()` uses is short, but quite clever.

```
>> n = 3;                  % degree of the polynomial
>> p = [1 -4 1 6];         % coefficents
>> A = diag(ones(n-1,1),-1)
```

```
A =
     0      0      0
     1      0      0
     0      1      0
>> A(1,:) = -p(2:n+1)./p(1)
A =
     4     -1     -6
     1      0      0
     0      1      0
>> r = eig(A)
r =
     3.0000
     2.0000
    -1.0000
```

The determinant of the characteristic equation of $\boldsymbol{A}$ has the same coefficients and thus the same roots as $f(x)$.

$$\begin{vmatrix} (4-\lambda) & -1 & -6 \\ 1 & (-\lambda) & 0 \\ 0 & 1 & (-\lambda) \end{vmatrix} = 0$$

$$\lambda^3 - 4\lambda^2 + \lambda + 6 = 0$$

Finding Eigenvectors

The eigenvectors, $\boldsymbol{x_i}$ (one per eigenvalue) lie in the same line as $\boldsymbol{A}\,\boldsymbol{x_i}$: $\boldsymbol{A}\,\boldsymbol{x_i} = \lambda_i\,\boldsymbol{x_i}$. Thus,

$$(\boldsymbol{A} - \lambda_i\,\boldsymbol{I})\boldsymbol{x_i} = 0.$$

The solution to the above equation is called the *null* solution because we are looking for a vector, $\boldsymbol{x_i}$, that sets the equation to zero. Given the matrix A and the eigenvalues, the eigenvectors can be found with elimination or with MATLAB's `null` function. See *Null Space*.

Now, we continue the previous example with elimination to find the eigenvectors.

$$\lambda_1 = -2:$$

$$\left[\begin{array}{cc|c} 4 & 2 & 0 \\ 2 & 1 & 0 \end{array}\right]$$

Add $-1/2$ of row 1 to row 2 and then divide row 1 by 4:

$$\left[\begin{array}{cc|c} 1 & 1/2 & 0 \\ 0 & 0 & 0 \end{array}\right]$$

$$x_{1a} + 1/2\,x_{1b} = 0$$

The second row of zeros occurs because it is a singular matrix. This means that we have a *free variable*, so we can set one variable to any desired value (usually 1).

$$\boldsymbol{x_1} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$\lambda_2 = 3\text{:}$$

$$\left[\begin{array}{cc|c} -1 & 2 & 0 \\ 2 & -4 & 0 \end{array}\right]$$

Add 2 of row 1 to row 2 and then divide row 1 by -1:

$$\left[\begin{array}{cc|c} 1 & -2 & 0 \\ 0 & 0 & 0 \end{array}\right]$$

$$x_{2a} - 2\,x_{2b} = 0$$

$$\boldsymbol{x_2} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Two things to note about the eigenvectors returned from `null` : First, MATLAB always normalizes the vector (unit length). Secondly, eigenvectors may always be multiplied by a scalar. It is the direction of the eigenvector that matters, not the magnitude. The definition of eigenvectors has the same eigenvectors on both sides of the equality, making them invariant to scale.

$$\begin{aligned} \boldsymbol{A}\,\boldsymbol{x_i} &= \lambda_i\,\boldsymbol{x_i} \\ c\,\boldsymbol{A}\,\boldsymbol{x_i} &= c\,\lambda_i\,\boldsymbol{x_i} \end{aligned}$$

```
>> A = [2 2;2 -1];
>> l1 = -2; l2 = 3;       % the eigenvalues
>> N1 = A - l1*eye(2)
N1 =
     4     2
     2     1
>> N2 = A - l2*eye(2)
N2 =
    -1     2
     2    -4
>> x1 = null(N1)       % normalized eigenvector
x1 =
   -0.4472
    0.8944
>> x1 = x1/x1(1)       % scaled eigenvector
x1 =
     1
    -2
>> x2 = null(N2)
x2 =
```

```
   -0.8944
   -0.4472
>> x2 = x2/X2(2)
X2 =
    2
    1
>> A*X1
ans =
    -2
     4
>> l1*X1
ans =
    -2
     4
>> A*X2
ans =
     6
     3
>> l2*X2
ans =
     6
     3
```

MATLAB has a function called `eig` that calculates both the eigenvalues and eigenvectors of a matrix. The results are returned as matrices, which are useful in some applications. The eigenvectors are the columns of `X` . As with the `null` function, the `eig` function always normalizes the eigenvectors (unit length). The eigenvalues are on the diagonal of `L` , the MATLAB function `diag(L)` will return the eigenvalues from the diagonal as a row vector.

```
>> A = [2 2;2 -1];
>> [X, L] = eig(A)
X =
    0.4472   -0.8944
   -0.8944   -0.4472
L =
   -2     0
    0     3
```

Passing the matrix as symbolic math variable will show data closer to what we found when we solved the problem by hand calculations.

```
>> [X, L] = eig(sym(A))
X =
[ -1/2, 2]
[    1, 1]
L =
[ -2, 0]
```

```
[  0, 3]
```