Eigen Problems and Diagonalization Using Matlab

An *Eigenproblem* for a given $n \times n$ matrix $\mathbf{A}$ requires finding the set of vectors, $\mathbf{x}$, and the scalar numbers λ such that

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}.$$

In other words, we want the vectors which, when operated on by A, are simply multiples of the orginal vector. *Geometrically, the eigenvectors of* $\mathbf{A}$ *are those vectors,* $\mathbf{x}$*, such that* $\mathbf{A}\mathbf{x}$ *lies in the same (or exactly opposite) direction as* $\mathbf{x}$*.* $\mathbf{A}$ *simply multiplies its "own" (in German "eigen") vectors. Multiplication by* $\mathbf{A}$ *changes the direction of all other vectors.*

Matlab allows for easy computation of the eigenvalues and eigenvectors of any square matrix. For example, consider the following Matlab commands:

```
> A = [-3 1 -3; -8 3 -6; 2 -1 2]
A =
    -3     1    -3
    -8     3    -6
     2    -1     2
```

To find the eigenvalues of `A` we could use the fact that the eigenvalues, λ satisfy the characteristic equation given by

$$det(\mathbf{A} - \lambda \mathbf{I}) = 0.$$

Matlab has an easy way of entering this. Simply use the `poly` command:

```
> p =  poly(A)
p =
      1            -2            -1            2
```

The result says that the characteristic polynomial is:

$$p(\lambda) = \lambda^3 - 2\lambda^2 - \lambda + 2 = 0$$

This can be factored into:

$$(\lambda - 1)(\lambda + 1)(\lambda - 2)$$

Which gives us the eigenvalues of $\mathbf{A}$ directly.

If you don't see the factorization easily, Matlab is equipped to solve the characteristic equation for you using the `roots()` command,

```
> eigs = roots(p)
eigs =
      2
      1
     -1
```

which gives the zeros (eigenvalues) of the polynomial directly.

Now we can solve for the eigenvectors of A. For each eigenvalue, we must solve

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

for the eigenvector $\mathbf{x}$. In Matlab the $n \times n$ identity matrix is given by `eye(n)`. To find the eigenvector associated with $\lambda = 2$ we could use:

```
> A1 = A - eigs(1)*eye(3) %Note: Use eigs(1) instead of '2' for accuracy

A1 =
     -5            1            -3
     -8            1            -6
      2           -1             0

> rref(A1)

ans =
      1            0             1
      0            1             2
      0            0             0
```

This gives us $\mathbf{x} = \alpha \begin{pmatrix} -1 \\ -2 \\ 1 \end{pmatrix}$ The same procedure could be used for the other two eigenvectors. Try it!

Seems complicated? Once again Matlab has a fast way of accomplishing the same task. The `eig()` command finds the eigenvalues and eigenvectors of a matrix directly. The output is given in two matrices. The first is a matrix whose columns contain the eigenvectors while the second is a diagonal matrix containing the eigenvalues.

```
> [V,E] = eig(A)

V =
    881/2158     1292/2889     -780/1351
    881/1079     2584/2889     -780/1351
   -881/2158         *          780/1351

E =
      2            0             0
      0           -1             0
      0            0             1
```

If the output looks a bit strange, its because matlab normalizes the eigenvectors so that $(V_i \cdot V_i) = 1$. For instance we can make the eigenvector corresponding to $\lambda = 2$ look like that given in our previous result:

```
> V1 = V(:,1)

V1 =
   881/2158
   881/1079
  -881/2158

> V1 = V1/V1(1)

V1 =
      1
      2
     -1
```

Diagonalization: Matlab's eigenvector output format is exactly what we need to diagonalize the input matrix, namely a transformation matrix $P = V$ whose columns are the eigenvectors of A. To see the utility of diagonalization, consider the following set of nonhomogeneous, coupled ODEs

$$\mathbf{x}' = A\mathbf{x} + \mathbf{F}$$

where $\mathbf{x}$ is the unknown vector of solutions and A is matrix of constant coefficients.

To solve the coupled set of equations via diagonalization, we first transform to new variables, $\mathbf{y}$ using the transformation matrix V:

$$\mathbf{x} = V\mathbf{y}$$

$$\mathbf{x}' = V\mathbf{y}' = A\mathbf{x} + \mathbf{F} = AV\mathbf{y} + \mathbf{F}$$

In terms of the new variable, $\mathbf{y}$,

$$\mathbf{y}' = V^{-1}AV\mathbf{y} + V^{-1}\mathbf{F}$$

Since $V^{-1}AV$ is just the diagonal matrix of eigenvalues of A, this last set is completely UNCOUPLED and easy to solve.

As an example, consider the coupled set of 1st order ODEs equivalent to the single 2nd order equation:

$$y'' + 3y' - 4y = 3e^{2t}$$

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 4 & -3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 3e^{2t} \end{pmatrix}$$

Lets solve the homogeneous 1st order problem using Matlab to do the matrix calculations.

First set up the matrix A and find its transformation matrix.

```
> A = [0 1;4 -3]
A =
      0             1
      4            -3

> [v,d] = eig(A)
v =
```

```
   985/1393    -528/2177
   985/1393    2112/2177

d =
      1            0
      0           -4

> v(:,1) = v(:,1)/v(1,1)  %Note: Can multiply an eigenvector by a scalar
v =                              Here we rescale the eigenvectors to make
      1        -528/2177         them 'prettier'
      1        2112/2177

> v(:,2) = v(:,2)/v(1,2)
v =
      1            1
      1           -4
```

We will also need the inverse, V^{-1}:

```
> inv(v)
ans =
     4/5          1/5
     1/5         -1/5
```

Now we have enough information to solve the problem. The uncoupled equations become:

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -4 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + \begin{pmatrix} 4/5 & 1/5 \\ 1/5 & -1/5 \end{pmatrix} \begin{pmatrix} 0 \\ 3e^{2t} \end{pmatrix}$$

Or, individually,

$$\begin{aligned} y_1' - y_1 - 3e^{2t}/5 &= 0 \\ y_2' + 4y_1 + 3e^{2t}/5 &= 0 \end{aligned}$$

The solution to these linear, 1st order ODEs are:

$$\begin{aligned} (e^{-t}y_1)' &= 3e^t/5 \\ y_1 &= c_1e^t + 3e^{2t}/5 \end{aligned}$$

and

$$\begin{aligned} (e^{4t}y_2)' &= 3e^{6t}/5 \\ y_2 &= c_2e^{-4t} - 3e^{2t}/30 \end{aligned}$$

To find the solution $\mathbf{x}$, simply transform back:

$$\mathbf{x} = V\mathbf{y} = \begin{pmatrix} 1 & 1 \\ 1 & -4 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

$$\mathbf{x} = \begin{pmatrix} c_1e^t + 3e^{2t}/5 + c_2e^{-4t} - 3e^{2t}/30 \\ c_1e^t + 3e^{2t}/5 - 4c_2e^{-4t} + 12e^{2t}/30 \end{pmatrix}$$

$$\mathbf{x} = \begin{pmatrix} c_1e^t + c_2e^{-4t} + e^{2t}/2 \\ c_1e^t - 4c_2e^{-4t} + e^{2t} \end{pmatrix}$$

Matrix Powers by Diagonalization: The work required to find the n^{th} power of a matrix is greatly reduced using diagonalization. As we showed in class,

$$A^k = VD^kV^{-1}$$

where V is the transformation matrix of A and D is the diagonal matrix of eigenvalues of A. Therefore D^n is simply the diagonal matrix containing λ^k on the diagonal. For example, consider the following matrix:

```
 A = [ 1 3 4; 3 -1 2; 4  2 2]

A =
     1     3     4
     3    -1     2
     4     2     2
```

The computationally fast way of calculating A^{10} is to use diagonalizaton.

```
> [V,D] = eig(A)

V =
    0.7040   -0.3182    0.6349
   -0.6521   -0.6437    0.4005
   -0.2812    0.6959    0.6607


D =
   -3.3764         0         0
         0   -1.6791         0
         0         0    7.0555

> A10 = V*D^10*inv(V)
A10 =

  1.0e+008 *
    1.2330    0.7763    1.2819
    0.7763    0.4911    0.8093
    1.2819    0.8093    1.3347
```

We can check by direct calculation:

```
> A^10
ans =
   123304096    77633408   128193568
    77633408    49109984    80925664
   128193568    80925664   133474944
```

Which is exactly the same result. Note: Matlab probably performed the direct calculation using diagonalization anyway!