Project 5 Help

The famous *Cayley-HamiltonTheorem* states that any matrix is a solution of its own characteristic polynomial. Recall that the characteritic polynomial of a matrix, M, is the polynomial that gives you the eigenvalues when set equal to zero:

$$\det(M - \lambda \cdot Id) = 0.$$

To explore this theorem, let's consider the following matrix:

```
M = {{1, 2, 3, 4}, {1, 1, 1, 1}, {0, -1, 2, 1}, {-1, 4, -2, 1}};
MatrixForm[M]
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \\ 0 & -1 & 2 & 1 \\ -1 & 4 & -2 & 1 \end{pmatrix}$$

Notice that you list the matrix entries as a list of lists in *Mathematica*. To get the characteristic polynomial, we can use the following command:

```
CharacteristicPolynomial[M, λ]
```

$$8 - 21\,\lambda + 10\,\lambda^2 - 5\,\lambda^3 + \lambda^4$$

To test the validity of the Cayley-Hamilton Theorem, we need to substitute the matrix M (using matrix multiplication) into the characteristic polynomial and verify that we get the zero matrix as a result. To do so, we use the MatrixPower command to multiply the matrix by itself. To display the final matrix in a nice form, we use the MatrixForm command. Notice that the constant term needs to come with the Identity Matrix so that everything can add together appropriately. The argument to the IdentityMatrix function is the size of the matrix.

```
MatrixForm[8 * IdentityMatrix[4] - 21 * MatrixPower[M, 1] +
  10 * MatrixPower[M, 2] - 5 * MatrixPower[M, 3] + MatrixPower[M, 4]]
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Notice that we get the zero matrix just as the Cayley-Hamilton Theorem predicted!

For part 2) of the project, we can use *Mathematica* to get the eigenvalues of any matrix. Unfortunately, our original matrix has fairly nasty eigenvalues. So, we will switch to

```
M = {{0, 0, 1, 0}, {0, 0, 0, 1}, {-5, 2, 0, 0}, {2, -2, 0, 0}};
MatrixForm[M]
Eigenvalues[M]
```

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -5 & 2 & 0 & 0 \\ 2 & -2 & 0 & 0 \end{pmatrix}$$

$$\left\{i\sqrt{6}, -i\sqrt{6}, i, -i\right\}$$

As we can see, this matrix has four imaginary eigenvalues! We will discuss how to find the corresponding eigenvectors in class, but for now we can ask *Mathematica* to find them for us:

```
Eigensystem[M]
```

$$\left\{\left\{i\sqrt{6}, -i\sqrt{6}, i, -i\right\},\right.$$
$$\left.\left\{\left\{i\sqrt{\frac{2}{3}}, -\frac{i}{\sqrt{6}}, -2, 1\right\}, \left\{-i\sqrt{\frac{2}{3}}, \frac{i}{\sqrt{6}}, -2, 1\right\}, \{-i, -2i, 1, 2\}, \{i, 2i, 1, 2\}\right\}\right\}$$

This output is a little harder to read. Note that the first list is just the eigenvalues again. The second list is a list of corresponding eigenvectors listed in the order of the eigenvalue they go with. So the first eigenvector $\left\{i\sqrt{\frac{2}{3}}, -\frac{i}{\sqrt{6}}, -2, 1\right\}$ goes with the eigenvalue $i\sqrt{6}$ and so on. Notice that since our matrix consists of all real numbers, the eigenvalues come in conjugate pairs. In addition, the eigenvectors also come in conjugate pairs! This will make finding them by hand a lot easier!

For part 3) of the project, we want to explore functions of a matrix. We will numerically compute the cosine of the original matrix M used above.

```
M = {{1, 2, 3, 4}, {1, 1, 1, 1}, {0, -1, 2, 1}, {-1, 4, -2, 1}};
MatrixForm[M]
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \\ 0 & -1 & 2 & 1 \\ -1 & 4 & -2 & 1 \end{pmatrix}$$

Since we enter a matrix as a list of lists, we have to be careful when asking *Mathematica* to perform matrix operations - especially matrix multiplication. It is possible to multiply two lists of the same length in *Mathematica*, and the result will be a list of the same size where the corresponding elements in the list have been multiplied. This is certainly not what we want to happen when thinking of these lists as matrices! For example, if we multiply M by itself, we get

```
M * M
```

```
{{1, 4, 9, 16}, {1, 1, 1, 1}, {0, 1, 4, 1}, {1, 16, 4, 1}}
```

On the other hand, we can multiply M by itself using the command MatrixPower:

```
MatrixForm[MatrixPower[M, 2]]
```

$$\begin{pmatrix} -1 & 17 & 3 & 13 \\ 1 & 6 & 4 & 7 \\ -2 & 1 & 1 & 2 \\ 2 & 8 & -5 & -1 \end{pmatrix}$$

Notice that the result of multiplying M by itself as a list is very different than multiplying M by itself as a matrix! We will only need to take powers of a single matrix here, but to specify that two matrices need to be multiplied together in *Mathematica*, enter both matrices as a list of rows and use Dot (an literal period '.') to specify matrix multiplication:

```
MatrixForm[M.M]
```

$$\begin{pmatrix} -1 & 17 & 3 & 13 \\ 1 & 6 & 4 & 7 \\ -2 & 1 & 1 & 2 \\ 2 & 8 & -5 & -1 \end{pmatrix}$$

Mathematica's extensive documentation can give you details for other matrix manipulations (including finding eigenvalues and eigenvectors). For this lab, we only need a few matrix commands.

To get a numerical approximation to the cosine of M, we define a function that will add up 100 terms of the power series for cosine (assuming a matrix argument). Since we need the first term to be the 4x4 identity matrix, we begin with the IdentityMatrix[4] command which produces this matrix. The rest of the powers are computed by MatrixPower[ ].

```
CosM[Mat_, n_] :=
 N[IdentityMatrix[4] + Sum[((-1)^j / (2 * j)!) * MatrixPower[Mat, 2 * j], {j, 1, n}]]
```

Note that we have enclosed the summation within an N[...] command. This will ensure that *Mathematica* returns a decimal approximation to the actual matrix. If you leave off this command, *Mathematica* will return the exact result of adding up n terms of the series (which in our case will be a matrix of particularly ugly rational numbers). Getting a numerical approximation for cos(*M*) is now particularly easy:

```
MatrixForm[CosM[M, 100]]
```

$$\begin{pmatrix} 2.70871 & -3.33225 & -1.85364 & -4.33553 \\ -0.298871 & 1.29437 & -2.40891 & -2.12639 \\ 1.24719 & -0.788815 & -0.0480486 & -1.79974 \\ -0.535138 & -2.57334 & 4.14332 & 3.82524 \end{pmatrix}$$

By adding an argument to the end of the N[...] command, we can ask *Mathematica* for more decimal places if need be. Check out the documentation of the N command to see the details.