LINEAR ALGEBRA IN PYTHON WITH THE SYMPY LIBRARY

There are unfortunately a number of incompatibilities between different versions of Python in how they handle linear algebra. The notes below refer to the version which you can run in your browser by visiting `http://live.sympy.org`. If you install or update Python and sympy on your own PC, then the instructions should work there as well. However, older versions may require different syntax.

You can enter the matrices

$$A = \begin{pmatrix} 5 & 6 & 7 \\ 4 & 3 & 2 \end{pmatrix} \qquad B = \begin{pmatrix} 9 & 5 \\ 5 & 9 \end{pmatrix}$$

like this:

```
A = Matrix([[5,6,7],[4,3,2]])
B = Matrix([[9,5],[5,9]])
```

You can now do some calculations:

- Enter `B*A` to calculate BA.
- If you enter `A*B` then you will get an error message, because AB is not defined. The error message is long and complicated, but the last line says `ShapeError: Matrices size mismatch`, which should not be too hard to understand.
- Enter `B**2` to calculate B^2, or `B**-1` to calculate the inverse matrix B^{-1}.
- Enter `transpose(A)` or `A.T` to find the transpose A^T.
- Enter `det(B)` or `B.det()` to find the determinant of B. The brackets are necessary here. If you enter `B.det` without the brackets, you will get

  ```
   boundmethodMutableDenseMatrix.detofMatrix([[9, 5],[5, 9]])
  ```

  which is not very helpful. In general, if you get a result starting with `boundmethod...`, it probably indicates that you left out a pair of brackets.
- Enter `eye(3)` for the 3×3 identity matrix.
- Enter `A.row(0)` to get the first row of A, or `A.row(1)` to get the second row. Python's indexing always starts with 0, so many things need to be shifted by 1 compared to how they are in the notes or in Maple.
- Similarly, you can enter `A.col(0)` or `A.col(1)` or `A.col(2)` to extract one of the columns of A.
- To get the top right entry in A (which is called A_{13} in the notes), enter `A[0,2]`.
- In many places, it is convenient to give names to the columns of a matrix. For example, we might take u_1, u_2 and u_3 to be the columns of the matrix A above. To do this in Python you can type

  ```
  u = [A.col(i) for i in range(3)]
  ```

  However, this gives the usual shift in indexing, so the columns are `u[0]`, `u[1]` and `u[2]`.
- To find the reduced row echelon form of A, enter `A.rref()[0]`. If you just enter `A.rref()` instead you will get a list of two things, the first of which is the RREF of A, and the second of which is the list of pivot columns. If you just enter `A.rref` then you will get a result like `boundmethod...` which you cannot use. You might like to define a more convenient syntax by entering

  ```
  def RREF(M):
      return M.rref()[0]
  ```

  You can then enter `RREF(A)` rather than `A.rref()[0]`.
- If you are going to use the RREF of A in further calculations, then you probably want to give it a name, like this:

  ```
  A1 = A.rref()[0]
  ```

  Note, however, that when you do this Python will not immediately print the result; you need to enter `A1` on a separate line if you want to see the RREF matrix.
- To find and factor the characteristic polynomial of B, enter

```
p = B.charpoly(t)
p
factor(p)
```

Python will print the characteristic polynomial as

$$\mathrm{PurePoly}(t^2 - 4t + 3, t, domain = \mathbb{Z})$$

rather than just $t^2 - 4t + 3$, but that should not cause a problem. It is best to stick with t as the name of the variable here. In particular, you should not try to use the symbol λ for this or anything else, because it would clash with a completely different use of that symbol in Python; we will not take the space to explain the background here.

- To find the eigenvalues and eigenvectors of B, enter `B.eigenvects()`. The result is

$$\left(4, 1, \left[\left[\begin{array}{c} -1 \\ 1 \end{array}\right]\right]\right), \quad \left(14, 1, \left[\left[\begin{array}{c} 1 \\ 1 \end{array}\right]\right]\right).$$

The numbers 4 and 14 are the eigenvalues, and the vectors $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ are the corresponding eigenvectors, and the ones in between the eigenvalues and the eigenvectors can be ignored. In cases where the matrix has repeated eigenvalues the result will need a bit more interpretation, but that will be discussed later.

Python with SymPy on your own machine

If you have installed Python (from `http://www.python.org/download/`) on your own PC then you can add SymPy by following the instructions at `http://sympy.org`. I recommend that you install version 2.7.5 of Python and version 1.4 of SymPy. After that you can start Python under IDLE as usual, and enter

```
from __future__ import division
from sympy import *
t = Symbol('t')
```

The first line effectively tells Python that you want to use exact fractions rather than decimal approximations everywhere. The second line says that you want to use SymPy. The third line tells SymPy to treat t as an abstract symbol, which we need when working with characteristic polynomials. All of these things are done automatically if you use `http://live.sympy.org`, but on your own PC you need to do them yourself.