Math 317: Computer Lab 4

Instructions

- This assignment must be completed and submitted on Blackboard by 2pm on

  Friday, April 29.

- Save your worksheet and name it Lab4.sws.
- **Submit your Lab4.sws worksheet file on Blackboard**.

1 Linear Transformations in Sage

In this lab we will learn how to create a linear transformation in Sage, and try out some of the many operations we can perform on such objects. As in the previous lab assignment, rather than work over the real or complex numbers as our base number field, we will work (at least initially) over $\mathbb{Q}$, the field of rational numbers, as this gives us better insight into the theory we have been studying.[1]

Let $V = \mathbb{Q}^3$ and $W = \mathbb{Q}^4$, and consider the linear transformation $T : V \to W$ defined as follows:

$$T(\mathbf{x}) = T\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -x_1 + 2x_3 \\ x_1 + 3x_2 + 7x_3 \\ x_1 + x_2 + x_3 \\ 2x_1 + 3x_2 + 5x_3 \end{bmatrix}. \tag{1}$$

To create this linear transformation in Sage, we will create a "symbolic" function. But this requires some symbolic variables which we will call `x1`, `x2` and `x3` in this case. This is done as follows:

```
x1, x2, x3 = var('x1, x2, x3')                                    (eq1)
```

Now we give a name to the transformation mentioned in (1), and define it (symbolically) as follows:

```
Tsymb(x1, x2, x3) = [-x1+2*x3, x1+3*x2+7*x3, x1+x2+x3, 2*x1+3*x2+5*x3]    (eq2)
```

[1] Working over $\mathbb{Q}$ makes it possible for Sage to perform exact arithmetic, which avoids the round-off errors associated with finite precision arithmetic over the real or complex numbers. This makes our demonstration of the theory much cleaner and clearer. In applications, we don't always have the luxury of working over $\mathbb{Q}$. Typically the results we observe when applying linear algebra "in the real-world" are not as elegant as the results presented in a first course in linear algebra. Nonetheless, the theoretical principles that we highlight (using exact arithmetic) in this lab assignment apply equally well in practical situations, even those that require real and complex numerical computation.

Exercise 1. Define and experiment with `Tsymb`, using the following steps to guide you:

(a) Create a new Sage worksheet if you haven't already, and name it something like `Lab4`. Enter the above expressions `(eq1)` and `(eq2)` in a worksheet cell and then evaluate the cell.

(b) Experiment with the `Tsymb` object. For example, you could evaluate the function at triples of rational numbers—say, `Tsymb(3, -1, 2/3)`. Alternatively, you could try something a little more exotic, like calculating the partial derivative of `Tsymb` with respect to `x3`. To see what methods are available to the `Tsymb` object, try the following: In a new cell of your worksheet, type `Tsymb.`, making sure to include the trailing dot. Then, with the cursor immediately after the dot, hit the Tab key. You should see a drop-down list of all the functions that can be applied to the object `Tsymb`. Some of the entries in the list begin with the string `Tsymb.is_` followed by the name of some property. Such commands allow us to interrogate `Tsymb` to find out whether it has the given property.

(c) Find out whether `Tsymb` is *immutable*. If you don't know what "immutable" means, enter the expression `Tsymb.is_immutable?` and evaluate it. By appending a question mark to a command, we are asking Sage to display the documentation page for that command. *The question mark is a very useful feature of Sage that you should keep in mind for future reference.*

(d) Another command that should have appeared in the drop-down list when you typed `Tsymb.`Tab is called `derivative`. You probably have an idea of what this function does, but just to be sure, evaluate `Tsymb.derivative?` in a worksheet cell, then compute the partial derivative of `Tsymb` with respect to `x3`.

1.1 Constructing a linear transformation

At this point, our `Tsymb` object is a vector of mathematical expressions that involve symbolic variables. We want to tell Sage to convert it into a linear transformation, as this will prompt Sage to make available a whole new set of functions that we can apply to `Tsymb`. We will use the `linear_transformation()` constructor, which requires us to carefully specify the domain and codomain, as vector spaces over the rational number field $\mathbb{Q}$.

```
V= QQ^3
W = QQ^4
T = linear_transformation(V, W, Tsymb)
```

Now, we can try out our newly constructed linear transformation `T` by evaluating it on some input vector, say, $\mathbf{x} = (3, -1, 2)$,

```
x = vector(QQ, [3, -1, 2])
T(x)
```

Enter these commands in a Sage worksheet cell and make sure you get the answer `(1, 14, 4, 13)`.

2 Representations of Linear Transformations

Recall the linear transformation from Homework Exercise 4.3.7,

$$T(\mathbf{x}) = T\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -x_1 + 2x_2 + x_3 \\ x_2 + 3x_3 \\ x_1 - x_2 + x_3 \end{bmatrix}. \tag{2}$$

Exercise 2. Use Sage to find the matrix representation of T by following these steps.

(a) In your Sage worksheet, construct the linear transformation described by (2). (Use the what you learned in Part 1.)

```
V = QQ^3
Tsymb = ...           (fill in the rest)
T = linear_transformation ...
                      (be careful to specify the right domain and codomain)
```

(b) By now you have a lot of experience with linear transformations, and with just a glance at Equation (2) you can surely write down a matrix that represents T. (Do it!) Of course, Sage has a command that will do this for you. Try evaluating `T.matrix()` in your Sage worksheet. Is the output of this command what you expected? If not, maybe there is an option that makes the `matrix()` function produce a more satisfying output. Try to find this option and use it. (Remember, you can get help with the `matrix` function by entering `T.matrix?`.)

(c) In this part, we will use Sage to find the matrix representation $[T]_{\mathcal{B}}$ of T with respect to the basis $\mathcal{B} = \left\{ \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$. We will do this in two different ways and then check that the results we get are the same.

Method 1: Define the three basis vectors in $\mathcal{B}$ using the `vector` constructor: `b1 = vector(QQ, [1,0,-1])`, etc. Construct the change-of-basis matrix P, and then compute $P^{-1}TP$. One way to construct P is with the command `matrix(QQ, [b1,b2,b3]).transpose()`. Don't forget to apply the `matrix` method to T before using it in matrix computations, as in `TB1 = P.inverse() * T.matrix(side="right") * P`

Method 2: Let `V = QQ^3` and define `VB = V.subspace_with_basis([b1, b2, b3])`. Use the `linear_transformation` constructor, as you did above, but this time replace each occurrence of `V` with `VB`, since `VB` is the vector space `QQ^3` endowed with the new basis. That is, compute

```
TB2 = linear_transformation(VB, VB, Tsymb)
```

Finally, check that the two methods give the same result:

```
TB1 == TB.matrix(side="right")      # (this should return True)
```

3 Diagonalization: finding the best basis possible

Given a linear transformation $T : V \to W$, we may want to find a basis that is in some sense optimal for representing T. Specifically, we often seek a matrix representation of T that is as close to diagonal as possible.

Let A be the standard basis representation of T, that is, $A = [T]_{\mathcal{E}}$. We learned that a necessary and sufficient condition for diagonalizability of A is that the algebraic multiplicity of each eigenvalue of A must be the same as the geometric multiplicity. We can interpret this in a few equivalent ways:

- A is similar to a diagonal matrix, $D = P^{-1}AP$.
- There is a change-of-basis that diagonalizes A.
- There is a basis $\mathcal{B}$ with respect to which the $\mathcal{B}$-basis representation $[T]_{\mathcal{B}}$ is a diagonal matrix.

These statements all say the same thing. Importantly, when these conditions hold, the basis $\mathcal{B}$ consists of eigenvectors of A, and D has the eigenvalues of A along the main diagonal.

Exercise 3. Example 5 on page 273 of our textbook involves two matrices,

$$A = \begin{bmatrix} -1 & 4 & 2 \\ -1 & 3 & 1 \\ -1 & 2 & 2 \end{bmatrix} \qquad \text{and} \qquad B = \begin{bmatrix} 0 & 3 & 1 \\ -1 & 3 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Use Sage to compute the characteristic polynomial and the eigenvalues/vectors of A. Check that A is diagonalizable and perform the similarity transformation that gives the diagonalization. Use the steps below as a guide. Later, come back and do the same B. What goes wrong?

(a) Construct the matrix A with the command `A = matrix(QQ, [[-1,4,2], [-1,3,1], [-1,2,2]])`, then use compute its characteristic polynomial.

```
p = A.characteristic_polynomial()
p.factor()
```

(b) Compute the roots of the characteristic polynomial with `p.roots()`. The output should be a list of two ordered pairs. Can you make sense of this output?

(c) You computed the eigenvalues of A indirectly in the last step. Now compute them directly with the Sage command `A.eigenvalues()`. Compare the output to the result from part (b).

(d) You found eigenvalues in the last two steps; name them `ev1` and `ev2`. Compute the eigenvectors of A using the `right_kernel` command that we learned in Lab 3. Recall that, for each eigenvalue λ, the eigenspace is the null space of $A - \lambda I$. Here's the first one:

```
I = identity_matrix(3)
E1 = (A-ev1*I).right_kernel(basis='pivot')
```

Finally, compute the change-of-basis matrix P whose columns are the eigenvectors of A and check that $P^{-1}AP$ gives a diagonal matrix with the eigenvalues of A along the main diagonal. Here's one way to do it:

```
b12 = E1.matrix().transpose()    # we want the eigenvectors as columns,
b3 = E2.matrix().transpose()     # not rows, so we apply transpose()
P = b12.augment(b3); print P
```

After checking that the columns of `P` are exactly the eigenvectors you found above, you can compute the diagonalization of A as usual: `P.inverse()*A*P`.

4 Notes for further study.

- (Exercise 2) To experiment more with change-of-basis, see if you can come up with an example of the more general version of the change-of-basis theorem—Theorem 4.2, page 231 of our textbook—and try it out in Sage. That is, given a linear transformation $T : V \to W$, and two bases $\mathcal{V}$, $\mathcal{V}'$ for V, and two bases $\mathcal{W}$, $\mathcal{W}'$ for W, use the formula $[T]_{\mathcal{V}',\mathcal{W}'} = Q^{-1}[T]_{\mathcal{V},\mathcal{W}}P$ to compute the $\mathcal{V}', \mathcal{W}'$-basis representation of T. You will have to generalize one of the two methods described in Exercise 2 (c) above, but by now it should be clear how to do this. If it isn't, then check out the examples on page 127 (Section MR) and page 138 (Section MRCB) of the document fcla-2.22-sage-4.7.1-preview.pdf, which is available for download from the Resources tab of our Piazza site.

- (Exercise 3) Of course, Sage has its own built-in commands for computing eigenvectors and eigenspaces. Here are some commands you could try in the context of Exercise 3 part (d): `A.eigenvectors_right()`, `A.eigenspaces_right()`. Compare the results of these commands with the results produced by the `right_kernel` that we used above.

  See also Section EE (p. 95) of the document fcla-2.22-sage-4.7.1-preview.pdf for more details about computing eigenvalues in Sage. In particular, we want to work over a field that contains the roots of the characteristic polynomial. (In Exercise 3, the eigenvalues were real numbers so this was not an issue.)