Sage and Linear Algebra Worksheet

FCLA Section SLT

Robert Beezer
Department of Mathematics and Computer Science
University of Puget Sound

Spring 2017

1 Surjective Linear Transformations

Two carefully-crafted linear transformations: `T` is surjective, `S` is not.

```
A = matrix(QQ, [[2, 2, 5, -2], [2, 3, 1, -4], [-3, -4, -4,
    5]])
T = linear_transformation(QQ^4, QQ^3, A, side='right')
```

```
T.is_surjective()
```

The range is known in Sage as the "image." For a surjective linear transformation, it will be the entire codomain. Note that the image is a vector space.

```
T.image()
```

```
T.image() == T.codomain()
```

```
B = matrix(QQ, [[1, -2, 0, 3], [3, -5, 1, 7], [-1, 4, 2,
    -7]])
S = linear_transformation(QQ^4, QQ^3, B, side='right')
```

```
S.is_surjective()
```

```
IM = S.image()
IM
```

```
IM == S.codomain()
```

2 Pre-Images

Exercise 1. We can create inputs associated with any output. First, we make an arbitrary output, but make sure it really is an output, as a linear combination of a basis of the image (see basis above). We print the two vectors in the opposite of what we would consider the "normal" order.

```
bas = IM.basis()
out = ()*bas[0] + ()*bas[1]
inp = S.preimage_representative(out)
out, inp
```

A check on our work.

```
S(inp)
```

Exercise 2. We can make other inputs, using the kernel.

Any value of `new_inp` is in the preimage of `out`, and every element of the preimage can be built this way. Notice the role the kernel plays, much like in the worksheet about injective linear transformations.

```
K = S.kernel()
K
```

```
z = K.random_element()
new_inp = inp + z
new_inp, S(new_inp)
```

Exercise 3. Elements outside the range (image) will have empty preimages. We mildly "wreck" an element of the range.

With two initial entries determined by the zeros and ones in the basis vectors, the third entry must be determined, so we can "twiddle" it just a bit to obtain a vector of the codomain that lies outside the range. We will ask Sage for a pre-image representative anyway and see what happens.

```
in_range = ()*bas[0] + ()*bas[1]
in_range
```

```
outside_range = vector(QQ, [ , , ])
S.preimage_representative(outside_range)
```