Sage and Linear Algebra Worksheet

FCLA Section B

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

Spring 2019

1 Bases

Five "random" vectors, each with 4 entries, collected into a set `S`.

```
v1 = vector(QQ, [-4, -2,  3, -11])
v2 = vector(QQ, [-2,  7,  3,   9])
v3 = vector(QQ, [ 6, -4, -7,   5])
v4 = vector(QQ, [-1,  0,  3,  -4])
v5 = vector(QQ, [-4,  5, -5,  11])
S = [v1, v2, v3, v4, v5]
```

Consider the subspace spanned by these five vectors. We will make these vectors the *rows* of a matrix and row-reduce to see a basis for the space (subspace, or row space, take your pick). This is an application of Theorem BRS.

```
A = matrix(S)
A
```

```
A.rref()
```

Sage does this semi-automatically, tossing zero rows for us.

```
W = span(S)
B = W.basis()
B
```

Demonstration 1 Construct a *random* vector, `w`, in this subspace by choosing scalars for a linear combination of the vectors we used to build `W` as a span originally.

Then use the three *basis* vectors in `B` to recreate the vector `w`. Question: how many ways can you do this? By Theorem VRRB there should always be exactly one way to create `w` using a linear combination of a basis of `W`.

```
w = *v1 + *v2 + *v3 + *v4 + *v5
w
```

```
w in W
```

```
*B[0] + *B[1] + *B[2]
```

2 Nonsingular Matrices

We will obtain a basis of $\mathbb{C}^{10}$ from the columns of a 10×10 nonsingular matrix.

```
entries = [[ 1,  1,  1, -1, -2,  4,  2, -3,  1, -6],
           [-2, -1, -2,  2,  4, -7, -4,  5, -1,  7],
           [ 1, -1,  2, -2, -5,  8,  5, -3,  4, -4],
           [-1, -2,  0,  1,  0, -5,  0, -3, -5,  6],
           [ 0, -2,  1, -1, -2,  3,  2,  3,  3,  7],
           [ 1,  0,  1, -1, -2,  4,  2,  0,  2,  0],
           [-1,  0, -1,  1,  3, -1, -2,  7,  5,  1],
           [ 1,  1,  1, -1, -2,  8,  3,  2,  8, -6],
           [ 0,  2, -1,  1,  2, -1, -2,  2,  2, -6],
           [ 1,  3,  0,  0,  1,  3,  0,  0,  3, -8]]
M = matrix(QQ, entries)
M
```

```
not M.is_singular()
```

A totally random vector with 10 entries:

```
v = random_vector(ZZ, 10, x=-9, y=9)
v
```

Demonstration 2 By Theorem CNMB, the columns of the matrix are a basis of $\mathbb{C}^{10}$. So the vector `v` should be a linear combination of the columns of the matrix. Verify this fact in three ways.

1. First, the old-fashioned way, thus exposing Theorem NMUS.
2. Then, the modern way, with an inverse, since a nonsingular matrix is invertible, thus exposing Theorem SNCM.
3. Finally, the Sage way, as described below.

```
aug = M.augment(v)
aug.rref()
```

```
M.inverse()*v
```

The Sage way: first create a space with a **user basis**.

```
X = (QQ^10).subspace_with_basis(M.columns())
X
```

Sage still carries an **echelonized basis**, in addition to the **user-installed** basis.

```
X.basis()
```

```
X.echelonized_basis()
```

Now ask for a coordinatization, relative to the basis in `X`, thus exposing Theorem VRRB.

```
X.coordinates(v)
```

3 Orthonormal Bases

A particularly simple orthonormal basis of $\mathbb{C}^3$, collected into the set `S`.

```
v1 = vector(QQ, [1/3, 2/3, 2/3])
v2 = vector(QQ, [2/3, -2/3, 1/3])
v3 = vector(QQ, [2/3, 1/3, -2/3])
S = [v1, v2, v3]
```

Demonstration 3 If these vectors are an orthonormal basis, then as the columns of a matrix they should create an orthonormal basis.

```
Q = column_matrix(S)
Q
```

```
Q.conjugate_transpose()*Q
```

```
Q.is_unitary()
```

Demonstration 4 Build a random vector of size 3 and find our ways to express the vector as a (unique) linear combination of the basis vectors. Which method is most efficient?

A totally random vector with 3 entries.

```
v = random_vector(ZZ, 3, x=-9, y=9)
v
```

First, the old-fashioned way, thus exposing Theorem NMUS.

```
aug = Q.augment(v)
aug.rref()
```

Now, the modern way, with an inverse, since a nonsingular matrix is invertible, thus exposing Theorem SNCM.

```
Q.inverse()*v
```

The Sage way. Create a space with a "user basis" and ask for a coordinatization, thus exposing Theorem VRRB.

```
X = (QQ^3).subspace_with_basis(Q.columns())
X.coordinates(v)
```

Finally, exploiting the orthonormal basis, and computing scalars for the linear combination with an inner product, thus exposing Theorem COB. (Sage's `.inner_product()` does not conjugate the entries of either vector, so we use the more careful `.hermitian_inner_product()` vector method instead.)

```
a1 = v1.hermitian_inner_product(v)
a2 = v2.hermitian_inner_product(v)
a3 = v3.hermitian_inner_product(v)
a1, a2, a3
```