Robert Beezer

Math 420, Spring 2014

1 Exact LU Decomposition

A 5×7 matrix, of rank 5.

```
A = matrix(QQ, [[ 1, 0,  1,  8,  0, -4,  4],
                [ 0, 1,  0,  5, -1,  6, -5],
                [-1, 0,  0, -4,  0,  0,  0],
                [-1, 1, -1, -3,  0,  6, -4],
                [-1, 0,  0, -4, -1,  5, -6]])
A
```

We use row operations, strictly of the third type, to "zero out" entries of columns *below* the diagonal.

Column one.

```
one = elementary_matrix(QQ, 5, row1=2, row2=0, scale=1)*elementary_matrix(QQ, 5, row1=3, ro
```

```
one*A
```

Column two.

```
two = elementary_matrix(QQ, 5, row1=3, row2=1, scale=-1)
```

```
two*one*A
```

Column three.

```
three = elementary_matrix(QQ, 5, row1=4, row2=2, scale=-1)
```

```
three*two*one*A
```

That's it. Done. Everything below the diagonal of this matrix is zero, so the matrix is upper-triangular. Notice that the result is *not in reduced row-echelon form* and *may not even have linearly independent rows*.

Let's collect the pieces.

First, the upper-triangular matrix U.

```
U = three*two*one*A
U
```

The elementary matrices are square and invertible, so have inverses individually, and so does their product. We will move the product to the other side of the equation by taking an inverse. As a product of elementary matrices of a very specific type, which are all lower-triangular with 1's on the diagonal, the inverse will also be lower-triangular with 1's on the diagonal.

```
L = (three*two*one).inverse()
L
```

Check our work, we should have $LU = A$.

```
L*U == A
```

Here is the Sage routine.

```
A.LU()
```

Read Sage online help about the *exact* LU method, especially pivoting and a storage efficient return value.

2 Numerical LU

`RDF` is the "Real Double Field", 64-bit imitations of real numbers (two 32-bit words). This gives 53 bits of precision in the significand (nee mantissa), 11 bits for the exponent, 1 bit for the sign. Also known as "double-precision floating point" numbers or IEEE 754. Do not confuse this with Sage's `RealField` which allows you to specify arbitraty precision, and which defaults to 53 bits as `RR`.

```
B = random_matrix(RDF, 15)
B
```

```
B.LU()
```

And a rectangular matrix.

```
C = random_matrix(RDF, 15, 18)
C
```

```
C.LU()
```

3 Solving Systems with an LU Decomposition

To solve $A\vec{x} = \vec{b}$, first solve $L\vec{y} = \vec{b}$, by successively solving for the first entries of $\vec{y}$ first. We will *simulate* this in Sage.

```
b = A*vector(QQ, (2,3,-1,4,5,0,0))
b
```

```
_, L, U = A.LU()
```

```
y = L.solve_right(b)
y
```

Now, solve for $\vec{x}$ from $U\vec{x} = \vec{y}$, since

$$A\vec{x} = LU\vec{x} = L\vec{y} = \vec{b}$$

Notice that this would be "backsolving", successively getting the entries of $\vec{x}$ with the last entries first, and in this case by setting the "extra" variables to zero as a convenience.

```
x = U.solve_right(y)
x
```

And check we have a solution:

```
A*x
```