Robert Beezer

Math 420, Spring 2014

1 Exact QR Decomposition

We form a QR decomposition of a random 4×4 nonsingular matrix, using the field of algebraic numbers for square roots. We use Householder reflections to progressively “zero out” the below-diagonal portions of columns of the matrix.

First we define a utility function which accepts a vector and returns the Householder matrix which will map it to a multiple of the first column of the identity matrix.

```
def house(x):
    """A vector in, Householder matrix out, converts
        vector to multiple of column 1 of identity
        matrix"""
    R = x.base_ring()
    e = zero_vector(R, len(x))
    e[0]=1
    v = x - (x.hermitian_inner_product(x)^(1/2))*e
    H = identity_matrix(R, len(v))
    H = H -
        (2/v.hermitian_inner_product(v))*matrix(v).transpose()*matrix(v).conjugate_t
    return H
```

A check that the function works as advertised.

```
v = vector(QQbar, [1,2,3])
W = house(v)
W*v
```

A random 4×4 matrix with determinant 1.

```
A = random_matrix(QQ, 4, algorithm="unimodular",
    upper_bound=9).change_ring(QQbar)
A
```

The first Householder matrix.

```
Q1 = block_diagonal_matrix(identity_matrix(0),
    house(A.column(0)) )
Q1
```

And its effect on A.

```
R1 = Q1*A
R1
```

Second iteration.

```
Q2 = block_diagonal_matrix(identity_matrix(1),
    house(R1.column(1)[1:4]) )
Q2
```

And its effect on A.

```
R2 = Q2*Q1*A
R2
```

third iteration.

```
Q3 = block_diagonal_matrix(identity_matrix(2),
    house(R2.column(2)[2:4]) )
Q3
```

And its effect on A.

```
R3 = Q3*Q2*Q1*A
R3
```

Done. `R3` is lower triangular. Since A was square, we do not need a fourth iteration.

Now we package up the unitary matrices properly, setting both Q and R. Remember Householder matrices are Hermitian, so we do not have to transpose them, and all our entries are real numbers, so we do not have to conjugate.

```
Q = Q1*Q2*Q3
R = R3
Q
```

```
Q.is_unitary()
```

```
Q*R
```

```
Q*R - A
```

```

```