Schur decomposition

matrix A can be expressed as similar transform of matrix U:

$$A = QUQ^{-1}$$

where U is upper triangular and is called the Schur form of A. Q is unitary, and U is upper diagonal, which means the eigenvalues of A will be on the diagonal of U. Find Schur decomp = expose eigenvalues.

Real analog of power method (7.3.1)

(7.4.1) is a real analog to the power method of finding eigenvalues. At a stage ($H_k = R_k U_k$), the machinery breaks down for matrices having complex eigenvalues. We must therefore calculate this term in a different way, namely by a *real Schur decomposition.*

Real Schur Decomposition

It is block upper diagonal. In the case of complex eigenvalues, the diagonal consists of 2x2 block matrices with complex conjugate eigenvalues.

Hessenberg QR step

The term $H_k = R_k U_k$ from the first section can now be calculated efficiently (order of magniute O(2) instead of O(3)) if we initialize the calculation with a matrix (H_0) on Hessenberg form (see next section).

The Hessenberg reduction

It reads

$$U_0^T A U_0 = H$$

everything below subdiagonal is zero in H. U_0 is a product of matrices P_k who's purpose is to zero the k'th column below the subdiagonal.

Code for solving this:

```
 #include <armadillo>
#include <iostream>
#include <math.h>

using namespace std;
using namespace arma;

void house(double & beta, colvec & v, colvec & x) {
    double sigma, mu;

    int n = x.n_rows;
```

```
    colvec xspan = x(span(1, n - 1));
    sigma = dot(xspan, xspan);

    v = zeros<colvec > (n);
    v(span(1, n - 1)) = xspan;

    //zero if sigma is zero
    if (sigma == 0) {
        beta = 0;
    } else {
        mu = sqrt(x(0) * x(0) + sigma);

        if (x(0) <= 0) {
            v(0) = x(0) - mu;
        } else {
            v(0) = -sigma / (x(0) + mu);
        }

        beta = 2 / (sigma / (v(0) * v(0)) + 1);
        v /= v(0);
    }
}

void houseRedHessenberg(mat & A) {

    double beta;
    colvec v, x;
    mat ImBvvT;

    int n = A.n_cols;
    mat I = eye<mat > (n - 1, n - 1);

    for (int k = 0; k < n - 2; k++) {
        x = A(span(k + 1, n-1), k);
        house(beta, v, x);
        A(span(k + 1, n - 1), k) = x;

        ImBvvT = I - beta * v * strans(v);
        A(span(k + 1, n - 1), span(k, n - 1)) = ImBvvT * A(span(k + 1, n - 1), span(k, n - 1)
        A(span(), span(k + 1, n - 1)) = A(span(), span(k + 1, n - 1)) * ImBvvT;
    }

}

int main() {

    mat A;
    A << 1 << 5 << 7 << endr
            << 3 << 0 << 6 << endr
```

```
            << 4 << 3 << 1 << endr;

    cout << A << endl;
    houseRedHessenberg(A);
    cout << A << endl;

    return 0;
}
```

Runtime:

```
 1.0000   5.0000   7.0000
3.0000        0   6.0000
4.0000   3.0000   1.0000

1.0000   8.6000  -0.2000
5.0000   4.9600  -0.7200
     0   2.2800  -3.9600
```

which is the example on page 345.
Flops: $10n^3/3$.

Hessenberg matrix properties

Hessenberg decompostion is not unique. However, if there are no subdiagonal zero elements it is unique. Then it is called *unreduced.*

Companion matrix form

Non unitary analog of hessenberg decomposition. Using *Krylov matrices*, we can express the companion matrix C as

$$K^{-1}AK = C = c_0 + c_1\lambda_A + .. + c_{n-1}\lambda_A^{n-1} + \lambda^n$$

which is the characteristic polynomial of A, which can be solved for the eigenvalues λ.

Hessenberg reduction via Gauss Transform

Uses only half the flops contra Householder, but chance that it uses more, since it uses partial pivoting which might or might not be slow. Eigenvalue condition numbers are not preserved, which gives a complicated error estimation process (unsure why..)