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 has all zeros below the diagonal, which means the eigenvalues of A will be on the diagonal of U. Find Schur decomp = expose eigenvalues.

Real Schur Decomposition

If A has real eigenvalues it is a pure schur form as above. [2x2]-block schur form for complex (conjugate) eigenvalues. [2x2] block i on the diagonal has eigenvalues $\lambda_{i,1}$ and $\lambda_{i,2} = \lambda_{i,1}^*$, where λ_* is eigenvalues of the original matrix A.

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 real matrices having complex eigenvalues (no schur form can be achieved).However, calculating this term in a different way can yield a *real Schur decomposition* (eigenvalues come in complex conjugate pairs) and save the day.

Hessenberg QR step

The term $H_k = R_k U_k$ from the first section can be calculated efficiently if we initialize the calculation with a matrix (H_0) on reduced hessenberg form (order of magniute O(2) instead of O(3)). if H_0 is on reduced hessenberg form, it will stay that way if we calculate $H_k = R_k U_k$ by a **Hessenberg QR step.**

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* K ($K^{-1} \neq K^H$), we can express the companion matrix C as

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

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. Chance that it uses more since it, like Gaussian elimination's partial pivoting, has a chance of $2^n growth$.

Eigenvalue condition numbers $s(\lambda)^{-}1$ are not preserved, which gives a complicated error estimation process. This is the case of non-orthogonal similarities. (Flipping rows, multiplying rows etc. changes the signs and factors of the eigenvalues, whilst orthogonal similarities preserve everything).