Class notes - Reduction to Hessenberg matrices

Kavita Sutar

1 Why care?

General purpose eigenvalue algorithms are based on the principle of computing *eigenvalue-revealing factorizations* of a given matrix A. We have seen a few of these earlier. These are computed by applying a series of transformations to A in order to introduce zeroes in the necessary places. Most of the eigenvalue algorithms proceed by computing the Schur factorization $X \mapsto Q_j^* X Q_j$ so that the product

$$\underbrace{Q_j^* \cdots Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 \cdots Q_j}_{Q}$$

converges to an upper triangular matrix T as $j \to \infty$.

This sequence of computations is usually split into two phases: in the first phase, a direct method is applied to the given matrix to produce an *upper Hessenberg* matrix, i.e. a matrix with zeroes below the first subdiagonal. In the second phase, an iteration is applied to generate a sequence of Hessenberg matrices that converge to a triangular form. Schematically:

$$A \neq A^* : \quad \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix} \xrightarrow{\text{Phase I}} \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{pmatrix} \xrightarrow{\text{Phase II}} \begin{pmatrix} * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \\ & & & & * \end{pmatrix}$$

The first phase requires $\mathcal{O}(m^3)$ flops; the second phase is infinite in principle, but achieves machine precision in $\mathcal{O}(m)$ flops. Notice how the preliminary step of reducing the matrix to Hessenberg form reduces the number of required computations: without it, Phase II would require $\mathcal{O}(m^3)$ in each iteration thus bringing the total count to $\mathcal{O}(m^4)$ flops (assuming that that convergence requires $\mathcal{O}(m)$ flops) or higher.

If A is symmetric, this approach is even faster. The intermediate matrix is symmetric Hessenberg (i.e. tridiagonal) and the final matrix is diagonal.

$$A = A^* : \quad \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix} \xrightarrow{\text{Phase I}} \begin{pmatrix} * & * & & & \\ * & * & * & & \\ & * & * & * & \\ & & * & * & * \\ & & & * & * \end{pmatrix} \xrightarrow{\text{Phase II}} \begin{pmatrix} * & & & & \\ & * & & & \\ & & * & & \\ & & & * & \\ & & & & * \end{pmatrix}$$

Here, Phase I requires the same (i.e. $\mathcal{O}(m^3)$ flops) since the reduction process is same as before. In Phase II, since the iterations are performed on a tridiagonal matrix, the number of computations comes down to $\mathcal{O}(m)$ flops per iteration, making it $\mathcal{O}(m^2)$ flops for the Phase II.

2 How is it done?

Its done by using Householder matrices in a clever manner. Remember that we want to find the eigenvalues of A, which means we do not want to lose them in the whole process, the final triangular(or diagonal) matrix should have the same eigenvalues as A. This means that we must resort to similarity transformations of the type $Q^{-1}AQ$ only.

First note that the obvious idea that we could kill off the subdiagonal entries of each column using Householder matrices and reach a desired upper triangular form directly, fails: suppose you use a Householder matrix Q_1^* to kill of the subdiagonal entries of the first column, so that

$$Q_1^* A = \begin{pmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix},$$

but the right multiplication by Q_1, namely $(Q_1^*A)Q_1$ will destroy the zeroes obtained in the previous step! Note also that this method is bound to fail since otherwise we would be able to find the eigenvalues in a finite number of steps, a contradiction.

The only merit in this idea is that the process eventually reduces the size of the entries below the diagonal, even if it does not make them zero. It is this idea that is exploited in the QR iteration.

The correct way to use Householder matrices is as follows:

- at the first step, choose a Householder matrix Q_1^* that leaves the first row unchanged.
  - when A is multiplied on the left by Q_1^*, it forms linear combinations of only rows $2, \dots, m$ to introduce zeroes in the spots $(3,1),\ (4,1), \dots, (m,1)$ of the first column;
  - when Q_1 is multiplied on the right of Q_1^*A, it replaces the columns $2, \dots, m$ by their linear combinations and leaves the first column unchanged.

  The result is:

$$\underset{A}{\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}} \xrightarrow{Q_1^*\cdot} \underset{Q_1^*A}{\begin{pmatrix} * & * & * & * & * \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{pmatrix}} \xrightarrow{\cdot Q_1} \underset{A_1 = Q_1^*AQ_1}{\begin{pmatrix} * & \times & \times & \times & \times \\ * & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{pmatrix}} \tag{1}$$

  The asteriks in the last two matrices denote the entries which are unchanged from the previous matrix.

- The second Householder matrix is chosen so that the first and second rows and columns of A_1 remain unchanged:

$$\underset{A_1}{\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix}} \xrightarrow{Q_2^*\cdot} \underset{Q_2^*A_1}{\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{pmatrix}} \xrightarrow{\cdot Q_2} \underset{A_2 = Q_2^*A_1Q_2}{\begin{pmatrix} * & * & \times & \times & \times \\ * & * & \times & \times & \times \\ 0 & * & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{pmatrix}} \tag{2}$$

- After repeating this process $(m-2)$ times, we have the Hessenberg form:

$$\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{pmatrix} \tag{3}$$
$$H = \underbrace{Q_{m-2}^* \cdots Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 \cdots Q_{m-2}}_{Q}$$

Reduction to Hessenberg form using Householder matrices: $A \in \mathbb{R}^{m\times m}$.

```
for $k = 1$ to $m-2$ do
    $x = A_{k+1:m,k}$
    $v_k = \text{sign}(x_1)||x||_2 e_1 + x$
    $v_k = v_k/||v_k||_2$
    $A_{k+1:m,k:m} = A_{k+1:m,k:m} - 2v_k(v_k^* A_{k+1:m,k:m})$
    $A_{1:m,k+1:m} = A_{1:m,k+1:m} - 2(A_{1:m,k+1:m}v_k)v_k^*$
end
```

As in the algorithm for Householder's method for solving system of equations, the matrix Q is never computed explicitly. The reflection vectors v_k are saved instead and can be used to reconstruct Q if necessary.

3 Operation count

The count is dominated by steps 4 and 5 of the for loop. These are two *for* loops in themselves.

Step 4 corresponds to left multiplication by Q^*. At the k^{th} step, the matrix operates on the last $m-k$ rows, each row has its first $(k-1)$ elements as zero, so the operations are done on $(m-k+1)$ elements per row, thus an $(m-k)\times(m-k+1)$ submatrix. Further we require 4 flops per operation, so counting over the outer for loop we are looking at

$$\sum_{k=1}^{m-2} 4(m-k)(m-k+1) \approx 4m^3/3 \text{ flops.}$$

Step 5 corresponds to right multiplication by Q. This involves operations on all entries (zero or non-zero) of the last $(m-k)$ columns. The reflector v_k operates by forming linear combinations of the $(m-k)$ columns. These are $m(m-k)$ entries for each value of k. For each entry we require 4 flops, so counting over the outer for loop, this makes a total of

$$\sum_{k=1}^{m-2} 4(m)(m-k) \approx 4(m^3/2) = 2m^3 \text{ flops.}$$

Thus the total amount of work for unitary reduction of an $m\times m$ matrix to Hessenberg form is $10m^3/3$ flops.

4 When A is hermitian

For a hermitian matrix, the above algorithm gives a hermitian Hessenberg i.e. a tridiagonal matrix.

Since there are zeroes in columns as well as rows, the number of operations required for steps 4 and 5 of the algorithm is the same i.e. $4m^3/3$ flops each. Further, the consideration of symmetry means that at each step, the matrix being acted upon is hermitian, so we can get by with doing half the computations (either left multiplication or right multiplication) to find the entries. Thus the total amount of work for unitary reduction of an $m\times m$ hermitian matrix to Hessenberg form is $4m^3/3$ flops.

5 Stability

The algorithm is backward stable in the following sense: let $\hat{H}$ be the Hessenberg matrix computed in floating point arithmetic and Q be the *exact unitary matrix* (i.e. defined mathematically, not by the computer) corresponding to the reflection vectors $\tilde{v_k}$ computed in floating point arithmetic

Theorem 1 *(Theorem 26.1 [1])*

$$Q\hat{H}Q^* = A + \delta A, \quad \text{where} \quad \frac{||\delta A||}{||A||} = \mathcal{O}(\epsilon_{\text{mach}}).$$

6 Details of operation count computation

6.1 Step 4

$$
\begin{aligned}
\sum_{k=1}^{m-2} 4(m-k)(m-k+1) &= 4\left[\sum_{k=1}^{m-2}(m-k)^2 + \sum_{1}^{m-2}(m-k)\right] \\
&= 4\left[\sum_{k=1}^{m-1} k^2 - 1 + \sum_{1}^{m-1} k - 1\right] \\
&= 4\left[\frac{(m-1)m(2m-1)}{6} - 1 + \frac{(m-1)m}{2} - 1\right] \\
&= 4\left[\frac{2m^3}{6} + \mathcal{O}(m^2)\right] \\
&= \frac{4m^3}{3} + \mathcal{O}(m^2)
\end{aligned}
$$

6.2 Step 5

$$
\begin{aligned}
\sum_{k=1}^{m-2} 4m(m-k) &= 4\left[\sum_{k=1}^{m-2} m^2 + \sum_{1}^{m-2} mk\right] \\
&= 4\left[m^2 \sum_{k=1}^{m-2} 1 + m\sum_{1}^{m-2} k\right] \\
&= 4\left[m^2(m-2) + \frac{m(m-2)(m-1)}{2}\right] \\
&= 4\left[m^3 - \frac{m^3}{2} + \mathcal{O}(m^2)\right] \\
&= 4m^3 - \frac{4m^3}{2} + \mathcal{O}(m^2) \\
&= 2m^3 + \mathcal{O}(m^2)
\end{aligned}
$$

References

[1] Trefethen and Bau. *Numerical linear algebra.*

[2] Golub and Loan. *Matrix computations*, fourth edition.