Numerical Analysis - Part II

Anders C. Hansen

Lecture 23

Eigenvalues and eigenvectors

Motivation – The Schrödinger equation

One of the word's most famous eigenvalue problems: The time independent Schrödinger equation

$$\left[\frac{-\hbar^2}{2m}\nabla^2 + V(\mathbf{r})\right]\Psi(\mathbf{r}) = E\Psi(\mathbf{r}).$$

Introduction to matrix eigenvalue calculations

Let A be a real $n \times n$ matrix. The eigenvalue equation is $A\mathbf{w} = \lambda\mathbf{w}$, where λ is a scalar, which may be complex if A is not symmetric. There exists a nonzero vector $\mathbf{w} \in \mathbb{R}^n$ satisfying this equation if and only if $\det(A - \lambda I) = 0$. The function $p(\lambda) = \det(A - \lambda I)$, $\lambda \in \mathbb{C}$, is a polynomial of degree n, but calculating the eigenvalues by finding the roots of p is a disaster area because of loss of accuracy due to rounding errors.

Introduction to matrix eigenvalue calculations

If the polynomial has some multiple roots and if A is not symmetric, then the number of linearly independent eigenvectors may be fewer than n, but there are always n mutually orthogonal real eigenvectors in the symmetric case.

We assume in all cases, however, that the eigenvalue equations $A\mathbf{w}_i = \lambda_i \mathbf{w}_i$, $i = 1..n$, are satisfied by eigenvectors $\mathbf{w}_i$ that are linearly independent, which can be achieved by making an arbitrarily small change to A if necessary.

The power method

The iterative algorithms that will be studied for the calculation of eigenvalues and eigenvectors are all closely related to the power method, which has the following basic form for generating a single eigenvalue and eigenvector of A.

We pick a nonzero vector $\mathbf{x}^{(0)}$ in $\mathbb{R}^n$. Then, for $k = 0, 1, 2, \ldots$, we let $\mathbf{x}^{(k+1)}$ be a nonzero multiple of $A\mathbf{x}^{(k)}$, typically to satisfy $\|\mathbf{x}^{(k+1)}\| = 1$ so that

$$\mathbf{x}^{(k+1)} = A\mathbf{x}^{(k)}/\|A\mathbf{x}^{(k)}\|, \qquad k = 0, 1, 2, \ldots$$

This method is oriented on finding an eigenvector corresponding to the largest eigenvalue as the the following theorem shows.

The power method – Theoretical results

Theorem 1

Let $A\mathbf{w}_i = \lambda_i \mathbf{w}_i$, where the eigenvalues of A satisfy $|\lambda_1| \le \cdots \le |\lambda_{n-1}| < |\lambda_n|$ and the eigenvectors are of the unit length $\|\mathbf{w}_i\| = 1$. Assume $\mathbf{x}^{(0)} = \sum_{i=1}^{n} c_i \mathbf{w}_i$ with $c_n \neq 0$. Then $\mathbf{x}^{(k)} \to \pm \mathbf{w}_n$ as $k \to \infty$.

Proof. Given $x^{(0)}$ as in the assumption, $\mathbf{x}^{(k)}$ is a multiple of

$$A^k \mathbf{x}^{(0)} = \sum_{i=1}^{n} c_i \lambda_i^k \mathbf{w}_i = c_n \lambda_n^k \left(\mathbf{w}_n + \sum_{i=1}^{n-1} \tfrac{c_i}{c_n} \left(\tfrac{\lambda_i}{\lambda_n} \right)^k \mathbf{w}_i \right).$$

Since $\|\mathbf{x}^{(k)}\| = \|\mathbf{w}_n\| = 1$, we conclude that $\mathbf{x}^{(k)} = \pm \mathbf{w}_n + \mathcal{O}(\rho^k)$, where the sign is that of $c_n \lambda_n^k$ and the ratio $\rho = \frac{|\lambda_{n-1}|}{|\lambda_n|} < 1$ characterizes the rate of convergence. $\square$

The power method in algorithmic form

Here are the details of an implementation of the procedure.

0. Pick $\mathbf{x}^{(0)} \in \mathbb{R}^n$ satisfying $\|\mathbf{x}^{(0)}\| = 1$. Let ε be a small positive tolerance. Set $k = 0$.

1. Calculate $\widetilde{\mathbf{x}}^{(k+1)} = A\mathbf{x}^{(k)}$ and set $\lambda = \frac{\mathbf{x}^{(k)T} A\mathbf{x}^{(k)}}{\mathbf{x}^{(k)T}\mathbf{x}^{(k)}}$.

(This λ is called the *Raleigh quotient* and it minimizes $f(\mu) = \|\widetilde{\mathbf{x}}^{(k+1)} - \mu\mathbf{x}^{(k)}\|$ over μ.

2. If $f(\lambda) \leq \varepsilon$, accept λ as an eigenvalue and $\mathbf{x}^{(k)}$ as the corresponding eigenvector.

3. Otherwise, let $\mathbf{x}^{(k+1)} = \widetilde{\mathbf{x}}^{(k+1)}/\|\widetilde{\mathbf{x}}^{(k+1)}\|$, increase k by one and go back to **1.**

The power method – Halting criterion

The termination occurs because, by the previous theorem, we have

$$\|\widetilde{\mathbf{x}}^{(k+1)} - \lambda \mathbf{x}^{(k)}\| = \min_{\mu} \|\widetilde{\mathbf{x}}^{(k+1)} - \mu \mathbf{x}^{(k)}\| \leq \|\widetilde{\mathbf{x}}^{(k+1)} - \lambda_n \mathbf{x}^{(k)}\|$$
$$= \|A\mathbf{x}^{(k)} - \lambda_n \mathbf{x}^{(k)}\| = \|A\mathbf{w}_n - \lambda_n \mathbf{w}_n\| + \mathcal{O}(\rho^k) = \mathcal{O}(\rho^k) \to 0.$$

Deficiencies of the power method

The power method may perform adequately if $c_n \neq 0$ and $|\lambda_{n-1}| < |\lambda_n|$, where we are using the notation of Theorem 1, but often it is unacceptably slow. The difficulty of $c_n = 0$ is that, theoretically, in this case the method should find an eigenvector $\mathbf{w}_m$ with the largest m such that $c_m \neq 0$, but practically computer rounding errors can introduce a small nonzero component of $\mathbf{w}_n$ into the sequence $\mathbf{x}^{(k)}$, and then $\mathbf{w}_n$ may be found eventually, but one has to wait for the small component to grow.

Moreover, $|\lambda_{n-1}| = |\lambda_n|$ is not uncommon when A is real and nonsymmetric, because the spectral radius of A may be due to a complex conjugate pair of eigenvalues. Next, we will study the inverse iterations (with *shifts*), because they can be highly useful, particularly in the more efficient methods for eigenvalue calculations that will be considered later.

Inverse iteration

This method is highly useful in practice. It is similar to the power method, except that, instead of $\mathbf{x}^{(k+1)}$ being a multiple of $A\mathbf{x}^{(k)}$, we make the choice

$$(A - sI)\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}, \qquad k = 0, 1, \dots, \tag{1}$$

where s is a scalar that may depend on k and $\|x^{(k)}\| = 1$. Therefore the calculation of $\mathbf{x}^{(k+1)}$ from $\mathbf{x}^{(k)}$ requires the solution of an $n \times n$ system of linear equations whose matrix is $(A-sI)$. Further, if s is a constant and if $A - sI$ is nonsingular, we deduce from (1) that $\mathbf{x}^{(k)}$ is a multiple of $(A - sI)^{-k}\mathbf{x}^{(0)}$.

Inverse iteration

We again let $\mathbf{x}^{(0)} = \sum_{i=1}^{n} c_i \mathbf{w}_i$, as in the proof of Theorem 1, assuming that $\mathbf{w}_i$, $i = 1..n$, are linearly independent eigenvectors of A that satisfy $A\mathbf{w}_i = \lambda_i \mathbf{w}_i$. Therefore we note that the eigenvalue equation implies $(A - sI)\mathbf{w}_i = (\lambda_i - s)\mathbf{w}_i$, which in turn implies $(A - sI)^{-1}\mathbf{w}_i = (\lambda_i - s)^{-1}\mathbf{w}_i$. It follows that $\mathbf{x}^{(k)}$ is a multiple of

$$(A - sI)^{-k}\mathbf{x}^{(0)} = \sum_{i=1}^{n} c_i (A - sI)^{-k}\mathbf{w}_i = \sum_{i=1}^{n} c_i (\lambda_i - s)^{-k}\mathbf{w}_i.$$

Thus, if the m-th number in the set $\{|\lambda_i - s|\}$ is the smallest and if c_m is nonzero, then $\mathbf{x}^{(k)}$ tends to be a multiple of $\mathbf{w}_m$ as $k \to \infty$.

We see that the speed of convergence can be excellent if s is very close to λ_m. Further, it can be made even faster by adjusting s during the calculation. Typical details are given in the following implementation.

Typical implementation of inverse iteration

0. Set s to an estimate of an eigenvalue of A. Prescribe $\mathbf{x}^{(0)} \neq 0$, let $0 < \varepsilon \ll 1$ and set $k = 0$.
1. Calculate (with pivoting if necessary) the LU factorization of $A - sI$.
2. Stop if U is singular because then s is an eigenvalue of A, while its eigenvector is any vector in the null space of U: it can be found easily, U being upper triangular.
3. Calculate $\mathbf{x}^{(k+1)}$ by solving $(A - sI)\mathbf{x}^{(k+1)} = LU\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$ using the LU factorization from **1.**
4. Set η to the number that minimizes $f(\mu) = \|\mathbf{x}^{(k)} - \mu\mathbf{x}^{(k+1)}\|$.
5. Stop if $f(\eta) \leq \varepsilon\|\mathbf{x}^{(k+1)}\|$. Since $f(\eta) = \|A\mathbf{x}^{(k+1)} - (s + \eta)\mathbf{x}^{(k+1)}\|$, we let $s + \eta$ be the calculated eigenvalue of A and $\mathbf{x}^{(k+1)}/\|\mathbf{x}^{(k+1)}\|$ be its eigenvector.
6. Otherwise, replace $\mathbf{x}^{(k+1)}$ by $\mathbf{x}^{(k+1)}/\|\mathbf{x}^{(k+1)}\|$, increase k by one, and either return to **3** without changing s or to **1** after replacing s by $s + \eta$.

Upper Hessenberg matrix

A matrix of the following form

$$H_n = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,n} \\ 0 & h_{3,2} & h_{3,3} & \cdots & h_{3,n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{n,n-1} & h_{n,n} \end{bmatrix}.$$

is called an upper Hessenberg matrix.

Further on inverse iteration

The inverse iteration algorithm is very efficient if A is an *upper Hessenberg matrix:* every element of A under the first subdiagonal is zero (i.e. $a_{ij} = 0$ if $j < i - 1$). In this case the LU factorization in **1** requires just $\mathcal{O}(n^2)$ or $\mathcal{O}(n)$ operations when A is nonsymmetric or symmetric, respectively.

Thus the replacement of s by $s + \eta$ in **6** need not be expensive, so fast convergence can often be achieved easily. There are standard ways of giving A this convenient form which will be considered later.

Spectrum preserved under similarity transforms

Theorem 2

Let A and S be $n \times n$ matrices, S being nonsingular. Then $\mathbf{w}$ is an eigenvector of A with eigenvalue λ if and only if $\widehat{\mathbf{w}} = S\mathbf{w}$ is an eigenvector of $\widehat{A} = SAS^{-1}$ with the same eigenvalue.

Proof.

$$A\mathbf{w} = \lambda\mathbf{w} \quad \Leftrightarrow \quad AS^{-1}(S\mathbf{w}) = \lambda\mathbf{w} \quad \Leftrightarrow \quad (SAS^{-1})(S\mathbf{w}) = \lambda(S\mathbf{w}).$$

$\square$

Deflation

Suppose that we have found one solution of the eigenvector equation $A\mathbf{w} = \lambda\mathbf{w}$, where A is again $n \times n$. Then *deflation* is the task of constructing an $(n-1) \times (n-1)$ matrix, B say, whose eigenvalues are the other eigenvalues of A. Specifically, we apply a similarity transformation S to A such that the first column of $\widehat{A} = SAS^{-1}$ is λ times the first coordinate vector $\mathbf{e}_1$, because it follows from the characteristic equation for eigenvalues and from Theorem 2 that we can let B be the bottom right $(n-1) \times (n-1)$ submatrix of $\widehat{A} = SAS^{-1}$. In particular,

$$SAS^{-1} = \widehat{A} = \begin{bmatrix} \lambda & \beta \\ 0 & B \end{bmatrix}.$$

Deflation

We write the condition on S as $(SAS^{-1})\mathbf{e}_1 = \lambda\mathbf{e}_1$. Then the last equation in the proof of Theorem 2 shows that it is sufficient if S has the property $S\mathbf{w} = c\mathbf{e}_1$, where c is any nonzero scalar.

Algorithm for deflation for symmetric A

Suppose that A is symmetric and $\mathbf{w} \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$ are given so that $A\mathbf{w} = \lambda\mathbf{w}$. We seek a nonsingular matrix S such that $S\mathbf{w} = c\mathbf{e}_1$ and such that SAS^{-1} is also symmetric. The last condition holds if S is orthogonal, since then $S^{-1} = S^T$. It is suitable to pick a *Householder reflection*, which means that S has the form

$$H_u = I - 2\mathbf{u}\mathbf{u}^T/\|\mathbf{u}\|^2, \quad \text{where} \quad \mathbf{u} \in \mathbb{R}^n.$$

Algorithm for deflation for symmetric A

Specifically, we recall from the Numerical Analysis IB course that Householder reflections are orthogonal and that, because $H_u\mathbf{u} = -\mathbf{u}$ and $H_u\mathbf{v} = \mathbf{v}$ if $\mathbf{u}^T\mathbf{v} = 0$, they reflect any vector in $\mathbb{R}^n$ with respect to the $(n-1)$-dimensional hyperplane orthogonal to $\mathbf{u}$. So, for any two vectors $\mathbf{x}$ and $\mathbf{y}$ of equal lengths,

$$H_{\mathbf{u}}\mathbf{x} = \mathbf{y}, \quad \text{where} \quad \mathbf{u} = \mathbf{x} - \mathbf{y}.$$

Hence,

$$\left(I - 2\frac{\mathbf{u}\mathbf{u}^T}{\|\mathbf{u}\|^2}\right)\mathbf{w} = \pm\|\mathbf{w}\|\mathbf{e}_1\,, \quad \text{where} \quad \mathbf{u} = \mathbf{w} \mp \|\mathbf{w}\|\mathbf{e}_1\,.$$

Since the bottom $n-1$ components of $\mathbf{u}$ and $\mathbf{w}$ coincide, the calculation of $\mathbf{u}$ requires only $\mathcal{O}(n)$ computer operations. Further, the calculation of SAS^{-1} can be done in only $\mathcal{O}(n^2)$ operations, taking advantage of the form $S = I - 2\mathbf{u}\mathbf{u}^T/\|\mathbf{u}\|^2$, even if all the elements of A are nonzero.

Algorithm for deflation for symmetric A

After deflation, we may find an eigenvector, $\widehat{\mathbf{w}}$ say, of SAS^{-1}. Then the new eigenvector of A, according to Theorem 2, is $S^{-1}\widehat{\mathbf{w}} = S\widehat{\mathbf{w}}$, because Householder matrices, like all symmetric orthogonal matrices, are *involutions*: $S^2 = I$.

Givens rotations

The notation $\Omega^{[i,j]}$ denotes the following $n \times n$ matrix

$$\Omega^{[i,j]} = \begin{array}{c} \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c & & s & \\ & & & \ddots & & \\ & & -s & & c & \\ & & & & & \ddots \\ & & \uparrow & & \uparrow & & 1 \end{bmatrix} \\ \begin{matrix} \quad & \quad & i & \quad & j & \quad \end{matrix} \end{array}, \quad c^2 + s^2 = 1.$$

Generally, for any vector $\mathbf{a}_k \in \mathbb{R}^n$, we can find a matrix $\Omega^{[i,j]}$ such that

$$\Omega^{[i,j]}\mathbf{a} = \begin{array}{c} \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c & & s & \\ & & & \ddots & & \\ & & -s & & c & \\ & & & & & \ddots \\ & & \uparrow & & \uparrow & & 1 \end{bmatrix} \\ \begin{matrix} \quad & \quad & i & \quad & j & \quad \end{matrix} \end{array} \begin{bmatrix} a_{1k} \\ \vdots \\ a_{ik} \\ \vdots \\ a_{jk} \\ \vdots \\ a_{nk} \end{bmatrix} = \begin{array}{cl} \begin{bmatrix} a_{1k} \\ \vdots \\ r \\ \vdots \\ 0 \\ \vdots \\ a_{nk} \end{bmatrix} & \begin{matrix} \\ \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \\ \end{matrix} \end{array}$$

$$c = \frac{a_{ik}}{\sqrt{a_{ik}^2 + a_{jk}^2}},$$

$$s = \frac{a_{jk}}{\sqrt{a_{ik}^2 + a_{jk}^2}},$$

$$r = \sqrt{a_{ik}^2 + a_{jk}^2}.$$

Givens rotations

1) We can choose $\Omega^{[i,j]}$ so that any prescribed element $\widetilde{a}_{jk}$ in the j-th row of $\widetilde{A} = \Omega^{[i,j]}A$ is zero.
2) The rows of $\widetilde{A} = \Omega^{[i,j]}A$ are the same as the rows of A, except that the i-th and j-th rows of the product are linear combinations of the i-th and j-th rows of A.
3) The columns of $\widehat{A} = \widetilde{A}\Omega^{[i,j]T}$ are the same as the columns of $\widetilde{A}$, except that the i-th and j-th columns of $\widehat{A}$ are linear combinations of the i-th and j-th columns of $\widetilde{A}$.
4) $\Omega^{[i,j]}$ is an orthogonal matrix, thus $\widehat{A} = \Omega^{[i,j]}A\Omega^{[i,j]T}$ inherits the eigenvalues of A.
5) If A is symmetric, then so is $\widehat{A}$.

Transformation to upper Hessenberg – Givens

Transformation to an upper Hessenberg form: We replace A by $\widehat{A} = SAS^{-1}$, where S is a product of Givens rotations $\Omega^{[i,j]}$ chosen to annihilate subsubdiagonal elements $a_{j,i-1}$ in the $(i-1)$-st column:

$$\begin{bmatrix} \ast\ast\ast\ast \\ \ast\ast\ast\ast \\ \ast\ast\ast\ast \\ \ast\ast\ast\ast \end{bmatrix} \xrightarrow{\Omega^{[2,3]}\times} \begin{bmatrix} \ast\ast\ast\ast \\ \bullet\bullet\bullet\bullet \\ 0\bullet\bullet\bullet \\ \ast\ast\ast\ast \end{bmatrix} \xrightarrow{\times\Omega^{[2,3]T}} \begin{bmatrix} \ast\bullet\bullet\ast \\ \ast\bullet\bullet\ast \\ 0\bullet\bullet\ast \\ \ast\bullet\bullet\ast \end{bmatrix} \xrightarrow{\Omega^{[2,4]}\times} \begin{bmatrix} \ast\ast\ast\ast \\ \bullet\bullet\bullet\bullet \\ 0\ast\ast\ast \\ 0\bullet\bullet\bullet \end{bmatrix} \xrightarrow{\times\Omega^{[2,4]T}} \begin{bmatrix} \ast\bullet\ast\bullet \\ \ast\bullet\ast\bullet \\ 0\bullet\ast\bullet \\ 0\bullet\ast\bullet \end{bmatrix} \xrightarrow{\Omega^{[3,4]}\times} \begin{bmatrix} \ast\ast\ast\ast \\ \ast\ast\ast\ast \\ 0\bullet\bullet\bullet \\ 00\bullet\bullet \end{bmatrix} \xrightarrow{\times\Omega^{[3,4]T}} \begin{bmatrix} \ast\ast\bullet\bullet \\ \ast\ast\bullet\bullet \\ 0\ast\bullet\bullet \\ 00\bullet\bullet \end{bmatrix}$$

The $\bullet$-elements have changed through a single transformation while the $\ast$-elements remained the same.

It is seen that every element that we have set to zero remains zero, and the final outcome is indeed an upper Hessenberg matrix. If A is symmetric then so will be the outcome of the calculation, hence it will be tridiagonal. In general, the cost of this procedure is $\mathcal{O}(n^3)$.

Transformation to upper Hessenberg – Householder

Alternatively, we can transform A to upper Hessenberg using *Householder reflections*, rather than Givens rotations. In that case we deal with a column at a time, taking $\mathbf{u}$ such that, with $H_u = I - 2\mathbf{u}\mathbf{u}^T/\|\mathbf{u}\|^2$, the i-th column of $\widetilde{B} = H_u B$ is consistent with the upper Hessenberg form. Such a $\mathbf{u}$ has its first i coordinates vanishing, therefore $\widehat{B} = \widetilde{B} H_u^T$ has the first i columns unchanged, and all new and old zeros (which are in the first i columns) stay untouched.

$$\begin{bmatrix} *&*&*&*&* \\ *&*&*&*&* \\ *&*&*&*&* \\ *&*&*&*&* \\ *&*&*&*&* \end{bmatrix} \overset{H_1\times}{\rightarrow} \begin{bmatrix} *&*&*&*&* \\ \bullet&\bullet&\bullet&\bullet&\bullet \\ 0&\bullet&\bullet&\bullet&\bullet \\ 0&\bullet&\bullet&\bullet&\bullet \\ 0&\bullet&\bullet&\bullet&\bullet \end{bmatrix} \overset{\times H_1^T}{\rightarrow} \begin{bmatrix} *&\bullet&\bullet&\bullet&\bullet \\ *&\bullet&\bullet&\bullet&\bullet \\ 0&\bullet&\bullet&\bullet&\bullet \\ 0&\bullet&\bullet&\bullet&\bullet \\ 0&\bullet&\bullet&\bullet&\bullet \end{bmatrix} \overset{H_2\times}{\rightarrow} \begin{bmatrix} *&*&*&*&* \\ *&*&*&*&* \\ 0&\bullet&\bullet&\bullet&\bullet \\ 0&0&\bullet&\bullet&\bullet \\ 0&0&\bullet&\bullet&\bullet \end{bmatrix} \overset{\times H_2^T}{\rightarrow} \begin{bmatrix} *&*&\bullet&\bullet&\bullet \\ *&*&\bullet&\bullet&\bullet \\ 0&*&\bullet&\bullet&\bullet \\ 0&0&\bullet&\bullet&\bullet \\ 0&0&\bullet&\bullet&\bullet \end{bmatrix} \overset{H_3\times}{\rightarrow} \begin{bmatrix} *&*&*&*&* \\ *&*&*&*&* \\ 0&*&*&*&* \\ 0&0&\bullet&\bullet&\bullet \\ 0&0&0&\bullet&\bullet \end{bmatrix} \overset{\times H_3^T}{\rightarrow} \begin{bmatrix} *&*&*&\bullet&\bullet \\ *&*&*&\bullet&\bullet \\ 0&*&*&\bullet&\bullet \\ 0&0&*&\bullet&\bullet \\ 0&0&0&\bullet&\bullet \end{bmatrix}$$

The QR algorithm

The “plain vanilla” version of the QR algorithm is as follows. Set $A_0 = A$. For $k = 0, 1, \ldots$ calculate the QR factorization $A_k = Q_k R_k$ (here Q_k is $n \times n$ orthogonal and R_k is $n \times n$ upper triangular) and set $A_{k+1} = R_k Q_k$. The eigenvalues of A_{k+1} are the same as the eigenvalues of A_k, since we have

$$A_{k+1} = R_k Q_k = Q_k^{-1}(Q_k R_k) Q_k = Q_k^{-1} A_k Q_k, \tag{2}$$

a similarity transformation. Moreover, $Q_k^{-1} = Q_k^T$, therefore if A_k is symmetric, then so is A_{k+1}.

If for some $k \geq 0$ the matrix A_{k+1} can be regarded as "deflated", i.e. it has the block form

$$A_{k+1} = \begin{bmatrix} B & C \\ D & E \end{bmatrix},$$

where B, E are square and $D \approx 0$, then we calculate the eigenvalues of B and E separately (again, with QR, except that there is nothing to calculate for 1×1 and 2×2 blocks). As it turns out, such a "deflation" occurs surprisingly often.

The QR iteration for upper Hessenberg matrices

If A_k is upper Hessenberg, then its QR factorization by means of the Givens rotations produces the matrix

$$R_k = Q_k^T A_k = \Omega^{[n-1,n]} \dots \Omega^{[2,3]} \Omega^{[1,2]} A_k \,,$$

which is upper triangular. The QR iteration sets $A_{k+1} = R_k Q_k = R_k \Omega^{[1,2]T} \Omega^{[2,3]T} \dots \Omega^{[n-1,n]T}$, and it follows that A_{k+1} is also upper Hessenberg, because

$$\begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix} \overset{\times\Omega^{[1,2]^T}}{\longrightarrow} \begin{bmatrix} \bullet & \bullet & * & * \\ \bullet & \bullet & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix} \overset{\times\Omega^{[2,3]T}}{\longrightarrow} \begin{bmatrix} * & \bullet & \bullet & * \\ * & \bullet & \bullet & * \\ 0 & \bullet & \bullet & * \\ 0 & 0 & 0 & * \end{bmatrix} \overset{\times\Omega^{[3,4]T}}{\longrightarrow} \begin{bmatrix} * & * & \bullet & \bullet \\ * & * & \bullet & \bullet \\ 0 & * & \bullet & \bullet \\ 0 & 0 & \bullet & \bullet \end{bmatrix}$$

Thus a strong advantage of bringing A to the upper Hessenberg form initially is that then, in every iteration in QR algorithm, Q_k is a product of just $n-1$ Givens rotations. Hence each iteration of the QR algorithm requires just $\mathcal{O}(n^2)$ operations.