CITED REFERENCES AND FURTHER READING:

Wilkinson, J.H., and Reinsch, C. 1971, *Linear Algebra*, vol. II of *Handbook for Automatic Computation* (New York: Springer-Verlag). [1]

Golub, G.H., and Van Loan, C.F. 1989, *Matrix Computations*, 2nd ed. (Baltimore: Johns Hopkins University Press), §7.5.

Smith, B.T., et al. 1976, *Matrix Eigensystem Routines — EISPACK Guide*, 2nd ed., vol. 6 of Lecture Notes in Computer Science (New York: Springer-Verlag). [2]

11.7 Improving Eigenvalues and/or Finding Eigenvectors by Inverse Iteration

The basic idea behind inverse iteration is quite simple. Let $\mathbf{y}$ be the solution of the linear system

$$(\mathbf{A} - \tau\mathbf{1}) \cdot \mathbf{y} = \mathbf{b} \tag{11.7.1}$$

where $\mathbf{b}$ is a random vector and τ is close to some eigenvalue λ of $\mathbf{A}$. Then the solution $\mathbf{y}$ will be close to the eigenvector corresponding to λ. The procedure can be iterated: Replace $\mathbf{b}$ by $\mathbf{y}$ and solve for a new $\mathbf{y}$, which will be even closer to the true eigenvector.

We can see why this works by expanding both $\mathbf{y}$ and $\mathbf{b}$ as linear combinations of the eigenvectors $\mathbf{x}_j$ of $\mathbf{A}$:

$$\mathbf{y} = \sum_j \alpha_j \mathbf{x}_j \qquad \mathbf{b} = \sum_j \beta_j \mathbf{x}_j \tag{11.7.2}$$

Then (11.7.1) gives

$$\sum_j \alpha_j (\lambda_j - \tau)\mathbf{x}_j = \sum_j \beta_j \mathbf{x}_j \tag{11.7.3}$$

so that

$$\alpha_j = \frac{\beta_j}{\lambda_j - \tau} \tag{11.7.4}$$

and

$$\mathbf{y} = \sum_j \frac{\beta_j \mathbf{x}_j}{\lambda_j - \tau} \tag{11.7.5}$$

If τ is close to λ_n, say, then provided β_n is not accidentally too small, $\mathbf{y}$ will be approximately $\mathbf{x}_n$, up to a normalization. Moreover, each iteration of this procedure gives another power of $\lambda_j - \tau$ in the denominator of (11.7.5). Thus the convergence is rapid for well-separated eigenvalues.

Suppose at the kth stage of iteration we are solving the equation

$$(\mathbf{A} - \tau_k \mathbf{1}) \cdot \mathbf{y} = \mathbf{b}_k \tag{11.7.6}$$

where $\mathbf{b}_k$ and τ_k are our current guesses for some eigenvector and eigenvalue of interest (let's say, $\mathbf{x}_n$ and λ_n). Normalize $\mathbf{b}_k$ so that $\mathbf{b}_k \cdot \mathbf{b}_k = 1$. The exact eigenvector and eigenvalue satisfy

$$\mathbf{A} \cdot \mathbf{x}_n = \lambda_n \mathbf{x}_n \tag{11.7.7}$$

so

$$(\mathbf{A} - \tau_k \mathbf{1}) \cdot \mathbf{x}_n = (\lambda_n - \tau_k)\mathbf{x}_n \tag{11.7.8}$$

Since $\mathbf{y}$ of (11.7.6) is an improved approximation to $\mathbf{x}_n$, we normalize it and set

$$\mathbf{b}_{k+1} = \frac{\mathbf{y}}{|\mathbf{y}|} \tag{11.7.9}$$

We get an improved estimate of the eigenvalue by substituting our improved guess $\mathbf{y}$ for $\mathbf{x}_n$ in (11.7.8). By (11.7.6), the left-hand side is $\mathbf{b}_k$, so calling λ_n our new value τ_{k+1}, we find

$$\tau_{k+1} = \tau_k + \frac{1}{\mathbf{b}_k \cdot \mathbf{y}} \tag{11.7.10}$$

While the above formulas look simple enough, in practice the implementation can be quite tricky. The first question to be resolved is *when* to use inverse iteration. Most of the computational load occurs in solving the linear system (11.7.6). Thus a possible strategy is first to reduce the matrix $\mathbf{A}$ to a special form that allows easy solution of (11.7.6). Tridiagonal form for symmetric matrices or Hessenberg for nonsymmetric are the obvious choices. Then apply inverse iteration to generate all the eigenvectors. While this is an $O(N^3)$ method for symmetric matrices, it is many times less efficient than the QL method given earlier. In fact, even the best inverse iteration packages are less efficient than the QL method as soon as more than about 25 percent of the eigenvectors are required. Accordingly, inverse iteration is generally used when one already has good eigenvalues and wants only a few selected eigenvectors.

You can write a simple inverse iteration routine yourself using LU decomposition to solve (11.7.6). You can decide whether to use the general LU algorithm we gave in Chapter 2 or whether to take advantage of tridiagonal or Hessenberg form. Note that, since the linear system (11.7.6) is nearly singular, you must be careful to use a version of LU decomposition like that in §2.3 which replaces a zero pivot with a very small number.

We have chosen not to give a general inverse iteration routine in this book, because it is quite cumbersome to take account of all the cases that can arise. Routines are given, for example, in [1,2]. If you use these, or write your own routine, you may appreciate the following pointers.

One starts by supplying an initial value τ_0 for the eigenvalue λ_n of interest. Choose a random normalized vector $\mathbf{b}_0$ as the initial guess for the eigenvector $\mathbf{x}_n$, and solve (11.7.6). The new vector $\mathbf{y}$ is bigger than $\mathbf{b}_0$ by a "growth factor" $|\mathbf{y}|$, which ideally should be large. Equivalently, the change in the eigenvalue, which by (11.7.10) is essentially $1/|\mathbf{y}|$, should be small. The following cases can arise:

- If the growth factor is too small initially, then we assume we have made a "bad" choice of random vector. This can happen not just because of a small β_n in (11.7.5), but also in the case of a defective matrix, when (11.7.5) does not even apply (see, e.g., [1] or [3] for details). We go back to the beginning and choose a new initial vector.

- The change $|\mathbf{b}_1 - \mathbf{b}_0|$ might be less than some tolerance ϵ. We can use this as a criterion for stopping, iterating until it is satisfied, with a maximum of 5 – 10 iterations, say.
- After a few iterations, if $|\mathbf{b}_{k+1} - \mathbf{b}_k|$ is not decreasing rapidly enough, we can try updating the eigenvalue according to (11.7.10). If $\tau_{k+1} = \tau_k$ to machine accuracy, we are not going to improve the eigenvector much more and can quit. Otherwise start another cycle of iterations with the new eigenvalue.

The reason we do not update the eigenvalue at every step is that when we solve the linear system (11.7.6) by LU decomposition, we can save the decomposition if τ_k is fixed. We only need do the backsubstitution step each time we update $\mathbf{b}_k$. The number of iterations we decide to do with a fixed τ_k is a trade-off between the quadratic convergence but $O(N^3)$ workload for updating τ_k at each step and the linear convergence but $O(N^2)$ load for keeping τ_k fixed. If you have determined the eigenvalue by one of the routines given earlier in the chapter, it is probably correct to machine accuracy anyway, and you can omit updating it.

There are two different pathologies that can arise during inverse iteration. The first is multiple or closely spaced roots. This is more often a problem with symmetric matrices. Inverse iteration will find only one eigenvector for a given initial guess τ_0. A good strategy is to perturb the last few significant digits in τ_0 and then repeat the iteration. Usually this provides an independent eigenvector. Special steps generally have to be taken to ensure orthogonality of the linearly independent eigenvectors, whereas the Jacobi and QL algorithms automatically yield orthogonal eigenvectors even in the case of multiple eigenvalues.

The second problem, peculiar to nonsymmetric matrices, is the defective case. Unless one makes a "good" initial guess, the growth factor is small. Moreover, iteration does not improve matters. In this case, the remedy is to choose random initial vectors, solve (11.7.6) once, and quit as soon as *any* vector gives an acceptably large growth factor. Typically only a few trials are necessary.

One further complication in the nonsymmetric case is that a real matrix can have complex-conjugate pairs of eigenvalues. You will then have to use complex arithmetic to solve (11.7.6) for the complex eigenvectors. For any moderate-sized (or larger) nonsymmetric matrix, our recommendation is to avoid inverse iteration in favor of a QR method that includes the eigenvector computation in complex arithmetic. You will find routines for this in [1,2] and other places.

CITED REFERENCES AND FURTHER READING:

Acton, F.S. 1970, *Numerical Methods That Work*; 1990, corrected edition (Washington: Mathematical Association of America).

Wilkinson, J.H., and Reinsch, C. 1971, *Linear Algebra*, vol. II of *Handbook for Automatic Computation* (New York: Springer-Verlag), p. 418. [1]

Smith, B.T., et al. 1976, *Matrix Eigensystem Routines — EISPACK Guide*, 2nd ed., vol. 6 of Lecture Notes in Computer Science (New York: Springer-Verlag). [2]

Stoer, J., and Bulirsch, R. 1980, *Introduction to Numerical Analysis* (New York: Springer-Verlag), p. 356. [3]