Rayleigh quotient iteration

From Wikipedia, the free encyclopedia

Rayleigh quotient iteration is an eigenvalue algorithm which extends the idea of the inverse iteration by using the Rayleigh quotient to obtain increasingly accurate eigenvalue estimates.

Rayleigh quotient iteration is an iterative method, that is, it must be repeated until it converges to an answer (this is true for all eigenvalue algorithms). Fortunately, very rapid convergence is guaranteed and no more than a few iterations are needed in practice. The Rayleigh quotient iteration algorithm converges cubically for Hermitian or symmetric matrices, given an initial vector that is sufficiently close to an eigenvector of the matrix that is being analyzed.

Contents [hide]

Algorithm [edit]

The algorithm is very similar to inverse iteration, but replaces the estimated eigenvalue at the end of each iteration with the Rayleigh quotient. Begin by choosing some value μ_0 as an initial eigenvalue guess for the Hermitian matrix A. An initial vector b_0 must also be supplied as initial eigenvector guess.

Calculate the next approximation of the eigenvector b_{i+1} by

$$b_{i+1} = \frac{(A-\mu_i I)^{-1} b_i}{||(A-\mu_i I)^{-1} b_i||},$$

where I is the identity matrix, and set the next approximation of the eigenvalue to the Rayleigh quotient of the current iteration equal to

$$\mu_i = \frac{b_i^* A b_i}{b_i^* b_i}.$$

To compute more than one eigenvalue, the algorithm can be combined with a deflation technique.

Example [edit]

Consider the matrix

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 1 \\ 3 & 2 & 1 \end{bmatrix}$$

for which the exact eigenvalues are $\lambda_1 = 3+\sqrt{5}$, $\lambda_2 = 3-\sqrt{5}$ and $\lambda_3 = -2$, with corresponding eigenvectors

$$v_1 = \begin{bmatrix} 1 \\ \varphi-1 \\ 1 \end{bmatrix}, v_2 = \begin{bmatrix} 1 \\ -\varphi \\ 1 \end{bmatrix} \text{ and } v_3 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}.$$

(where $\varphi = \frac{1+\sqrt{5}}{2}$ is the golden ratio).

The largest eigenvalue is $\lambda_1 \approx 5.2361$ and corresponds to any eigenvector proportional to

$$v_1 \approx \begin{bmatrix} 1 \\ 0.6180 \\ 1 \end{bmatrix}.$$

We begin with an initial eigenvalue guess of

$$b_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \mu_0 = 200.$$

Then, the first iteration yields

$$b_1 \approx \begin{bmatrix} -0.57927 \\ -0.57348 \\ -0.57927 \end{bmatrix}, \quad \mu_1 \approx 5.3355$$

the second iteration,

$$b_2 \approx \begin{bmatrix} 0.64676 \\ 0.40422 \\ 0.64676 \end{bmatrix}, \quad \mu_2 \approx 5.2418$$

and the third,

$$b_3 \approx \begin{bmatrix} -0.64793 \\ -0.40045 \\ -0.64793 \end{bmatrix}, \quad \mu_3 \approx 5.2361$$

from which the cubic convergence is evident.

Octave Implementation [edit]

The following is a simple implementation of the algorithm in Octave.

```
function x = rayleigh(A,epsilon,mu,x)
  x = x / norm(x);
  y = (A-mu*eye(rows(A))) \ x;
  lambda = y'*x;
  mu = mu + 1 / lambda
  err = norm(y-lambda*x) / norm(y)
  while err > epsilon
    x = y / norm(y);
    y = (A-mu*eye(rows(A))) \ x;
    lambda = y'*x;
    mu = mu + 1 / lambda
    err = norm(y-lambda*x) / norm(y)
  end
end
```

See also [edit]

- Power iteration
- Inverse iteration

References [edit]

- Lloyd N. Trefethen and David Bau, III, *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, 1997. ISBN 0-89871-361-7.
- Rainer Kress, "Numerical Analysis", Springer, 1991. ISBN 0-387-98408-9