Matrix Functions in the NAG Library

Further functionality has been added the suite of matrix function routines for Mark 24 of the NAG Library. In this article we will briefly discuss some of the theoretical background and applications of the functions of matrices and give an overview of the functionality in the NAG Library.

Motivation

The ordinary differential equation

$$\frac{dy}{dt} = ay,$$

where a is a constant scalar, has the general solution $y = y_0 \exp(at)$. Suppose instead that we replace y with a vector $\boldsymbol{y}$ and the scalar a with a square matrix A. If the definition of the exponential could be extended to square matrices, then the solution could be written succinctly as $y = y_0 \exp(tA)$.

Defining Matrix Functions

Given a scalar function *f(x),* the matrix function *f(A)* can be defined using the Taylor series expansion of *f* (there are several other equivalent definitions, but this one is the most intuitive). In the case of the matrix exponential, for example, we can define

$$\exp A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots$$

Provided that the eigenvalues of *A* lie within the radius of convergence of *f,* the Taylor series will converge when evaluated at *A*.

Some matrix functions are multivalued. For example, a matrix logarithm of *A* can be defined to be any solution *X* to the equation $A = \exp(X)$. Similarly, a p^{th} root of *A* is any solution to $X^p = \mathrm{A}$. For matrices with no eigenvalues on the negative real line, it is possible to define a unique *principal logarithm* and a unique *principal* p^{th} *root* whose eigenvalues lie within certain regions of the complex plane.

Applications of Matrix Functions

Matrix functions play an important role in financial mathematics, where Markov chains are used to model phenomena such as asset prices. Such models are governed by a *transition probability matrix, P(t),* whose *(i,j)* entry is equal to the probability that an individual in state *i* will move to state *j* in a given time step *t*. Given *P(t),* the transition probability matrix for a smaller time step *t/p* can be obtained by computing p^{th} roots of *P(t).* An associated matrix is the *transition intensity matrix, Q(t),* defined via $P(t) = \exp(Q(t))$. Similar applications of matrix functions can be found in population models of mathematical biology.

Matrices can be used to represent graphs. For example, given a network of *n* nodes, suppose that the *(i,j)* element of *A* is equal to 1 if nodes *i* and *j* are connected and 0 otherwise. Then A^m is the matrix containing the number of routes of length *m* between nodes. The matrix exponential $\exp(A)$

can then be used as a measure of the “connectedness” of the graph by weighting in favour of shorter routes.

There are many other applications of matrix functions, including optics, control theory, particle physics, computer graphics and NMR spectroscopy.

Computing Matrix Functions

If A has a full set of eigenvectors *V*, then it can be factorized as

$$A = VDV^{-1},$$

where *D* is the diagonal matrix whose diagonal elements, d_i , are the eigenvalues of *A.* The matrix function *f(A)* is then given by

$$f(A) = Vf(D)V^{-1},$$

where *f(D)* is the diagonal matrix whose *i*th diagonal element is $f(d_i)$. In general, however, this method is unstable and computing a function of a matrix is a nontrivial problem. Naively evaluating the Taylor series can be highly inefficient and can introduce numerical errors due to the limitations of floating-point arithmetic. At the University of Manchester, a team lead by Professor Nick Higham has developed many state-of-the-art algorithms for computing matrix functions and these algorithms have been implemented in the NAG Library through a Knowledge Transfer Partnership.

Matrix Functions in the NAG Library

One of the most commonly encountered matrix functions is the exponential. This is available via f01ec and f01fc for real and complex matrices respectively. If the matrix is real symmetric or complex Hermitian then f01ed and f01fd will take advantage of the symmetry. If the *action* of the exponential is required on a matrix, that is $\exp(tA)B$, then f01ga and f01ha are available. These routines avoid explicitly forming the matrix exponential. If *A* is not stored in dense form then the reverse communication equivalents, f01gb and f01hb, can be used that require the user to compute matrix products involving *A*. The principal matrix logarithm is computed by f01ej and f01fj. The routines f01ek and f01fk compute the exponential, sine, cosine, or the hyperbolic sine or cosine of general real and complex matrices.

Mark 24 also contains general purpose matrix function routines. The user need only provide a scalar function to f01ef and f01ff to compute a function of a real symmetric or a complex Hermitian matrix respectively. For general matrices the required derivatives are either computed by the user supplied function, f01em and f01fm, or computed via numerical differentiation, f01el and f01fl.

New routines have been added to compute matrix square roots. They are f01en* that computes a real matrix square root and f01ep* specifically for real upper quasi-triangular matrices, which have may have some 2x2 blocks on the diagonal. The complex counterparts are f01fn* and f01fp*. Routines are also proved for general powers of a matrix, f01eq* for real matrices and f01fq* for complex. They compute the p^{th} power, for any real *p*.

The *condition number* of a matrix function is a measure of the sensitivity of the computed solution to small changes in the input data. The Library contains many routines for estimating matrix function condition numbers along with the function value. Another interesting quantity that can be computed is the Fréchet Derivative, which describes the first-order effect on the matrix function of perturbations to the input matrix.

* These routines are initially available for the NAG C Library (Mark 24).