Matrix Functions in the NAG Library

Further functionality has been added to Mark 24 of the NAG Fortran Library in the area of matrix functions. In this article we will briefly discuss some of the theoretical background concerning functions of matrices. We will then describe some of the many applications that matrix functions have found in science and engineering, due to the succinct way that they allow problems to be formulated and solutions to be expressed. Finally we will give an overview of the new 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 $\mathbf{y}$ 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 using the NAG Fortran Library

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. Even 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. A Knowledge Transfer Partnership has been set up between NAG and the University of Manchester to implement these algorithms in the NAG Library.

One of the most commonly encountered matrix functions is the exponential. This is available via f01ecf and f01fcf for real and complex matrices respectively. If the matrix is real symmetric or complex Hermitian then f01edf and f01fdf will take advantage of the symmetry. The principal matrix logarithm is computed by f01ejf and f01fjf.

Mark 24 also contains a selection of general purpose matrix function routines. f01eff and f01fff will compute a user-provided function of a real symmetric or a complex Hermitian matrix respectively. f01ekf and f01fkf compute the exponential, sine, cosine, or the hyperbolic sine or cosine of general real and complex matrices. Finally f01elf, f01emf, f01flf and f01fmf allow the user to provide a scalar function via a subroutine which returns function values, before computing the corresponding matrix function. The required derivatives are either returned by the user supplied subroutine or computed via numerical differentiation.

The *condition number* of a matrix function is a measure of the sensitivity of the computed solution to small changes in the input data. FL24 contains six routines for estimating matrix function condition numbers. f01jaf and f01kaf return condition numbers for the exponential, logarithm, sine, cosine, or hyperbolic sine or cosine of general real and complex matrices. f01jbf, f01jcf, f01kbf and f01kcf return condition numbers for matrix functions corresponding to user-supplied subroutines, using either user supplied derivatives or numerical differentiation.

Future marks of the NAG Library will include further functionality such as square roots, p^{th} powers for non-integer *p* and Fréchet derivatives.