Summary of "Nineteen Dubious Ways to Compute the Matrix Exponential, Twenty-Five Years Later"

Kyle Kastner

September 12, 2013

Why Do We Care?

Linear, time invariant systems can be described by the equations

$$\dot{x}(t) = Ax(t) x(0) = x_0 \tag{1}$$

where A is an n-by-n matrix of real or complex values. This means that solutions to this system of equations can be described with

$$\dot{x}(t) = e^{At} x_0 \tag{2}$$

Solving these equations algorithmically requires an efficient way of calculating the matrix exponential accurately. The article "Nineteen Dubious Ways to Compute the Matrix Exponential, Twenty-Five Years Later", describes many different ways to perform this calculation, trading off algorithm characteristics in attempt to balance efficiency, memory requirements, and computational complexity. While this paper focuses on "in memory" sized matrices, there are

different algorithms for distributed processing of matrices, with a special focus on methods for sparse matrices. One of the most famous examples may be the Google PageRank algorithm, though there are other useful applications for distributed processing of sparse matrices.

Identifying Characteristics

Generality
: Applicable to a variety of matrix types

Reliability
: Gives warning of excessive errors

Stability
Does not introduce additional sensitivity to changes

Accuracy
Error introduced by by truncation of the series

Efficiency
Algorithmic complexity, usually described in "big O" notation or "flops".

Near-singular matrices will be contaminated with higher error most of the time, regardless of the attributes of the algorithm of choice. There is also a "stability" term related to the decay of the e^{At} - if e^{At} decays to 0 as t goes to infinity, then the e^{At} term can be considered stable. Eigenvalue and eigenvector

algorithms tend to be more efficient, but suffer serious issues when eigenvalues are not unique. Round off errors and near identical eigenvalues can also cause inaccuracy.

MATLAB

There are also three methods shown in pure MATLAB, in addition to discussions about EigTool and ExpoKit for computing matrix exponentials.

Conclusions

There is no "best" method for all matrices - rather, the method chosen should be based on the type of input data expected. These expectations, along with tolerances to instability and error, should help guide the algorithm selection process. Polynomial and splitting approaches are typically reserved for niche application with unique matrix classes, so the typical approaches will be either series (scaling and squaring), specialized ODE methods, or matrix decompositions. Decompositions are generally efficient ($O(n^3)$ for the examples shown)

and are an area of active research, making matrix decompositions a preferred approach. However, the MATLAB builtin $expm()$ is speculated to use a scaling and squaring algorithm with Pade approximation, so decompositions are not always the "right answer"!