Matlab codes to compute the matrix exponential with an optimized Taylor polynomial approximation

Philipp Bader Sergio Blanes Fernando Casas

November 27, 2018

In the paper

- P. Bader, S. Blanes, F. Casas, *Computing the matrix exponential with an optimized Taylor polynomial approximation*

we propose a new implementation of the Taylor polynomial approximating the exponential of a matrix in such a way that the number of matrix products is reduced with respect to the Patterson-Stockmeyer technique. This technique is used in combination with the scaling and squaring procedure in such a way that the algorithm is superior in performance to the standard approach based on Padé approximants, such as it is used by the Matlab function `expm`.

We next list the provided functions and their usage to generate the graphics in the manuscript. The code has been tested with Matlab 2016a and Mathematica 10.

1 Generate graphics

`generate_data.m`
This Matlab script runs core functions to generate sets of test matrices and compute the exponentials thereof. Notice that the reference solution has to be found separately using Mathematica or else.

`create_all_figures.m`
This Matlab script calls core functions below to generate all experimental figures in the manuscript once the computations have been performed and the results stored. The figures are exported using matlab2tikz.

2 Core functions

`generate_matrices.m`
This Matlab function generates the test matrices for the plots and stores them on the disk for further manipulation. Notice that the random seed it fixed to 0 to allow reproducible results. The input arguments are: dimension of the random matrices, number of matrices, and an offset since the type of matrix that is generated depends on the running index. For details, see the implementation. E.g. an offset of 700 skips any special matrices and goes directly to random matrices.

`compute_exponentials.m`
This Matlab script tests a set of methods for the exponential on the test matrices generated using `generate_matrices.m`. For large matrices, they are generated on the fly since they would occupy too much disk space using the same routine.

`plot_performance_profile.m`

Plots performance profiles for different methods with data loaded generated through the functions above.

`plot_vs_condition.m`
Plots error and other information vs the condition number.

`generate_reference_solutions.nb`
This Mathematica notebook has been used to load the matrix data generated above, compute a reference exponential and store it to disk.

3 Helper functions

`load_reference.m`
This script converts the reference solution from Mathematica which was saved as a text file to a Matlab cell structure. Essentially, it is an import function.

4 The algorithms

The folder `expms` contains several files containing Matlab implementations of the algorithms tested in the manuscript, namely:

- `expm2.m`: the simplest implementation of the new algorithm based on Taylor polynomials (algorithm **expm2** in the text).
- `expm3.m`: a most sophisticated implementation to deal with overscaling based on Taylor polynomials (algorithm **expm3** in the text).
- `expm2005.m`: the simplest implementation of the scaling and squaring procedure with Padé approximants (`expm` in Matlab from R2013a).
- `Higham2009paper.m`: algorithm based on Padé approximants and norm estimators to avoid overscaling (called **Higham2009[2]** in the manuscript).
- `expm_with_output.m`: algorithm based on Padé approximants, norm estimators to avoid overscaling and special devices to deal with small matrices, symmetric matrices, Schur-decompositions and block structures (called **expm2016** in the manuscript, `expm` in Matlab R2016a).