Contribution ID: **15** Type: **Plenary**

Computational graphs for matrix functions

Wednesday, 27 September 2023 09:00 (45 minutes)

Numerical methods for evaluating a function f at an $n \times n$ matrix A can be based on a variety of different approaches, but for a large class of algorithms the matrix $f(A)$ is approximated by using only three operations:

1. $Z \leftarrow c_X X + c_Y Y$, linear combination of matrices),
2. $Z \leftarrow X \cdot Y$, (matrix multiplication),
3. $Z \leftarrow X^{-1}Y$, (solution of a linear system with n right-hand sides),

where X, Y, and Z are $n \times n$ matrices and c_X and c_Y are scalars.

Algorithms that combine only these three basic building blocks are particularly attractive, as they correspond to functions that are easy to work with: if an expression for the scalar function g features only linear combinations, multiplications, and inversions, and g is defined on the spectrum of A, then a formula for $g(A)$ can be obtained by replacing all occurrences of z in the formula for $g(z)$ with A.

Rephrasing these methods as directed acyclic graphs (DAGs) is a particularly effective approach to study existing techniques, improve them, and eventually derive new ones. The accuracy of these matrix techniques can be characterized by the accuracy of their scalar counterparts, thus designing algorithms for matrix functions can be regarded as a scalar-valued optimization problem. The derivatives needed during the optimization can be calculated automatically by exploiting the structure of the DAG, in a fashion analogous to backpropagation.

GraphMatFun.jl is a Julia package that offers the means to generate and manipulate computational graphs, optimize their coefficients, and generate Julia, MATLAB, and C code to evaluate them efficiently at a matrix argument. The software also provides tools to estimate the accuracy of a graph-based algorithm and thus obtain numerically reliable methods. For the exponential, for example, using a particular form (degree-optimal) of polynomials produces implementations that in many cases are cheaper, in terms of computational cost, than the Padé-based techniques typically used in mathematical software.

Primary author: FASI, Massimiliano (Durham University)

Co-authors: Prof. JARLEBRING, Elias (KTH Royal Institute of Technology); Dr RINGH, Emil (Ericsson Research)

Presenter: FASI, Massimiliano (Durham University)

Session Classification: Invited talks