Computation of matrix functions with fully automatic Schur-Parlett and Rational Krylov methods.

Roberto Zanotto | PD2GGALN 2018

About me

Computer Science student at the University of Pisa.

Master's degree thesis under supervision of professor Federico Poloni.

MatFun

A Julia package for computing dense and sparse matrix functions automatically (no user input other than f and A).

https://github.com/robzan8/MatFun.jl

Matrix Functions

$\sin(A), \exp(A), f(A)$ ... where A is a square matrix.

Key factor: values of f and its derivatives on A's spectrum.

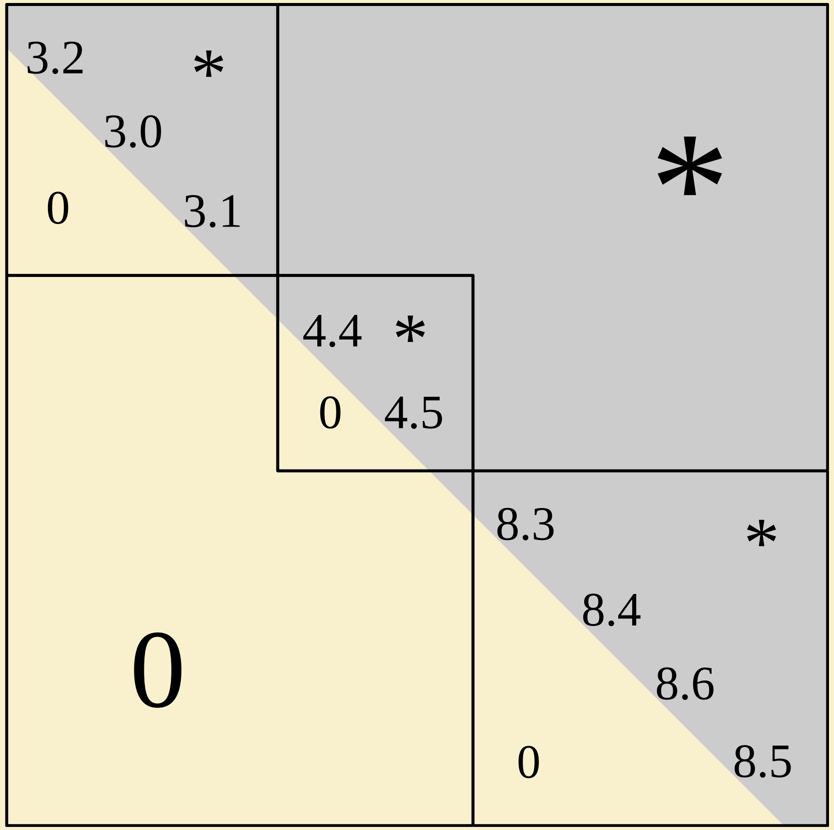
Some specialized methods ($expm, logm, sqrtm$), we are looking for generic ones.

Schur-Parlett – dense matrix functions

$A = QTQ^* \Rightarrow f(A) = Qf(T)Q^*$

Need to compute $F = f(T)$:

- Group eigenvalues in blocks by proximity;
- Compute $f(T_{ii})$ of diagonal blocks with Taylor;
- Use the Parlett recurrence: $T_{ii}\boldsymbol{F_{ij}} - \boldsymbol{F_{ij}}T_{jj} = F_{ii}T_{ij} - T_{ij}F_{jj} + \sum_{k=i+1}^{j-1}(F_{ik}T_{kj} - T_{ik}F_{kj})$



Schur-Parlett – automatic differentiation

Derivatives of f are required for Taylor.

Dual numbers: $x + y\varepsilon$ with $\varepsilon \neq 0, \varepsilon^2 = 0$

$$f(x + y\varepsilon) = f(x) + f'(x)y\varepsilon \Rightarrow$$

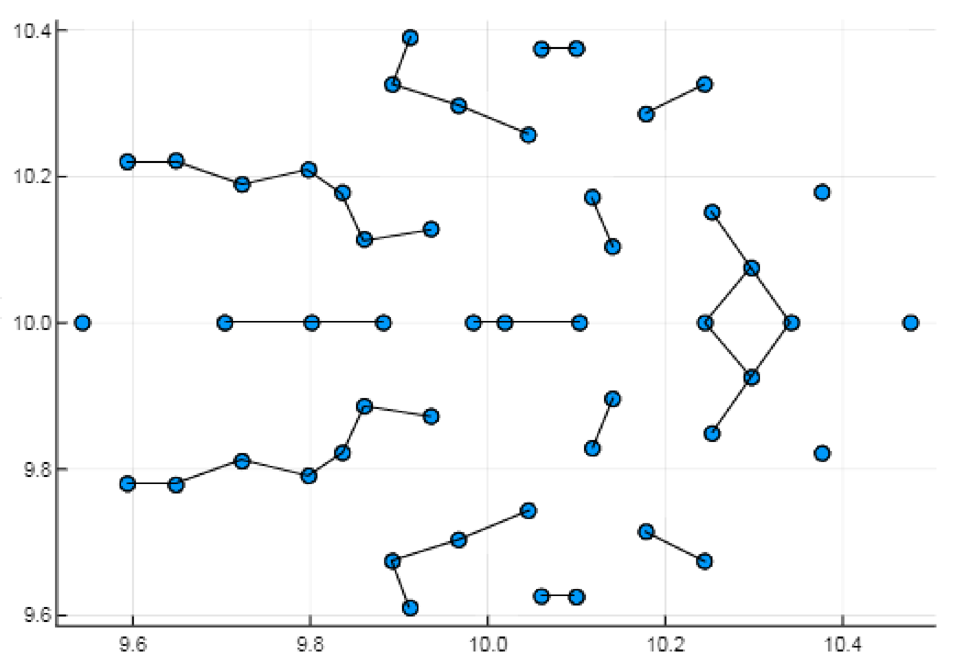
$$f'(x) = Eps(f(x + \varepsilon))$$

In Julia, a function accepting any real number also accepts dual numbers and is therefore automatically differentiable.

We use TaylorSeries.jl for higher-order differentiation.

Schur-Parlett – numerical accuracy

Boils down to how well the eigenvalues can be clustered:



$\Rightarrow$

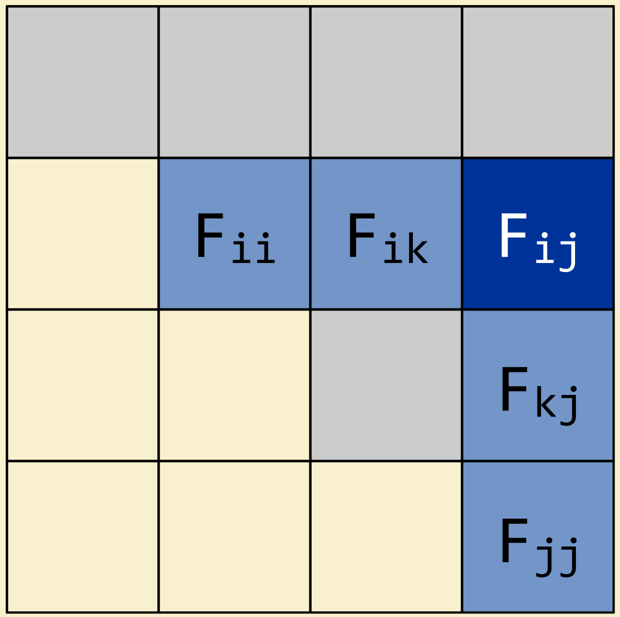
Relative error versus MatLab's Symbolic Toolbox (on 50x50 random matrix):

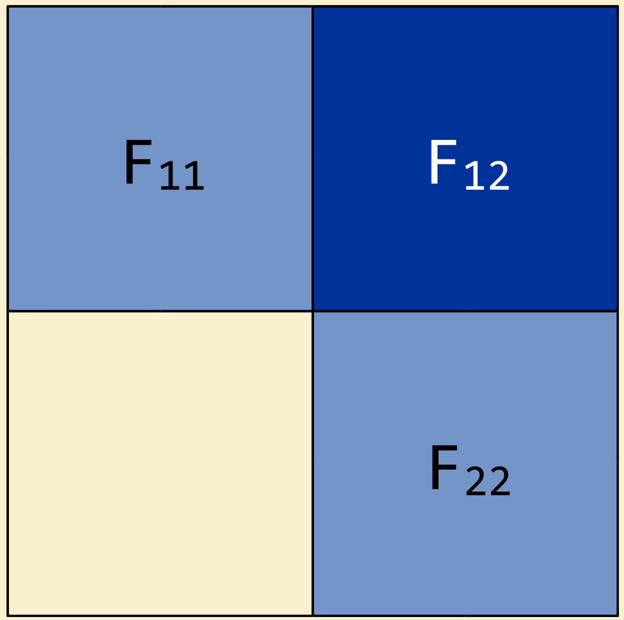
	Specialized method	Schur-Parlett
exp	1.0e-14	1.5e-13
log	4.0e-14	4.1e-14
sqrt	4.3e-14	4.5e-14

Can behave badly (7e-4) with "snake" eigenvalues (as shown in original paper, experiment 4)

Schur-Parlett – performance improvements

Parlett recurrence made recursive and cache-oblivious.





$$T_{ii}F_{ij} - F_{ij}T_{jj} = F_{ii}T_{ij} - T_{ij}F_{jj} + \sum_{k=i+1}^{j-1}(F_{ik}T_{kj} - T_{ik}F_{kj})$$

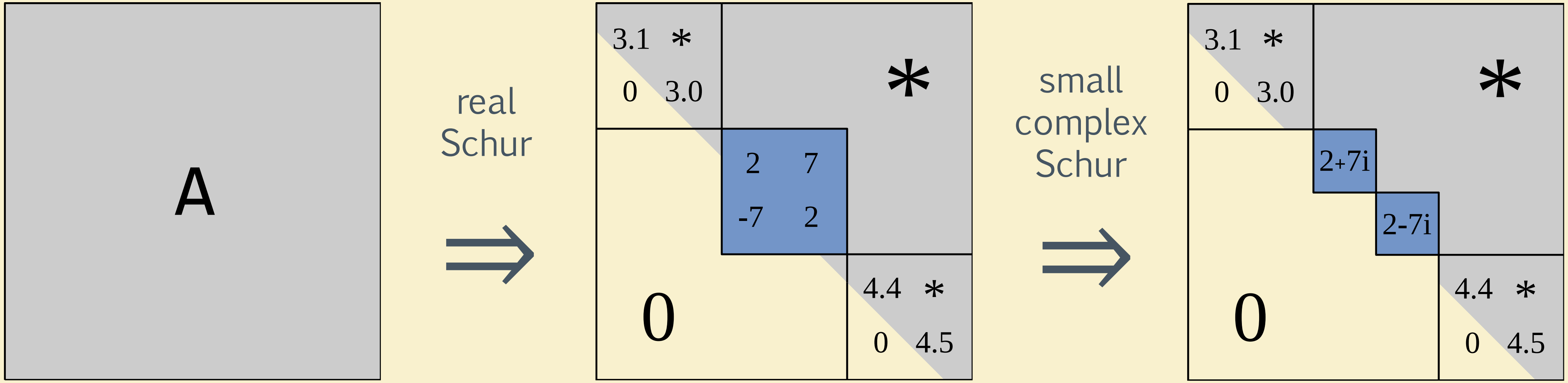
$$T_{11}F_{12} - F_{12}T_{22} = F_{11}T_{12} - T_{12}F_{22}$$

~3x speedup for n = 2500

Problem: conjugated eigenvalues with big imaginary part must go in different blocks, even with real A.

Original solution: do everything in complex arithmetic.

Our solution: complex Schur factorization can be “delayed” and done on small blocks:



Also allows for Parlett recurrence with real Sylvester equations, ~2x speedup for n = 2500 on whole Schur-Parlett.

Schur-Parlett – performance results

The whole procedure typically spends ~2/3 of the time doing A's Schur decomposition (varies depending on the eigenvalues' distribution).