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 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 exist ($expm, logm, sqrtm$),

we are looking for generic ones.

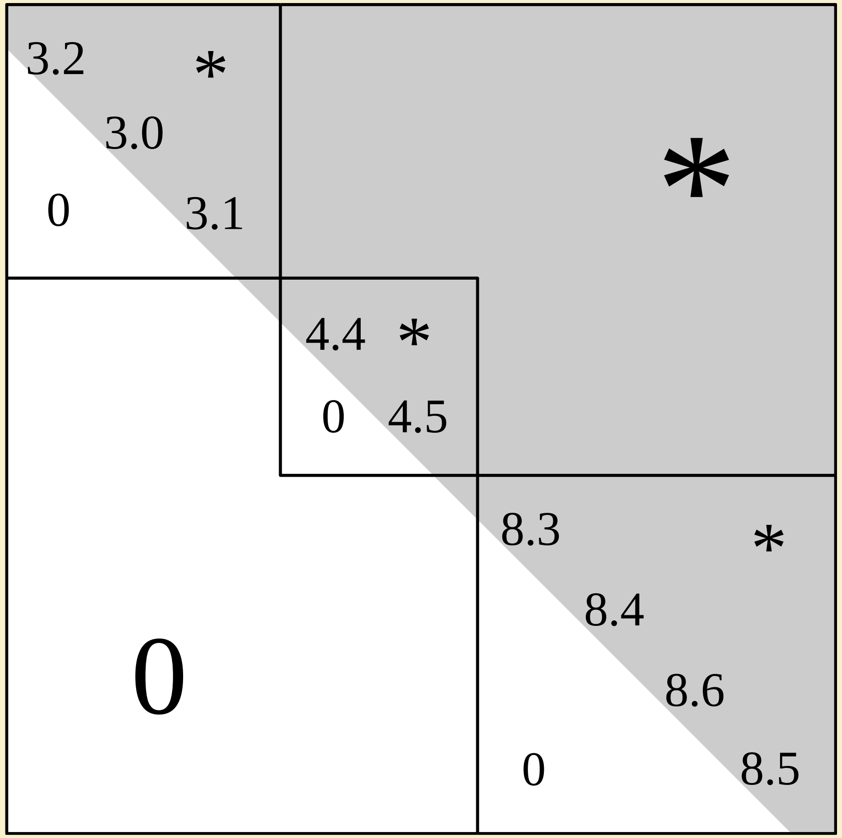
Schur-Parlett – dense matrix functions

Proposed by Higham and Davies in 2003.

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

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 $\varepsilon \neq 0, \varepsilon^2 = 0$

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

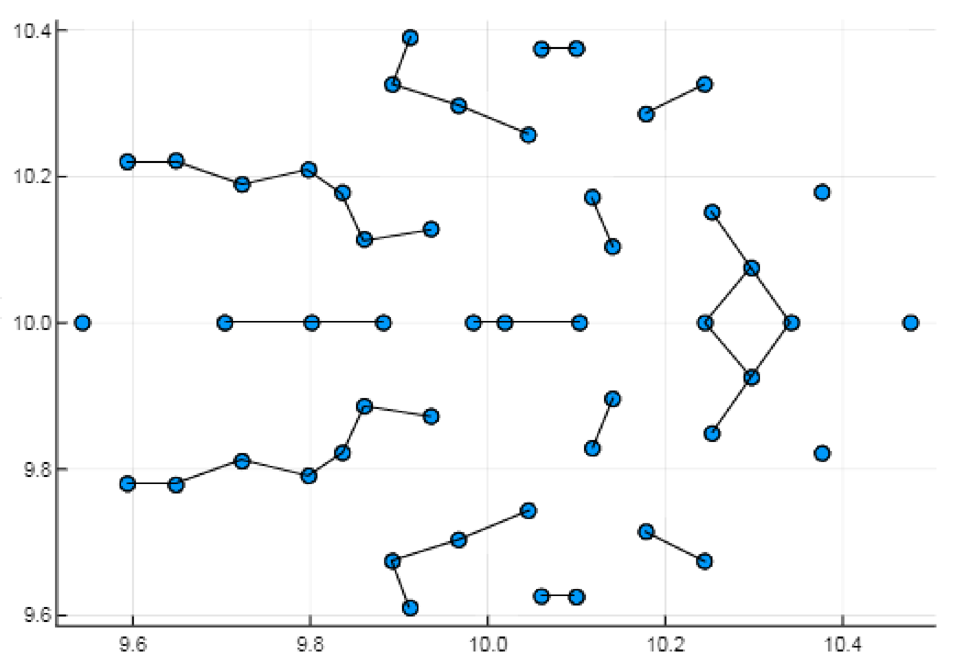
$$f'(x) = Eps(f(x + \varepsilon))$$

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:



⇒

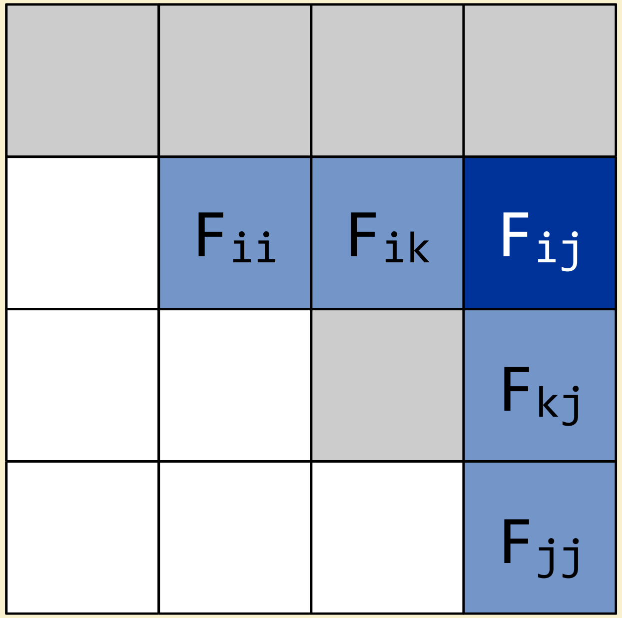
Relative error versus MatLab's Symbolic Toolbox (on 50x50 random matrix):

	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

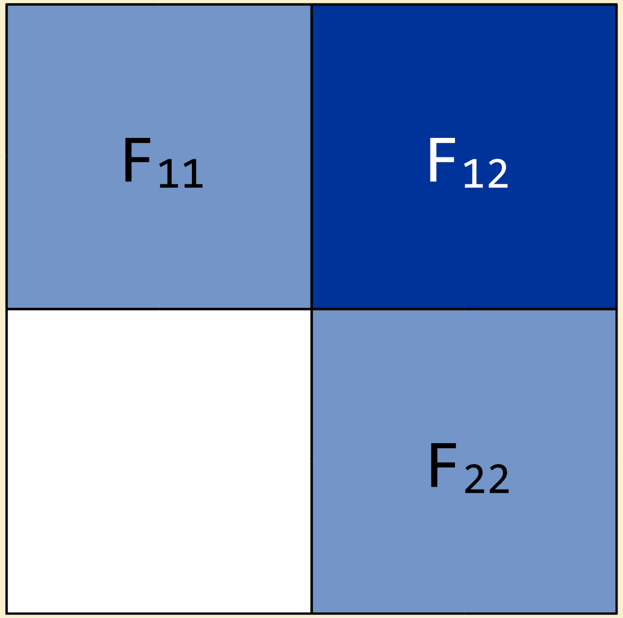
Can behave badly (7e-4) with "snake" eigenvalues (as shown in original paper, experiment 4)

Schur-Parlett – performance improvements

Parlett recurrence made recursive and cache-oblivious.



$\Rightarrow$



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

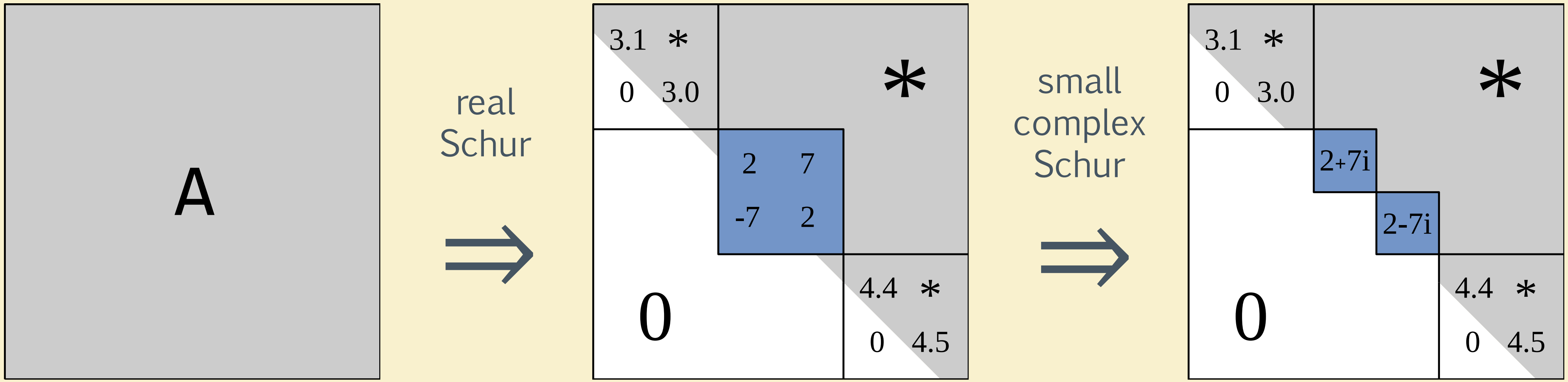
$$T_{11}F_{12} - F_{12}T_{22} = F_{11}T_{12} - T_{12}F_{22}$$

~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.

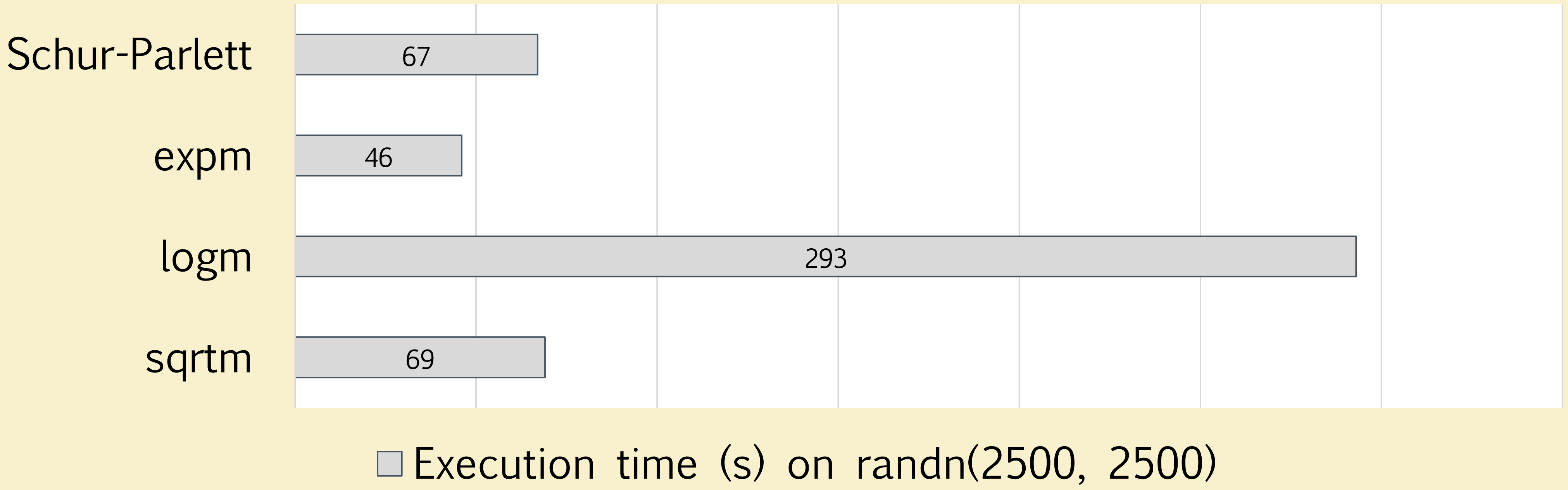
Our solution: complex Schur factorization can be "delayed" and done on small blocks:



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).



Rational Krylov – sparse matrix functions

Proposed by Güttel in 2013.

Goal: compute $f(A)b$ for a sparse A.

Approximate $f(A)b$ with $r_m(A)b$ where $r_m = p_{m-1}/q_{m-1}$

Denominator is factored as $q_{m-1}(z) = \prod_{j=1}^{m-1}(1 - z/\xi_j)$
with poles ξ_j provided by the user.

Rational Krylov space is defined as:

$$Q_m(A, b) = q_{m-1}(A)^{-1} span\{b, Ab, \dots, A^{m-1}b\}$$

Obtained with Ruhe's rational Arnoldi algorithm:

$$v_1 = b/||b||, \quad v_{j+1} = orthonormalize((I - A/\xi_j)^{-1} A v_j)$$

$r_m(A)b$ is computed by projecting A into the Krylov space:

$$A_m = V^* A V, \quad r_m(A)b = V f(A_m) V^*$$

Rational Krylov – approximation accuracy

Depends on two factors:

- How well f can be approximated on A's spectrum by a rational function, hopefully with low degree (ill-posed problem);
- How well we choose the poles ξ_j.

We use the AAA algorithm for rational approximation to find good poles automatically.

AAA algorithm for rational approximation

Proposed by Nakatsukasa, Sète, Trefethen in 2017.

Input: function samples (real or complex).
Output: rational barycentric function of type $(m-1, m-1)$:

$$r(z) = \frac{n(z)}{d(z)} = \sum_{j=1}^{m} \frac{w_j f_j}{z - z_j} \Bigg/ \sum_{j=1}^{m} \frac{w_j}{z - z_j}$$

$r(z_j) = \infty/\infty$ but $\lim_{z \to z_j} r(z) = f_j$

Support/interpolation points (z_j) are chosen incrementally from samples in a greedy way, to avoid instabilities:

next z_j is chosen where $|f(z_j) - r(z_j)|$ is maximized.

After a new support point is found, weights are recomputed to minimize the approximation error:

$$f(z) \approx \frac{n(z)}{d(z)} \rightarrow minimize||f(z)d(z) - n(z)||, \qquad z \in Z^{(m)}$$

Is a least squares problem solvable with SVD:

$$minimize||Aw||, \qquad ||w|| = 1$$

When approximation error is small, we are done.

Poles can be then retrieved by solving a generalized eigenvalue problem, with accuracy up to machine precision.

Performance of Rational Krylov + AAA

Rational Krylov: m times sparse linear system + orthogonalization: $O(m(L + mN))$

AAA: m times SVD: $O(m(m^2 M))$

Setting AAA's number of samples $M = \sim nnz(A)$ balances execution times.

Accuracy of Rational Krylov + AAA

Matrices are from the SuiteSparse Matrix Collection.

$\exp(A)b$ with Krylov is compared with dense *expm*:

Problem type	name	size(A)	cond(A)	# poles	accuracy	# poles	accuracy
2D/3D problem	jagmesh	1089	1168	9	1.5e-6	15	6.0e-14
Fluid dynamics	sherman4	1104	2178	11	6.6e-6	19	6.7e-10
Structural problem	can_1072	1072	2.0e34	11	7.3e-6	23	1.7e-14
Directed graph	SmaGri	1059	Inf	9	2.6e-7	21	7.3e-13

I haven't implemented the estimator of $cond(f(A))$ yet ☹

Try it out at:
https://github.com/robzan8/MatFun.jl

Questions?