Mixed precision recursive block diagonalization for bivariate functions of matrices

Leonardo Robol <leonardo.robol@unipi.it>,
Stefano Massei <s.massei@tue.nl>

Due Giorni di Algebra Lineare, Napoli, 2022.

Univariate vs bivariate matrix functions

We may define univariate matrix functions in several ways, for $f(z) = \sum_{i \geq 0} f_i z^i$,

$$f(A)v = \sum_{i \geq 0} f_i A^i v = \frac{1}{2\pi i} \int_\Gamma f(z)(zI - A)^{-1} v \, dz.$$

Univariate vs bivariate matrix functions

We may define univariate matrix functions in several ways, for $f(z) = \sum_{i \geq 0} f_i z^i$,

$$f(A)v = \sum_{i \geq 0} f_i A^i v = \frac{1}{2\pi i} \int_\Gamma f(z)(zI - A)^{-1} v \, dz.$$

This can be generalized to the bivariate case, by setting for $f(z, w) = \sum_{i,j \geq 0} f_{ij} z^i w^j$:

$$f\{A, B^T\}(C) = \sum_{i,j \geq 0} f_{ij} A^i C B^j = -\frac{1}{4\pi^2} \int_{\Gamma_A} \int_{\Gamma_B} f(z, w)(zI - A)^{-1} C (zI - B^T)^{-1} \, dz \, dw.$$

- Most of the properties of univariate matrix functions carry over to this more general setting.
- Similarities on A, B behave well:

$$f\{A, B^T\}(C) = V \cdot f\{V^{-1}AV, (W^{-1}BW)^T\}(V^{-1}CW) \cdot W^{-1},$$

for any invertible matrices V, W.

Applications

- if X satisfies $AX + XB = C$, then

$$C = f\{A, B^T\}(X), \qquad f(x, y) = x + y,$$

and therefore the solution of a Sylvester equation is expressed as:

$$X = g\{A, B^T\}(C), \qquad g(x, y) = \frac{1}{x + y}.$$

Applications

- if X satisfies $AX + XB = C$, then

$$C = f\{A, B^T\}(X), \qquad f(x, y) = x + y,$$

  and therefore the solution of a Sylvester equation is expressed as:

$$X = g\{A, B^T\}(C), \qquad g(x, y) = \frac{1}{x + y}.$$

- Similar ideas apply for generalized Sylvester equations of the form $p\{A, B^T\}(X) = C$, whose solution is expressed using $f(x, y) := \frac{1}{p(x,y)}$.

Applications

- if X satisfies $AX + XB = C$, then

$$C = f\{A, B^T\}(X), \qquad f(x, y) = x + y,$$

  and therefore the solution of a Sylvester equation is expressed as:

$$X = g\{A, B^T\}(C), \qquad g(x, y) = \frac{1}{x + y}.$$

- Similar ideas apply for generalized Sylvester equations of the form $p\{A, B^T\}(X) = C$, whose solution is expressed using $f(x, y) := \frac{1}{p(x,y)}$.
- $f\{A, A^T\}(H)$ is the Frechét derivative of $g(z)$ at A in the direction H, if $f(z, w)$ is the divided difference of $g(z)$.
- There is a nice connection with Kronecker sums; if $\mathcal{A} = B^T \otimes I + I \otimes A$ then

$$\mathrm{vec}(X) = f(\mathcal{A})(\mathrm{vec}(C)), \implies X = g\{A, B^T\}(C), \qquad g(x, y) = f(x + y).$$

Evaluation in the diagonalizable case

$$f(x,y) = \sum_{ij} f_{ij} x^i y^j \implies f\{A, B^T\}(C) = \sum_{ij} f_{ij} A^i C B^j.$$

When A, B are diagonalizable, i.e., $A = V_A D_A V_A^{-1}$ and $B = V_B D_B V_B^{-1}$:

$$\begin{aligned} f\{A, B^T\}(C) &= V_A \sum_{ij} f_{ij} D_A^i V_A^{-1} C V_B D_B^j V_B^{-1}, \\ &= V_A f\{D_A, D_B\}(V_A^{-1} C V_B) V_B^{-1}. \end{aligned}$$

Hence, we have ($\circ$ is the Hadamard product):

$$f\{A, B^T\}(C) = V_A \left(\begin{bmatrix} f(\lambda_1, \mu_1) & \dots & f(\lambda_1, \mu_n) \\ \vdots & & \vdots \\ f(\lambda_m, \mu_1) & \dots & f(\lambda_m, \mu_n) \end{bmatrix} \circ V_A^{-1} C V_B \right) V_B^{-1}.$$

How do we compute $f\{A, B^T\}(C)$ for generic functions and non–normal matrices?

A bivariate evaluation scheme

Our aim: evaluating $f\{A, B^T\}(C)$.

- We can assume A, B triangular by taking Schur forms.
- We can partition the diagonal blocks of A, B so that their spectra are separated.
- We now need a formula for

$$F := f\left\{ \begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix}, \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix}^T \right\}(C).$$

The generic case is then obtained by divide–and–conquer.

Block diagonalization

Let A, B be block upper triangular:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix}, \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix}.$$

Block diagonalization

Let A, B be block upper triangular:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix}, \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix}.$$

If V, W verify $A_{11}V - A_{22}V = A_{12}$ and $B_{11}W - B_{22}W = B_{12}$, then:

$$\underbrace{\begin{bmatrix} I & V \\ & I \end{bmatrix}}_{\widetilde{V}} A \begin{bmatrix} I & -V \\ & I \end{bmatrix} = \begin{bmatrix} A_{11} & \\ & A_{22} \end{bmatrix}, \qquad \underbrace{\begin{bmatrix} I & W \\ & I \end{bmatrix}}_{\widetilde{W}} B \begin{bmatrix} I & -W \\ & I \end{bmatrix} = \begin{bmatrix} B_{11} & \\ & B_{22} \end{bmatrix}.$$

Block diagonalization

Let A, B be block upper triangular:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix}, \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix}.$$

If V, W verify $A_{11}V - A_{22}V = A_{12}$ and $B_{11}W - B_{22}W = B_{12}$, then:

$$\underbrace{\begin{bmatrix} I & V \\ & I \end{bmatrix}}_{\widetilde{V}} A \begin{bmatrix} I & -V \\ & I \end{bmatrix} = \begin{bmatrix} A_{11} & \\ & A_{22} \end{bmatrix}, \qquad \underbrace{\begin{bmatrix} I & W \\ & I \end{bmatrix}}_{\widetilde{W}} B \begin{bmatrix} I & -W \\ & I \end{bmatrix} = \begin{bmatrix} B_{11} & \\ & B_{22} \end{bmatrix}.$$

So that:

$$\begin{bmatrix} I & V \\ & I \end{bmatrix} f\{A, B^T\}(C) \begin{bmatrix} I & -W \\ & I \end{bmatrix} = \underbrace{f\left\{ \begin{bmatrix} A_{11} & \\ & A_{22} \end{bmatrix}, \begin{bmatrix} B_{11}^T & \\ & B_{22}^T \end{bmatrix} \right\} (\tilde{V} C \tilde{W}^{-1})}_{\text{decouple into 4 function evaluations of smaller matrices}}$$

Algorithm 1 Evaluates $f\{A, B^T\}(C)$

procedure fun2m(f, A, B, C)
$[Q_A, T_A] = \mathrm{schur}(A)$, $[Q_B, T_B] = \mathrm{schur}(B)$
$\widetilde{C} \leftarrow Q_A^* C Q_B$
$F \leftarrow$fun2m_rec($f, T_A, T_B, \widetilde{C}$)
return $Q_A F Q_B^*$
end procedure

procedure fun2m_rec(f, A, B, C)
if A, B are small **then return** $f\{A, B^T\}(C)$
else
    Partition A, B and C as:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix}, \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix}, \qquad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

    Retrieve V and W by solving Sylvester equations
    Compute $\begin{bmatrix} \widetilde{C}_{11} & \widetilde{C}_{12} \\ \widetilde{C}_{21} & \widetilde{C}_{22} \end{bmatrix} = \begin{bmatrix} I & V \\ & I \end{bmatrix} \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} I & -W \\ & I \end{bmatrix}$
    $F_{ij} \leftarrow$ fun2m_rec($f, A_{ii}, B_{jj}, \widetilde{C}_{ij}$), for $i, j = 1, 2$
    return $\begin{bmatrix} I & -V \\ & I \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \begin{bmatrix} I & W \\ & I \end{bmatrix}$
end if
end procedure

Algorithm 2 Evaluates $f\{A, B^T\}(C)$

procedure fun2m(f, A, B, C)
$[Q_A, T_A] = \mathrm{schur}(A)$, $[Q_B, T_B] = \mathrm{schur}(B)$
$\widetilde{C} \leftarrow Q_A^* C Q_B$
$F \leftarrow$fun2m_rec($f, T_A, T_B, \widetilde{C}$)
return $Q_A F Q_B^*$
end procedure

procedure fun2m_rec(f, A, B, C)
if A, B are small **then return** $f\{A, B^T\}(C)$
else
    Partition A, B and C as:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix}, \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix}, \qquad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

    Retrieve V and W by solving Sylvester equations
    Compute $\begin{bmatrix} \widetilde{C}_{11} & \widetilde{C}_{12} \\ \widetilde{C}_{21} & \widetilde{C}_{22} \end{bmatrix} = \begin{bmatrix} I & V \\ & I \end{bmatrix} \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} I & -W \\ & I \end{bmatrix}$
    $F_{ij} \leftarrow$ fun2m_rec($f, A_{ii}, B_{jj}, \widetilde{C}_{ij}$), for $i, j = 1, 2$
    return $\begin{bmatrix} I & -V \\ & I \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \begin{bmatrix} I & W \\ & I \end{bmatrix}$
end if
end procedure

Blocking strategy

Each recursive call needs the matrices $\begin{bmatrix} I & V \\ & I \end{bmatrix}$, $\begin{bmatrix} I & W \\ & I \end{bmatrix}$ to be not so ill–conditioned. This is equivalent to keep under control the norm of the solutions of

$$A_{11}V - VA_{22} = A_{12}, \qquad B_{11}W - WB_{22} = B_{12}.$$

Blocking strategy

Each recursive call needs the matrices $\begin{bmatrix} I & V \\ & I \end{bmatrix}$, $\begin{bmatrix} I & W \\ & I \end{bmatrix}$ to be not so ill–conditioned. This is equivalent to keep under control the norm of the solutions of

$$A_{11}V - VA_{22} = A_{12}, \qquad B_{11}W - WB_{22} = B_{12}.$$

As in the Schur–Parlett algorithm [4], the blocking is based on clustering the eigenvalues of A (resp. B) such that, for a given $\delta > 0$:

- For each eigenvalue λ in a cluster $\exists \mu$ in the same cluster s.t. $|\lambda - \mu| \leq \delta$.
- Each pair of eig. λ, μ that belong to different clusters verifies $|\lambda - \mu| > \delta$

Blocking strategy

Each recursive call needs the matrices $\left[\begin{smallmatrix} I & V \\ & I \end{smallmatrix}\right]$, $\left[\begin{smallmatrix} I & W \\ & I \end{smallmatrix}\right]$ to be not so ill–conditioned. This is equivalent to keep under control the norm of the solutions of

$$A_{11}V - VA_{22} = A_{12}, \qquad B_{11}W - WB_{22} = B_{12}.$$

As in the Schur–Parlett algorithm [4], the blocking is based on clustering the eigenvalues of A (resp. B) such that, for a given $\delta > 0$:

- For each eigenvalue λ in a cluster $\exists \mu$ in the same cluster s.t. $|\lambda - \mu| \le \delta$.
- Each pair of eig. λ, μ that belong to different clusters verifies $|\lambda - \mu| > \delta$

Since this criterion is only heuristic, we also check a posteriori whether $\|V\|_2 > \gamma \|A_{12}\|$ (resp. $\|W\|_2 > \gamma \|B_{12}\|$) for a moderate $\gamma \ge \delta^{-1}$.

In that case the two clusters are merged.

[4] Davies, Higham. *A Schur–Parlett algorithm for computing matrix functions*. SIMAX, 2003.

Evaluating the function of the triangular atomic blocks

Core idea [5,6]: Consider small diagonal random perturbations E_A, E_B and compute

$$f\{A + E_A, B + E_B\}(C)$$

via diagonalization with higher precision.

Evaluating the function of the triangular atomic blocks

Core idea [5,6]: Consider small diagonal random perturbations E_A, E_B and compute

$$f\{A + E_A, B + E_B\}(C)$$

via diagonalization with higher precision.

Main issue: V_A, V_B such that $A + E_A = V_A D_A V_A^{-1}$, $B + E_B = V_B D_B V_B^{-1}$ might have large condition numbers.

Evaluating the function of the triangular atomic blocks

Core idea [5,6]: Consider small diagonal random perturbations E_A, E_B and compute

$$f\{A+E_A, B+E_B\}(C)$$

via diagonalization with higher precision.

Main issue: V_A, V_B such that $A+E_A = V_A D_A V_A^{-1}, B+E_B = V_B D_B V_B^{-1}$ might have large condition numbers.

- Lower the unit round–off to u^2 and compute $\widetilde{A} = A + E_A, \widetilde{B} = B + E_B$ with $\|E_A\| = u\|A\|$, $\|E_B\| = u\|B\|$.
- Set the unit round–off to $u_h \leq u$ and retrieve triangular V_A, V_B by solving shifted linear systems with $\widetilde{A}$ and $\widetilde{B}$.
- Evaluate $V_A f\{D_A, D_B\}(V_A^{-1} C V_B) V_B^{-1}$ using u_h
- Go back to unit round–off u.

[5] Davies. *Approximate diagonalization*. SIMAX, 2008.

[6] Higham, Liu. *A multiprecision derivative–free Schur–Parlett algorithm for computing matrix functions*. MIMS EPrint 2020.19, 2020.

Choosing u_h

The following Lemma suggests the choice $\boxed{u_h \le \frac{u}{\kappa(V_A)\kappa(V_B)}}$.

Lemma

Let $Y = V_A f\{D_A, D_B\}(V_A^{-1} C V_B) V_B^{-1}$, and let $\hat{Y}$ be the corresponding quantity computed in floating point arithmetic. If the matrix multiplications are performed exactly, and $f(\lambda_i^A, \lambda_j^B)$ is computed with relative error bounded by u_h, then

$$\|F - \hat{F}\| \le \kappa(V_A)\kappa(V_B)\|C\| \max_{i,j} |f(\lambda_i, \mu_j)| u_h.$$

Choosing u_h

The following Lemma suggests the choice $\boxed{u_h \le \frac{u}{\kappa(V_A)\kappa(V_B)}}$.

Lemma
Let $Y = V_A f\{D_A, D_B\}(V_A^{-1} C V_B) V_B^{-1}$, and let $\hat{Y}$ be the corresponding quantity computed in floating point arithmetic. If the matrix multiplications are performed exactly, and $f(\lambda_i^A, \lambda_j^B)$ is computed with relative error bounded by u_h, then

$$\|F - \hat{F}\| \le \kappa(V_A)\kappa(V_B)\|C\| \max_{i,j} |f(\lambda_i, \mu_j)| u_h.$$

Problem: How to estimate $\kappa(V_A), \kappa(V_B)$ before their computation?

Choosing u_h

The following Lemma suggests the choice $\boxed{u_h \leq \frac{u}{\kappa(V_A)\kappa(V_B)}}$.

Lemma
Let $Y = V_A f\{D_A, D_B\}(V_A^{-1} C V_B) V_B^{-1}$, and let $\hat{Y}$ be the corresponding quantity computed in floating point arithmetic. If the matrix multiplications are performed exactly, and $f(\lambda_i^A, \lambda_j^B)$ is computed with relative error bounded by u_h, then

$$\|F - \hat{F}\| \leq \kappa(V_A)\kappa(V_B)\|C\| \max_{i,j} |f(\lambda_i, \mu_j)| u_h.$$

Problem: How to estimate $\kappa(V_A), \kappa(V_B)$ before their computation?

$\kappa(V_A)$ (analogously $\kappa(V_B)$) can be estimated from the entries of $\widetilde{A}$:

$$\kappa(V_A) \lesssim m\zeta(\zeta + 1)^{m+1}, \qquad \zeta = \frac{\max_{i<j} |\widetilde{A}_{ij}|}{\min_{i \neq j} |\widetilde{A}_{ii} - \widetilde{A}_{jj}|} \tag{1}$$

Choosing u_h

The following Lemma suggests the choice $\boxed{u_h \leq \frac{u}{\kappa(V_A)\kappa(V_B)}}$.

Lemma

Let $Y = V_A f\{D_A, D_B\}(V_A^{-1} C V_B) V_B^{-1}$, and let $\hat{Y}$ be the corresponding quantity computed in floating point arithmetic. If the matrix multiplications are performed exactly, and $f(\lambda_i^A, \lambda_j^B)$ is computed with relative error bounded by u_h, then

$$\|F - \hat{F}\| \leq \kappa(V_A)\kappa(V_B)\|C\| \max_{i,j} |f(\lambda_i, \mu_j)| u_h.$$

Problem: How to estimate $\kappa(V_A), \kappa(V_B)$ before their computation?

$\kappa(V_A)$ (analogously $\kappa(V_B)$) can be estimated from the entries of $\widetilde{A}$:

$$\kappa(V_A) \lesssim m\zeta(\zeta+1)^{m+1}, \qquad \zeta = \frac{\max_{i<j} |\widetilde{A}_{ij}|}{\min_{i \neq j} |\widetilde{A}_{ii} - \widetilde{A}_{jj}|} \tag{1}$$

This is usually too pessimistic; practically, we apply the blocking method with a parameter $\delta_1 < \delta$ and we compute the maximum of (1) for the diagonal blocks.

Numerical results: highly non normal matrices

Setting

- $m = n = 64$, $f(x, y) = (x + y)^{-\frac{1}{2}}$.
- diag: diagonalization (no blocking and no HP).
- diag_hp: HP diagonalization (no blocking).
- Err: relative error with respect to $f\{A, B\}(C)$ evaluated with diag_hp using 128 digits.
- n_A, n_B: number of atomic blocks in A and B.
- κ_f: estimate of

$$\lim_{h\to 0} \sup_{\frac{\|\Delta A\|}{\|A\|}, \frac{\|\Delta B\|}{\|B\|} \le h} \frac{\|f\{A + \Delta A, B^T + \Delta B^T\}(C) - f\{A, B^T\}(C)\|}{h}.$$

	FUN2M					DIAG		DIAG_HP			
Test	Err	Time	nA	nB	Digits	Err	Time	Time	Err	Digits	$\kappa_f \cdot u$
jordbloc	$2.0 \cdot 10^{-9}$	0.02	15	15	48	$3.9 \cdot 10^{-1}$	0.008	1.65	$2.0 \cdot 10^{-9}$	51	$3.2 \cdot 10^{-10}$
grcar	$1.5 \cdot 10^{-13}$	1.53	1	1	40	$7.7 \cdot 10^{-8}$	0.008	1.54	$1.5 \cdot 10^{-13}$	40	$1.0 \cdot 10^{-9}$
smoke	$3.5 \cdot 10^{-9}$	1.46	1	1	35	$1.1 \cdot 10^{-8}$	0.002	1.45	$1.8 \cdot 10^{-9}$	35	$5.0 \cdot 10^{-1}$
kahan	$3.4 \cdot 10^{-16}$	1.37	1	1	43	$6.8 \cdot 10^{-7}$	0.002	1.36	$4.5 \cdot 10^{-16}$	43	$1.4 \cdot 10^{-7}$
lesp	$4.4 \cdot 10^{-15}$	0.23	9	9	35	$1.6 \cdot 10^{-1}$	0.003	1.32	$3.5 \cdot 10^{-15}$	36	$1.9 \cdot 10^{-15}$
sampling	$1.0 \cdot 10^{-7}$	0.41	10	9	49	$2.2 \cdot 10^{-2}$	0.006	2.04	$1.0 \cdot 10^{-7}$	49	$8.2 \cdot 10^{-8}$
grcar-rand	$5.2 \cdot 10^{-12}$	0.39	1	16	29	$7.8 \cdot 10^{-8}$	0.009	1.45	$5.2 \cdot 10^{-12}$	31	$3.7 \cdot 10^{-6}$

Numerical results: random matrices

Test $=$ randn, $f(x,y) = \frac{1}{\sqrt{x+y}(x-y)}$

Size	fun2m Time	fun2m nA	fun2m nB	diag Time
64	0.01	16	16	0.01
128	0.08	32	32	0.06
256	0.27	64	64	0.2
512	1.42	128	128	1.07
1,024	5.47	256	256	4.09
2,048	29.12	512	512	22.01
4,096	243.41	1,024	1,024	132.48

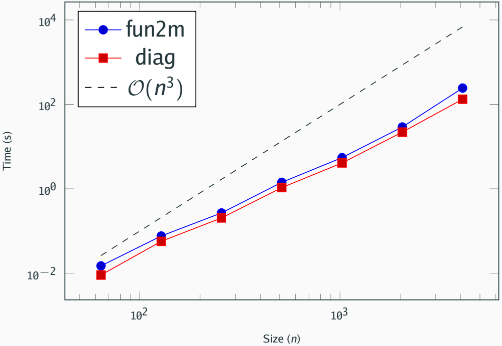

Figure 1: Timings of fun2m and diag for well–conditioned A and B.

Is there code available?

Yes, we have the Julia package BivMatFun.

```
julia> import Pkg;
julia> Pkg.add(url = "https://github.com/numpi/BivMatFun.git");
julia> using BivMatFun;

# Only complex matrices are implemented
julia> n = 1024;
julia> A = complex(randn(n,n)); B = complex(randn(n,n));
julia> C = complex(randn(n,n));
julia> f = (z,w,i,j) -> 1 / (z + w);
julia> X, _ = fun2m(f, A, B, C);
julia> using LinearAlgebra;
julia> opnorm(A*X + X*B - C) / opnorm(X)

     3.277465131019034e-13
```

Conclusions

Reference:

- S. Massei., L. R. *Mixed precision recursive block diagonalization for bivariate functions of matrices*, to appear on SIMAX, 2022.

Remarks:

- A perturb–and–diagonalize approach combined with high precision can be a workaround when dealing with linear algebra tasks related to (nearly) non diagonalizable matrices.
- An effective blocking strategy is necessary in order to mitigate the impact of high precision arithmetic on timings.

Possible applications/extensions

- Projection methods for function of Kronecker sum structured matrices.
- Multivariate matrix functions $\to$ operations on tensors.