Lecture 2:
Computing functions of dense matrices

Paola Boito and Federico Poloni

Università di Pisa

Pisa - Hokkaido - Roma2 Summer School
Pisa, August 27 - September 8, 2018

Introduction

In this lecture we focus on the problem of computing functions of dense, small- or medium-sized matrices.

- Computations are performed in finite precision. Therefore:
  - we can only expect to compute an approximation to $f(A)$,
  - error and conditioning analyses are generally required.
- We do not expect A to have any special structure or sparsity pattern.
- The size of A is such that we can store and work on all of its entries.
- Main reference: N. Higham, *Functions of Matrices. Theory and Computation.* SIAM 2008 (will be denoted in the following as [Higham]).
- Software/programming language: we will mostly use MATLAB.

Conditioning

Condition numbers measure the sensitivity of matrix functions to perturbation in the data.

Well-known example: conditioning of matrix inversion. Take

$$A = \begin{bmatrix} 1 + 10^{-6} & 1 \\ 1 & 1 \end{bmatrix}$$

Then $\|\texttt{inv(A)} - A^{-1}\| \approx 10^{-5}$.

Conditioning of scalar functions

The relative condition number for a scalar function $f(x)$ is defined as

$$\mathrm{cond}_{\mathrm{rel}}(f,x) := \lim_{\epsilon\to 0} \sup_{|\Delta x|\le\epsilon|x|} \left| \frac{f(x+\Delta x)-f(x)}{\epsilon f(x)} \right|.$$

If $f(x)$ is twice continuously differentiable, this can be rewritten as

$$\mathrm{cond}_{\mathrm{rel}}(f,x) = \left| \frac{xf'(x)}{f(x)} \right|$$

because

$$\frac{f(x+\Delta x)-f(x)}{f(x)} = \left(\frac{xf'(x)}{f(x)}\right)\frac{\Delta x}{x} + o(\Delta x)$$

Conditioning of matrix functions

For a matrix function $f : \mathbb{C}^{n\times n} \longrightarrow \mathbb{C}^{n\times n}$ evaluated at X we define the relative condition number:

$$\mathrm{cond}_{\mathrm{rel}}(f, X) := \lim_{\epsilon\to 0} \sup_{\|E\|\le\epsilon\|X\|} \frac{\|f(X+E) - f(X)\|}{\epsilon\|f(X)\|}.$$

As a consequence we have a bound for small perturbations:

$$\frac{\|f(X+E) - f(X)\|}{\|f(X)\|} \le \mathrm{cond}_{\mathrm{rel}}(f, X)\frac{\|E\|}{\|X\|} + o(\|E\|).$$

The absolute condition number is

$$\mathrm{cond}_{\mathrm{abs}}(f, X) := \lim_{\epsilon\to 0} \sup_{\|E\|\le\epsilon} \frac{\|f(X+E) - f(X)\|}{\epsilon}$$

with the property

$$\mathrm{cond}_{\mathrm{rel}}(f, X) = \mathrm{cond}_{\mathrm{abs}}(f, X)\frac{\|X\|}{\|f(X)\|}.$$

The Fréchet derivative

The Fréchet derivative of f at X is a linear mapping

$$L_f(X) : \mathbb{C}^{n\times n} \longrightarrow \mathbb{C}^{n\times n}$$
$$E \longrightarrow L_f(X, E)$$

such that for all $E \in \mathbb{C}^{n\times n}$

$$f(X + E) - f(X) - L_f(X, E) = o(\|E\|).$$

Examples:

$$(X + E)^2 - X^2 = XE + EX + E^2 \Rightarrow L_{X^2}(X, E) = XE + EX.$$

$$(X+E)^{-1} - X^{-1} = -X^{-1}EX^{-1} + \mathcal{O}(E^2) \Rightarrow L_{X^{-1}}(X, E) = -X^{-1}EX^{-1}.$$

The Fréchet derivative

More properties:

- If it exists, the Fréchet derivative is unique.
- If f is $2n-1$ times continuously differentiable on $\mathcal{D} \subset \mathbb{C}$ and $X \in \mathbb{C}^{n\times n}$ has spectrum in $\mathcal{D}$, then $L_f(X,E)$ exists and is continuous in X and E.
- The norm of the Fréchet derivative is defined as

$$\|L_f(X)\| := \max_{Z\neq 0} \frac{\|L_f(X,Z)\|}{\|Z\|}.$$

Condition numbers

Theorem (Rice)

The absolute and relative condition numbers for $f(X)$ are given by

$$\text{cond}_{\text{abs}}(f, X) = \|L_f(X)\|,$$
$$\text{cond}_{\text{rel}}(f, X) = \frac{\|L_f(X)\| \, \|X\|}{\|f(X)\|}.$$

Proof:

$$\begin{aligned}
\text{cond}_{\text{abs}}(f, X) &= \lim_{\epsilon \to 0} \sup_{\|E\| \le \epsilon} \frac{\|f(X+E) - f(X)\|}{\epsilon} \\
&= \lim_{\epsilon \to 0} \sup_{\|E\| \le \epsilon} \left\| \frac{L_f(X, E) + o(\|E\|)}{\epsilon} \right\| \\
&= \lim_{\epsilon \to 0} \sup_{\|E\| \le \epsilon} \|L_f(X, E/\epsilon) + o(\|E\|)/\epsilon\| \\
&= \sup_{\|Z\| \le 1} \|L_f(X, Z)\| = \|L_f(X)\|.
\end{aligned}$$

Conditioning of the matrix exponential

As an example, consider $f(A) = e^A$. The Fréchet derivative is

$$L_{\exp}(A, E) = \int_0^1 e^{A(1-s)} E\, e^{As} ds$$

from which we deduce

$$\|A\| \leq \mathrm{cond}_{\mathrm{rel}}(\exp, A) \leq \frac{e^{\|A\|}\|A\|}{\|e^A\|}.$$

Proof:

- $\|L_{\exp}(A, E)\| \leq \|E\| \int_0^1 e^{\|A\|(1-s)} e^{\|A\|s} ds = \|E\| \int_0^1 e^{\|A\|} ds = \|E\| e^{\|A\|}$
- $\|L_{\exp}(A)\| \geq \|L_{\exp}(A, I)\| = \left\| \int_0^1 e^A ds \right\| = \|e^A\|$.

Normal matrices have minimal condition number in 2-norm (Van Loan).

Estimating the Fréchet derivative

There are several ways to compute or estimate the Fréchet derivative of a matrix function – and therefore its condition number.

One useful property is:

Theorem (Mathias)

Let f be $2n-1$ times continuously differentiable on $\mathcal{D} \subset \mathbb{C}$. For $A \in \mathbb{C}^{n\times n}$ with spectrum contained in $\mathcal{D}$ we have

$$f\left(\begin{bmatrix} A & E \\ 0 & A \end{bmatrix}\right) = \begin{bmatrix} f(A) & L_f(A,E) \\ 0 & f(A) \end{bmatrix}.$$

Estimating the Fréchet derivative

Since L_f is a linear operator, there exists a matrix $K_f(A) \in \mathbb{C}^{n^2 \times n^2}$, known as Kronecker form of the Fréchet derivative, such that

$$\mathrm{vec}(L_f(A, E)) = K_f(A)\mathrm{vec}(E)$$

Note that if (λ, V) is an eigenpair of $L_f(A)$, then $(\lambda, \mathrm{vec}(V))$ is an eigenpair of $K_f(A)$.

Theorem

The eigenvalues of $L_f(A)$ are

$$f[\lambda_i, \lambda_j], \qquad i, j = 1, \ldots n,$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A.

Here square brackets denote divided differences, i.e.,

$$f[\lambda, \mu] = \begin{cases} \frac{f(\lambda) - f(\mu)}{\lambda - \mu} & \text{if } \lambda \neq \mu, \\ f'(\lambda) & \text{if } \lambda = \mu. \end{cases}$$

Estimating the Fréchet derivative

Let λ be an eigenvalue of $L_f(A)$; then for all matrix norms it holds

$$\|L_f(A)\| \geq \lambda$$

and therefore we have

Theorem

For any norm,

$$\text{cond}_{\text{abs}}(f, A) \geq \max_{\lambda,\mu \in \Lambda(A)} |f[\lambda, \mu]|.$$

Equality holds for normal A in the Frobenius norm.

Estimating the Fréchet derivative

We can compute the condition number in Frobenius norm using the Kronecker form of L_f:

$$\begin{aligned} \|L_f(A)\|_F &= \max_{E\neq 0} \frac{\|L_f(A,E)\|_F}{\|E\|_F} = \max_{E\neq 0} \frac{\|\mathrm{vec}(L_f(A,E))\|_2}{\|\mathrm{vec}(E)\|_2} \\ &= \max_{E\neq 0} \frac{\|K_f(A)\mathrm{vec}(E)\|_2}{\|\mathrm{vec}(E)\|_2} = \|K_f(A)\|_2. \end{aligned}$$

Since

$$\|K_f(A)\|_2 = \|K_f(A)^* K_f(A)\|_2^{1/2} = \lambda_{\max}(K_f(A)^* K_f(A))^{1/2},$$

we can also estimate the condition number of $f(A)$ using, for instance, the power method.

How to compute $f(A)$?

Many techniques are available. Some of them work for general functions and matrices. Others are well-suited to treat certain functions and/or matrices with special properties.

Some general techniques are obtained directly from the definitions of $f(A)$. For instance, suppose that A is diagonalisable. Then

1. compute the factorization $A = M^{-1}\Lambda M$,
2. compute $f(A) = M^{-1} f(\Lambda) M$.

Works well numerically for A symmetric or Hermitian. See MATLAB function

```
F = funm(A,f).
```

Matrix powers

- A sequence of powers A^2, A^3, A^4 ... is computed in the obvious way (repeated multiplication by A).
- If we only need A^m, repeated squaring is more efficient.
  Let $m = \sum_{i=0}^{t} \beta_i 2^i$ with $\beta_t \neq 0$ and do the following:

```
1  P = A
2  i = 0
3  while β_i = 0
4        P = P^2
5        i = i + 1
6  end
7  X = P
8  for j = i + 1 : t
9        P = P^2
10       if β_j = 1
11             X = XP
12       end
13 end
```

  Cost bounded by $2 \lfloor \log_2 m \rfloor \mathcal{M}$ flops ($\mathcal{M}$ cost of matrix multiplication).

Polynomials

Several competing methods are available to compute

$$p_m(A) = \sum_{k=0}^{m} b_k A^k, \qquad A \in \mathbb{C}^{n\times n}.$$

- Horner's method (nested multiplication); not suitable when m is not known from the start (e.g., truncated series).
- Explicit powers (it isn't more expensive than Horner in the matrix case).
- Factored form $p_m(A) = b_m(A - \alpha_1 I_n)\ldots(A - \alpha_m I_n)$.
- Paterson and Stockmeyer's method, e.g.,

$$p_6(A) = b_6 I_n (A^3)^2 + (b_5 A^2 + b_4 A + b_3 I_n)A^3 + (b_2 A^2 + b_1^A + b_0 I_n).$$

Computational cost is $(m-1)\mathcal{M}$ flops for the first three methods.

Taylor series

One strategy to approximate $f(A)$ consists in applying a truncated Taylor expansion of $f(x)$ to A. But first we need to make sure that the matrix series converges.

Theorem

Suppose that $f(x)$ has a Taylor series expansion

$$f(x) = \sum_{k=0}^{\infty} a_k (x - \alpha)^k, \qquad a_k = \frac{f^{(k)}(\alpha)}{k!}$$

with radius of convergence r. If $A \in \mathbb{C}^{n \times n}$ then $f(A)$ is defined and given by

$$f(A) = \sum_{k=0}^{\infty} a_k (A - \alpha I)^k$$

iff each (distinct) eigenvalue λ_i of A satisfies:

(i) $|\lambda_i - \alpha| < r$, *or*

(ii) $|\lambda_i - \alpha| = r$ *and the series for $f^{(n_i - 1)}(\lambda)$ is convergent at each λ_i.*

Taylor series

Next question: where should we truncate the series?

An error bound is needed: see e.g., [Golub & Van Loan, *Matrix Computations*] or the following result [Mathias, *Approximation of matrix-valued functions*, SIMAX 1993]:

$$\left\| f(A) - \sum_{k=0}^{s} a_k (A - \alpha I)^k \right\| \leq \frac{1}{s!} \max_{0 \leq t \leq 1} \|(A - \alpha I)^s f^{(s)}(\alpha I + t(A - \alpha I))\|.$$

An example

Consider the matrix

$$A = \begin{bmatrix} 0 & \alpha \\ -\alpha & 0 \end{bmatrix}.$$

What is e^A?

An example

Consider the matrix

$$A = \begin{bmatrix} 0 & \alpha \\ -\alpha & 0 \end{bmatrix}.$$

What is e^A? It turns out that

$$e^A = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}.$$

An example

Consider the matrix

$$A = \begin{bmatrix} 0 & \alpha \\ -\alpha & 0 \end{bmatrix}.$$

What is e^A? It turns out that

$$e^A = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}.$$

Now, take $\alpha = 25$ and compute e^A numerically via truncated Taylor expansion. What happens?

Rational approximation

- General idea: if $r(x) = p(x)/q(x) \approx f(x)$, take $f(A) \approx r(A)$.
- What about approximation errors? Suppose A diagonalisable. Then:

$$f(A) \approx r(A) = M^{-1} r(\Lambda) M.$$

  A good approximation of f on the spectrum of A is crucial.
- But this is not enough! Let $e(x) := f(x) - r(x)$; we have

$$e(A) = M^{-1} e(\Lambda) M$$

  and therefore

$$\|e(A)\| \leq \kappa(M) \|e(\Lambda)\|.$$

  If M is ill-conditioned, the approximation error can be large.

Rational approximation

Commonly used classes of rational approximations include

- Best L_∞ (minimax, Chebyshev) approximations:

$$\|r(x) - f(x)\|_\infty = \min_{s \in \mathcal{R}_{k,m}} \|s(x) - f(x)\|_\infty,$$

  where the norm is $\|g(x)\|_\infty = \max_{x \in [a,b]} |g(x)|$.
  Usually employed for Hermitian matrices (see previous bound).

- Padé approximations:
  $r_{k,m}(x) = p_{k,m}(x)/q_{k,m}(x)$ is a $[k/m]$ Padé approximant of f if

$$r_{k,m} \in \mathcal{R}_{k,m} \quad \text{with} \quad q_{k,m}(0) = 1$$

  and

$$f(x) - r_{k,m}(x) = \mathcal{O}(x^{k+m+1}).$$

Padé approximation

Some properties:

- if the $[k/m]$ Padé approximant of f exists, it is unique;
- potential for lower computational cost w.r.t. polynomial/Taylor approximation;
- x should be close to 0 for good approximation,
- well-developed theory, Padé approximants known for several important functions (e.g., exponential),
- code available in Maple, Mathematica, MATLAB (Extended Symbolic Math Toolbox)...

The matrix exponential

- The most studied matrix function!
- Lots of applications, starting from differential equations (we'll see a different example in the fourth lecture).
- Many methods for its computation. Crucial reference:
  - C. B. Moler and C. F. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix*, SIAM Rev. 20(4):801-836, 1978.
  - C. B. Moler and C. F. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev. 45(1):3-49, 2003.
- Three candidates to "best method":
  - methods for ODEs,
  - scaling and squaring (implemented in MATLAB function `expm`),
  - use of Schur form.

Scaling and squaring

This approach to the computation of e^A relies on the property

$$e^A = (e^{A/\sigma})^\sigma, \qquad \text{for} \quad \sigma \in \mathbb{C},$$

and on the fact that e^A is well approximated via Taylor or Padé for small $\|A\|$.
The main idea goes as follows:

1. choose $\sigma = 2^s$ such that $\|A/\sigma\| \approx 1$,
2. approximate $e^{A/\sigma} \approx r(A/\sigma)$, where r is a Taylor or Padé approximant to the exponential,
3. take $e^A \approx r(A/\sigma)^\sigma$ via repeated squarings.

Scaling and squaring

$[k, m]$-Padé approximants for e^x are known explicitly:

$$r_{km}(x) = p_{km}(x)/q_{km}(x) \text{with}$$

$$p_{km}(x) = \sum_{j=0}^{k} \frac{(k+m-j)!k!}{(k+m)!(k-j)!} \frac{x^j}{j!}, \quad q_{km}(x) = \sum_{j=0}^{m} \frac{(k+m-j)!m!}{(k+m)!(m-j)!} \frac{(-x)^j}{j!}$$

(Note that $q_m(x) = p_m(-x)$.)
How should we choose k and m (and the scaling parameter s)?

- Better take $k = m$ (diagonal approximant), because r_{km} with $k \neq m$ is less accurate than $r_{\max(k,m),\max(k,m)}$, but evaluation at a matrix argument has the same cost;
- m and s should be such that the computation of e^A has backward error bounded by $\mathbf{u}$ and minimal cost.

Scaling and squaring

A backward error bound is given as follows.

Theorem

Let $r(x)$ be a rational approximation of e^x such that

$$e^{-2^{-s}A} r(2^{-s}A) = I + G$$

with $\|G\| < 1$ in any consistent matrix norm. Then

$$r(2^{-s}A)^{2^s} = e^{A+E},$$

where E commutes with A and

$$\frac{\|E\|}{\|A\|} \leq \frac{-\log(1 - \|G\|)}{\|2^{-s}A\|}.$$

Scaling and squaring

Now, let us take $r = r_m$ the m-th diagonal Padé approximant. By definition of Padé approximant, we have $e^x - r_m(x) = \mathcal{O}(x^{2m+1})$. Therefore we can write the power series expansion

$$\rho(x) := e^{-x} r_m(x) - 1 = \sum_{j=2m+1}^{\infty} c_i x^i$$

which converges absolutely for $|x| < \nu_m := \min\{|t| : q_m(t) = 0\}$. Hence

$$\|G\| = \|\rho(2^{-s}A)\| \leq \sum_{j=2m+1}^{\infty} |c_i| \theta^i =: f(\theta),$$

where $\theta = \|2^{-s}A\|$. Using the previous theorem we obtain the bound

$$\frac{\|E\|}{\|A\|} \leq \frac{-\log(1 - f(\theta))}{\theta}.$$

For each m we compute $\theta_m := \max\{\theta : \frac{\|E\|}{\|A\|} < \mathbf{u}\}$ and determine s.

Scaling and squaring

What is the cost of evaluating $r_m(A)$ for different values of m? (We will measure computational cost as the required number of matrix multiplications π_m.)
We need to:

1. evaluate $p_m(A)$ and $q_m(A)$,
2. solve the matrix equation $q_m(A)r_m(A) = p_m(A)$.

Since $q_m(x) = p_m(-x)$, an efficient approach to item 1 relies on explicit computation of even powers of A. For instance, if $m = 2\ell$:

$$p_{2\ell}(A) = b_{2\ell}A^{2\ell} + \cdots + b_2A^2 + b_0 I + A(b_{2\ell-1}A^{2\ell-2} + \cdots + b_3A^2 + b_1 I) =: U + V.$$

$$q_{2\ell}(A) = U - V.$$

This scheme can be improved for $m \geq 12$.

Scaling and squaring

m	1	2	3	4	5	6	7
π_m	0	1	2	3	3	4	4
θ_m	3.7e-8	5.3e-4	1.5e-2	8.5e-2	2.5e-1	5.4e-1	9.5e-1

m	8	9	10	11	12	13	14
π_m	5	5	6	6	6	6	7
θ_m	1.5e0	2.1e0	2.8e0	3.6e0	4.5e0	5.4e0	6.3e0

m	15	16	17	18	19	20	21
π_m	7	7	7	8	8	8	8
θ_m	7.3e0	8.4e0	9.4e0	1.1e1	1.2e1	1.3e1	1.4e1

Table taken from [Higham].
Recall: π_m = computational cost, θ_m = bound on $2^{-s}\|A\|$.

We can discard some cases by inspection of π_m.

Scaling and squaring

m	1	2	3	5	7
π_m	0	1	2	3	4
θ_m	3.7e-8	5.3e-4	1.5e-2	2.5e-1	9.5e-1

m	9	13
π_m	5	6
θ_m	2.1e0	5.4e0

m	17	21
π_m	7	8
θ_m	9.4e0	1.4e1

Table taken from [Higham].
Recall: π_m = computational cost, θ_m = bound on $2^{-s}\|A\|$.

If θ_m increases of a factor 2 this saves us one matrix multiplication...
therefore $m = 13$ is the best choice.

Scaling and squaring algorithm

Input: matrix A. ***Output***: $X \approx e^A$.

1 for $m = [3, 5, 7, 9]$
2 if $\|A\|_1 \leq \theta_m$
3 evaluate U and V and solve $(-U + V)X = U + V$
4 quit
5 end
6 end
7 $A \leftarrow A/2^s$ with s minimal integer such that $\|A/2^s\|_1 \leq \theta_{13}$
8 $A_2 = A^2$, $A_4 = A_2^2$, $A_6 = A_2 A_4$
9 $U = A[A_6(b_{13}A_6 + b_{11}A_4 + b_9 A_2) + b_7 A_6 + b_5 A_4 + b_3 A_2 + b_1 I]$
10 $V = A_6(b_{12}A_6 + b_{10}A_4 + b_8 A_2) + b_6 A_6 + b_4 A_4 + b_2 A_2 + b_0 I$
11 solve $(-U + V)Y = U + V$
12 compute $X = Y^{2^s}$ by repeated squaring.

Scaling and squaring algorithm

- Cost: $(\pi_m + \log_2 \lceil \|A\|_1/\theta_m) \rceil \, \mathcal{M} + D$.

But there is more... we should also:

- check whether $\theta_k < \nu_k$ (it is!);
- study the effect of rounding errors on the evaluation of $p_m(A)$ and $q_m(A)$: errors turn out to be nicely bounded;
- estimate $\|q_m(A)^{-1}\|$: it grows with m, but conditioning is very good for m up to 13;
- perform error analysis for the squaring steps. Although much has been done, this point is still unclear and, in principle, a potential source of instability. However, the algorithm is known to be forward stable if A is normal.

See [N. Higham, *The Scaling and Squaring Method for the Matrix Exponential Revisited*, SIAM Rev. 51(4), 747 – 764, 2009] for a detailed discussion.

Schur-Parlett algorithm

General purpose algorithm for computing $f(A)$, based on the Schur decomposition of $A \in \mathbb{C}^{n\times n}$:

$$A = QTQ^*,$$

with $Q \in \mathbb{C}^{n\times n}$ unitary and $T \in \mathbb{C}^{n\times n}$ upper triangular. The Schur decomposition can be computed in a backward stable way via the QR algorithm.
The idea is:

- given A, compute its Schur decomposition $A = QTQ^*$,
- form $f(A) = Q\,f(T)Q^*$.

We have shifted the problem to computing functions of triangular matrices.

Schur-Parlett algorithm

How to compute $f(T)$? Explicit formulas exist:

Function of a triangular matrix

Let $T = (t_{ij})_{i,j=1,\ldots,n}$ and $F = f(T) = (f_{ij})_{i,j=1\ldots n}$. Then:

$$f_{ii} = f(t_{ii}), \qquad i = 1, \ldots, n$$

$$f_{ij} = \sum_{(s_0,\ldots,s_k)\in\mathcal{S}_{ij}} t_{s_0 s_1} t_{s_1 s_2} \ldots t_{s_{k-1} s_k} f[\lambda_{s_0}, \ldots, \lambda_{s_k}],$$

where $\lambda_i = t_{ii}$ and $S_{i,j}$ denotes the set of all strictly increasing sequences of integers that start at i and end at j, whereas $f[\lambda_{s_0}, \ldots, \lambda_{s_k}]$ is the k-th order divided difference of f at $\lambda_{s_0}, \ldots, \lambda_{s_k}$.

But these formulas are computationally too expensive: $\mathcal{O}(2^n)$ flops.

Schur-Parlett algorithm

Idea by Parlett:

- $F = f(T)$ commutes with T (general property of matrix functions),
- diagonal entries of F are known ($f_{ii} = f(t_{ii})$),
- therefore the equation $TF = FT$ can be solved for off-diagonal entries of F.

Indeed we have

$$\sum_{k=i}^{j} t_{ik} f_{kj} = \sum_{k=i}^{j} f_{ik} t_{kj},$$

that is,

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

If $t_{ii} \neq t_{jj}$:

$$f_{ij} = t_{ij} \frac{f_{ii} - f_{jj}}{t_{ii} - t_{jj}} + \sum_{k=i+1}^{j-1} \frac{f_{ik} t_{kj} - t_{ik} f_{kj}}{t_{ii} - t_{jj}}, \qquad i < j.$$

[B. N. Parlett, *A recurrence among the elements of functions of triangular matrices*, Linear Algebra Appl. 14:117–121, 1976]

Schur-Parlett algorithm

Input: triangular matrix T. ***Output***: $F = f(T)$.

1 $f_{ii} = f(t_{ii}), \qquad i = 1 : n$
2 for $j = 2 : n$
3 for $i = j - 1 : -1 : 1$
4 $f_{i,j} = t_{ij} \frac{f_{ii} - f_{jj}}{t_{ii} - t_{jj}} + \frac{\sum_{k=i+1}^{j-1} f_{ik} t_{kj} - t_{ik} f_{kj}}{t_{ii} - t_{jj}}$
5 end
6 end

Cost: $2n^3/3$ flops.

Main drawback: breaks down when $t_{ii} = t_{jj}$ for some i, j.

Schur-Parlett algorithm: block version

Idea: write the recurrence in block form.

- Suppose T has upper triangular block form $T = (T_{ij})$.
- Then $F = f(T) = (F_{ij})$ has the same block structure.
- For diagonal blocks we have $F_{ii} = f(T_{ii})$.
- For off-diagonal blocks (i.e., $i < j$), the block recurrence is

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

  This is a Sylvester equation in the unknown F_{ij}. It is nonsingular iff T_{ii} and T_{jj} have no eigenvalues in common.
- Suitable reordering techniques should be applied beforehand.

Note that we still have to compute $f(T_{ii})$.

Schur-Parlett algorithm: atomic blocks

Each atomic block T_{ii} is assumed to be an upper triangular matrix with clustered eigenvalues. Denote $T = T_{ii} \in \mathbb{R}^{m\times m}$. Here is an idea for computing $f(T)$:

- Write $T = \sigma I + M, \qquad \sigma = \text{trace}(T)/m.$
- Suppose $f(\sigma + z) = \sum_{k=0}^{\infty} \frac{f^{(k)}(\sigma)}{k!} z^k$.
- Then $f(T) = \sum_{k=0}^{\infty} \frac{f^{(k)}(\sigma)}{k!} M^k$.
- After $m-1$ terms, the powers of M should decay quickly, so a suitable truncation of the series should be sufficiently accurate. (All of this can be made more precise!)
- Cost $\mathcal{O}(m^4)$ flops.
- Potential danger: cancellation.

Schur-Parlett algorithm: error analysis

Back to $TF - FT = 0$. Let $\hat{F}$ be the computed solution:

$$T\hat{F} - \hat{F}T = R, \qquad R = \text{residual}.$$

Let $\hat{F} = F + \Delta F$. By subtraction:

$$T\Delta F - \Delta F\, T = R.$$

Taking blocks:

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

from which we deduce

$$\|\Delta F_{ij}\|_F \leq \text{sep}(T_{ii}, T_{jj})^{-1} \|B_{ij}\|_F$$

where the separation of T_{ii} and T_{jj} is

$$\text{sep}(T_{ii}, T_{jj}) = \min_{X \neq 0} \frac{\|T_{ii}X - XT_{jj}\|_F}{\|X\|_F}.$$

Schur-Parlett algorithm: error analysis

Some comments:

- The block R_{ij} represents the errors introduced during the computation of F_{ij}. These can lead to an error ΔF_{ij} of norm proportional to $\mathrm{sep}(T_{ii}, T_{jj})^{-1}\|R_{ij}\|$.
- The blocks ΔF_{ij} in the rhs represent the errors introduced during previous computations of diagonal or off-diagonal blocks in the recurrence. These can be magnified by a factor $\mathrm{sep}(T_{ii}, T_{jj})^{-1}$.
- The separation of atomic blocks clearly plays a crucial role. However, error growth is also possible if some T_{ii} is large.
- Reordering applied prior to Schur-Parlett should maximize separation between atomic blocks while keeping block sizes reasonably small.