4 Numerical methods for matrix functions

As the name suggests, a *matrix function* is a function mapping a matrix to a matrix:

$$f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}. \tag{4.1}$$

In our setting, the term *matrix function* will refer to not just any such matrix mapping, but will refer to a specific class of matrix-valued functions. In a sense, we mean the "natural generalization" of a scalar-valued function f to matrices. With a slight abuse of notation we can let $f(z)$ represent a scalar-valued function and let $f(A)$ represent the associated matrix-valued function. The matrix function associated with the scalar function $f(z) = z^2$ is the matrix function $f(A) = AA$. The matrix function associated with $f(z) = (1-z)/(1+z)$ is $f(A) = (I+A)^{-1}(I-A)$. The matrix exponential which is the matrix function extension of $f(z) = \exp(z)$, is commonly used in the study and computation of ordinary differential equations. Other matrix functions that we discuss further in this chapter is the matrix sign function $f(z) = \text{sign}(z)$, the square root function $f(z) = \sqrt{z}$ and φ-functions, $\varphi(z) = (e^z - 1)/z$. In fact, a large number of elementary scalar functions have been extended to matrices and used in various scientific fields.

Matrix functions appear in many scientific fields. The current theoretical and numerical techniques for matrix functions form a fundamental tool-set to analyze and solve many matrix problems. In 2011, the mathematics panel of the European Research Council granted a large five-year research project on *"Functions of Matrices: Theory and Computation"* with principal investigator Nicholas Higham (http://en.wikipedia.org/wiki/Nicholas_Higham). We are in this course mainly concerned with numerical methods and theory required to derive and understand the main properties of numerical methods.

Following some definitions and general properties (Section 4.1) the chapter is separated into *general methods* Section 4.2 and *specialized methods* Section 4.3. For very large-scale problems we will see that we can use Krylov methods (in Section 4.4), similar to the Krylov methods we have seen for eigenvalue problems and linear systems of equations.

Note: If $f(z) = (1-z)/(1+z)$ the two matrix function definitions $f(A) = (I+A)^{-1}(I-A)$ and $f(A) = (I-A)(I+A)^{-1}$ are equal since $I-A$ and $(I+A)^{-1}$ commute.

4.1 Definitions and general properties

There are several ways to define matrix functions. We give three definitions, with slightly different domains of definition. The definitions are equivalent for very large classes of functions. In particular, they are equivalent for matrix function extension of a scalar-valued function

Not all matrix-valued functions are matrix functions in our sense. The function that multiplies a matrix by a constant matrix $f(A) = BA$ is a function of the form (4.1) but it is not a natural generalization of a scalar-valued function and is not a matrix function in our sense. Other examples, such as $f(A) = \text{diag}(A)$ and $f(A) = A^T$ are also not matrix functions in our sense.

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

which is analytic in $\mathbb{C}$ (entire functions), as we show in Section 4.1.4. All three definitions lead to numerical methods.

4.1.1 *Taylor expansion definition*

If we want a definition which is consistent with matrix-matrix multiplications, there is essentially only one way to extend polynomials to matrices. The matrix function corresponding to the polynomial $p(z) = \alpha_0 + \cdots + \alpha_k z^k$ is

$$p(A) = \alpha_0 I + \cdots + \alpha_k A^k. \tag{4.2}$$

Throughout this course we use the notation: $D(\mu, r)$ denotes an open disk of radius r centered at $\mu \in \mathbb{C}$,

$$D(\mu, r) = \{z \in \mathbb{C} : |z - \mu| < r\}$$

and $\bar{D}(\mu, r)$ is the corresponding closed disk.

Suppose now that the Taylor series of the scalar function f is convergent for expansion point μ:

$$f(z) = \sum_{i=0}^{\infty} \frac{f^{(i)}(\mu)}{i!}(z - \mu)^i. \tag{4.3}$$

If we generalize polynomials using (4.2), the matrix generalization of $(z - \mu)^i$ is $(A - \mu I)^i$. Therefore, in complete analogy with the polynomial matrix function (4.2) we can define the matrix function via a Taylor series. The following definition is the matrix generalization of (4.2) and (4.3).

Definition 4.1.1 (Taylor definition)**.** *The Taylor definition with expansion point $\mu \in \mathbb{C}$ of the matrix function associated with $f(z)$ is given by*

$$f(A) = \sum_{i=0}^{\infty} \frac{f^{(i)}(\mu)}{i!}(A - \mu I)^i. \tag{4.4}$$

This definition is valid for functions with convergent matrix-valued Taylor series, which turns out to be the case if the function is analytic in a sufficiently large domain. This is illustrated in the following theorem. In particular, a consequence of the following result is that (4.4) is well-defined if the scalar-valued function is analytic in the complex plane.

In Theorem 4.1.2, we make an assumption about $\|A - \mu I\|$. With a slightly more advanced version of the proof, the assumption can relaxed to the condition that the eigenvalues A are contained in $D(\mu, r)$.

Theorem 4.1.2 (Convergence of Taylor definition)**.** *Suppose $f(z)$ is analytic in $\bar{D}(\mu, r)$ and suppose $r > \|A - \mu I\|$. Then, with definition (4.4) of $f(A)$ and $\gamma := \|A - \mu I\|/r < 1$, there exists a constant $C > 0$ independent of N such that*

$$\|f(A) - \sum_{i=0}^{N} \frac{f^{(i)}(\mu)}{i!}(A - \mu I)^i\| \leq C\gamma^N \to 0 \text{ as } N \to \infty.$$

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

Proof. From functional analysis we know that analyticity implies that the derivatives of an analytic function cannot grow arbitrarily fast. More precisely, for any function which is analytic in $\bar{D}(\mu, r)$ there exists a constant C_r independent of i such that

$$|f^{(i)}(\mu)| \leq C_r \frac{i!}{r^i}. \tag{4.5}$$

The bound follows from (4.5), the triangle inequality and geometric series:

$$\begin{aligned} \|f(A) - \sum_{i=0}^{N} \frac{f^{(i)}(\mu)}{i!}(A-\mu I)^i\| &= \|\sum_{i=N+1}^{\infty} \frac{f^{(i)}(\mu)}{i!}(A-\mu I)^i\| \\ &\leq \sum_{i=N+1}^{\infty} \frac{|f^{(i)}(\mu)|}{i!}\|A-\mu I\|^i \\ &\leq C_r \sum_{i=N+1}^{\infty} \frac{\|A-\mu I\|^i}{r^i} \\ &\leq C_r \frac{\gamma^{N+1}}{1-\gamma} \end{aligned}$$

Use: $\|B^i\| \leq \|B\|^i$.

Use: (4.5)

Geometric series: If $|\gamma| < 1$,

$$\sum_{i=p}^{\infty} \gamma^i = \frac{\gamma^p}{1-\gamma}.$$

The proof is complete by defining $C := C_r\gamma/(1-\gamma)$. □

Taylor definition example

The following program computes $\sin(A)$ for a particular matrix A, by using the Taylor expansion of $\sin(z)$. The same example is used below in the other definitions.

```
>> A=[1,2; -5, 4]
A =
     1     2
    -5     4
>> m=21;  FT=zeros(2);
>> for k=1:2:m; FT=FT+(A^k)*(-1)^((k-1)/2)/factorial(k); end
>> FT
FT =
   8.339880980278327  -4.638979409410489
  11.597448523526220   1.381411866162594
```

○

4.1.2 *Jordan form definition*

Although the definition (4.4) is very natural, it turns out that matrix functions can be more generally defined in a different way. First note that a matrix function stemming from the Taylor definition commutes with similarity transformation in the sense that

$$f(XBX^{-1}) = Xf(B)X^{-1} \tag{4.6}$$

and it commutes with the diag operator in the sense that

$$f(\text{diag}(F_1,\ldots,F_k)) = \text{diag}(f(F_1),\ldots,f(F_k)), \tag{4.7}$$

where $F_1,\ldots,F_k$ are square matrices. Therefore, if we want a definition which is consistent with the Taylor definition, it must satisfy (4.6) and (4.7). The definition of matrix functions that follows is based on (4.6) and (4.7), where we select the transformation as the transformation associated with the Jordan canonical form (JCF). Suppose $A \in \mathbb{C}^{n\times n}$ and let

$$A = X\,\text{diag}(J_1,\ldots,J_q)X^{-1} \tag{4.8}$$

be the Jordan canonical form with

$$J_i = \begin{bmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix} \in \mathbb{C}^{n_i\times n_i}. \tag{4.9}$$

In order to be consistent with the Taylor defintion, the matrix function $f(A)$ must satisfy

$$f(A) := X\,\text{diag}(f(J_1),\ldots,f(J_q))X^{-1}.$$

If $J_1,\ldots,J_q$ are scalars, which is the case if the matrix is diagonalizable, this forms a definition since $f(J_1),\ldots,f(J_q)$ are well defined. For non-diagonalizable matrices, we need to define the meaning of a matrix function applied to a Jordan block. We justify the definition of a matrix function of a Jordan block with an example.

Definition of diag operator:

$$\text{diag}(F_1,\ldots,F_k) := \begin{bmatrix} F_1 & & \\ & \ddots & \\ & & F_k \end{bmatrix}$$

In matlab, the corresponding command is `blkdiag`.

Jordan block example

Suppose $p(A)$ is a polynomial of degree four $p(A) = A^4$. Let us investigate this function when $A = J$ is a Jordan block

$$J = \begin{bmatrix} \lambda & 1 & 0 \\ 0 & \lambda & 1 \\ 0 & 0 & \lambda \end{bmatrix}$$

where $\lambda = 1$, $\lambda = 2$ and $\lambda = 10$.

```
>> p=@(A) A*A*A*A;
>> s=1; p([s 1 0; 0 s 1; 0 0 s])
ans =
     1     4     6
     0     1     4
     0     0     1
>> s=2; p([s 1 0; 0 s 1; 0 0 s])
ans =
    16    32    24
```

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

```
     0    16    32
     0     0    16
>> s=10; p([s 1 0; 0 s 1; 0 0 s])
ans =
       10000        4000         600
           0       10000        4000
           0           0       10000
```

From the results of the simulation, we identify that, at least for the this Jordan block and this matrix function, the result of the matrix function applied to a Jordan block satisfies

$$p(J) = \begin{bmatrix} p(\lambda) & p'(\lambda) & \frac{1}{2}p''(\lambda) \\ 0 & p(\lambda) & p'(\lambda) \\ 0 & 0 & p(\lambda) \end{bmatrix}.$$

The conclusion in the example turns out to be a general property, and the matrix function of a Jordan block of size r can be consistently defined via the first $r-1$ derivatives of the function f evaluated in the eigenvalue.

Definition 4.1.3 (Jordan canonical form (JCF) definition)**.** *Suppose $A \in \mathbb{C}^{n\times n}$ and let X and $J_1, \ldots, J_q$ be the Jordan canonical form (4.8)-(4.9). The JCF-definition of the matrix function $f(A)$ is given by*

$$f(A) := X \operatorname{diag}(F_1, \ldots, F_q) X^{-1}, \tag{4.10}$$

where

$$F_i = f(J_i) := \begin{bmatrix} f(\lambda_i) & \frac{f'(\lambda_i)}{1!} & \cdots & \frac{f^{(n_i-1)}(\lambda_i)}{(n_i-1)!} \\ & \ddots & \ddots & \vdots \\ & & \ddots & \frac{f'(\lambda_i)}{1!} \\ & & & f(\lambda_i) \end{bmatrix} \in \mathbb{C}^{n_i \times n_i}. \tag{4.11}$$

The Jordan form definition is more general than the Taylor definition. The Jordan form definition only requires a finite number of derivatives in the eigenvalues, whereas the Taylor definition requires all derivatives in the expansion point and a convergent Taylor series. They are equivalent for many functions, for instance all entire functions as is illustrated later (in Theorem 4.1.5).

Jordan definition example

We continue the example for the Taylor definition.

```
>> [V,D]=eig(A);
>> F=diag(sin(diag(D)));
>> FJ=V*F*inv(V)
FJ =
   8.339880979874099  -4.638979409584841
  11.597448523962106   1.381411865496835
```

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

4.1.3 *Cauchy integral definition*

Suppose f is analytic inside and on a simple, closed, piecewise-smooth curve Γ, which encloses a point $z = x \in \mathbb{C}$ counter-clockwise. From complex analysis, in particular Cauchy's integral formula, we know that

$$f(x) = \frac{1}{2i\pi} \oint_\Gamma \frac{f(z)}{z-x}\, dz. \tag{4.12}$$

Note that the only way the right-hand side of (4.12) depends on x, is in the expression $1/(z-x)$. We now want to generalize this formula when x is a matrix. In order to reach a different definition of matrix functions we replace $1/(z-x)$ by $(zI - X)^{-1}$.

The Cauchy integral definition is the basis of the derivation of Krylov methods for matrix functions in Section 4.4

Definition 4.1.4 (Cauchy integral definition)**.** *Suppose f is analytic inside and on a simple, closed, piecewise-smooth curve Γ, which encloses the eigenvalues of A once counter-clockwise. The Cauchy integral definition of matrix functions is given by*

$$f(A) := \frac{1}{2i\pi} \oint_\Gamma f(z)(zI - A)^{-1}\, dz.$$

Cauchy integral definition example

We illustrate the Cauchy integral definition by integrating over a circle of radius r. The parameterization $z = re^{i\theta}$ of the circle gives, $dz/d\theta = ire^{i\theta}$ and

$$f(A) = \frac{1}{2i\pi} \int_0^{2\pi} f(re^{i\theta})((re^{i\theta} I - A)^{-1})\, ire^{i\theta}\, d\theta,$$

which we for illustration purposes integrate numerically:

```
>> g=@(z) sin(z)*inv(z*eye(size(A))-A)
>> r=5;
>> FC=quadv(@(t) g(exp(1i*t)*r)*1i*r*exp(1i*t),0,2*pi)/(2i*pi)
FC =
   8.3399 + 0.0000i  -4.6390 + 0.0000i
  11.5974 - 0.0000i   1.3814 + 0.0000i
```

If we integrate over a circle of radius $r = 5$ we get the correct matrix function value. When the radius is smaller than the absolute value of the largest eigenvalue, we get a different result. The eigenvalues of the matrix in this example are $\lambda_i \approx 2.5 \pm 2.8i$.

We get the wrong result when the contour does not include the eigenvalues.

```
>> r=3;
>> FC2=quadv(@(t) g(exp(1i*t)*r)*1i*r*exp(1i*t),0,2*pi)/(2i*pi)
FC2 =
   1.0e-08 *
   0.5648 - 0.0000i  -0.1300 - 0.0000i
   0.3251 + 0.0000i   0.3697 + 0.0000i
```

4.1.4 Equivalence of definitions

The definitions above have different domains of definitions. However, for a large class of functions the definitions are equivalent.

Equivalence illustration

We observe directly from the examples in the definitions above that the definitions are equal up to approximation errors, for this specific example.

```
>> should_be_zero=norm(FT-FJ)
should_be_zero =
     8.008177121446691e-10
>> should_be_zero=norm(FJ-FC)
should_be_zero =
     3.527726719978948e-08
>> should_be_zero=norm(FT-FC)
should_be_zero =
     3.519037727771136e-08
```

○

More formally, a sufficient condition for the definitions to be equivalence is that the functions are analytic in $\mathbb{C}$.

Theorem 4.1.5 (Equivalence of the matrix function definitions)**.** *Suppose f is an entire function and suppose $A \in \mathbb{C}^{n\times n}$. Then, the matrix function definitions (Definition 4.1.1, Definition 4.1.3 and Definition 4.1.4) are equivalent.*

The proof of equivalance in Theorem 4.1.5 is available in appendix but not a part of the course.

4.2 Methods for general matrix functions

4.2.1 Truncating the Taylor series (naive approach)

We saw in Section 4.1.1 that matrix functions can be defined with Taylor series, and that the truncated Taylor series converges for analytic functions. The truncated Taylor series can be used as a method to compute matrix functions when the derivatives of the scalar-valued function are explicitly available.

In general we need N matrix vector products, which (unless there is particular structure) can be computed on $\mathcal{O}(n^3)$. Hence, the total computation time of a truncated Taylor series is

$$\mathcal{O}(Nn^3).$$

The value N can be large if the series converges slowly. In comparison to other general methods below, the computation time of truncated Taylor series is mostly not competitive.

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

4.2.2 *Eigenvalue-eigenvector approach (naive approach)*

If the matrix is diagonalizable, the Jordan form consists of diagonal elements, and the columns of X consists of the eigenvectors. Hence, the formula

$$f(A) = X \operatorname{diag}(f(\lambda_1), \ldots, f(\lambda_n)) X^{-1}$$

gives a procedure to compute $f(A)$ if we can compute X and $\lambda_1, \ldots, \lambda_n$.

From previous parts of the course: The standard rough estimate for the computation of the eigenpairs of a dense matrix with a QR-method $\mathcal{O}(n^3)$.

Under the assumption that computing all eigenpairs has computational cost $\mathcal{O}(n^3)$, an eigen-decomposition approach has computational cost

$$\mathcal{O}(n^3).$$

For large-scale problems, the eigendecomposition approach is in general faster than the truncated Taylor series. However, we learned in other parts of this course that the Jordan canonical form is not numerically stable with respect to unstructured rounding errors. Very small rounding errors can generate very large errors in the output. This is often the case for non-symmetric matrices where the eigenvalues are close to eachother. This can seriously jeopardize the reliability of the method.

4.2.3 *The Schur-Parlett method*

An advanced version of the Schur-Parlett method is implemented in the matlab command `funm`.

The Schur form does not suffer from the same rounding errors disadvantages as the Jordan form. For this reason, in the parts of the course where we learned about eigenvalue computations, in particular when we learned the QR-method, we focused the Schur factorization, rather than a Jordan decomposition. The Schur-Parlett method is analogously based on the Schur factorization rather than the Jordan decomposition, which we used (for diagonalizable matrices) in Section 4.2.2.

Suppose $A = Q^* T Q$ is a (complex) Schur factorization, where $Q \in \mathbb{C}^{n \times n}$ is an orthogonal matrix and T an upper triangular matrix. From the fact that all definitions commute with similarity transformation (4.6) we can use that

$$f(A) = Q^* f(T) Q. \tag{4.13}$$

Hence, an approach based on the Schur factorization can be implemented in a straightforward way with (4.13) if we know how to compute the matrix function for the triangular matrix T. This can be done as follows.

As a first step to a general method for triangular matrices, we derive an explicit formula for two-by-two triangular matrices. All of the matrix-function definitions satisfy

$$FT = TF \tag{4.14}$$

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

where $F = f(T)$. Consequently,

$$\begin{bmatrix} f_{11} & f_{12} \\ 0 & f_{22} \end{bmatrix}\begin{bmatrix} t_{11} & t_{12} \\ 0 & t_{22} \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ 0 & t_{22} \end{bmatrix}\begin{bmatrix} f_{11} & f_{12} \\ 0 & f_{22} \end{bmatrix}$$

The diagonal elements are explicitly $f_{11} = f(t_{11})$ and $f_{22} = f(t_{22})$. The equality corresponding to the (1,2) entry is

$$f_{11}t_{12} + f_{12}t_{22} = t_{11}f_{12} + t_{12}f_{22}.$$

This scalar equation can be solved for f_{12},

$$f_{12} := t_{12}\frac{f(t_{22}) - f(t_{11})}{t_{22} - t_{11}}.$$

The formula is a special case of the more general result.

Theorem 4.2.1. *Suppose $T \in \mathbb{C}^{n\times n}$ is an upper triangular matrix with distinct eigenvalues and denote $F := f(T)$ with elements f_{ij}, $i = 1,\dots,n$, $j = 1,\dots,n$. Then, for any i and any $j > i$,*

$$f_{ij} = \frac{s}{t_{jj} - t_{ii}} \tag{4.15}$$

where

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

Proof. Consider the ith row and jth column of the equality $0 = FT - TF$,

$$0 = \sum_{k=1}^{n} f_{ik}t_{kj} - \sum_{k=1}^{n} t_{ik}f_{kj} = \sum_{k=i}^{j} f_{ik}t_{kj} - t_{ik}f_{kj}$$

In the second equality we used that F and T are triangular matrices $t_{ik} = f_{ik} = 0$ when $k < i$ and $t_{kj} = f_{kj} = 0$ when $k > j$. We separate the terms $k = i$ and $k = j$ from the sum and rearrange the terms

$$f_{ii}t_{ij} + f_{ij}t_{jj} - t_{ii}f_{ij} - t_{ij}f_{jj} = \sum_{k=i+1}^{k=j-1} t_{ik}f_{kj} - f_{ik}t_{kj}. \tag{4.16}$$

The expression (4.15) follows from solving (4.16) for f_{ij}. □

Clearly the diagonal elements of F are known from the start since they are explicitly $f_{ii} = f(t_{ii})$, $i = 1,\dots,n$. We can now compute the elements of F by working one subdiagonal at a time. Let □ denote values that we do not know yet and let + denote quantities we have already computed. By applying Theorem 4.2.1 repeatedly for different i and j we can proceed as follows.

Graphical illustration of the elements of $F = f(T)$ and T required to compute f_{ij} in Theorem 4.2.1:

F:

$$\begin{array}{cc} & \\ & \\ i & \rightarrow \end{array}\begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{c} \\ \\ j \\ \downarrow \\ \\ \\ \\ \\ \end{array}\begin{bmatrix} + & + & + & + & + & \square & \square & \square \\ 0 & + & + & + & + & + & \square & \square \\ 0 & 0 & + & + & + & + & f_{ij} & \square \\ 0 & 0 & 0 & + & + & + & + & \square \\ 0 & 0 & 0 & 0 & + & + & + & + \\ 0 & 0 & 0 & 0 & 0 & + & + & + \\ 0 & 0 & 0 & 0 & 0 & 0 & + & + \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & + \end{bmatrix}$$

T:

$$\begin{array}{cc} & \\ & \\ i & \rightarrow \end{array}\begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{c} \\ \\ j \\ \downarrow \\ \\ \\ \\ \\ \end{array}\begin{bmatrix} + & + & + & + & + & + & + & + \\ 0 & + & + & + & + & + & + & + \\ 0 & 0 & + & + & + & + & + & + \\ 0 & 0 & 0 & + & + & + & + & + \\ 0 & 0 & 0 & 0 & + & + & + & + \\ 0 & 0 & 0 & 0 & 0 & + & + & + \\ 0 & 0 & 0 & 0 & 0 & 0 & + & + \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & + \end{bmatrix}$$

$$\begin{bmatrix} + & \square & \square & \square \\ & + & \square & \square \\ & & + & \square \\ & & & + \end{bmatrix} \overset{\substack{i=1\\j=2}}{\Rightarrow} \begin{bmatrix} + & + & \square & \square \\ & + & \square & \square \\ & & + & \square \\ & & & + \end{bmatrix} \overset{\substack{i=2\\j=3}}{\Rightarrow} \begin{bmatrix} + & + & \square & \square \\ & + & + & \square \\ & & + & \square \\ & & & + \end{bmatrix} \overset{\substack{i=3\\j=4}}{\Rightarrow} \begin{bmatrix} + & + & \square & \square \\ & + & + & \square \\ & & + & + \\ & & & + \end{bmatrix} \overset{\substack{i=1\\j=3}}{\Rightarrow} \begin{bmatrix} + & + & + & \square \\ & + & + & \square \\ & & + & + \\ & & & + \end{bmatrix} \overset{\substack{i=2\\j=4}}{\Rightarrow} \begin{bmatrix} + & + & + & \square \\ & + & + & + \\ & & + & + \\ & & & + \end{bmatrix} \overset{\substack{i=1\\j=4}}{\Rightarrow} \begin{bmatrix} + & + & + & + \\ & + & + & + \\ & & + & + \\ & & & + \end{bmatrix}$$

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

The complete more general process is summarized in Algorithm 1.

Input: A triangular matrix $T \in \mathbb{C}^{n\times n}$ with distinct eigenvalues
Output: The matrix function $F = f(T)$
for $i = 1, \ldots, n$ **do**
  $f_{ii} = f(t_{i,i})$
end
for $p = 1, \ldots, n-1$ **do**
  for $i = 1, \ldots, n-p$ **do**
    $j = i + p$
    $s = t_{ij}(f_{jj} - f_{ii})$
    for k=i+1,. . . ,j-1 **do**
      $s = s + t_{ik}f_{kj} - f_{ik}t_{kj}$
    end
    $f_{ij} = s/(t_{jj} - t_{ii})$
  end
end

Algorithm 1: Simplified Schur-Parlett method

We call Algorithm 1 *simplified* Schur-Parlett since in practice one needs to consider a number of issues. Most importantly, the reasoning above is only for matrices with distinct eigenvalues. If the eigenvalues are close to eachother, the approach may suffer from stability issues. This can be resolved by using a different block Schur form and a different method when the distance between eigenvalues is small.

4.3 *Methods for specialized matrix functions*

4.3.1 *Scaling-and-squaring for the matrix exponential*

We saw above that the Taylor approximation can give approximations, which match the derivatives of the function in an expansion point μ. In our derivation of the most important method for the matrix exponential, we will use another useful class of approximants is the Padé approximants, which are rational functions

$$R_{pq}(z) = \frac{N_{pq}(z)}{D_{pq}(z)}$$

where N_{pq} and D_{pq} are appropriate polynomials. Similar to the Taylor expansion, the polynomials N_{pq} and D_{pq} are constructed such that they match the derivative of the function (at the origin).

For the matrix exponential, it can be shown that the Padé approxi-

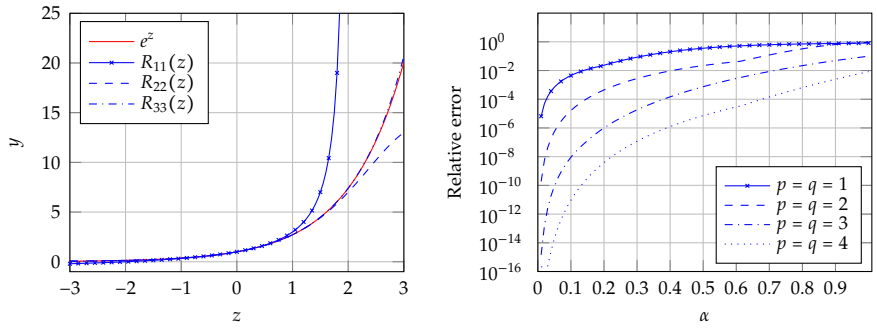
The matrix exponential is the most commonly used matrix function. The scaling-and-squaring algorithm is a very robust algorithm for the matrix exponential and it is implemented in the matlab function `expm`. A robust implementation of scaling-and-squaring was available already in one of the very first matlab versions. According to some expert in numerical linear algebra, this was one of the main reasons for the early success of matlab.

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

mants are

$$N_{pq}(z) = \sum_{k=0}^{p} \frac{(p+q-k)!p!}{(p+q)!k!(p-k)!} z^k$$
$$D_{pq}(z) = \sum_{k=0}^{q} \frac{(p+q-k)!q!}{(p+q)!k!(p-k)!} (-z)^k.$$

These rational functions R_{pq} are constructed such that $p+q$ derivatives of R_{pq} match the derivatives of e^z.



(a) Error in Padé approximation

(b) $\|R_{pq}(\alpha A) - \exp(\alpha A)\| / \|\exp(\alpha A)\|$

Figure 4.1: Illustration of the Padé approximation for the matrix exponential.

This type of Padé approximants are only good approximations near the origin. They can be used to directly good approximations of the matrix exponential

$$\exp(A) \approx R_{pq}(A) = D_{pq}(A)^{-1} N_{pq}(A),$$

when $\|A\|$ is small.

It will in general not be a good approximation if $\|A\|$ is large. Fortunately, this problem can be handled by exploiting the fact that

$$\exp(A) = \exp(A/m)^m, \tag{4.17}$$

for any m. In particular, we may scale A by m such that $R_{pq}(A/m)$ is an accurate approximation of $\exp(A/m)$. The approximation of $\exp(A)$, can subsequently be computed by forming $R_{pq}(A/m)^m$. If m is a power of two, $m = 2^j$, then this last stage can be done very efficiently, with j matrix-matrix multiplications.

By further considerations of the approximant it can be shown that if

$$\frac{\|A\|_\infty}{2^j} \leq 1/2$$

The property (4.17) can be derived from the simple fact that

$$f(A)g(A) = h(A)$$

where $h(z) = f(z)g(z)$.

Further details of the error analysis can be found in references [4, Chapter 11.3]

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

then there exists $E \in \mathbb{R}^{n\times n}$ such that

$$F_{pq} = e^{A+E} \tag{4.18}$$
$$AE = EA \tag{4.19}$$
$$\|E\|_\infty \leq \varepsilon(p,q)\|A\|_\infty \tag{4.20}$$
$$\varepsilon(p,q) = 2^{3-(p+q)}\frac{p!q!}{(p+q)!(p+q+1)!} \tag{4.21}$$

By using these bounds we arrive at the main algorithm for the matrix exponential. The algorithm consists of first computing the rational function and subsequently repeatedly squaring the result.

Input: $\delta > 0$ and $A \in \mathbb{R}^{n\times n}$
Output: $F = \exp(A+E)$ where $\|E\|_\infty \leq \delta\|A\|_\infty$.
begin
  $j = \max(0, 1 + \text{floor}(\log_2(\|A\|_\infty)))$
  $A = A/2^j$
  Let q be the smallest non-negative integer such that
   $\varepsilon(q,q) \leq \delta$.
  $D = I; N = I; X = I; c = 1$
  for $k = 1 : q$ **do**
    $c = c(q-k+1)/((2q-k+1)k)$
    $X = AX; N = N + cX; D = D + (-1)^k cX$
  end
  Solve $DF = N$ for F
  for $k = 1 : j$ **do**
    $F = F^2$
  end
end

Algorithm 2: Scaling-and-squaring for the matrix exponential

4.3.2 *Matrix square root*

The matrix square root can be used to solve partial-differential equations with certain types of absorbing boundary conditions. It has applications in for instance quantum physics (density functional theory) and it can be used to compute matrix logarithms and polar decompositions.

We have now seen many examples of matrix functions which stemming from entire functions. The matrix square root is an important example of a function which is not analytic in the entire plane. In fact, it is usually defined by a system of equations rather than as an extension of a scalar valued function. Any matrix X that satisfies $X^2 = A$ is called a matrix square root of A. If A does not have any eigenvalues in $(-\infty, 0] \subset \mathbb{C}$, there exists a specific matrix square root called the principal square root, which we can define with the definitions above as follows. Suppose $F = f(A)$ is defined with Definition 4.1.3 for $f(x) = \sqrt{x}$. Then,

$$F^2 = X\,\text{diag}(F_1,\ldots,F_q)X^{-1}X\,\text{diag}(F_1,\ldots,F_q)X^{-1} = X\,\text{diag}(F_1^2,\ldots,F_q^2)X^{-1} = X\,\text{diag}(J_1,\ldots,J_q)X^{-1} = A,$$

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

since

$$F_i^2 = J_i, \quad i = 1, \ldots, q.$$

For the scalar case, $x = \sqrt{a}$ is obviously a root to the nonlinear scalar equation $g(x) = x^2 - a = 0$. Newton-Raphson's method for $g(x)$ is, $x_{k+1} = x_k - g(x_k)/g'(x_k) = x_k - (x_k^2 - a)/2x_k = x_k/2 + a/2x_k$.

The matrix function generalization is

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}A). \tag{4.22}$$

Although the iteration (4.22) can be shown to be equivalent to Newton's method in matrix form and therefore exhibits local quadratic convergence in general, it is in a certain sense numerically unstable. This can be explained by the fact that local convergence of (4.22) requires commutativity of $X_0, X_1, \ldots$. Due to round-off errors the commutativity is not satisfied. Suppose we have an accurate approximation of $A^{1/2}$ such that $X_k = A^{1/2} + \varepsilon\Delta$, where ε will be thought of as a small scalar parameter. Then,

$$\begin{aligned} X_{k+1} &= \frac{1}{2}\left(A^{1/2} + \varepsilon\Delta + (A^{1/2} + \varepsilon\Delta)^{-1}A\right) && \text{(4.23a)} \\ &= A^{1/2} + \frac{\varepsilon}{2}(\Delta - A^{-1/2}\Delta A^{1/2}) + \mathcal{O}(\varepsilon^2). && \text{(4.23b)} \end{aligned}$$

The reasoning in (4.23) suggests at least loss of quadratic convergence due to round-off errors, and justifies the bad performance of (4.22).

The term $\Delta - A^{-1/2}\Delta A^{1/2}$ vanishes if Δ commutes with A, otherwise it will in general not vanish.

Denman and Beavers proposed [3] an equivalent variant of (4.22) which does not exhibit the same sensitivity with respect to rounding errors. The Denman-Beavers iteration is derived as follows. Due to the fact that $X_0 = A$, the matrices $X_0, X_1, \ldots$ commute. Hence, $AX_k^{-1} = A^{1/2}X_k^{-1}A^{1/2}$ and equation (4.22) can be rewritten as

$$X_{k+1} = \frac{1}{2}(X_k + A^{1/2}X_k^{-1}A^{1/2}).$$

If we want to use the Schur-Parlett method (in Section 4.2.3) to compute the principal matrix square root we need to compute a Schur factorization of A. The Denman-Beavers iteration (4.24) does not require a Schur factorization, but instead requires an inverse. In some situations the inverse is easier to compute than the Schur factorization, for instance if A is tridiagonal.

By setting $Y_k = A^{-1/2}X_kA^{-1/2}$, we have an iteration with two matrices

$$\begin{aligned} X_{k+1} &= \frac{1}{2}(X_k + Y_k^{-1}) && \text{(4.24a)} \\ Y_{k+1} &= \frac{1}{2}(Y_k + X_k^{-1}) && \text{(4.24b)} \end{aligned}$$

with initial conditions $X_0 = A$ and $Y_0 = I$. By construction, if $X_k \to A^{1/2}$, $Y_k \to A^{-1/2}$.

It can be shown that this iteration is not as sensitive to rounding errors. In contrast to (4.22) the convergence of the Denman-Beavers iteration (4.24) does not depend in the same way on commutativity.

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

4.3.3 *Matrix sign function*

In systems and control, the underlying method to solve the Riccati equation is based on the matrix sign function.

One of the leading methods in quantum chemistry is derived from the matrix sign function.

The (scalar-valued) sign function is for real numbers defined as

$$\operatorname{sign}(a) = \begin{cases} -1 & a < 0 \\ 1 & a > 0 \end{cases}$$

and normally undefined for $a = 0$. The matrix sign function is another matrix function which appears frequently in various fields. In order to construct a matrix function generalization, we first generalize the sign to complex numbers. We here work only with the following property which generalizes consistently to complex numbers,

$$\operatorname{sign}(a) = \frac{|a|}{a} = \frac{\sqrt{a^2}}{a}.$$

We have in the previous section seen how to define the matrix square root, so we can in a consistent way generalize the matrix sign as follows

$$\operatorname{sign}(A) := A^{-1}\sqrt{A^2}. \tag{4.25}$$

The previous section directly also gives us an iterative approach for the matrix sign function, by first computing $B = A^2$, $C = \sqrt{B}$ multiplying by $A^{-1}C$. However, rather than taking these separate steps it turns out to be advantageous to integrate the operations of such a Newton approach (4.22) into a quite simple iteration. If we apply (4.22) to compute the square root of the square, we have, $X_0 = A^2$ and

$$X_{k+1} = \frac{1}{2}(X_k + A^2 X_k^{-1}) \tag{4.26}$$

and $\operatorname{sign}(A) = A^{-1}X_\infty$. We now rephrase the iteration by defining

$$S_k = A^{-1}X_k, \tag{4.27}$$

such that $\operatorname{sign}(A) = S_\infty$. Note that $S_0 = A^{-1}X_0 = A^{-1}A^2 = A$. Moreover, after carrying out the substitution (4.27), (4.26) can be equivalenty phrased as

$$S_{k+1} = \frac{1}{2}(S_k + S_k^{-1}). \tag{4.28}$$

The iteration (4.28) is usually referred to as the Newton method for matrix sign function.

* Theorem of equivalence *

It has global quadratic convergence in exact arithmetic, in the following sense.

Theorem 4.3.1 (Global quadratic convergence of (4.28))**.** *Suppose $A \in \mathbb{R}^{n\times n}$ has no eigenvalues on the imaginary axis. Let $S = \operatorname{sign}(A)$, and S_k be generated by* (4.28)*. Let*

$$G_k := (S_k - S)(S_k + S)^{-1}. \tag{4.29}$$

Then,

- $S_k = S(I + G_k)(I - G_k)^{-1}$ *for all* k,
- $G_k \to 0$ *as* $k \to \infty$,
- $S_k \to S$ *as* $k \to \infty$, *and*
- $$\|S_{k+1} - S\| \leq \frac{1}{2}\|S_k^{-1}\|\|S_k - S\|^2. \tag{4.30}$$

Proof. From the fact that $S_k S = S S_k$ and $S^2 = I$, we have

$$\begin{aligned}(S_k \pm S)^2 &= S_k^2 \pm 2SS_k + I \\ &= S_k(S_k + S_k^{-1} \pm 2S) \\ &= 2S_k(S_{k+1} \pm S)\end{aligned} \tag{4.31}$$

The inequality (4.30) follows by taking norms of (4.31) with the minus sign. Now note that (4.31) imply that

$$G_{k+1} = (S_{k+1} - S)(S_{k+1} + S)^{-1} = S_k^{-1}(S_k - S)^2(S_k + S)^2 S_k = (S_k - S)^2(S_k + S)^{-2} = G_k^2 \tag{4.32}$$

such that $G_k = G_0^{2^k}$. The eigenvalues of G_0 are $\mu = (\lambda - \text{sign}(\lambda))/(\lambda + \text{sign}(\lambda))$ such that $|\mu| < 1$ and $G_k \to 0$. Finally, the proof is completed by noting that

$$S_k = S(I + G_k)(I - G_k)^{-1}.$$

□

If all eigenvalues of a matrix B are less than one in modulus, we have

$$\|B^m\| = \|V^{-1}\Lambda^m V\| \leq \|V^{-1}\|\|V\|\|\Lambda^m\| \to 0$$

when $m \to \infty$.

Sign function and Riccati equation

An important application of matrix sign function is its use in a solution method for quadratic matrix equation called the Riccati equation

$$G + A^T X + XA - XFX = 0. \tag{4.33}$$

The Riccati equation (4.33) is an important equation in systems and control, in particular stochastic and optimal control. Riccati equations can be solved with the MATLAB command `care`.

It can be solved for X when F and G symmetric positive definite as follows. For illustative purposes we computed the matrix sign with formula (4.25).

The matrix sign function approach is considered one of the leading numerical methods for the Riccati equation. A derivation of the formula in the example can be found in [2].

```
>> A=[2 1 ; 2 2]; F=[5 4; 4 6];G=[1 -1 ; -1 3];
>> K=[A' G; F -A];
>> signm=@(X) inv(X)*sqrtm(X^2);
>> W2=signm(K)-eye(4);
>> [Q,R]=qr(W2(:,1:2));
>> X=-R\Q'*W2(:,3:4);
>> norm(G+A'*X+X*A-X*F*X)
ans =
   4.2717e-15
```

4.4 Krylov methods for $f(A)b$

We have above seen several methods to compute $f(A)$ for specialized f as well as general situations. These methods are mostly suitable for dense matrices. We will now see a method which is suitable for large and sparse problems. In many applications, it is sufficient to compute the action of the matrix function on a vector:

$$f(A)b \tag{4.34}$$

where $b \in \mathbb{C}^n$. The method we present next computes (4.34) directly, instead of first computing $f(A)$ and then multiplying by b. It is based on Arnoldi's method and the only way A is accessed in the algorithm is via matrix-vector products.

4.4.1 Derivation of Arnoldi approximation of matrix functions

The Cauchy integral formula definition (Definition 4.1.4) can be combined with the quantity we want to compute (4.34) such that

$$f(A)b = \frac{-1}{2i\pi} \oint_\Gamma f(z)((A - zI)^{-1}b)\, dz. \tag{4.35}$$

In the approach we take now, the shifted linear system of equations

$$x = (A - zI)^{-1}b, \tag{4.36}$$

in (4.35) is approximated by means of Krylov subspace approximations.

Arnoldi decomposition for shifted linear systems

Our approximation of (4.36) is computed by applying Arnoldi's method to the shifted linear system. We first establish some properties of Arnoldi's method when it is applied to a linear systems.

Due to the fact that the span is unchanged by adding or subtracting multiples of the vectors, from the definition of a Krylov subspace one can verify that

$$\mathcal{K}_m(A - \sigma I, b) = \mathcal{K}_m(A, b).$$

This implies that the Krylov subspace is independent of shift. The Arnoldi method is a procedure which generates an Arnoldi factorization,

$$AQ_m = Q_m H_{m+1,m} + e_m^T q_{m+1} h_{m+1,m}, \tag{4.37}$$

where the columns of Q_m span the associated Krylov subspace. The shift-invariance holds also for the Arnoldi factorization.

Lemma 4.4.1. *Suppose $Q_m \in \mathbb{C}^{n\times m}$, $\underline{H}_m \in \mathbb{C}^{(m+1)\times m}$ is an Arnoldi factorization (4.37) associated with $\mathcal{K}_m(A, b)$.*

As we have seen earlier in the course, a Krylov subspace is defined by the span of the iterates of the power method,

$$\mathcal{K}_m(A, b) := \text{span}(b, Ab, \ldots, A^{m-1}b).$$

The shift-invariance illustrated in Lemma 4.4.1 shows that from the Arnoldi decomposition of a matrix, we can construct the Arnoldi decomposition of the shifted matrix without any additional cost.

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

Then, for any $\sigma \in \mathbb{C}$, $Q_m \in \mathbb{C}^{n\times m}$ and $\underline{H}_m - \sigma I_{m+1,m}$ is an Arnoldi factorization associated with $\mathcal{K}_m(A-\sigma I, b)$,

$$(A-\sigma I)Q_m = Q_{m+1}(\underline{H}_m - \sigma I_{m+1,m}). \tag{4.38}$$

Notational convenience in (4.38):

$$I_{m+1,m} = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \\ 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{(m+1)\times m}.$$

Proof. The conclusion (4.38) follows from subtracting σQ_m from the Arnoldi factorization $AQ_m = Q_{m+1}\underline{H}_m = Q_m H_m + h_{m+1,m} q_{m+1} e_m^T$. □

Hence, the Arnoldi factorization associated with $\mathcal{K}_m(A-\sigma I, b)$ can be easily reconstructed from the Arnoldi factorization associated with $\mathcal{K}_m(A,b)$, by shifting the Hessenberg matrix H_m and using the same basis matrix Q_m.

FOM for shifted linear system of equations

In the part in this course on sparse linear systems, we learned that GMRES was a natural procedure to extract an approximation of a linear system by means of minimizing the residual over approximations in a Krylov subspace. For the purpose of approximating $f(A)b$, via approximation of $(A-zI)^{-1}b$ we will work with a slightly different way to extract an approximation from the Krylov subspace, called the FOM-approximation.

FOM is an abbreviation for *full orthogonalization method*. The name was historically given to distinguish it from other (now less used) methods based on incomplete orthogonalization in the underlying Gram-Schmidt process.

We define the FOM-approximation of the linear system $Ax = b$ by

$$\tilde{x} = Q_m H_m^{-1} Q_m^T b = Q_m H_m^{-1} e_1 \|b\|. \tag{4.39}$$

GMRES vs FOM: The approximation (4.39) corresponds to an element of $\mathcal{K}_m(A,b)$ such that the residual satisfies $Q_m^T(A\tilde{x}-b) = 0$, whereas the GMRES approximation corresponds to an element of $\mathcal{K}_m(A,b)$ which minimizes $\min_{x\in\mathcal{K}_m(A,b)} \|Ax-b\|_2 = \|A\tilde{x}-b\|_2$.

This approximation can be derived by assuming that $\tilde{x} \in \mathcal{K}_m(A,b)$ and imposing that the residual is orthogonal to the mth Krylov subspace, $Q_m^T(A\tilde{x}-b) = 0$.

Due to Lemma 4.4.1, the Krylov approximation of the shifted linear system $(A-\sigma I)x = b$ is

$$\tilde{x} = Q_m(H_m - \sigma I)^{-1} e_1 \|b\|. \tag{4.40}$$

Note that this approximation can be computed for many σ by only computing one Arnoldi factorization.

The Krylov approximation of a matrix function

By using the approximation stemming from FOM applied to a linear system expressed in (4.40) with the Cauchy integral formulation (4.35), we have

$$f(A)b \approx \frac{-1}{2i\pi} \oint_\Gamma f(z) Q_m (H_m - zI)^{-1} e_1 \|b\| \, dz =$$
$$Q_m \frac{1}{2i\pi} \oint_\Gamma f(z)(zI - H_m)^{-1} \, dz (e_1\|b\|) = Q_m f(H_m) e_1 \|b\|.$$

This serves as a justification of our approximation scheme.

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

Definition 4.4.2 (Krylov approximation of a matrix function)**.** *Let $Q_m \in \mathbb{C}^{n\times m}$ and $H_{m,m} \in \mathbb{C}^{m\times m}$ correspond to an Arnoldi factorization (4.37) of the matrix A with $q_1 = b/\|b\|$. The Krylov approximation of a matrix function is defined as*

$$f_m := Q_m f(H_m) e_1 \|b\|. \tag{4.41}$$

Note that the Krylov approximation (4.41) also requires the evaluation of a matrix function. However, the Hessenberg matrix H_m is in general much smaller than the original problem and computing $f(H_m)$ is relatively inexpensive in comparison to carrying out the Arnoldi method, at least if $m \ll n$, which is typically the case for large and sparse problems.

Example

For many problems, the convergence is superlinear in practice. See the video demonstration

`http://www.math.kth.se/~eliasj/krylov_matfun_approx.mp4`

4.4.2 *Convergence theory of Krylov approximation of matrix functions*

Similar to the Arnoldi method for eigenvalue problems and GMRES, the convergence can be characterized with a min-max expression. To illustrate the convergence, we present only a result for normal matrices. Unlike the other min-max bounds in this course, the maximum is not taken over a discrete set, but a continuous convex compact set Ω containing the eigenvalues of A.

Recall: A *normal matrix* is a matrix satisfying $A^TA = AA^T$, which is the case for instance if the matrix is symmetric. Not all normal matrices are symmetric matrices.

Theorem 4.4.3. *Suppose $A \in \mathbb{C}^{n\times n}$ is a normal matrix and suppose $\Omega \subset \mathbb{C}$ is a convex compact set such that $\lambda(A) \subset \Omega$. Let f_m be the Krylov approximation of $f(A)b$ defined by (4.41). Then,*

$$\|f(A)b - f_m\| \leq 2\|b\| \min_{p\in P_{m-1}} \max_{z\in\Omega} |f(z) - p(z)|.$$

The proof of Theorem 4.4.3 is beyond the scope of the contents of the course. A proof can be found in [*Error estimation and evaluation of matrix functions via the Faber transform*, Beckermann, Reichel, SIAM J. Numer. Anal., 47:3849-3883, 2009]

Interpretation of bound

The bound gives several qualitative interpretations of the error. Sufficient conditions for fast convergence can be easily identified: The method will work well if

- $f(z)$ can be well approximated with low-order polynomials
- $\lambda(A)$ are clustered together such that Ω can be chosen small

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

Similar to other min-max bounds in this course, qualitative understanding can be found by bounding using particular choices of the polynomials. Let q_m be the truncated Taylor expansion

$$q_m(z) := \sum_{i=0}^{m} \frac{f^{(m)}(0)}{i!} z^m.$$

Hence, $r_m(z) = f(z) - q_m$ is the remainder term in the Taylor expansion of f. Suppose now that Ω is a subset of a disk of radius ρ centered at the origin $\Omega \subset D(\rho, 0)$. Then

$$\max_{z \in \Omega} r_m(z) \sim \frac{\rho^{m-1}}{(m-1)!}$$

and $\|f(A)b - f_m\| \leq e_m \sim \frac{\rho^{m-1}}{(m-1)!} \to 0$ as $m \to \infty$. This shows that the method is convergent. The speed of convergence is usually much faster than what is predicted by this Taylor series bound.

4.4.3 *Application of Krylov methods for exponential integrators*

The Krylov approximation techniques for matrix functions have turned out to be useful in combination with techniques for differential equations. We can use them to numerically compute solutions to the ordinary differential equation (ODE),

$$y'(t) = g(y(t)), \quad y(0) = y_0 \tag{4.42}$$

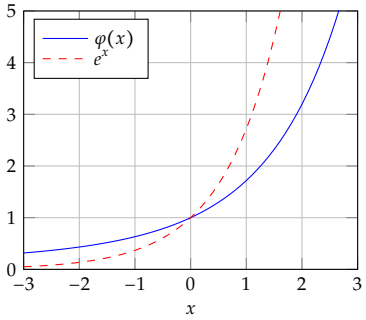
Typically, in our setting, (4.42) stems from the discretization of a partial differential equation and $n \gg 1$

where $y(t) \in \mathbb{C}^n$. In the method class called *exponential integrators*, we use the solution of an approximating linear ODE as a method to integrate the nonlinear ODE (4.42).

A simple exponential integrator can derived by explicitly solving the inhomogeneous linear ODE. For the formulation we use the matrix function corresponding to φ given by

$$\varphi(z) := \frac{e^z - 1}{z}. \tag{4.43}$$

The function (4.43) has a removable singularity at $z = 0$. With a slight abuse of notation we remove it by redefining $\varphi(0) := 1$ such that φ is an entire function. Graphically:



Lemma 4.4.4 (Explicit solution linear inhomogeneous ODE)**.** *The linear inhomogeneous ODE with right-hand side $g(y) = g_1(y) := Ay + b$ and*

$$y'(t) = Ay(t) + b = g_1(y(t)), \quad y(0) = y_0, \tag{4.44}$$

has a solution explicitly given by

$$y(t) = y_0 + t\varphi(tA)g_1(y_0). \tag{4.45}$$

Proof. Although the proof can be done by directly differentiating (4.45) and collecting terms, it is more instructive to carry out a constructive proof. Note that the solution to (4.44) satisfies

$$y(t) = \exp(tA)\left(y(0) + \int_0^t \exp(-\tau A) b \, d\tau\right) \tag{4.46}$$

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

From the Taylor definition of the matrix exponential we can integrate explicitly,

$$\int_0^t \exp(-\tau A)\, d\tau = -A^{-1}(\exp(-tA) - I)$$

such that (4.46) simplifies to

$$\begin{aligned} y(t) &= \exp(tA)y(0) - \exp(tA)A^{-1}(\exp(-tA) - I)b \\ &= \big(y(0) + A^{-1}(\exp(tA) - I)Ay(0)\big) + \big(A^{-1}(\exp(tA) - I)b\big) \\ &= y(0) + t\varphi(tA)g_1(y(0)), \end{aligned} \tag{4.47}$$

Use: $A^{-1}\exp(tA)A = \exp(tA)$

which proves (4.45). □

The formula (4.45) is exact for the ODE (4.44), but in general not for the nonlinear ODE (4.42). The simplest exponential integrator for (4.42) is based on applying (4.45) as a time-stepping method for (4.42). We approximate $y_1 \approx y(h)$ by

$$y_1 = y_0 + h\varphi(hA)g(y_0) \tag{4.48}$$

where $A = g'(y_0)$ is the Jacobian of g. The approximation techniques can be repeated.

Definition 4.4.5 (Forward Euler exponential integrator)**.** *Let* $0 = t_0 < t_1 < \cdots < t_N$. *The* forward Euler exponential integrator *associated with* (4.45) *for* (4.42) *generate the approximations* $y_k \approx y(t_k)$, $k =, \ldots, N$ *defined as*

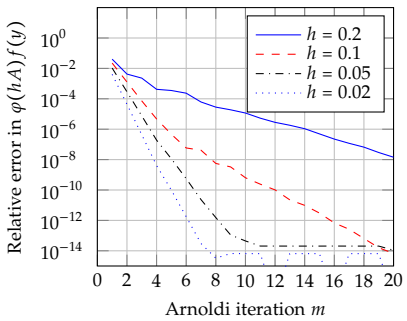
$$y_{k+1} = y_k + h_k\varphi(h_kA_k)g(y_k) \tag{4.49}$$

where $h_k = t_{k+1} - t_k$ *and* $A_k := g'(y_k)$.

The method in Definition 4.4.5 is exact for the linear inhomogeneous case (4.44), and one step can be proven to be second order in h in the general case.

The computationally dominating part in the evaluation of (4.49) is the computation of $\varphi(hA)g(y_0)$. This quantity is exactly what we computed in with the Krylov subspace techniques in the previous section.

Illustration of how well $\varphi(hA)f$ is approximated with the Krylov method in one step of the exponential Euler method. Faster convergence is observed for smaller h.



Example of exponential integrator

Let $f(y) = Ay + c(b^Ty)^2$ and $y(0), b, c \in \mathbb{R}^n$. Then, the Jacobian is given by

$$f'(y) = A + 2(b^Ty)cb^T.$$

A simple implementation of the forward Euler exponential integrator for a fixed number of Arnoldi steps (m) and equidistant time-step ($h = 1/N$) can be done as follows.

```
>> m=20;  % Number of Arnoldi steps
>> randn('seed',0); rand('seed',0); % reproducability of example
>> A=-gallery('wathen',100,100); n=length(A);
```

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

```
>> b=zeros(n,1); b(round(n/2))=1; b=sparse(b); c=b;
>> N=10; tv=linspace(0,1,N+1); h=tv(2)-tv(1);
>> J=@(y) A+2*(b'*y)*(c*b');;
>> g=@(y) A*y+c*(b'*y)^2;
>> y0=eye(n,1); y=y0;
>> for k=1:N
>>     v=g(y);
>>     [Q,H]=arnoldi(J(y),v,m);
>>     phig=Q(:,1:m)*(varphi(h*H(1:m,1:m))*eye(m,1));
>>     y=y+h*phig*norm(v);
>> end
>> sol=ode45(@(t,v) g(v), [0,1],y0,{'RelTol',1e-10});
>> rel_err=norm(sol.y(:,end)-y)/norm(y)
rel_err =

   2.9768e-06
```

A more thorough convergence analysis (beyond the scope of this course) shows that the error behaves as

$$\|\varphi(tA)b - f_m\| = \mathcal{O}(t^m) \tag{4.50}$$

As we have seen above, designing an exponential integrator requires the choice of many quantities. In particular, the choice of step-length is inherently very difficult. We have the trade-offs in order obtain fast convergence and accurate results. We want

- small h, because it leads to fast convergence in the Krylov method as is illustrated in (4.50);
- small h, because it leads to smaller discretization error as the discretization error (per step) is quadratic in h; but
- large h, because then we can complete the integration in $N \sim 1/h$ steps.

The typical approach to balance these quantities often involves problem-specific a posteriori error analysis of the Krylov method as well as the integrator.

4.5 *Further reading and literature*

The most complete source of information for matrix functions is the works of Nick Higham, in particular his monograph [5] but also summary papers such as [6]. The work of Horn and Johnson [9] contains a thorough and more theoretical approach to matrix functions. Matrix functions also appear in concise formats in standard literature in matrix computations [4, 1]. The presentation of the matrix square root

in Secion 4.3.2 was inspired by sections from [1] and [5], whereas the scaling-and-squaring in Section 4.3.1 was based on [4]. Further information about Section 4.4.3 can be found in [8, 7] and references therein. The topic of matrix functions is a very active topic. There is active research on rational Krylov methods for matrix functions, conditioning numbers and development of new methods for other matrix functions. Considerable research is carried out on convergence of iterative methods for matrix functions and also specialized preconditioning techniques.

4.6 *Appendix - Omitted proofs*

Proof of Theorem 4.1.5. First note that all three definitions satisfy (4.6) and (4.7). If we select X and B as the Jordan canonical form of A, we have

$$\begin{aligned} f_T(A) &= f_T(XBX^{-1}) = X\,\mathrm{diag}(f_T(J_1),\ldots,f_T(J_i))X^{-1} \\ f_J(A) &= f_J(XBX^{-1}) = X\,\mathrm{diag}(f_J(J_1),\ldots,f_J(J_i))X^{-1} \\ f_C(A) &= f_C(XBX^{-1}) = X\,\mathrm{diag}(f_C(J_1),\ldots,f_C(J_i))X^{-1}. \end{aligned}$$

Hence, the definitions are equivalent if and only if $f_T(J) = f_J(J) = f_C(J)$ for any Jordan block $J = J_i$. Therefore, it is sufficient to prove equivalence for an arbitrary Jordan block $J \in \mathbb{C}^{m\times m}$.

Definition 4.1.1 $\Leftrightarrow$ Definition 4.1.3. We first show the result for monomials $p_k(z) := z^k$. By induction we see that $p_k^{(\ell)}(z) = \ell p_{k-1}^{(\ell-1)}(z) + zp_{k-1}^{(\ell)}(z)$ for any ℓ and k. By again using induction we have that

$$(J-\mu I)^k = p_k(J-\mu I) = \begin{bmatrix} p_k(\lambda-\mu) & \frac{p_k'(\lambda-\mu)}{1!} & \cdots & \frac{p_k^{(m-1)}(\lambda-\mu)}{(m-1)!} \\ & \ddots & \ddots & \vdots \\ & & \ddots & \frac{p_k'(\lambda-\mu)}{1!} \\ & & & p_k(\lambda-\mu) \end{bmatrix}. \tag{4.51}$$

The function is analytic, so we can directly differentiate the Taylor expansion j times

$$f^{(j)}(\lambda) = \sum_{\ell=0}^{\infty} \frac{f^{(\ell)}(\mu)}{\ell!} p_\ell^{(j)}(\lambda-\mu). \tag{4.52}$$

We now combine (4.51) and (4.52) in the Taylor definition (4.4) to

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018

establish the conclusion

$$f_T(J) = \sum_{\ell=0}^{\infty} \frac{f^{(\ell)}(\mu)}{\ell!}(J-\mu I)^{\ell} =$$
$$\sum_{\ell=0}^{\infty} \frac{f^{(\ell)}(\mu)}{\ell!} \begin{bmatrix} p_\ell(\lambda-\mu) & \frac{p_\ell'(\lambda-\mu)}{1!} & \cdots & \frac{p_\ell^{(m-1)}(\lambda-\mu)}{(m-1)!} \\ & \ddots & \ddots & \vdots \\ & & \ddots & \frac{p_\ell'(\lambda-\mu)}{1!} \\ & & & p_\ell(\lambda-\mu) \end{bmatrix} =$$
$$\begin{bmatrix} f(\lambda) & \frac{f'(\lambda)}{1!} & \cdots & \frac{f^{(m-1)}(\lambda)}{(m-1)!} \\ & \ddots & \ddots & \vdots \\ & & \ddots & \frac{f'(\lambda)}{1!} \\ & & & f(\lambda) \end{bmatrix}.$$

The implication holds in both direction since the definitions are unique.

Definition 4.1.3 ⇔ Definition 4.1.4. It is easy to verify that that the expression involving an inverse in the Cauchy integral formula is explicitly

$$(zI-J)^{-1} = \begin{bmatrix} 1/(z-\lambda) & 1/(z-\lambda)^2 & \cdots & 1/(z-\lambda)^m \\ & \ddots & \ddots & \vdots \\ & & \ddots & 1/(z-\lambda)^2 \\ & & & 1/(z-\lambda) \end{bmatrix} \tag{4.53}$$

Now let

$$F := \frac{1}{2i\pi} \oint_\Gamma f(z)(zI-J)^{-1}\,dz.$$

From the (scalar) Cauchy integral formula and (4.53) we have

$$F_{i,j} = \frac{1}{2i\pi} \oint_\Gamma \frac{f(z)}{(z-\lambda)^j}\,dz = \frac{f^{(j-i)}(\lambda)}{(j-i)!}, \quad i = 1,\ldots,m, j = i,\ldots,m,$$

which coincides with the JCF-definition (4.11)

□

4.7 *Bibliography*

[1] Å. Björck. *Numerical methods in matrix computations*. Springer, 2015.

[2] R. Byers. Solving the algebraic Riccati equation with the matrix sign function. *Linear Algebra Appl.*, 85:267–279, 1987.

[3] E. Denman and A. N. Beavers. The matrix sign function and computations in systems. *Appl. Math. Comput.*, 2:63–94, 1976.

[4] G. Golub and C. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 2013. 4th edition.

[5] N. J. Higham. *Functions of Matrices. Theory and Computation*. SIAM, 2008.

[6] N. J. Higham and A. H. Al-Mohy. Computing matrix functions. *Acta Numerica*, 19:159–208, 2010.

[7] M. Hochbruck, C. Lubich, and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM J. Sci. Comput.*, 19(5):1552–1574, 1998.

[8] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.

[9] R. Horn and C. Johnson. *Matrix analysis*. Cambridge University Press, New York, 1985.

 version:2018-10-25, Elias Jarlebring - copyright 2015-2018