Leja interpolation for matrix functions

Peter Kandolf

Innovative Time Integrators

May 14, 2012

Aim

Problem

Exponential integrator schemes require the evaluation or approximation of a matrix function acting on vectors. We denote this action by

$$\phi(A)v, \quad A \in \mathbb{R}^{d\times d}, \quad v \in \mathbb{R}^{d},$$

for some analytic function ϕ.

Ansatz

Polynomial interpolation with Leja points.

Aim

Problem

Exponential integrator schemes require the evaluation or approximation of a matrix function acting on vectors. We denote this action by

$$\phi(A)v, \quad A \in \mathbb{R}^{d\times d}, \quad v \in \mathbb{R}^d,$$

for some analytic function ϕ.

Ansatz

Polynomial interpolation with Leja points.

Overview

1. Polynomial interpolation with Newton's scheme
2. Introduction to Leja points
3. Extension to the matrix case
4. Computation
5. Future work

Newton interpolation

Ansatz

Let f be a function, analytic in an open set $K \subset \mathbb{C}$ and $\{\xi_i\}_{i=0}^n$ points in K. The Newton interpolation uses the ansatz

$$p(x) = a_0 + a_1(x - \xi_0) + a_2(x - \xi_0)(x - \xi_1) + \dots \\ \dots + a_n(x - \xi_0) \cdots (x - \xi_n).$$

Divided differences

We define the divided differences $f[\xi_j, \dots, \xi_k]$ of f at the points $\{\xi_i\}_{i=0}^n$ as

$$f[\xi_j, \dots, \xi_k] := \frac{f[\xi_{j+1}, \dots, \xi_k] - f[\xi_j, \dots, \xi_{k-1}]}{\xi_k - \xi_j}, \quad f[\xi_i] = f(\xi_i).$$

Newton interpolation

Ansatz

Let f be a function, analytic in an open set $K \subset \mathbb{C}$ and $\{\xi_i\}_{i=0}^n$ points in K. The Newton interpolation uses the ansatz

$$p(x) = a_0 + a_1(x - \xi_0) + a_2(x - \xi_0)(x - \xi_1) + \dots \\ \dots + a_n(x - \xi_0)\cdots(x - \xi_n).$$

Divided differences

We define the divided differences $f[\xi_j, \dots, \xi_k]$ of f at the points $\{\xi_i\}_{i=0}^n$ as

$$f[\xi_j, \dots, \xi_k] := \frac{f[\xi_{j+1}, \dots, \xi_k] - f[\xi_j, \dots, \xi_{k-1}]}{\xi_k - \xi_j}, \quad f[\xi_i] = f(\xi_i).$$

Newton interpolation 2

Rewritten

With the divided differences the scheme can be written in a more stable and compact from (reduction of round-off errors)

$$p_n(x) = f[\xi_0] + \sum_{j=1}^{n} f[\xi_0, \ldots, \xi_j] \prod_{k=0}^{j-1} (x - \xi_k).$$

Remark

- *It is easy to compute p_{n+1} from p_n.*
- *Optimally conditioned interpolation for Chebyshev points.*

Newton interpolation 2

Rewritten

With the divided differences the scheme can be written in a more stable and compact from (reduction of round-off errors)

$$p_n(x) = f[\xi_0] + \sum_{j=1}^{n} f[\xi_0, \ldots, \xi_j] \prod_{k=0}^{j-1} (x - \xi_k).$$

Remark

- *It is easy to compute p_{n+1} from p_n.*
- *Optimally conditioned interpolation for Chebyshev points.*

Newton interpolation 2

Rewritten

With the divided differences the scheme can be written in a more stable and compact from (reduction of round-off errors)

$$p_n(x) = f[\xi_0] + \sum_{j=1}^{n} f[\xi_0, \ldots, \xi_j] \prod_{k=0}^{j-1} (x - \xi_k).$$

Remark

- *It is easy to compute p_{n+1} from p_n.*
- *Optimally conditioned interpolation for Chebyshev points.*

Overview

Definition of Leja points

Aim

Define a sequence of points with same convergence properties as Chebyshev points, but computational advantages

Definition (Leja points)

Let $K \subset \mathbb{C}$ be a compact set then the sequence of Leja points $\{\xi_i\}_{i=0}^{\infty}$ for K is defined recursively as:

- ξ_0 *can be chosen arbitrary, normally* $|\xi_0| = \max_{z \in K} |z|$
- ξ_m *is then defined as*

$$\xi_m \in \arg\max_{z \in K} \prod_{i=0}^{m-1} |z - \xi_i|, \quad m > 0.$$

Definition of Leja points

Aim

Define a sequence of points with same convergence properties as Chebyshev points, but computational advantages

Definition (Leja points)

Let $K \subset \mathbb{C}$ be a compact set then the sequence of Leja points $\{\xi_i\}_{i=0}^{\infty}$ for K is defined recursively as:

- *ξ_0 can be chosen arbitrary, normally $|\xi_0| = \max_{z \in K} |z|$*
- *ξ_m is then defined as*

$$\xi_m \in \arg\max_{z \in K} \prod_{i=0}^{m-1} |z - \xi_i|, \quad m > 0.$$

Definition of Leja points

Aim

Define a sequence of points with same convergence properties as Chebyshev points, but computational advantages

Definition (Leja points)

Let $K \subset \mathbb{C}$ be a compact set then the sequence of Leja points $\{\xi_i\}_{i=0}^{\infty}$ for K is defined recursively as:

- ξ_0 *can be chosen arbitrary, normally* $|\xi_0| = \max_{z \in K} |z|$
- ξ_m *is then defined as*

$$\xi_m \in \arg\max_{z \in K} \prod_{i=0}^{m-1} |z - \xi_i|, \quad m > 0.$$

Definition of Leja points

Aim

Define a sequence of points with same convergence properties as Chebyshev points, but computational advantages

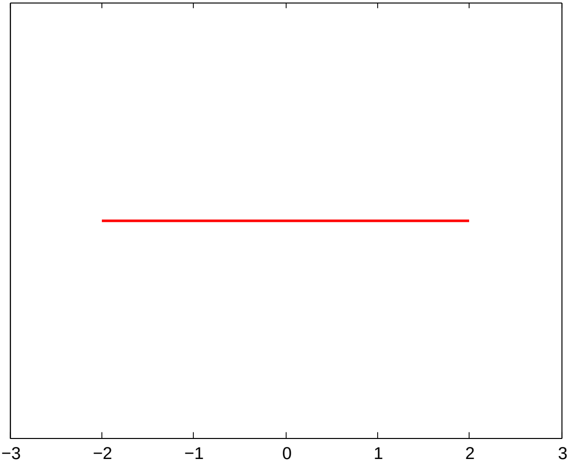
Definition (Leja points)

Let $K \subset \mathbb{C}$ be a compact set then the sequence of Leja points $\{\xi_i\}_{i=0}^{\infty}$ for K is defined recursively as:

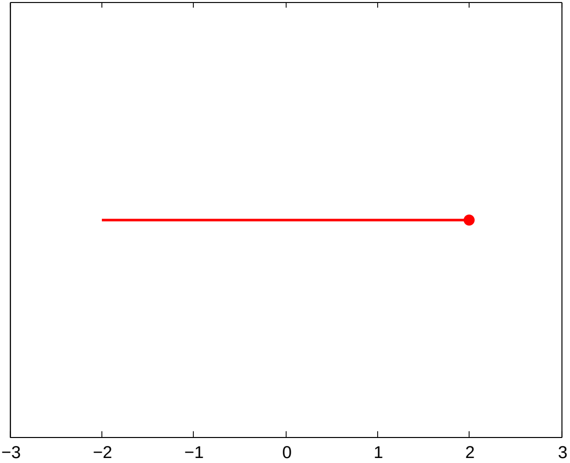
- ξ_0 *can be chosen arbitrary, normally* $|\xi_0| = \max_{z \in K} |z|$
- ξ_m *is then defined as*

$$\xi_m \in \arg\max_{z \in K} \prod_{i=0}^{m-1} |z - \xi_i|, \quad m > 0.$$

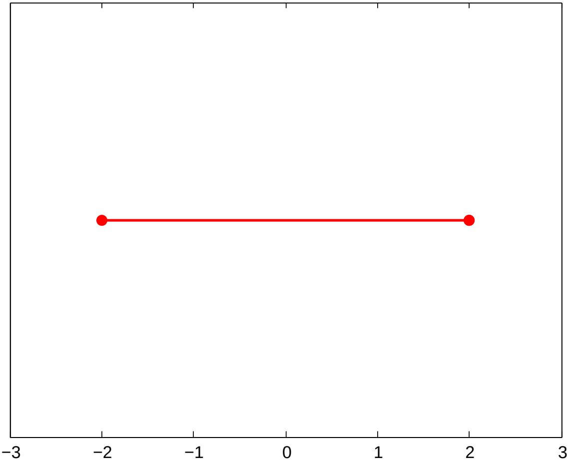
Example for $K = [-2, 2]$



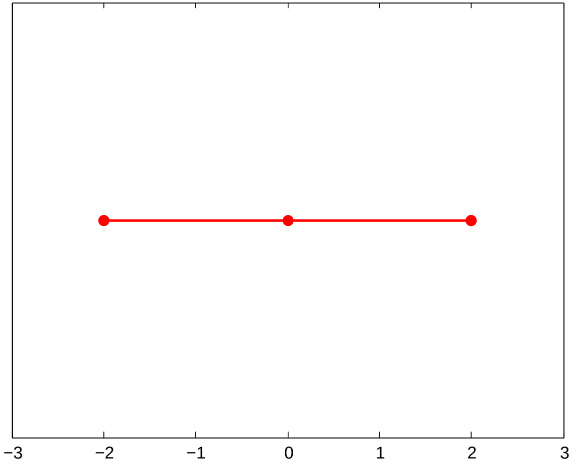
Example for $K = [-2, 2]$



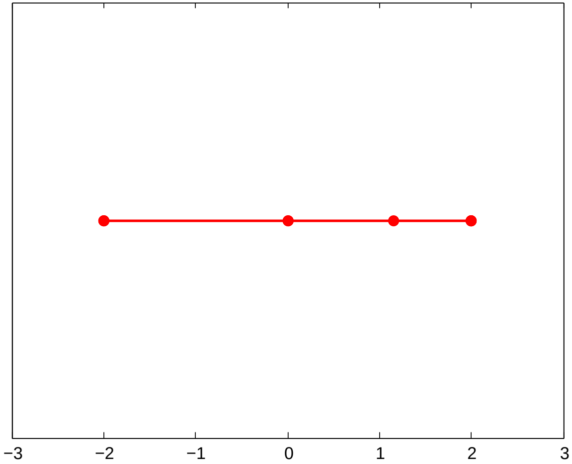
Example for $K = [-2, 2]$



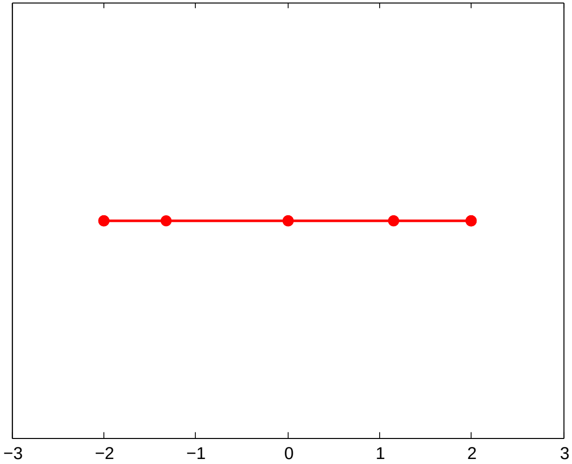
Example for $K = [-2, 2]$



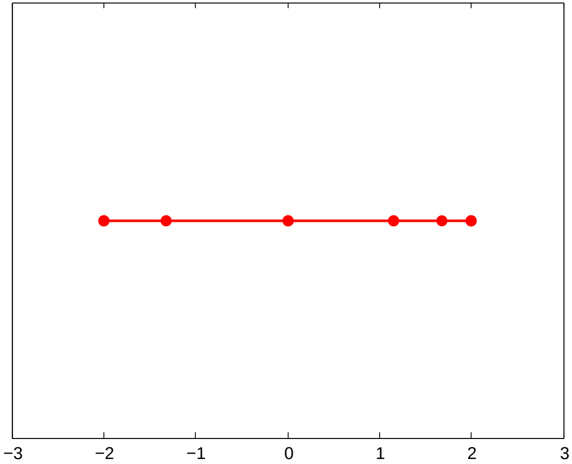
Example for $K = [-2, 2]$



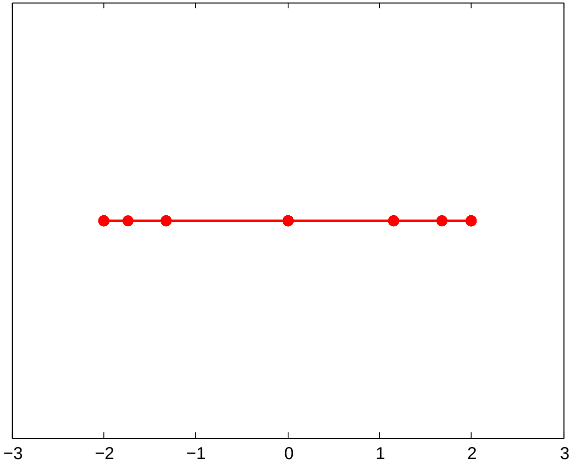
Example for $K = [-2, 2]$



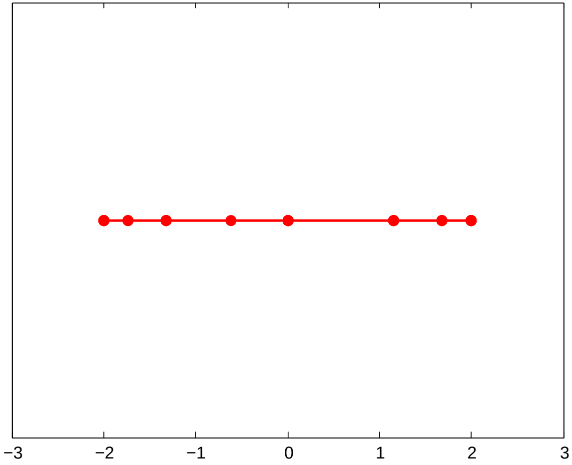
Example for $K = [-2, 2]$



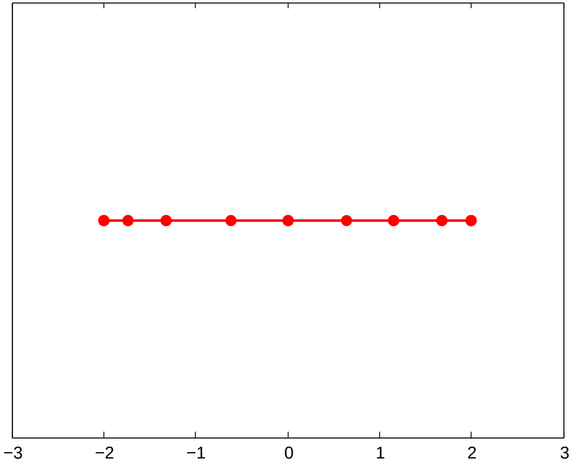
Example for $K = [-2, 2]$



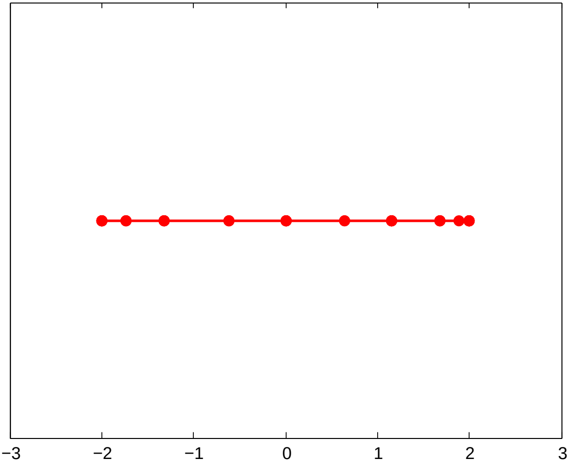
Example for $K = [-2, 2]$



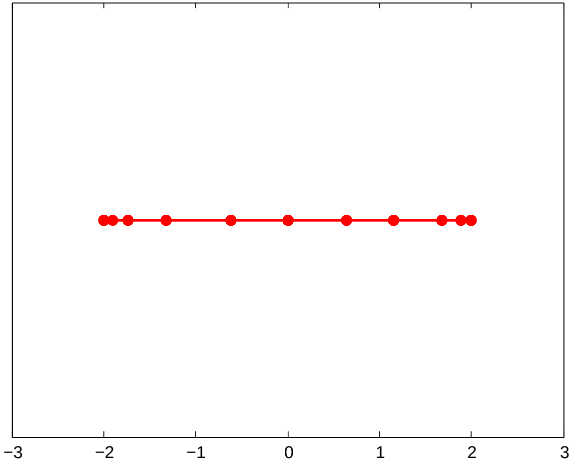
Example for $K = [-2, 2]$



Example for $K = [-2, 2]$



Example for $K = [-2, 2]$



Computation

Computation of Leja points

- For $K \subset \mathbb{C}$ compact, $\{\xi_i\} \in \partial K$.
- Replace K by discrete $S_M \subset \partial K$ with $|S_M| = M$.

Remark

- *For our purposes m Leja points in* $[-2, 2]$.

Computation

Computation of Leja points

- For $K \subset \mathbb{C}$ compact, $\{\xi_i\} \in \partial K$.
- Replace K by discrete $S_M \subset \partial K$ with $|S_M| = M$.

Remark

- *For our purposes m Leja points in* $[-2, 2]$.

Computation

Computation of Leja points

- For $K \subset \mathbb{C}$ compact, $\{\xi_i\} \in \partial K$.
- Replace K by discrete $S_M \subset \partial K$ with $|S_M| = M$.

Remark

- *For our purposes m Leja points in* $[-2, 2]$.

Convergence and stability

Convergence

For entire functions super-linear convergence can be shown.

Stability

Let $T : \mathbb{C}^{m+1} \to \mathbb{P}_m$ map a sequence of points $\{\xi_i\}$ to the interpolation polynomial in Newton form.

- $\mathrm{cond}(T)$ grows exponentially for arbitrary points.
- For Leja points we get $\lim_{m\to\infty} \mathrm{cond}(T)^{1/m} = 1$.

Convergence and stability

Convergence

For entire functions super-linear convergence can be shown.

Stability

Let $T : \mathbb{C}^{m+1} \to \mathbb{P}_m$ map a sequence of points $\{\xi_i\}$ to the interpolation polynomial in Newton form.

- $\mathrm{cond}(T)$ grows exponentially for arbitrary points.
- For Leja points we get $\lim_{m\to\infty} \mathrm{cond}(T)^{1/m} = 1$.

Convergence and stability

Convergence

For entire functions super-linear convergence can be shown.

Stability

Let $T : \mathbb{C}^{m+1} \to \mathbb{P}_m$ map a sequence of points $\{\xi_i\}$ to the interpolation polynomial in Newton form.

- $\mathrm{cond}(T)$ grows exponentially for arbitrary points.
- For Leja points we get $\lim_{m\to\infty} \mathrm{cond}(T)^{1/m} = 1$.

Convergence and stability

Convergence

For entire functions super-linear convergence can be shown.

Stability

Let $T : \mathbb{C}^{m+1} \to \mathbb{P}_m$ map a sequence of points $\{\xi_i\}$ to the interpolation polynomial in Newton form.

- $\mathrm{cond}(T)$ grows exponentially for arbitrary points.
- For Leja points we get $\lim_{m\to\infty} \mathrm{cond}(T)^{1/m} = 1$.

Example for $\mathrm{cond}(T)$ on $[-2, 2]$

Nodes	equidistant	rand. chosen	Chebyshev	Leja
10	1.79e+01	8.27e+06	2.43e+00	4.49e+00
50	1.86e+12	3.88e+40	3.45e+00	1.40e+01
100	8.94e+26	8.18e+74	3.89e+00	2.01e+01
150	6.19e+41	8.54e+115	4.15e+00	3.99e+01

Example for $\mathrm{cond}(T)$ on $[-2, 2]$

Nodes	equidistant	rand. chosen	Chebyshev	Leja
10	1.79e+01	8.27e+06	2.43e+00	4.49e+00
50	1.86e+12	3.88e+40	3.45e+00	1.40e+01
100	8.94e+26	8.18e+74	3.89e+00	2.01e+01
150	6.19e+41	8.54e+115	4.15e+00	3.99e+01

Example for $\mathrm{cond}(T)$ on $[-2, 2]$

Nodes	equidistant	rand. chosen	Chebyshev	Leja
10	1.79e+01	8.27e+06	2.43e+00	4.49e+00
50	1.86e+12	3.88e+40	3.45e+00	1.40e+01
100	8.94e+26	8.18e+74	3.89e+00	2.01e+01
150	6.19e+41	8.54e+115	4.15e+00	3.99e+01

Example for $\mathrm{cond}(T)$ on $[-2, 2]$

Nodes	equidistant	rand. chosen	Chebyshev	Leja
10	1.79e+01	8.27e+06	2.43e+00	4.49e+00
50	1.86e+12	3.88e+40	3.45e+00	1.40e+01
100	8.94e+26	8.18e+74	3.89e+00	2.01e+01
150	6.19e+41	8.54e+115	4.15e+00	3.99e+01

Example for $\mathrm{cond}(T)$ on $[-2, 2]$

Nodes	equidistant	rand. chosen	Chebyshev	Leja
10	1.79e+01	8.27e+06	2.43e+00	4.49e+00
50	1.86e+12	3.88e+40	3.45e+00	1.40e+01
100	8.94e+26	8.18e+74	3.89e+00	2.01e+01
150	6.19e+41	8.54e+115	4.15e+00	3.99e+01

Overview

1. Polynomial interpolation with Newton's scheme
2. Introduction to Leja points
3. **Extension to the matrix case**
4. Computation
5. Future work

Convergence for matrix functions

Preliminaries

- Let f be analytic in $K = B(0, R_{max}) \subset \mathbb{C}$
- Let $\{p_m\} \in \mathbb{P}_m$ be the polynomial interpolation to f in the Leja points of K

The sequence $\{p_m\}$ converges asymptotically like the best uniform approximation polynomial to f in K.

Convergence for matrix functions

Preliminaries

- Let f be analytic in $K = B(0, R_{max}) \subset \mathbb{C}$
- Let $\{p_m\} \in \mathbb{P}_m$ be the polynomial interpolation to f in the Leja points of K

The sequence $\{p_m\}$ converges asymptotically like the best uniform approximation polynomial to f in K.

Convergence for matrix functions

Preliminaries

- Let f be analytic in $K = B(0, R_{max}) \subset \mathbb{C}$
- Let $\{p_m\} \in \mathbb{P}_m$ be the polynomial interpolation to f in the Leja points of K

The sequence $\{p_m\}$ converges asymptotically like the best uniform approximation polynomial to f in K.

Convergence for matrix functions

Convergence

Let $A \in \mathbb{R}^{n\times n}$, $v \in \mathbb{R}^n$ and $\sigma(A) \subset B(0, R)$ for some $0 < R < R_{max}$. Then $\{p_m(A)v\}$ converges to $f(A)v$.

Corollary

For an entire functions f this means that

$$\limsup_{m\to\infty} \|f(A)v - p_m(A)v\|_2^{1/m} = 0$$

and therefore **super-linear** convergence.

Convergence for matrix functions

Convergence

Let $A \in \mathbb{R}^{n \times n}$, $v \in \mathbb{R}^n$ and $\sigma(A) \subset B(0, R)$ for some $0 < R < R_{max}$. Then $\{p_m(A)v\}$ converges to $f(A)v$.

Corollary

For an entire functions f this means that

$$\limsup_{m \to \infty} \|f(A)v - p_m(A)v\|_2^{1/m} = 0$$

and therefore **super-linear** convergence.

Overview

Computation of φ-functions

Problem

We want to compute $\varphi_k(hA)v$ for the entire functions

$$\varphi_k(\tau z) = \tau^{-k} \int_0^{\tau} e^{(\tau - s)z} \frac{s^{k-1}}{(k-1)!}\, ds, \quad k \geq 1.$$

up to a specified tolerance.

Algorithm

Leja interpolation of φ-functions

Input: A, v, h, tol, k

Compute rough estimate of spectrum $h\sigma(A)$:

$$(-a, -\mathrm{i}b), (0, -\mathrm{i}b), (0, \mathrm{i}b), (a, \mathrm{i}b) \quad a, b \geq 0$$

If: $a \geq b$ Newton interpolation on real Leja points in $[-a, 0]$

$a < b$ Newton interpolation on conjugate pairs of Leja points in $D = \{z \in D : \Re z = -a/2, \Im z \in [-b, b]\}$ (real arithmetic).

Output: approximation of $\varphi_k(hA)v$ with accuracy tol

Algorithm

Leja interpolation of φ-functions

Input: A, v, h, tol, k

Compute rough estimate of spectrum $h\sigma(A)$:

$$(-a, -\mathrm{i}b), (0, -\mathrm{i}b), (0, \mathrm{i}b), (a, \mathrm{i}b) \quad a, b \geq 0$$

If: $a \geq b$ Newton interpolation on real Leja points in $[-a, 0]$

$a < b$ Newton interpolation on conjugate pairs of Leja points in $D = \{z \in D : \Re z = -a/2, \Im z \in [-b, b]\}$ (real arithmetic).

Output: approximation of $\varphi_k(hA)v$ with accuracy tol

Algorithm

Leja interpolation of φ-functions

Input: A, v, h, tol, k

Compute rough estimate of spectrum $h\sigma(A)$:

$$(-a, -\mathrm{i}b), (0, -\mathrm{i}b), (0, \mathrm{i}b), (a, \mathrm{i}b) \quad a, b \geq 0$$

If: $a \geq b$ Newton interpolation on real Leja points in $[-a, 0]$

$a < b$ Newton interpolation on conjugate pairs of Leja points in $D = \{z \in D : \Re z = -a/2, \Im z \in [-b, b]\}$ (real arithmetic).

Output: approximation of $\varphi_k(hA)v$ with accuracy tol

Algorithm

Leja interpolation of φ-functions

Input: A, v, h, tol, k

Compute rough estimate of spectrum $h\sigma(A)$:

$$(-a, -\mathrm{i}b), (0, -\mathrm{i}b), (0, \mathrm{i}b), (a, \mathrm{i}b) \quad a, b \geq 0$$

If: $a \geq b$ Newton interpolation on real Leja points in $[-a, 0]$

$a < b$ Newton interpolation on conjugate pairs of Leja points in $D = \{z \in D : \Re z = -a/2, \Im z \in [-b, b]\}$ (real arithmetic).

Output: approximation of $\varphi_k(hA)v$ with accuracy tol

Algorithm

Leja interpolation of φ-functions

Input: A, v, h, tol, k

Compute rough estimate of spectrum $h\sigma(A)$:

$$(-a, -\mathrm{i}b), (0, -\mathrm{i}b), (0, \mathrm{i}b), (a, \mathrm{i}b) \quad a, b \geq 0$$

If: $a \geq b$ Newton interpolation on real Leja points in $[-a, 0]$

$a < b$ Newton interpolation on conjugate pairs of Leja points in $D = \{z \in D : \Re z = -a/2, \Im z \in [-b, b]\}$ (real arithmetic).

Output: approximation of $\varphi_k(hA)v$ with accuracy tol

Algorithm

Newton interpolation

Computation of divided differences d_i at shifted $\varphi_k(h(c-\gamma\xi_i))$.

Initialise: $q_0 = v$, $p_0(hA)v = d_0 q_0$

while: $\|e_m\| \leq tol$

$$q_m = (hA - \xi_{m-1})q_{m-1},$$
$$p_m(hA)v = p_{m-1}(hA)v + d_m q_m,$$
$$\|e_m\| = \|p_m(hA)v - p_{m-1}(hA)v\| = |d_m| \cdot \|q_m\|$$

return: $p_m(hA)v$, $\|e_m\|$

Algorithm

Newton interpolation

Computation of divided differences d_i at shifted $\varphi_k(h(c - \gamma\xi_i))$.

Initialise: $q_0 = v$, $p_0(hA)v = d_0 q_0$

while: $\|e_m\| \leq tol$

$$
\begin{aligned}
q_m &= (hA - \xi_{m-1})q_{m-1}, \\
p_m(hA)v &= p_{m-1}(hA)v + d_m q_m, \\
\|e_m\| &= \|p_m(hA)v - p_{m-1}(hA)v\| = |d_m| \cdot \|q_m\|
\end{aligned}
$$

return: $p_m(hA)v$, $\|e_m\|$

Algorithm

Newton interpolation

Computation of divided differences d_i at shifted $\varphi_k(h(c - \gamma\xi_i))$.
Initialise: $q_0 = v$, $p_0(hA)v = d_0 q_0$
while: $\|e_m\| \leq tol$

$$q_m = (hA - \xi_{m-1})q_{m-1},$$
$$p_m(hA)v = p_{m-1}(hA)v + d_m q_m,$$
$$\|e_m\| = \|p_m(hA)v - p_{m-1}(hA)v\| = |d_m| \cdot \|q_m\|$$

return: $p_m(hA)v$, $\|e_m\|$

Algorithm

Newton interpolation

Computation of divided differences d_i at shifted $\varphi_k(h(c-\gamma\xi_i))$.
Initialise: $q_0 = v$, $p_0(hA)v = d_0 q_0$
while: $\|e_m\| \leq tol$

$$q_m = (hA - \xi_{m-1})q_{m-1},$$
$$p_m(hA)v = p_{m-1}(hA)v + d_m q_m,$$
$$\|e_m\| = \|p_m(hA)v - p_{m-1}(hA)v\| = |d_m| \cdot \|q_m\|$$

return: $p_m(hA)v$, $\|e_m\|$

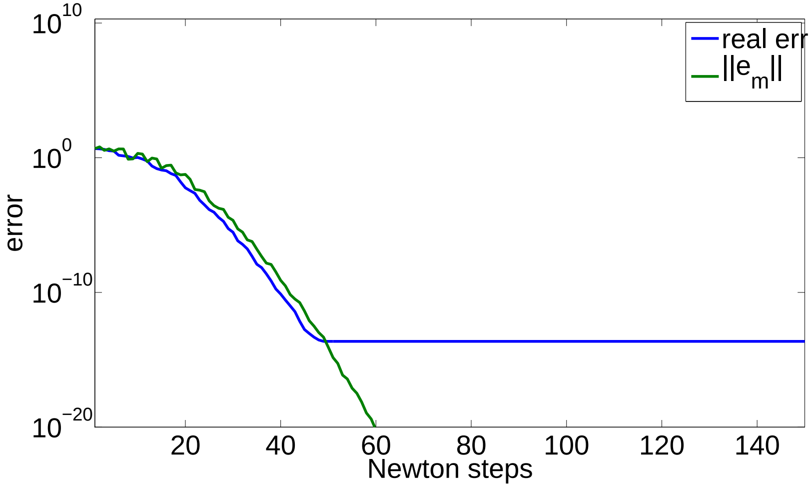
Example

Example (2D advection-diffusion)

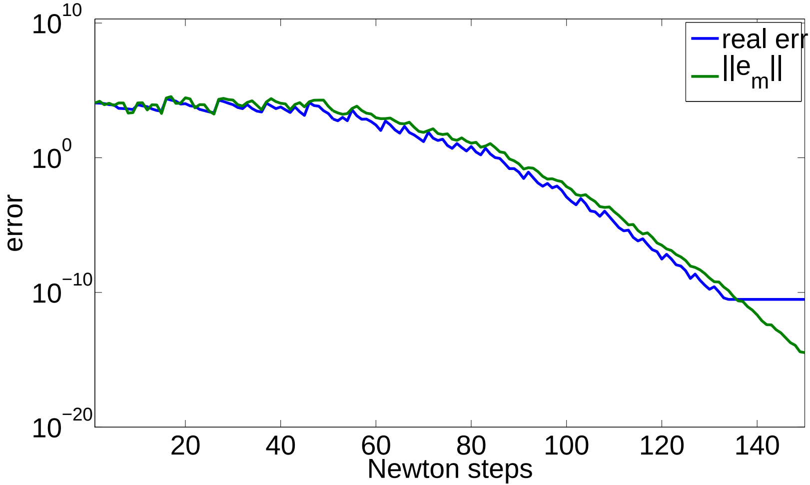
Computation of $\mathrm{e}^{hA}v$ *with polynomial interpolation in Leja points for:*

- *dimension of A:* 10.000×10.000,
- *Peclet number* 0.303,
- $v = [1, \ldots, 1]^{\mathsf{T}}$,
- $h = 5e\text{-}4$,
- *tolerance* $1e\text{-}12$.

Example - 2D advection-diffusion, h =5e-4



Example - 2D advection-diffusion, $h = $30e-4



Substepping

Substeps

To overcome the humb problem we compute L substeps and recover $\varphi_k(hA)v$ from $\varphi_k(\tau hA)v$ with $\tau = 1/L$.

Example

$y = v$
for $j = 1 : L$
    $y = \mathrm{e}^{\frac{h}{L}A} y$
end

Substepping

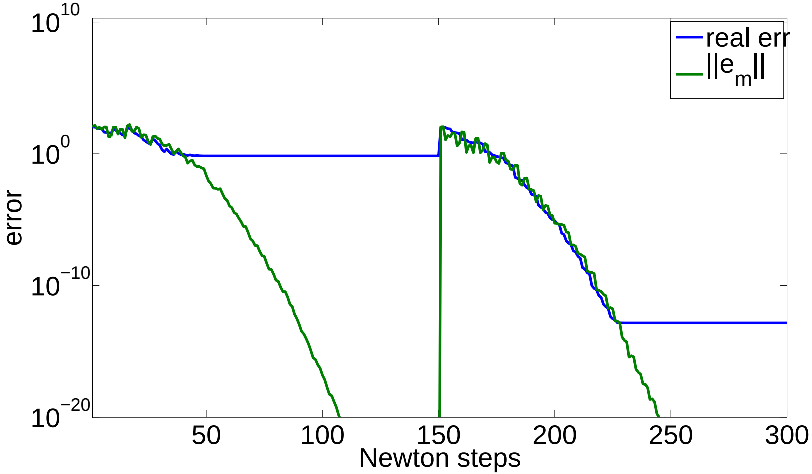
Substeps

To overcome the humb problem we compute L substeps and recover $\varphi_k(hA)v$ from $\varphi_k(\tau hA)v$ with $\tau = 1/L$.

Example

$y = v$
for $j = 1 : L$
$\quad y = \mathrm{e}^{\frac{h}{L}A} y$
end

Example - 2D advection-diffusion, h =30e-4



Overview

Future work

New error estimate

Develop a new error estimate that:

- shows correct asymptotic behaviour,
- obtains the accurate number of substeps,
- obtains the minimal degree of interpolation.

Future work

New error estimate

Develop a new error estimate that:

- shows correct asymptotic behaviour,
- obtains the accurate number of substeps,
- obtains the minimal degree of interpolation.

Future work

New error estimate

Develop a new error estimate that:

- shows correct asymptotic behaviour,
- obtains the accurate number of substeps,
- obtains the minimal degree of interpolation.

Future work

New error estimate

Develop a new error estimate that:

- shows correct asymptotic behaviour,
- obtains the accurate number of substeps,
- obtains the minimal degree of interpolation.

Future work

Parallelisation

Parallelisation on GPU:

- primary matrix-vector multiplications
- multiply-accumulate functionality
- parallel data access
- data transfer
- data types

Future work

Parallelisation

Parallelisation on GPU:

- primary matrix-vector multiplications
- multiply-accumulate functionality
- parallel data access
- data transfer
- data types

Future work

Parallelisation

Parallelisation on GPU:

- primary matrix-vector multiplications
- multiply-accumulate functionality
- parallel data access
- data transfer
- data types

Future work

Parallelisation

Parallelisation on GPU:

- primary matrix-vector multiplications
- multiply-accumulate functionality
- parallel data access
- data transfer
- data types

Future work

Parallelisation

Parallelisation on GPU:

- primary matrix-vector multiplications
- multiply-accumulate functionality
- parallel data access
- data transfer
- data types

Future work

Parallelisation

Parallelisation on GPU:

- primary matrix-vector multiplications
- multiply-accumulate functionality
- parallel data access
- data transfer
- data types

Computation of divided differences

Standard computation

$$f[\xi_j, \ldots, \xi_k] := \frac{f[\xi_{j+1}, \ldots, \xi_k] - f[\xi_j, \ldots, \xi_{k-1}]}{\xi_k - \xi_j}, \quad f[\xi_i] = f(\xi_i).$$

- easy to implement with low storage demands
- vulnerable to round-off errors
- stable computation via matrix function

Computation of divided differences

Standard computation

$$f[\xi_j, \ldots, \xi_k] := \frac{f[\xi_{j+1}, \ldots, \xi_k] - f[\xi_j, \ldots, \xi_{k-1}]}{\xi_k - \xi_j}, \quad f[\xi_i] = f(\xi_i).$$

- easy to implement with low storage demands
- vulnerable to round-off errors
- stable computation via matrix function

Computation of divided differences

Standard computation

$$f[\xi_j, \ldots, \xi_k] := \frac{f[\xi_{j+1}, \ldots, \xi_k] - f[\xi_j, \ldots, \xi_{k-1}]}{\xi_k - \xi_j}, \quad f[\xi_i] = f(\xi_i).$$

- easy to implement with low storage demands
- vulnerable to round-off errors
- stable computation via matrix function

Computation of divided differences

Standard computation

$$f[\xi_j, \dots, \xi_k] := \frac{f[\xi_{j+1}, \dots, \xi_k] - f[\xi_j, \dots, \xi_{k-1}]}{\xi_k - \xi_j}, \quad f[\xi_i] = f(\xi_i).$$

- easy to implement with low storage demands
- vulnerable to round-off errors
- stable computation via matrix function

Stable dd via matrix function [M. Caliari 2007]

Matrix computation

Divided differences $\{d_i\}$ of $f(h(c+\gamma\xi_i))$ at $\xi_i \in [-2,2]$ are the first column of the matrix function $f(H_m)$ for

$$H_m = h(cI_{m+1} + \gamma T_m), \quad T_m = \begin{bmatrix} \xi_0 & & & & \\ 1 & \xi_2 & & & \\ & 1 & \ddots & & \\ & & \ddots & \ddots & \\ & & & 1 & \xi_m \end{bmatrix}.$$

Stable dd via matrix function [M. Caliari 2007]

Computation of $\varphi(H_m)$

- scale H_m by $\tau = 1/L$ s.t. $\max_i |\tau x_i| < 1.59$ for $x_i = h(c + \gamma\xi_i)$,
- compute $\varphi(\tau H_m)$ by Taylor expansion,
- recover, in L steps, $\varphi(H_m)e_1$ via recurrence relation.

Stable dd via matrix function [M. Caliari 2007]

Computation of $\varphi(H_m)$

- scale H_m by $\tau = 1/L$ s.t. $\max_i |\tau x_i| < 1.59$ for $x_i = h(c + \gamma\xi_i)$,
- compute $\varphi(\tau H_m)$ by Taylor expansion,
- recover, in L steps, $\varphi(H_m)e_1$ via recurrence relation.

Stable dd via matrix function [M. Caliari 2007]

Computation of $\varphi(H_m)$

- scale H_m by $\tau = 1/L$ s.t. $\max_i |\tau x_i| < 1.59$ for $x_i = h(c + \gamma\xi_i)$,
- compute $\varphi(\tau H_m)$ by Taylor expansion,
- recover, in L steps, $\varphi(H_m)e_1$ via recurrence relation.

Thank you for your attention. Enjoy the wine!

- L. Reichel, Newton interpolation at Leja points, *BIT. Numerical Mathematics*, 1990
- M. Caliari, M. Vianello, L. Bergamaschi, Interpolating discrete advection-diffusion propagators at Leja sequences, *J. Comput. Appl. Math.*, 2004
- M. Caliari, Accurate evaluation of divided differences for polynomial interpolation of exponential propagators, *Computing*, 2007