MANCHESTER 1824
The University of Manchester

M\crNA
Manchester Numerical Analysis

Efficient and Accurate Evaluation of Polynomials of Matrices

Massimiliano Fasi

Numerical Linear Algebra Group Meeting

Manchester, 11 December 2018

Outline

Joint work with Steven Elsworth & Gian Maria Negri Porzio.

Polynomials of matrices

We want to evaluate (accurately and efficiently)

$$p(A) = \sum_{i=0}^{k} c_i A^i = c_0 I + c_1 A + c_2 A^2 + \cdots + c_k A^k,$$

where

- $k \in \mathbb{N}$,
- $c_i \in \mathbb{C}$ and mostly nonzero,
- $A \in \mathbb{C}^{N \times N}$.

Motivation

- Evaluation of series expansions:
  - computation of matrix functions (Taylor series),
  - solution of matrix equations.
- Evaluation of matrix rational functions $q(A)^{-1}p(A)$.

Beginner's algorithm

Algorithm: Explicit powers.

Input: $A \in \mathbb{C}^{N \times N}$, $c_0, \ldots, c_k \in \mathbb{C}$

Output: $S = p(A)$

$P \leftarrow A$

$S \leftarrow c_0 I + c_1 A$

for $i \leftarrow 2$ **to** k **do**

  $P \leftarrow PA$

  $S \leftarrow S + c_i P$

return S

Beginner's algorithm

Algorithm: Explicit powers.

Input: $A \in \mathbb{C}^{N\times N}$, $c_0, \dots, c_k \in \mathbb{C}$

Output: $S = p(A)$

$P \leftarrow A$

$S \leftarrow c_0 I + c_1 A$

for $i \leftarrow 2$ **to** k **do**

  $P \leftarrow PA$

  $S \leftarrow S + c_i P$

return S

- 1 diagonal sum
- k scalings, $k-1$ sums
- $k-1$ matrix products
- $2n^2$ additional storage

$$\approx 2(k-1)n^3$$

Avoiding scalings

Algorithm: Horner's method.

Input: $A \in \mathbb{C}^{N \times N}$, $c_0, \ldots, c_k \in \mathbb{C}$

Output: $S = p(A)$

$S \leftarrow c_k A + c_{k-1} I$

for $i \leftarrow k-2$ **down to** 0 **do**

  $S \leftarrow SA + c_i I$

return S

Avoiding scalings

Algorithm: Horner's method.

Input: $A \in \mathbb{C}^{N \times N}$, $c_0, \ldots, c_k \in \mathbb{C}$

Output: $S = p(A)$

$S \leftarrow c_k A + c_{k-1} I$

for $i \leftarrow k-2$ **down to** 0 **do**

  $S \leftarrow SA + c_i I$

return S

- k diagonal sums
- 1 scaling, 0 sums
- $k-1$ matrix products
- n^2 additional storage

$$\approx 2(k-1)n^3$$

Are faster algorithms possible?

Theorem [Paterson & Stockmeyer, 1973]

For any $k \in \mathbb{N}$ there are polynomials whose evaluation requires at least $\sqrt{k}$ nonscalar multiplications.

Therefore:

- $\sqrt{k}$ matrix multiplications are required in general
- Can we do better than $k - 1$?

Complicating an easy task

For a positive integer s, we can rewrite

$$p(A) = \sum_{i=0}^{r} (A^s)^i B_i, \qquad r = \lfloor k/s \rfloor,$$

where

$$B_i = \begin{cases} \sum_{j=0}^{s-1} c_{si+j} A^j, & i = 0, \ldots, r-1, \\ \sum_{j=0}^{k \bmod s} c_{si+j} A^j, & i = r. \end{cases}$$

Complicating an easy task

For a positive integer s, we can rewrite

$$p(A) = \sum_{i=0}^{r} (A^s)^i B_i, \qquad r = \lfloor k/s \rfloor,$$

where

$$B_i = \begin{cases} \sum_{j=0}^{s-1} c_{si+j} A^j, & i = 0, \ldots, r-1, \\ \sum_{j=0}^{k \bmod s} c_{si+j} A^j, & i = r. \end{cases}$$

Remark: $p(A)$ is a polynomial in A^s with coefficients B_i.

The Paterson–Stockmeyer method

Input: $A \in \mathbb{C}^{N \times N}$, $c_0, \ldots, c_k \in \mathbb{C}$

Output: $S = p(A)$

The Paterson–Stockmeyer method

Input: $A \in \mathbb{C}^{N \times N}$, $c_0, \ldots, c_k \in \mathbb{C}$

Output: $S = p(A)$

$\mathcal{A}_0 \leftarrow I$, $\mathcal{A}_1 \leftarrow A$

for $i \leftarrow 2$ **to** s **do**

  $\mathcal{A}_i \leftarrow A\mathcal{A}_{i-1}$

The Paterson–Stockmeyer method

Input: $A \in \mathbb{C}^{N \times N}$, $c_0, \ldots, c_k \in \mathbb{C}$

Output: $S = p(A)$

$\mathcal{A}_0 \leftarrow I$, $\mathcal{A}_1 \leftarrow A$

for $i \leftarrow 2$ **to** s **do**

  $\mathcal{A}_i \leftarrow A\mathcal{A}_{i-1}$

if s *divides* k **then** $S \leftarrow c_k$

else $S \leftarrow \sum_{j=0}^{k \bmod s} c_{sr+j}\mathcal{A}_j$

The Paterson–Stockmeyer method

Input: $A \in \mathbb{C}^{N \times N}$, $c_0, \ldots, c_k \in \mathbb{C}$

Output: $S = p(A)$

$\mathcal{A}_0 \leftarrow I$, $\mathcal{A}_1 \leftarrow A$

for $i \leftarrow 2$ **to** s **do**

  $\mathcal{A}_i \leftarrow A\mathcal{A}_{i-1}$

if s *divides* k **then** $S \leftarrow c_k$

else $S \leftarrow \sum_{j=0}^{k \bmod s} c_{sr+j}\mathcal{A}_j$

for $i \leftarrow r-1$ **down to** 0 **do**

  $S \leftarrow S\mathcal{A}_s + \sum_{j=0}^{s-1} c_{si+j}\mathcal{A}_j$

return S

Cost of the Paterson–Stockmeyer algorithm

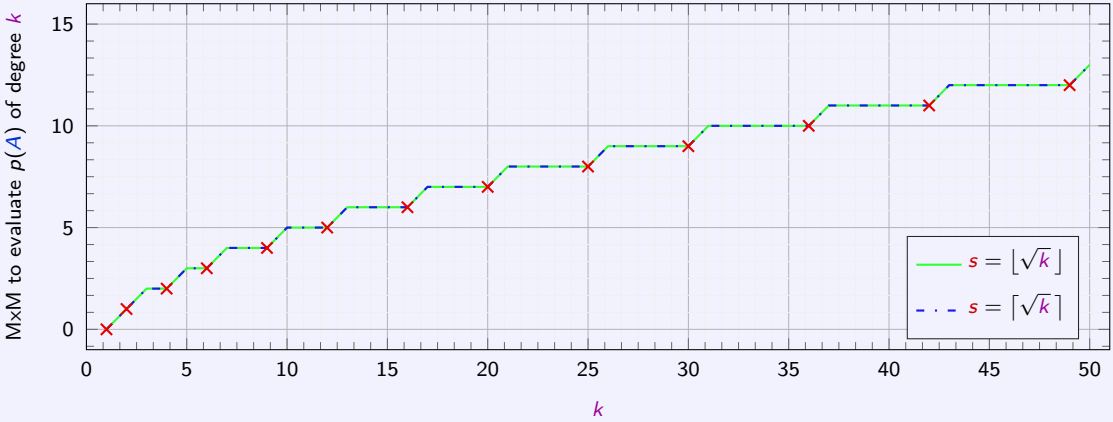
The algorithm requires (recall that $r = \lfloor k/s \rfloor$):

- $r + 1$ diagonal scalings, $r + 1$ diagonal sums,
- $k - r - 1$ scalings, $k - r - 1$ sums,
- **about** $s + r - 1$ matrix products,
- $(s - 1)n^2$ additional elements of storage.

Optimal choice of the parameter

For $s \in \mathbb{R}$ the number of matrix multiplication is minimized by $s = \sqrt{k}$.
For $s \in \mathbb{N}$ one can choose either $s = \lfloor \sqrt{k} \rfloor$ or $s = \lceil \sqrt{k} \rceil$.

Computational cost



Matrix multiplications to evaluate $p(A)$ of degree k for $s = \lfloor \sqrt{k} \rfloor$ and $s = \lceil \sqrt{k} \rceil$.

The Paterson–Stockmeyer method is optimal

Equivalence of the methods [Hargreaves, 2005]

Both choices $s = \lfloor \sqrt{k} \rfloor$ or $s = \lceil \sqrt{k} \rceil$ yield the same number of matrix multiplications.

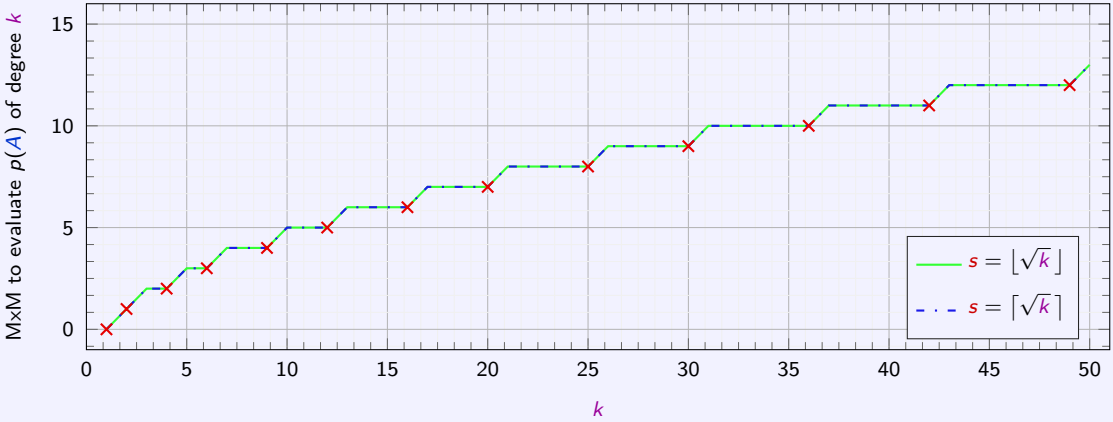
Optimality [Working note, 2018]

The choice $s = \lfloor \sqrt{k} \rfloor$ $\left(\text{or } s = \lceil \sqrt{k} \rceil\right)$ minimizes the number of matrix multiplications required to evaluate $p(A)$ over all choices of s.

Feel of the proofs

Proofs by exhaustion, with plenty of cases.

Slow growth of the computational cost



Matrix multiplications to evaluate $p(A)$ of degree k for $s = \lfloor \sqrt{k} \rfloor$ and $s = \lceil \sqrt{k} \rceil$.

Effect of rounding errors

Bound on forward error [Higham, 2002]

For a polynomial p of degree k, let $\widehat{p}(A)$ be the computed polynomial. Then:

- $|p(A) - \widehat{p}(A)| \leq \gamma_{kn}\widetilde{p}(|A|)$,
- $\|p(A) - \widehat{p}(A)\|_1 \leq \gamma_{kn}\widetilde{p}(\|A\|_1)$,
- $\|p(A) - \widehat{p}(A)\|_\infty \leq \gamma_{kn}\widetilde{p}(\|A\|_\infty)$,

where $\gamma_n = cnu/(1 - cnu)$, and $\widetilde{p}(A) = \sum_{i=0}^{k} |c_i| A^i$.

If significant cancellation, the error in computing $p(A)$ can be large.

Deriving new coefficients

- Evaluation in factored form, $2(k-1)n^3$ flops.
- Recent algorithm [Sastre, 2018], [Sastre, Ibáñez, & Defez, 2018].
- Modified Paterson–Stockmeyer [Paterson & Stockmeyer, 1973]:
  - only $\sqrt{2k} + \log_2 k$ nonscalar multiplications,
  - only rational preprocessing,
  - never seen discussed/used.

Deriving new coefficients

- Evaluation in factored form, $2(k-1)n^3$ flops.
- Recent algorithm [Sastre, 2018], [Sastre, Ibáñez, & Defez, 2018].
- Modified Paterson–Stockmeyer [Paterson & Stockmeyer, 1973]:
  - only $\sqrt{2k} + \log_2 k$ nonscalar multiplications,
  - only rational preprocessing,
  - never seen discussed/used.

Open question

Is any of these methods worth considering at all?

Is Paterson–Stockmeyer optimal?

Summary

- Theoretical lower bound of $\sqrt{k}$ MxM.
- Paterson–Stockmeyer requires $2\sqrt{k}$ MxM with original coefficients.
- Algorithm that requires $\sqrt{2k}$ MxM with modified coefficients.

Open questions

1. Is $2\sqrt{k}$ a lower bound without modifying the coefficients?
2. Is there an algorithm that requires only $\sqrt{k}$ MxM?

Most accurate evaluation

Paterson–Stockmeyer requires computing s consecutive powers of A.

Since A commutes with its powers, we have several algorithms:

- AA, AA^2, AA^3, AA^4, ... (A on the left)
- AA, A^2A, A^3A, A^4A, ... (A on the right)
- AA, A^2A, AA^3, A^4A, ... (alternating)
- AA, AA^2, A^2A^2, A^2A^3, ... (balanced)

Open questions

1. Select most accurate algorithm by looking at A and s only?
2. If A and B commute, when is $fl(AB)$ more accurate than $fl(BA)$?

Evaluation of sparse polynomials

Sparse polynomial

A polynomial with enough zero coefficients to make exploiting the structure worth it.

Open questions

1. What is the best way of evaluating sparse polynomials?
2. When is the sparsity worth exploiting?

Take home messages

1. **Evaluating polynomials of matrices is not entirely trivial.**
2. Not clear how fast reasonably accurate algorithms are.
3. Not clear how accurate faster algorithms are.
4. Still plenty of open questions waiting to be solved.