Algorithms for Structured Linear Systems Solving and their Implementation

Seung Gyu Hyun, Éric Schost

University of Waterloo

Romain Lebreton

Université de Montpellier

Séminaire Eco-Escape – LIRMM

June 6th 2018

Hermite-Padé approximants

Given $r_1, \ldots, r_s \in k[x]$, compute $f_1, \ldots, f_s \in k[x]$ such that

$$f_1 r_1 + \cdots + f_s r_s = 0 \bmod x^\delta \qquad + \text{ degree conditions}$$

Example

If

$$\begin{cases} r_0 = 8x^4 - 8x^2 + 1 \\ r_1 = 16x^5 - 20x^3 + 5x \\ r_2 = 32x^6 - 48x^4 + 18x^2 - 1 \end{cases}$$

then we find the relation $r_0 - 2x r_1 + r_2 = 0 \bmod x^3$ $\rightsquigarrow$ Chebyshev polynomials

Hermite-Padé approximants

Given $r_1, \ldots, r_s \in k[x]$, compute $f_1, \ldots, f_s \in k[x]$ such that

$$f_1 r_1 + \cdots + f_s r_s = 0 \bmod x^\delta \qquad + \text{ degree conditions}$$

Solved by order basis or **structured linear algebra** :

$$\left[\begin{array}{ccc|c|ccc} r_{1,0} & 0 & 0 & \cdots & r_{s,0} & 0 & 0 \\ r_{1,1} & r_{1,0} & 0 & \cdots & r_{s,1} & r_{s,0} & 0 \\ \vdots & r_{1,1} & \ddots & \cdots & \vdots & r_{s,1} & \ddots \\ \vdots & \vdots & \ddots & \cdots & \vdots & \ddots & \ddots \\ r_{1,\delta} & r_{1,\delta-1} & \ddots & \cdots & r_{s,\delta} & \ddots & \ddots \end{array}\right] \left[\begin{array}{c} f_{1,0} \\ \vdots \\ f_{2,0} \\ \vdots \\ f_{s,0} \\ \vdots \end{array}\right] = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \end{array}\right]$$

Special case:

- algebraic approximants $r_{i+1} = r^i$
- differential approximants $r_{i+1} = \dfrac{d^i r}{dx^i}$

Outline of the talk

Goal. Design an implementation and study its practical performance in symbolic computation.

Outline.

1. **Structured matrices basic operations**
2. Structured matrices inversion over $\mathbb{F}_p$
   a. Cauchy-like inversion: iterative & divide-and-conquer
   b. Application to Hermite-Padé approximants
   c. Implementations
3. Structured matrices inversion over $\mathbb{Q}$
   a. Modular techniques
   b. Special case of algebraic approximants

Toeplitz

$$\mathsf{T} = \begin{bmatrix} t_0 & t_{-1} & \cdots & \cdots & t_{-(n-1)} \\ t_1 & t_0 & t_{-1} & \ddots & \vdots \\ \vdots & t_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \cdots & \cdots & t_1 & t_0 \end{bmatrix}$$

Hankel

$$\mathsf{H} = \begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_{(n-1)} \\ h_1 & h_2 & \iddots & \iddots & \vdots \\ h_2 & \iddots & \iddots & \iddots & \vdots \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ h_{n-1} & \cdots & \cdots & \cdots & h_{2n-2} \end{bmatrix}$$

Vandermonde

$$\mathsf{V} = \begin{bmatrix} 1 & u_1 & (u_1)^2 & \cdots & u_1^{n-1} \\ 1 & u_2 & (u_2)^2 & \iddots & \vdots \\ 1 & \vdots & \vdots & \iddots & \vdots \\ 1 & u_n & (u_n)^2 & \cdots & u_n^{n-1} \end{bmatrix}$$

Cauchy

$$\mathsf{C} = \begin{bmatrix} \frac{1}{u_1 - v_1} & \cdots & \frac{1}{u_1 - v_n} \\ \vdots & & \vdots \\ \frac{1}{u_n - v_1} & \cdots & \frac{1}{u_n - v_n} \end{bmatrix}$$

Remark: Hermite-Padé matrix was block Toeplitz

Matrix $\times$ vector product:

1. Toeplitz, Hankel: Cost $\sim\mathcal{M}(n)$ arithmetic operations
2. Vandermonde, Cauchy: Cost $\mathcal{O}(\mathcal{M}(n)\log(n))$ or $\mathcal{O}(\mathcal{M}(n))$ if u, v are geometric sequences

Example – Toeplitz

Let $Z = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ 1 & \ddots & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$. Notice that $Z\,T = (T\downarrow) \simeq (T\leftarrow) = T\,Z$.

So consider $\nabla_{Z,Z}(T) = Z\,T - T\,Z = (T\downarrow) - (T\leftarrow)$ and

$$\nabla_{Z,Z}\left(\begin{bmatrix} t_0 & \cdots & t_{-(n-1)} \\ \vdots & \ddots & \vdots \\ t_{n-1} & \cdots & t_0 \end{bmatrix}\right) = \begin{bmatrix} t_{-1} & \cdots & t_{-(n-1)} & 0 \\ 0 & \cdots & 0 & t_{-(n-1)} \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & t_{-1} \end{bmatrix}$$

We say that T has $\nabla_{Z,Z}$-displacement rank $\alpha = 2$.

Definition of displacement rank [Kailath, Kung, Morf '79]

Displacement operator: $A \in \mathbb{K}^{n\times n} \mapsto \nabla(A) \in \mathbb{K}^{n\times n}$

∇-displacement rank α of A: $\operatorname{rank}(\nabla(A))$

Definition of displacement rank [Kailath, Kung, Morf '79]

Sylvester displacement operator: $\nabla_{M,N}(A) = MA - AN$

$\nabla_{M,N}$-displacement rank α of A: $\alpha := \operatorname{rank}(\nabla_{M,N}(A))$

Let $D_u = \operatorname{Diag}(u_1, \ldots, u_n)$, $Z_\varphi = \begin{bmatrix} & & \varphi \\ 1 & \ddots & \\ & & 1 \end{bmatrix}$ and $Z = Z_0$

Informally, A is **structured** for $\nabla_{M,N}$ if the displacement rank α satisfies $\alpha \ll n$

Examples:

$\nabla_{Z,Z}$: Toeplitz-like, ∇_{Z,Z^t}: Hankel-like, ∇_{D_u,D_v}: Cauchy-like, $\nabla_{D_u,Z}$: Vandermonde-like

In this talk, $M, N \in \{D_u, Z, Z^t\}$

There exists a more general theory with block companion M, N, that applies to Padé-Hermite generalization $f_1 R_{i,1} + \cdots + f_s R_{i,s} = 0 \bmod P_i$. [Olshevsky, Shokrollahi '00] [Bostan, Jeannerod, Mouilleron, Schost '17]

Pan's motto: Compress, Operate, Decompress

1. Compression: Store A with $\ll n^2$ elements
2. Operate: Sum, product, inversion...
3. Decompression: Recover A

Compression

If $\nabla_{M,N}(A)$ has rank α, A is stored as

$$\nabla_{M,N}(A) = G \cdot H^t$$

where $G, H \in \mathbb{K}^{n \times \alpha}$.

Compressed size: $2\,\alpha\,n \ll n^2$

Decompression – Inversibility

Is $\nabla_{M,N}$ always invertible ?

- No, consider $\nabla_{Z,Z}(A) = ZA - AZ$ then $\nabla_{Z,Z}(Z^i) = 0$ for all i.
- **Theorem.** $\nabla_{M,N}$ is invertible if and only if $\mathrm{Spec}(M) \cap \mathrm{Spec}(N) = \varnothing$.

Decompression – Cauchy

Inversibility. If $\{u_i\} \cap \{v_j\} = \mathrm{Spec}(M) \cap \mathrm{Spec}(N) = \varnothing$ then ∇_{D_u,D_v} is invertible.

Reconstruction formula. If $\nabla_{D_u,D_v}(A) = (s_{i,j})$ then

$$A = (s_{i,j}/(u_i - v_j))$$

Decompression – Reconstruction formulae

Theorem. [Gohberg, Olshevsky '92], [Wood '93], [Pan et al. '02]

If M *invertible,* N *nilpotent and* $\nabla_{\mathsf{M},\mathsf{N}}(\mathsf{A}) = \mathsf{G} \cdot \mathsf{H}^t$*, then* $\nabla_{\mathsf{M},\mathsf{N}}$ *is invertible and*

$$\mathsf{A} = \mathsf{M}^{-1} \cdot \mathrm{Krylov}(\mathsf{M}^{-1}, \mathsf{G}) \cdot \mathrm{Krylov}(\mathsf{N}^t, \mathsf{H})^t$$

where $\mathrm{Krylov}(\mathsf{M}, \mathsf{G}) = [\ \mathsf{M}^{n-1}\mathsf{G} \,|\, \cdots \,|\, \mathsf{M}^2\mathsf{G} \,|\, \mathsf{M}\mathsf{G} \,|\, \mathsf{G}\ ]$.

Decompression – Vandermonde

Since $\mathsf{M} = \mathsf{D}_u$ is invertible and $\mathsf{N} = \mathsf{Z}$ is nilpotent, apply the theorem.

Decompression – Toeplitz, Hankel

Consider $\nabla_{\mathsf{Z}_1,\mathsf{Z}}$ instead of $\nabla_{\mathsf{Z},\mathsf{Z}}$:

- $\nabla_{\mathsf{Z}_1,\mathsf{Z}}$ is invertible — Indeed Z_1 is invertible and Z is nilpotent
- $|\alpha_{\nabla_{\mathsf{Z}_1,\mathsf{Z}}} - \alpha_{\nabla_{\mathsf{Z},\mathsf{Z}}}| \leqslant 1$ — Since $\nabla_{\mathsf{Z}_1,\mathsf{Z}}(A) - \nabla_{\mathsf{Z},\mathsf{Z}}(A) = \underbrace{(\mathsf{Z}_1 - \mathsf{Z})}_{\text{rank } 1}\mathsf{A}$

Stability of basic operations

Let $\mathrm{rank}(\nabla_{M,N}(A)) = \alpha$ and $\mathrm{rank}(\nabla_{M,N}(B)) = \beta$,

1. **Addition**

   $$\nabla_{M,N}(A+B) = \nabla_{M,N}(A) + \nabla_{M,N}(B) \qquad \mathrm{rank} \leqslant \alpha + \beta$$

2. **Transpose**

   $$\nabla_{N^t,M^t}(A^t) = (MA - AN)^t = (\nabla_{M,N}(A))^t \qquad \mathrm{rank} = \alpha$$

3. **Inverse**

   $$A^{-1}\nabla_{M,N}(A)A^{-1} = A^{-1}M - NA^{-1} = -\nabla_{N,M}(A^{-1}) \qquad \mathrm{rank} = \alpha$$

4. **Multiplication**

   $$\nabla_{M,P}(AB) = (MA - AN)B + A(NB - BP) = \nabla_{M,N}(A)B + A\nabla_{N,P}(B) \qquad \mathrm{rank} \leqslant \alpha + \beta$$

   Note: Requires (structured matrix)$\times$(dense matrix) product

Stability of basic operations

Let $\operatorname{rank}(\nabla_{M,N}(A)) = \alpha$ and $\operatorname{rank}(\nabla_{M,N}(B)) = \beta$,

1. **Addition**

   $$\nabla_{M,N}(A+B) = \nabla_{M,N}(A) + \nabla_{M,N}(B) \qquad \operatorname{rank} \leqslant \alpha + \beta$$

2. **Transpose**

   $$\nabla_{N^t,M^t}(A^t) = (MA - AN)^t = (\nabla_{M,N}(A))^t \qquad \operatorname{rank} = \alpha$$

3. **Inverse**

   $$A^{-1}\nabla_{M,N}(A)A^{-1} = A^{-1}M - NA^{-1} = -\nabla_{N,M}(A^{-1}) \qquad \operatorname{rank} = \alpha$$

4. **Multiplication**

   $$\nabla_{M,P}(AB) = (MA - AN)B + A(NB - BP) = \nabla_{M,N}(A)B + A\nabla_{N,P}(B) \qquad \operatorname{rank} \leqslant \alpha + \beta$$

   Note: Requires (structured matrix)$\times$(dense matrix) product

Stability of basic operations

Let $\mathrm{rank}(\nabla_{M,N}(A)) = \alpha$ and $\mathrm{rank}(\nabla_{M,N}(B)) = \beta$,

1. **Addition**

$$\nabla_{M,N}(A+B) = \nabla_{M,N}(A) + \nabla_{M,N}(B) \qquad \mathrm{rank} \leqslant \alpha + \beta$$

2. **Transpose**

$$\nabla_{N^t,M^t}(A^t) = (MA - AN)^t = (\nabla_{M,N}(A))^t \qquad \mathrm{rank} = \alpha$$

3. **Inverse**

$$A^{-1}\nabla_{M,N}(A)A^{-1} = A^{-1}M - NA^{-1} = -\nabla_{N,M}(A^{-1}) \qquad \mathrm{rank} = \alpha$$

4. **Multiplication**

$$\nabla_{M,P}(AB) = (MA - AN)B + A(NB - BP) = \nabla_{M,N}(A)B + A\nabla_{N,P}(B) \qquad \mathrm{rank} \leqslant \alpha + \beta$$

Note: Requires (structured matrix)$\times$(dense matrix) product

Stability of basic operations

Let $\mathrm{rank}(\nabla_{M,N}(A)) = \alpha$ and $\mathrm{rank}(\nabla_{M,N}(B)) = \beta$,

1. **Addition**

   $$\nabla_{M,N}(A+B) = \nabla_{M,N}(A) + \nabla_{M,N}(B) \qquad \mathrm{rank} \leqslant \alpha + \beta$$

2. **Transpose**

   $$\nabla_{N^t,M^t}(A^t) = (M\,A - A\,N)^t = (\nabla_{M,N}(A))^t \qquad \mathrm{rank} = \alpha$$

3. **Inverse**

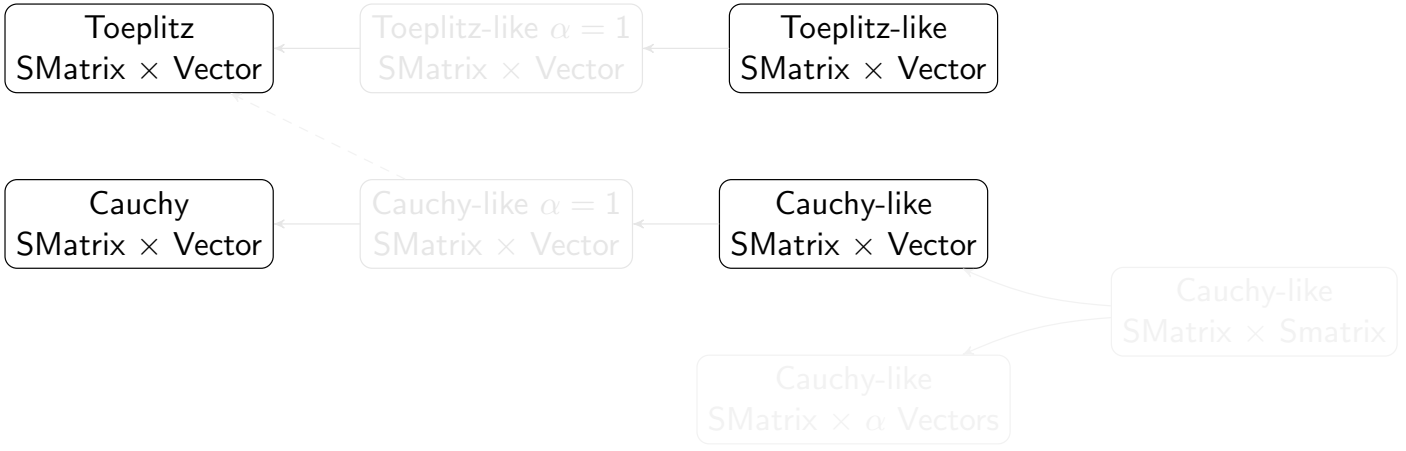
   $$A^{-1}\,\nabla_{M,N}(A)\,A^{-1} = A^{-1}\,M - N\,A^{-1} = -\nabla_{N,M}(A^{-1}) \qquad \mathrm{rank} = \alpha$$

4. **Multiplication**

   $$\nabla_{M,P}(A\,B) = (M\,A - A\,N)\,B + A\,(N\,B - B\,P) = \nabla_{M,N}(A)\,B + A\,\nabla_{N,P}(B) \qquad \mathrm{rank} \leqslant \alpha + \beta$$

   Note: Requires (structured matrix)$\times$(dense matrix) product

Structured matrix – vector product

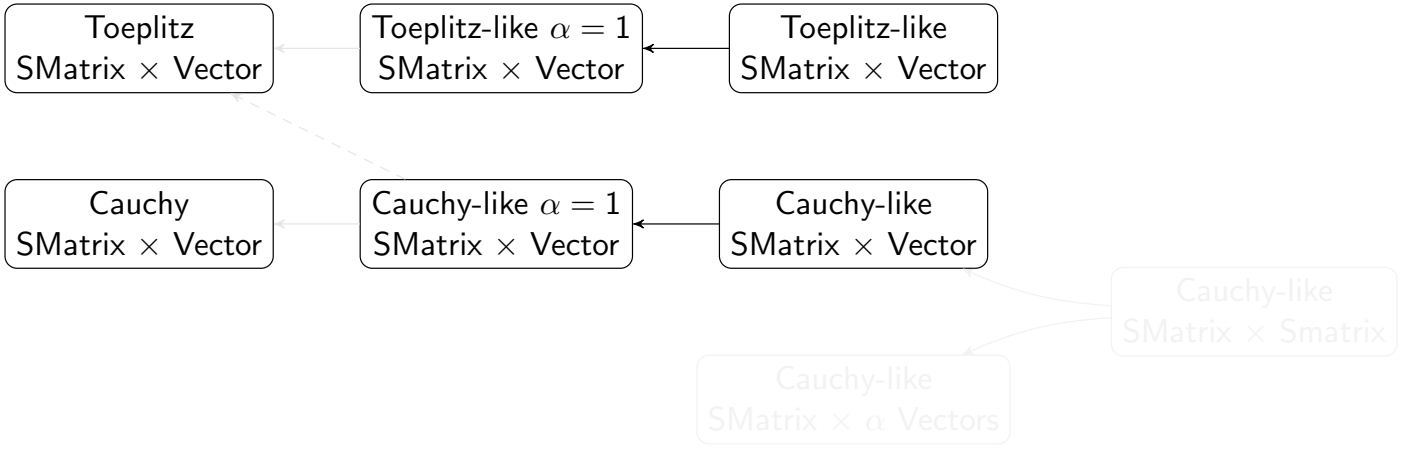


Reminder:

1. Toeplitz matrix $\times$ vector product costs $\sim\mathcal{M}(n)$
2. Cauchy matrix $\times$ vector product costs $\mathcal{O}(\mathcal{M}(n)\log(n))$ (or $\mathcal{O}(\mathcal{M}(n))$ in special case)

How do we perform Toeplitz/Cauchy-like matrix – vector multiplication ?

Structured matrix – vector product



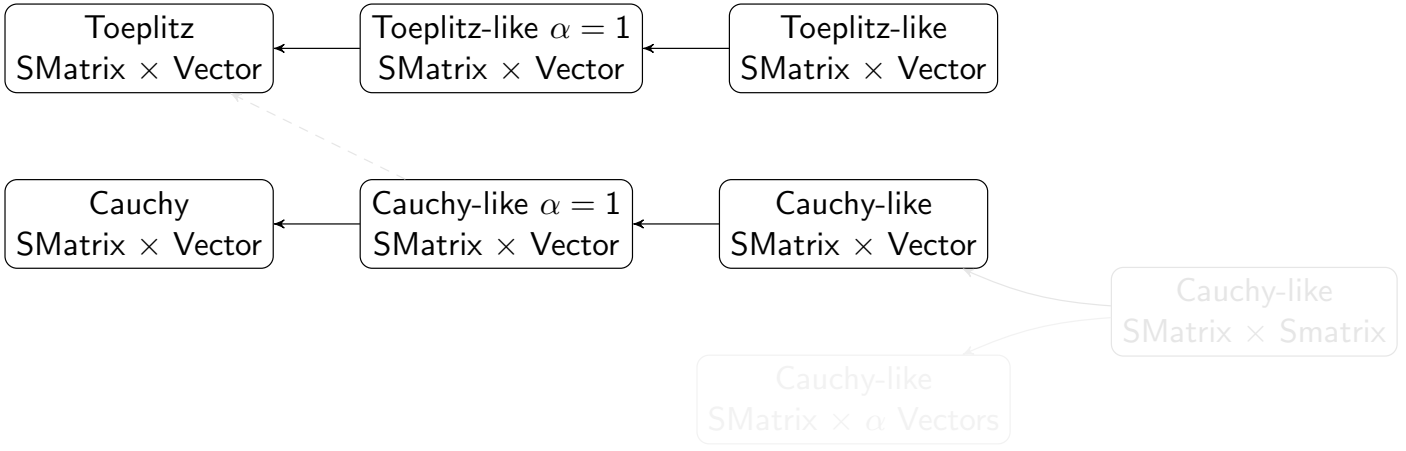
Remark. A is the sum of displacement rank 1 matrices

Decompose $\mathsf{G} \cdot \mathsf{H}^t = \underbrace{\mathsf{G}_1 \cdot \mathsf{H}_1^t}_{\text{rank } 1} + \cdots + \underbrace{\mathsf{G}_\alpha \cdot \mathsf{H}_\alpha^t}_{\text{rank } 1}$ where $\mathsf{G}_i, \mathsf{H}_i$ are the i-th column

Then A is the sum of displacement rank 1 matrices:

$$\mathsf{A} = \nabla^{-1}(\mathsf{G} \cdot \mathsf{H}^t) = \underbrace{\nabla^{-1}(\mathsf{G}_1 \cdot \mathsf{H}_1^t)}_{\text{displacement rank } 1} + \cdots + \underbrace{\nabla^{-1}(\mathsf{G}_\alpha \cdot \mathsf{H}_\alpha^t)}_{\text{displacement rank } 1}$$

Structured matrix – vector product



Toeplitz-like $\alpha = 1$

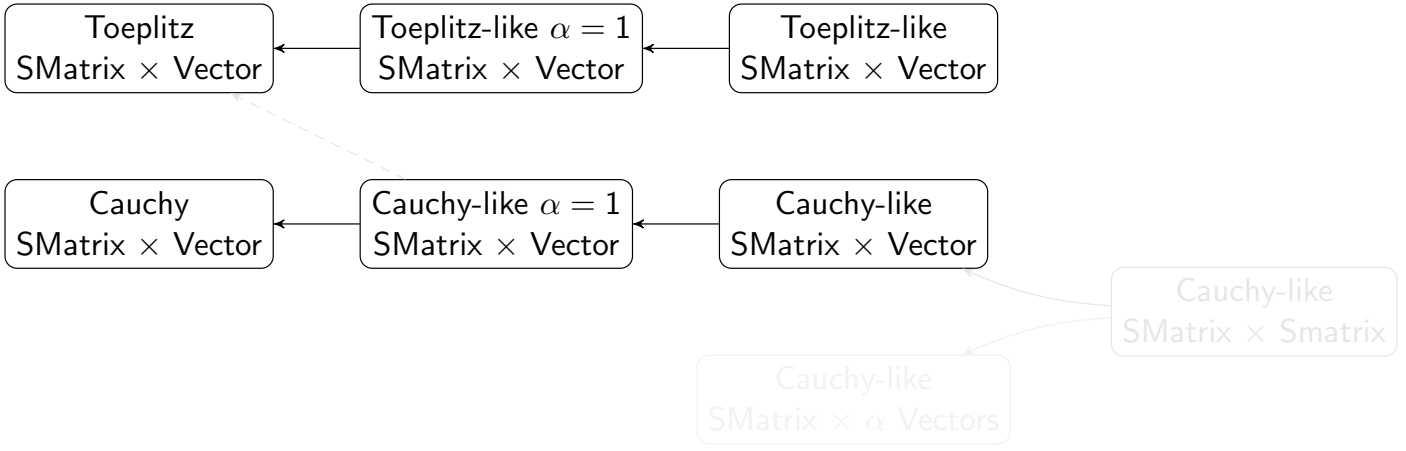
Since $\alpha = 1$, write $\nabla_{Z_1,Z}(A) = g \cdot h^t = (g_i h_j)_{i,j}$. Then

$$A = Z_1^{-1} \cdot \mathrm{Krylov}(Z_1^{-1}, g) \cdot \mathrm{Krylov}(Z, h)^t = \underbrace{\left[\begin{array}{c|c|c|c} g_1 & g_n & \ddots & g_2 \\ g_2 & g_1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & g_n \\ g_n & g_{n-1} & \ddots & g_1 \end{array}\right]}_{\text{Toeplitz}} \cdot \underbrace{\left[\begin{array}{c|c|c|c} h_n & 0 & \cdots & 0 \\ \hline \vdots & h_n & \ddots & \vdots \\ \hline h_2 & \ddots & \ddots & 0 \\ \hline h_1 & h_2 & .... & h_n \end{array}\right]}_{\text{Toeplitz}}$$

So reduction to Toeplitz matrix-vector product.

Cost: $\sim 2\,\mathcal{M}(n)$

Structured matrix – vector product



Cauchy-like $\alpha = 1$

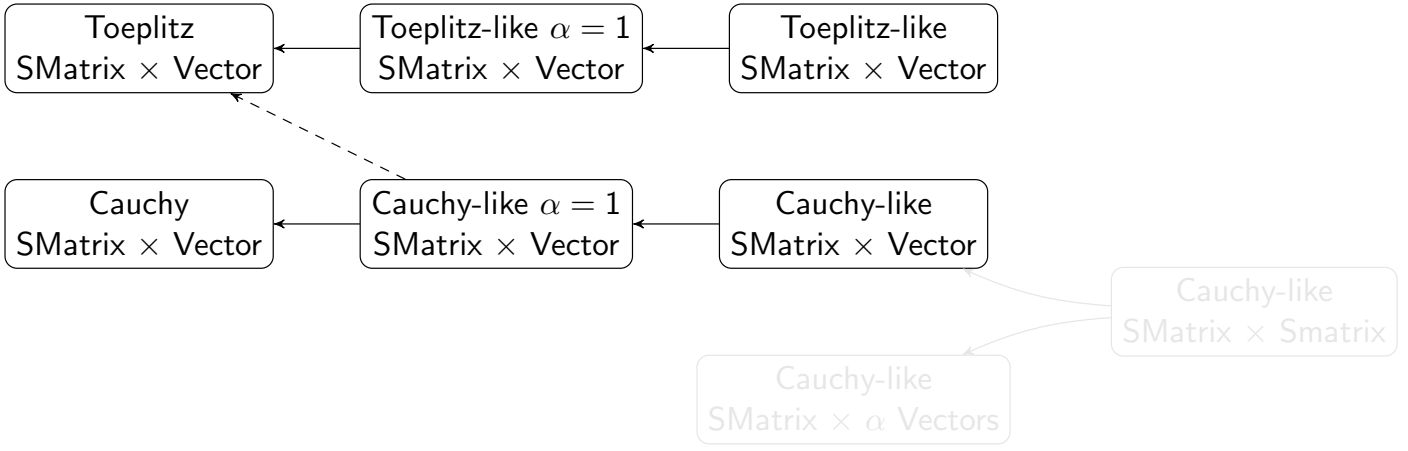
If $\nabla_{\mathsf{D_u},\mathsf{D_v}}(\mathsf{A}) = (g_i\, h_j)_{i,j}$ then

$$\mathsf{A} = (g_i\, h_j / (u_i - v_j))_{i,j} = \mathsf{D_g}\, \mathsf{C_{u,v}}\, \mathsf{D_h}$$

So reduction to Cauchy matrix-vector product.

Cost for matrix-vector product: $\mathcal{O}(\mathcal{M}(n)\log(n))$

Structured matrix – vector product



Cauchy-like $\alpha = 1$ – Trick for geometric sequences with same ratio

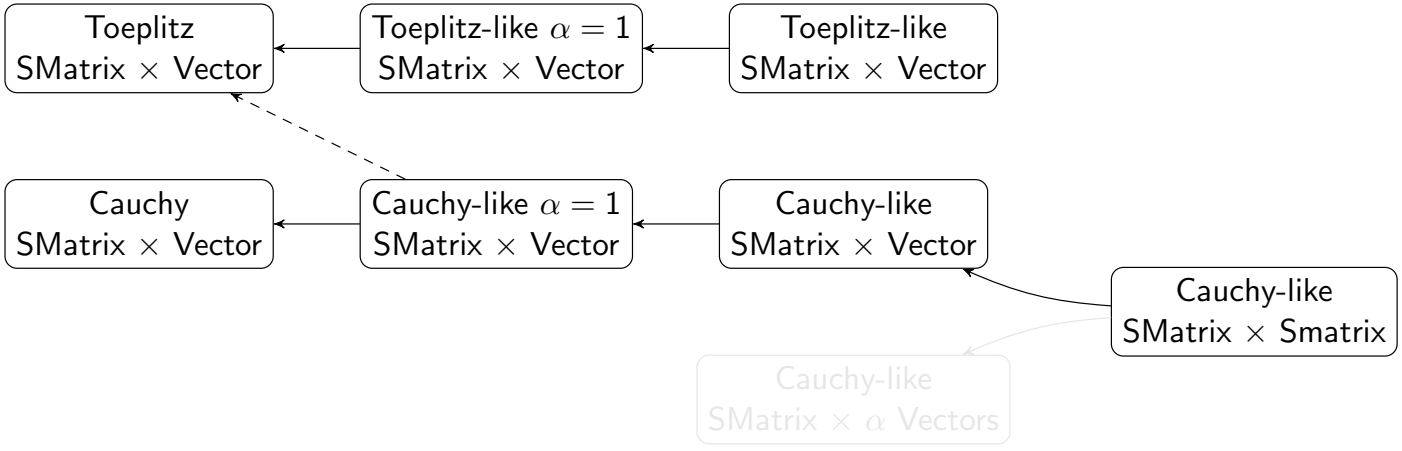
If $u_i = \tau^{i-1} u_1$, $v_j = \tau^{j-1} v_1$, then

$$C = \left[\frac{1}{u_i - v_j} \right] = \left[\frac{1}{\tau^{i-1} u_1 - \tau^{j-1} v_1} \right] = \left[\frac{1}{\tau^{i-1}} \frac{1}{u_1 - \tau^{j-i} v_1} \right] = D_{1,\tau^{-1},\ldots} \cdot \underbrace{\left[\frac{1}{u_1 - \tau^{j-i} v_1} \right]}_{\text{Toeplitz}}$$

Reduce to Toeplitz matrix vector-product

Cost for matrix-vector product: $\sim \mathcal{M}(n)$ (vs. $\sim 3\,\mathcal{M}(n)$ for classical approach)

Structured matrix – vector product



Consequence to product of structured matrices

If $\nabla_{M,N}(A) = G \cdot H^t$, $\nabla_{N,P}(B) = Y \cdot Z^t$ with $G, H, Y, Z \in \mathbb{K}^{n \times \alpha}$ then

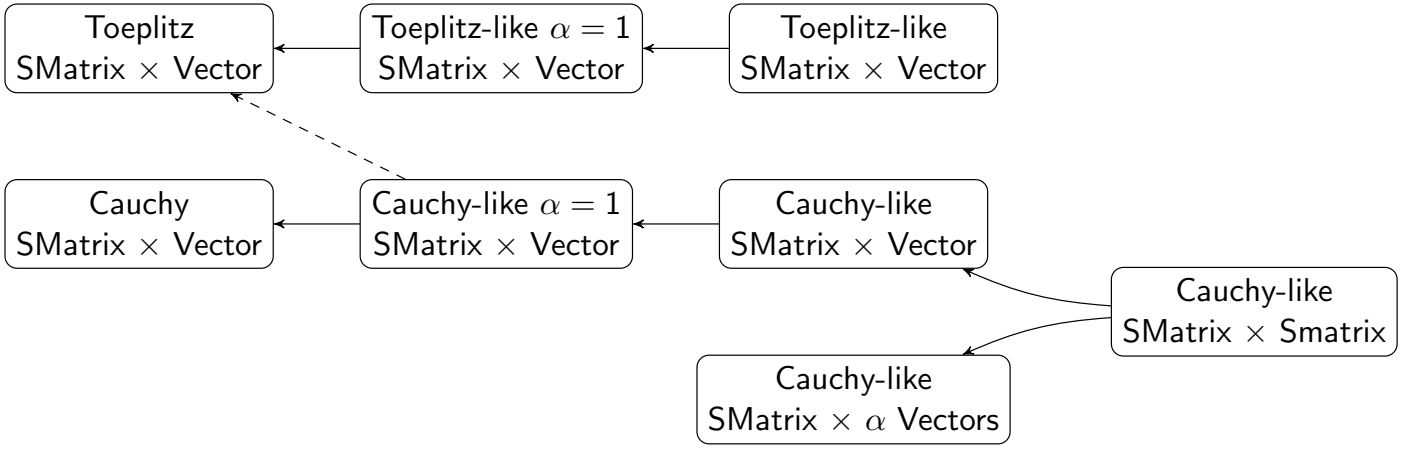
$$\nabla_{M,P}(A\,B) = \nabla_{M,N}(A)\,B + A\,\nabla_{N,P}(B) = G \cdot (B^t\,H)^t + (A\,Y) \cdot Z^t$$

$A\,Y$ and $B^t\,H$ are (structured matrix)$\times$(α vectors) product

Solution 1: Decompose into α (structured matrix)$\times$(vector) products, each cost $\tilde{\mathcal{O}}(\alpha n)$

Total cost: $\tilde{\mathcal{O}}(\alpha^2 n)$

Structured matrix – vector product



Consequence to product of structured matrices

If $\nabla_{M,N}(A) = G \cdot H^t$, $\nabla_{N,P}(B) = Y \cdot Z^t$ with $G, H, Y, Z \in \mathbb{K}^{n \times \alpha}$ then

$$\nabla_{M,P}(AB) = \nabla_{M,N}(A)\,B + A\,\nabla_{N,P}(B) = G \cdot (B^t H)^t + (AY) \cdot Z^t$$

AY and $B^t H$ are (structured matrix)$\times$(α vectors) product

Solution 2: [Bostan, Jeannerod, Schost '08], [Bostan, Jeannerod, Mouilleron, Schost '17]

One (structured matrix)$\times$(α vectors) product in cost $\tilde{\mathcal{O}}(\alpha^{\omega-1} n)$

Total cost: $\tilde{\mathcal{O}}(\alpha^{\omega-1} n)$

Outline

1. Structured matrices basic operations

2. **Structured matrices inversion over $\mathbb{F}_p$**
   a. Cauchy-like inversion: iterative & divide-and-conquer
   b. Application to Hermite-Padé approximants
   c. Implementations

3. Structured matrices inversion over $\mathbb{Q}$
   a. Modular techniques
   b. Special case of algebraic approximants

Cauchy-like inversion – $\mathsf{S}^{(i)}$

Intermediate inverse matrices $\mathsf{S}^{(i)}$

[Cardinal '99], [Jeannerod, Mouilleron '10]

For $0 \leqslant i \leqslant n$, cut $\mathsf{A} = \begin{bmatrix} \mathsf{A}_{00} & \mathsf{A}_{01} \\ \mathsf{A}_{10} & \mathsf{A}_{11} \end{bmatrix}$ so that $\mathsf{A}_{00} \in \mathbb{K}^{i \times i}$. Define

$$\mathsf{S}^{(i)} = \mathsf{PartialInversion}_i(\mathsf{A}) = \left[\begin{array}{c|c} \mathsf{A}_{11} - \mathsf{A}_{10}\mathsf{A}_{00}^{-1}\mathsf{A}_{01} & \mathsf{A}_{10}\mathsf{A}_{00}^{-1} \\ \hline -\mathsf{A}_{00}^{-1}\mathsf{A}_{01} & \mathsf{A}_{00}^{-1} \end{array}\right]$$

Facts.

1. $\mathsf{S}^{(0)} = \mathsf{A}$, $\mathsf{S}^{(n)} = \mathsf{A}^{-1}$
2. We have

$$\mathsf{PartialInversion}_i = \mathsf{PartialInversion}_1 \circ \cdots \circ \mathsf{PartialInversion}_1$$

which implies that

$$\mathsf{S}^{(0)} = \mathsf{A} \xrightarrow{\mathsf{PartialInversion}_\beta} \mathsf{S}^{(\beta)} \xrightarrow{\mathsf{PartialInversion}_\beta} \mathsf{S}^{(2\beta)} \xrightarrow{\mathsf{PartialInversion}_\beta} \cdots \longrightarrow \mathsf{S}^{(k\beta)}$$

Cauchy-like inversion – generators of $S^{(i)}$

How can we compute PartialInversion with structured matrices ?

$$S^{(i)} = \text{PartialInversion}_i\left(\begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix}\right) = \begin{bmatrix} A_{11} - A_{10}A_{00}^{-1}A_{01} & A_{10}A_{00}^{-1} \\ -A_{00}^{-1}A_{01} & A_{00}^{-1} \end{bmatrix}$$

Cauchy magic – Submatrices are structured

For instance if $G = \begin{bmatrix} G_0 \\ G_1 \end{bmatrix}$, $H = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}$ then G_0, H_0 are generators of A_{00} (rank α).

Cardinal's formula – $S^{(i)}$ has displacement rank α !

$$\nabla_{[u_1,v_0],[v_1,u_0]}(S^{(i)}) = \begin{bmatrix} -A_{10}G_0' + G_1 \\ G_0' \end{bmatrix} \cdot \begin{bmatrix} -A_{01}^t H_0' + H_1 \\ H_0' \end{bmatrix}^t$$

where G_0', H_0' are ∇_{v_0,u_0}-generators of A_{00}^{-1}.

Partial inversion using structured operations

Algorithm PartialInversion$_i$-Structured

Input: $0 \leqslant i \leqslant n$

$\mathsf{G}, \mathsf{H}, \mathsf{u}, \mathsf{v}$ s.t. $\nabla_{\mathsf{u},\mathsf{v}}(\mathsf{A}) = \mathsf{G} \cdot \mathsf{H}^t$

$\mathsf{G}'_0, \mathsf{H}'_0, \mathsf{u}_0, \mathsf{v}_0$ s.t. $\nabla_{\mathsf{v}_0,\mathsf{u}_0}(\mathsf{A}_{00}^{-1}) = \mathsf{G}'_0 \cdot \mathsf{H}'^t_0$ where $\mathsf{A}_{00} \in \mathbb{K}^{i \times i}$

Output: $\mathsf{Y}, \mathsf{Z}, \mathsf{u}', \mathsf{v}'$ such that $\nabla_{\mathsf{u}',\mathsf{v}'}(\mathsf{S}^{(i)}) = \mathsf{Y} \cdot \mathsf{Z}^t$

1. Cut $\mathsf{G}, \mathsf{H}, \mathsf{u}, \mathsf{v}$ in size i
2. Compute $\mathsf{A}_{10}\,\mathsf{G}'_0$ and $\mathsf{A}_{01}^t \mathsf{H}'_0$ using (structured matrix)$\times$(α vectors) product
3. **return**

$$\mathsf{Y} = \begin{bmatrix} -\mathsf{A}_{10}\,\mathsf{G}'_0 + \mathsf{G}_1 \\ \mathsf{G}'_0 \end{bmatrix}, \mathsf{Z} = \begin{bmatrix} -\mathsf{A}_{01}^t \mathsf{H}'_0 + \mathsf{H}_1 \\ \mathsf{H}'_0 \end{bmatrix}, \mathsf{u}' = [\mathsf{u}_1, \mathsf{v}_0], \mathsf{v}' = [\mathsf{v}_1, \mathsf{u}_0]$$

Cost:

- (structured matrix) $\times$ (α vectors) product in $\mathcal{O}(\alpha^{\omega-1}\,\mathcal{M}(n)\log(n))$
- Total cost: $\mathcal{O}(\alpha^{\omega-1}\,\mathcal{M}(n)\log(n))$

Algorithm Struct-inverse-DAC

Input: G, H, u, v such that $\nabla_{u,v}(A) = G \cdot H^t$

Output: Y, Z such that $\nabla_{v,u}(A^{-1}) = Y \cdot Z^t$

1. **if** $n \leqslant \text{threshold}$ **then return** Struct-inverse-iter(G, H, u, v)
2. $S^{(0)} = A \xrightarrow{\text{PartialInversion}_{n/2}} S^{(n/2)}$
   a. Invert leading minor of $S^{(0)}$ using Struct-inverse-DAC
   b. Compute generators Y, Z of $S^{(n/2)}$ using PartialInverse-Structured
3. $S^{(n/2)} \xrightarrow{\text{PartialInversion}_{n/2}} S^{(n)} = A^{-1}$
   a. Invert leading minor of $S^{(n/2)}$ using Struct-inverse-DAC
   b. **return** generators Y, Z of $S^{(n)}$ using PartialInverse-Structured

Total cost:

$$\mathcal{S}(n) = 2\,\mathcal{S}(n/2) + \mathcal{O}(\alpha^{\omega-1}\,\mathcal{M}(n)\log(n)) \quad \Rightarrow \quad \mathcal{S}(n) = \mathcal{O}(\alpha^{\omega-1}\,\mathcal{M}(n)\log^2(n))$$

Cauchy-like divide-and-conquer inversion

Algorithm Struct-inverse-DAC

Input: G, H, u, v such that $\nabla_{u,v}(A) = G \cdot H^t$

Output: Y, Z such that $\nabla_{v,u}(A^{-1}) = Y \cdot Z^t$

1. **if** $n \leqslant$ threshold **then return** Struct-inverse-iter(G, H, u, v)
2. $S^{(0)} = A \xrightarrow{\text{PartialInversion}_{n/2}} S^{(n/2)}$
   a. Invert leading minor of $S^{(0)}$ using Struct-inverse-DAC
   b. Compute generators Y, Z of $S^{(n/2)}$ using PartialInverse-Structured
3. $S^{(n/2)} \xrightarrow{\text{PartialInversion}_{n/2}} S^{(n)} = A^{-1}$
   a. Invert leading minor of $S^{(n/2)}$ using Struct-inverse-DAC
   b. **return** generators Y, Z of $S^{(n)}$ using PartialInverse-Structured

History

- Need compression: [Morf/Bitmead-Anderson '80], [Kaltofen '94]
- Compressed formulae: [Cardinal '99], [Pan '00], [Jeannerod, Mouilleron '10]

Algorithm Struct-inverse-DAC

Input: G, H, u, v such that $\nabla_{u,v}(A) = G \cdot H^t$

Output: Y, Z such that $\nabla_{v,u}(A^{-1}) = Y \cdot Z^t$

1. **if** $n \leqslant \text{threshold}$ **then return** Struct-inverse-iter(G, H, u, v)
2. $S^{(0)} = A \xrightarrow{\text{PartialInversion}_{n/2}} S^{(n/2)}$
   a. Invert leading minor of $S^{(0)}$ using Struct-inverse-DAC
   b. Compute generators Y, Z of $S^{(n/2)}$ using PartialInverse-Structured
3. $S^{(n/2)} \xrightarrow{\text{PartialInversion}_{n/2}} S^{(n)} = A^{-1}$
   a. Invert leading minor of $S^{(n/2)}$ using Struct-inverse-DAC
   b. **return** generators Y, Z of $S^{(n)}$ using PartialInverse-Structured

Contributions in [Hyun, L., Schost '17]

Check the required generic rank profile property

Compute $r = \text{rank}(A)$ and invert only the leading principal minor

Algorithm PartialInversion$_\beta$-Dense

Input: $0 \leqslant \beta \leqslant \alpha$ and G, H, u, v s.t. $\nabla_{u,v}(A) = G \cdot H^t$

Output: Y, Z, u', v' such that $\nabla_{u',v'}(S^{(\beta)}) = Y \cdot Z^t$

1. Cut G, H, u, v in size β
2. Decompress A_{00}, A_{01} and A_{10}
3. Invert A_{00} — Dense inversion
4. Compress $A_{00}^{-1} \rightsquigarrow G'_0, H'_0$
5. Compute $A_{10} G'_0$ and $A_{01}^t H'_0$ — Dense matrix multiplication
6. **return**

$$Y = \begin{bmatrix} -A_{10} G'_0 + G_1 \\ G'_0 \end{bmatrix}, Z = \begin{bmatrix} -A_{01}^t H'_0 + H_1 \\ H'_0 \end{bmatrix}, u' = [u_1, v_0], v' = [v_1, u_0]$$

Why dense matrices ?

$A_{00}, A_{10}, A_{01}^t \in \mathbb{K}^{* \times \beta}$ are too small to take advantage of structured algorithms.

Cost analysis for one iteration when $\beta = \alpha$: $\mathcal{O}(\alpha^{\omega-1} n)$

Algorithm Struct-inverse-iter

Input: $\mathsf{G}, \mathsf{H}, \mathsf{u}, \mathsf{v}$ such that $\nabla_{\mathsf{u},\mathsf{v}}(\mathsf{A}) = \mathsf{G} \cdot \mathsf{H}^t$ and step size β $(1 \leqslant \beta \leqslant \alpha)$

Output: Y, Z such that $\nabla_{\mathsf{v},\mathsf{u}}(\mathsf{A}^{-1}) = \mathsf{Y} \cdot \mathsf{Z}^t$

1. **for** $(i = 0; i < n; i = i + \beta)$
   a. $\mathsf{S}^{(i\beta)} \xrightarrow{\text{PartialInversion}_\beta} \mathsf{S}^{((i+1)\beta)}$
      Compute generators Y, Z of $\mathsf{S}^{((i+1)\beta)}$ using PartialInversion-Dense
2. **return** Y, Z

Idea:

$$\mathsf{S}^{(0)} = \mathsf{A} \xrightarrow{\text{PartialInversion}_\beta} \mathsf{S}^{(\beta)} \xrightarrow{\text{PartialInversion}_\beta} \mathsf{S}^{(2\beta)} \xrightarrow{\text{PartialInversion}_\beta} \cdots \longrightarrow \mathsf{S}^{(n)} = \mathsf{A}^{-1}$$

Total cost when $\beta = \alpha$: $\underbrace{n/\alpha}_{\#\text{iterations}} \times \underbrace{\mathcal{O}(\alpha^{\omega-1} n)}_{\text{PartialInversion-Dense}} = \mathcal{O}(\alpha^{\omega-2} n^2)$

Cauchy-like iterative inversion

Algorithm Struct-inverse-iter

Input: $\mathsf{G}, \mathsf{H}, \mathsf{u}, \mathsf{v}$ such that $\nabla_{\mathsf{u},\mathsf{v}}(\mathsf{A}) = \mathsf{G} \cdot \mathsf{H}^t$ and step size β $(1 \leqslant \beta \leqslant \alpha)$

Output: Y, Z such that $\nabla_{\mathsf{v},\mathsf{u}}(\mathsf{A}^{-1}) = \mathsf{Y} \cdot \mathsf{Z}^t$

1. **for** $(i = 0; i < n; i = i + \beta)$
   a. $\mathsf{S}^{(i\beta)} \xrightarrow{\mathsf{PartialInversion}_\beta} \mathsf{S}^{((i+1)\beta)}$
      Compute generators Y, Z of $\mathsf{S}^{((i+1)\beta)}$ using PartialInversion-Dense
2. **return** Y, Z

History

$\beta = 1$: Cost $\mathcal{O}(\alpha\, n^2)$	[Levinson '47], [Durbin '60], [Trench '64],[Kailath '99], [Mouilleron '08]
$\beta = \alpha$: Cost $\mathcal{O}(\alpha^{\omega-2}\, n^2)$	[Hyun, L., Schost '17]

Original motivation – Hermite Padé approximants

Find a non-zero vector in the kernel of T Toeplitz-like

Transformation between classes of structured matrices [Pan '90] [Heinig '94], [Gohberg et al. '95]

We can reduce questions about Toeplitz-like matrices to ones about Cauchy-like matrices.

Advantages:

1. Benefit from efficient Cauchy-like inversion
2. Can choose u, v geometric sequences with same ratio $\rightsquigarrow$ Gain factor 3
3. The generated C should have generic rank profile

Note: [Jeannerod, Mouilleron '10] extend Cardinal's compressed formulae to Toeplitz.

Toeplitz-like to Cauchy-like

Transformation

If T is Toeplitz-like of displacement rank α then

$$C = V_u\, T\, W_v \text{ is Cauchy-like with displacement rank } \leqslant \alpha + 2$$

Why ? V_u is $\nabla_{D_u,Z}$-structured, T is $\nabla_{Z,Z}$-structured and W_v is ∇_{Z,D_v}-structured

Moderate cost overhead: $\mathcal{O}(\alpha\, \mathcal{M}(n) \log(n))$

Regularization

[Pan '01]: Regularization of Cauchy-like A $\qquad A' = D_x\, C_{a,u}\, A\, C_{v,b}\, D_y$

[Hyun, L., Schost '17]: $A = V_u\, T\, W_v$ has generic rank profile for generic u, v in $\mathbb{K}$

Implementation details:

- C++ implementation based on Shoup's NTL

  NTL provides efficient polynomial arithmetic and matrix multiplication over small prime fields

- Available at https://github.com/romainlebreton/structured_linear_system_solving

Iterative inversion: $\mathcal{O}(\alpha\, n^2)$ (step size $\beta = 1$) vs. $\mathcal{O}(\alpha^{\omega-2}\, n^2)$ ($\beta = \alpha$).

Note: $\mathcal{O}(\alpha^{\omega-2}\, n^2)$ algorithm is faster as soon as $\alpha \geqslant 10$

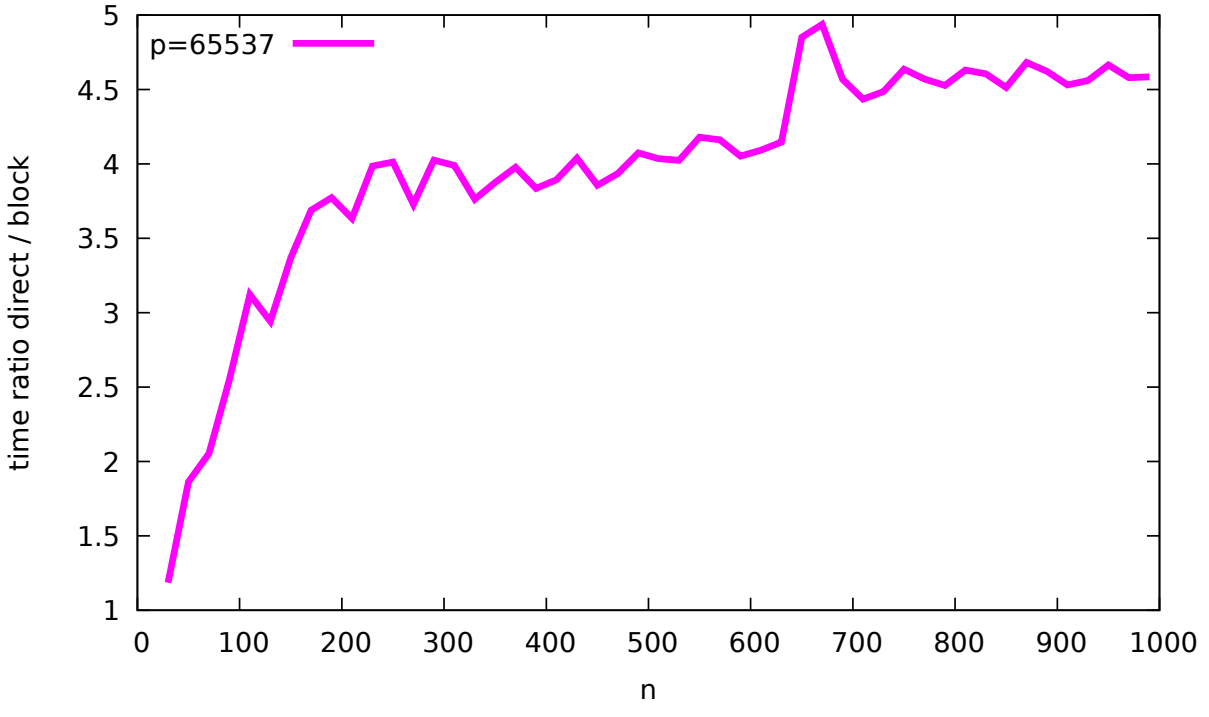
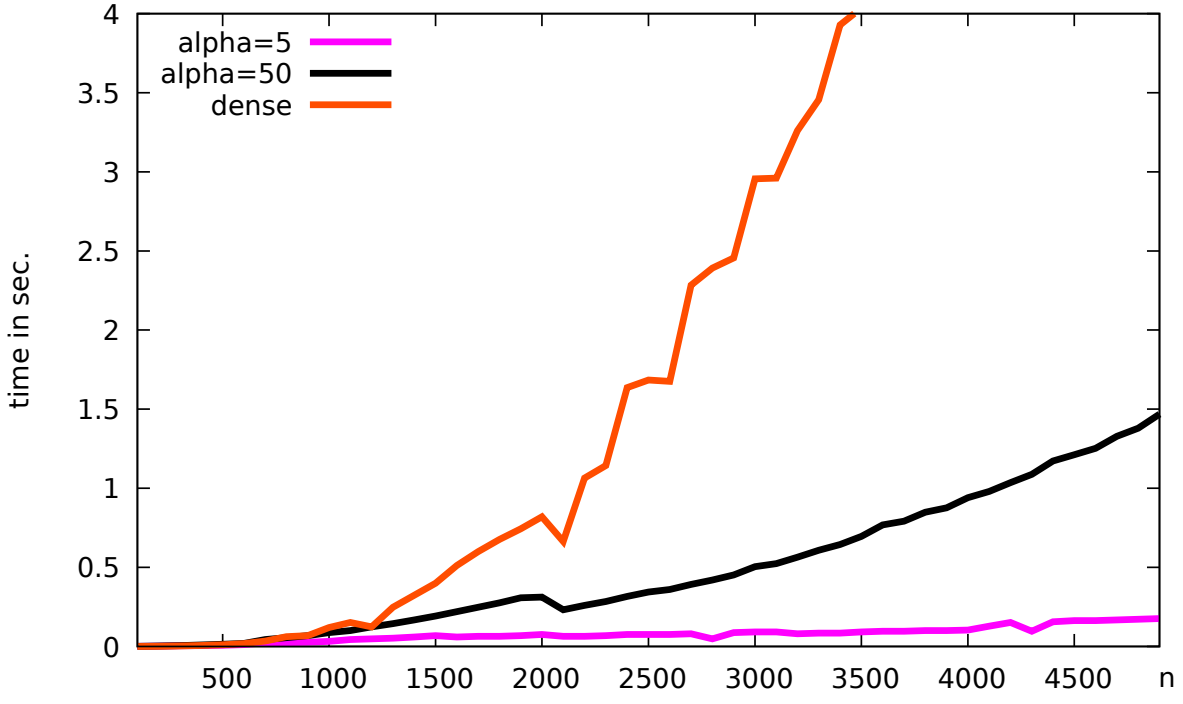


Figure. Time ratio "$(\alpha\, n^2)\,/\,(\alpha^{\omega-2}\, n^2)$" for $\alpha = 30$

Implementation – DAC Cauchy-like inversion

DAC algorithm: Dense $\mathcal{O}(n^\omega)$ vs structured $\mathcal{O}(\alpha^{\omega-1}\,\mathcal{M}(n)\log(n))$ ($\alpha=5$ and $\alpha=50$)

Note: Structured algorithms are faster as soon as $\alpha \leqslant 0.2\,n$



For Pade-Hermite and Toeplitz-like inversion: overhead of $\sim 5\%$

Outline

1. Structured matrices basic operations

2. Structured matrices inversion over $\mathbb{F}_p$
   a. Cauchy-like inversion: iterative & divide-and-conquer
   b. Application to Hermite-Padé approximants
   c. Implementations

3. **Structured matrices inversion over $\mathbb{Q}$**
   a. Modular techniques
   b. Special case of algebraic approximants

Algorithms over the rationals

Initial goal: Pade-Hermite approximant over $\mathbb{Q}$, so find x s.t. $T\,x = 0$ (T Toeplitz-like)

Reduce the problem:

1. Cauchy matrices: $C = V_u\,T\,W_v$ and $C\,y = 0 \Leftrightarrow T\,(W_v\,y) = 0$
2. Kernel to linear system solving: $\ker C = \left\{ \begin{bmatrix} C_{00}^{-1}\,C_{01}\,r \\ -r \end{bmatrix} \right\}_r$ where C_{00} maximal minor
3. Modular techniques: Solve modulo N, then apply rational reconstruction

Current Goal – Modular Cauchy system soving

Given C invertible Cauchy-like matrix, solve

$$C\,x = b \bmod N$$

Cost model: From now on, cost model will be bit complexity

Notation: $\mathcal{I}(t)$ – cost of multiplying t-bit integers

Newton Iteration [Pan '92] $N = p^t$

- Compute generators of C^{-1} modulo $p, p^2, p^4, ..., p^t$
- Bit complexity: $\mathcal{O}\left(\underbrace{\alpha^{\omega-1}\,\mathcal{M}(n)\,\mathcal{I}(t)}_{\text{lifting}} + \underbrace{\alpha^{\omega-1}\,\mathcal{M}(n)\log(n)}_{\mathsf{C}^{-1}\bmod p}\right)$

Divide-and-Conquer $N = p^t$

- Recursively compute the solution at half the precision and cancel the residue
- Particular case of a faster online algorithm [Hoeven '02], [L. '12]
- Bit complexity: $\mathcal{O}\left(\underbrace{\alpha\,\mathcal{M}(n)\,\mathcal{I}(t)\log(t)}_{\text{lifting}} + \underbrace{\alpha^{\omega-1}\,\mathcal{M}(n)\log(n)}_{\mathsf{C}^{-1}\bmod p}\right)$

Chinese Remainder Theorem $N = p_1 \cdots p_t$

- Solve $\mathsf{C}\,\mathsf{x} = \mathsf{b}$ modulo t primes $p_1, ..., p_t$ of the same magnitude
- Bit complexity: $\mathcal{O}\left(\underbrace{\alpha^{\omega-1}\,\mathcal{M}(n)\log(n)\,t}_{\mathsf{C}^{-1}\bmod p_1,\ldots,p_t} + \underbrace{n\,\mathcal{I}(t)\log(t)}_{\text{Chinese remaindering}}\right)$

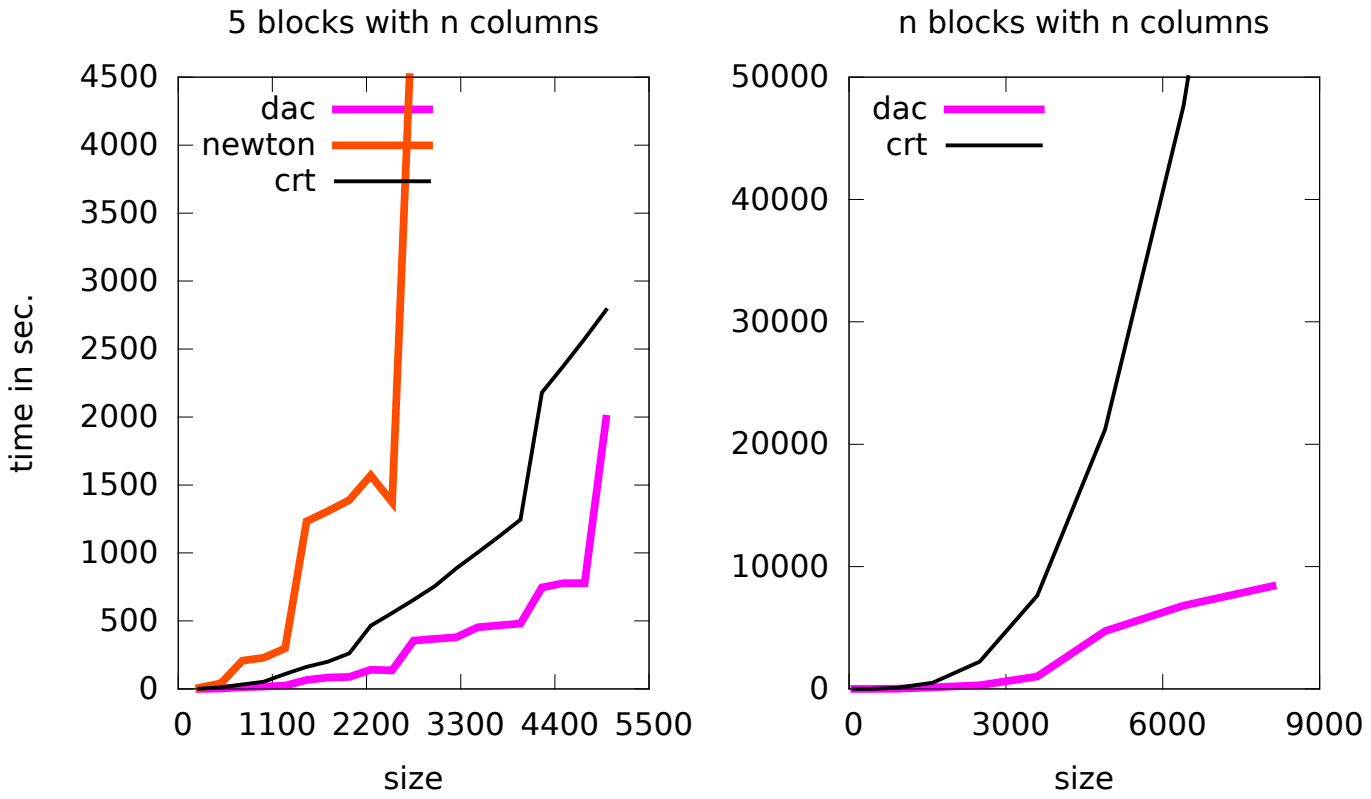


Figure. Pade-Hermite approximants over $\mathbb{Q}$ with degree bounds $\left(\underbrace{n,n,n,n,n}_{5\text{ times}}\right)$ and $\left(\underbrace{n,\ldots,n}_{n\text{ times}}\right)$

Bottleneck in DAC solving are matrix-vector products

Special case of algebraic approximants

Find $f_1, \ldots, f_s$ s.t. $f_1 + f_2\, r + f_3\, r^2 + \cdots + f_s\, r^{s-1} = 0 \bmod x^\delta$ and $\deg f_i < d$

Cost $\mathcal{C}_\mathsf{T}$ of matrix-vector product $\mathsf{T}\,\mathbf{r}$

1. Structured algorithms: $\alpha = s, n = \mathcal{O}(d\,s)$ $\qquad \mathcal{C}_\mathsf{T} = \mathcal{O}(\alpha\,\mathcal{M}(n)) = \tilde{\mathcal{O}}(s^2\,d)$
2. Bivariate modular composition: [Brent, Kung '78], [Nüsken, Ziegler '04]

   Compute $F(x, r(x)) \bmod x^\delta$ for $F(x, y) = \sum f_i\, y^{i-1}$ $\qquad \mathcal{C}_\mathsf{T} = \tilde{\mathcal{O}}(s^{(\omega+1)/2}\,d)$

Note: Same trick does not apply for Newton iteration, nor for chinese remaindering.

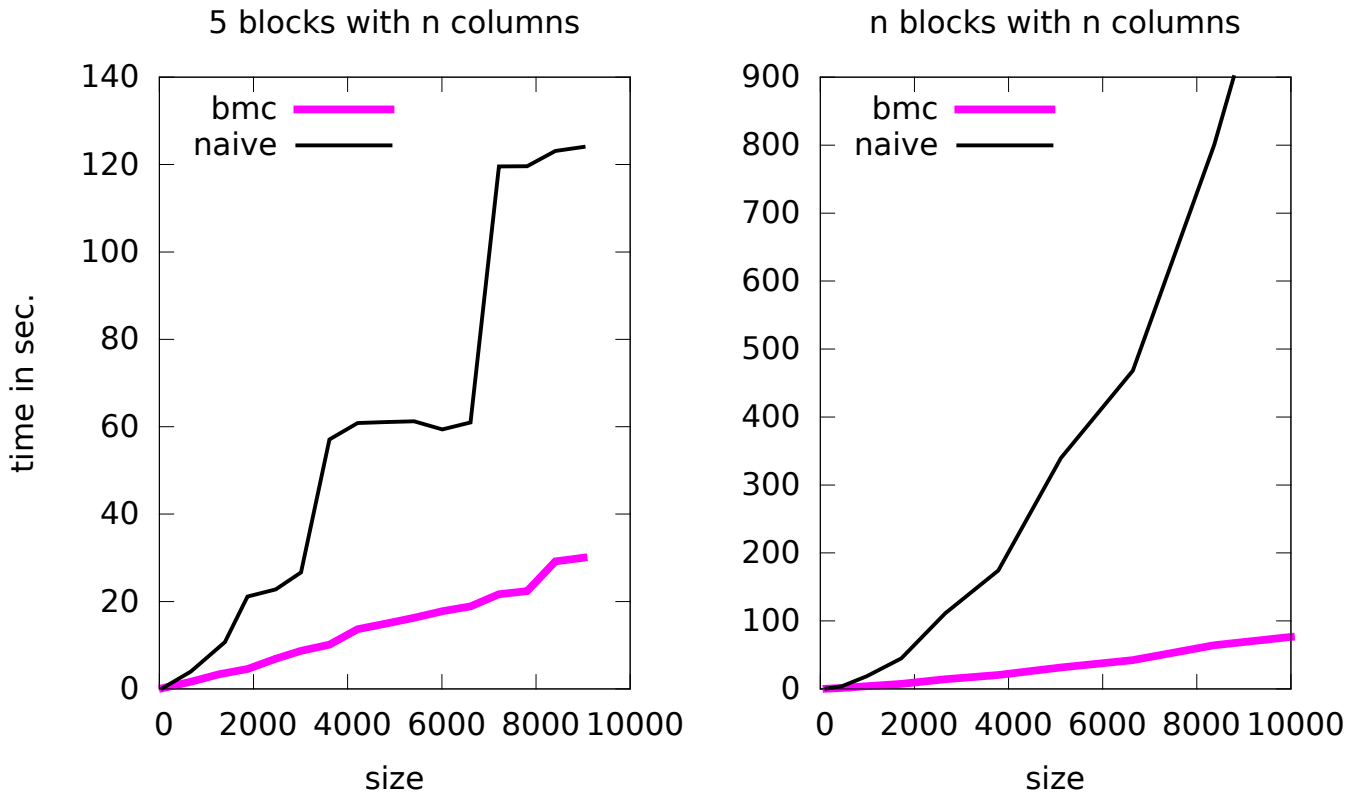


Figure. Algebraic approximants over $\mathbb{Q}$ with degree bounds $\left(\underbrace{n, n, n, n, n}_{5 \text{ times}}\right)$ and $\left(\underbrace{n, \ldots, n}_{n \text{ times}}\right)$

Conclusion

Summary:

- Implemented many different algorithms using efficient libraries
- Improvements over $\mathbb{F}_p$ with significant practical benefits:
  - Faster iterative algorithm
  - New class of Cauchy-like matrices with faster matrix-vector product
  - Simpler regularization process

- Improvements over $\mathbb{Q}$ with significant practical benefits:
  - DAC is most efficient in practice
  - Exploited the additional structure of the algebraic approximants

Perspective:

- Similar improvements over $\mathbb{Q}$ for the differential case $(r, r', r'', \ldots)$

  Bivariate modular composition in this context [Bostan, Schost '09]

Thank you for your attention ;-)