

Algorithms for Structured Linear Systems Solving and their Implementation

SEUNG GYU HYUN, ÉRIC SCHOST

University of Waterloo

ROMAIN LEBRETON

Université de Montpellier

Structured Matrix Days – ENS de Lyon

May 15th 2018

Hermite-Padé approximants

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

$$f_1 r_1 + \dots + f_s r_s = 0 \pmod{x^\delta} \quad + \text{ degree conditions}$$

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

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

Special case:

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

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

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

Hankel

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

Vandermonde

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

Cauchy

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

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

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 :

$$\alpha := \text{rank}(\nabla_{M,N}(A))$$

In this talk, $M, N \in \{D_u, Z, Z^t\}$ where $D_u = \text{Diag}(u_1, \dots, u_n)$, $Z_\varphi = \begin{bmatrix} 1 & & & \varphi \\ & \ddots & & \\ & & 1 & \\ & & & \ddots \end{bmatrix}$ and $Z = Z_0$

Informally, A is **structured** for $\nabla_{M,N}$ if $\alpha \ll n$

Examples:

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

There exists a more general theory with block companion M, N , that applies to Padé-Hermite generalization $f_1 R_{i,1} + \dots + f_s R_{i,s} = 0 \pmod{P_i}$.

[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 $\text{Spec}(M) \cap \text{Spec}(N) = \emptyset$.

Decompression – Cauchy

Inversibility. If $\{u_i\} \cap \{v_j\} = \text{Spec}(M) \cap \text{Spec}(N) = \emptyset$ 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. If M invertible, N nilpotent and $\nabla_{M,N}(A) = G \cdot H^t$, then $\nabla_{M,N}$ is invertible and

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

where $\text{Krylov}(M, G) = [M^{n-1}G \mid \dots \mid M^2G \mid MG \mid G]$.

Decompression – Vandermonde

Since $M = D_u$ is invertible and $N = Z$ is nilpotent, apply the theorem.

Decompression – Toeplitz, Hankel

Consider $\nabla_{Z_1, Z}$ instead of $\nabla_{Z, Z}$:

- $\nabla_{Z_1, Z}$ is invertible: Indeed Z_1 is invertible and Z is nilpotent
- $|\alpha_{\nabla_{Z_1, Z}} - \alpha_{\nabla_{Z, Z}}| \leq 1$ since $\nabla_{Z_1, Z}(A) - \nabla_{Z, Z}(A) = \underbrace{(Z_1 - Z)}_{\text{rank 1}} A$

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

1. Addition

$$\nabla_{M,N}(A + B) = \nabla_{M,N}(A) + \nabla_{M,N}(B) \quad \text{rank} \leq \alpha + \beta$$

2. Transpose

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

3. Inverse

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

4. Multiplication

$$\nabla_{M,P}(A B) = (M A - A N) B + A (N B - B P) = \nabla_{M,N}(A) B + A \nabla_{N,P}(B) \quad \text{rank} \leq \alpha + \beta$$

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

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

1. Addition

$$\nabla_{M,N}(A + B) = \nabla_{M,N}(A) + \nabla_{M,N}(B) \quad \text{rank} \leq \alpha + \beta$$

2. Transpose

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

3. Inverse

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

4. Multiplication

$$\nabla_{M,P}(A B) = (M A - A N) B + A (N B - B P) = \nabla_{M,N}(A) B + A \nabla_{N,P}(B) \quad \text{rank} \leq \alpha + \beta$$

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

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

1. Addition

$$\nabla_{M,N}(A + B) = \nabla_{M,N}(A) + \nabla_{M,N}(B) \quad \text{rank} \leq \alpha + \beta$$

2. Transpose

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

3. Inverse

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

4. Multiplication

$$\nabla_{M,P}(A B) = (M A - A N) B + A (N B - B P) = \nabla_{M,N}(A) B + A \nabla_{N,P}(B) \quad \text{rank} \leq \alpha + \beta$$

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

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

1. Addition

$$\nabla_{M,N}(A + B) = \nabla_{M,N}(A) + \nabla_{M,N}(B) \quad \text{rank} \leq \alpha + \beta$$

2. Transpose

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

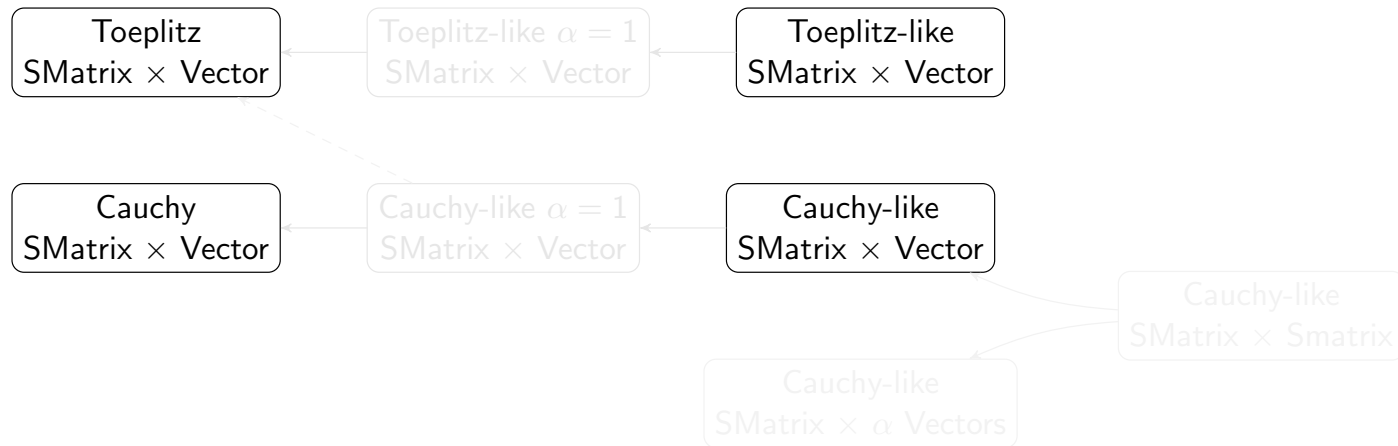
3. Inverse

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

4. Multiplication

$$\nabla_{M,P}(A B) = (M A - A N) B + A (N B - B P) = \nabla_{M,N}(A) B + A \nabla_{N,P}(B) \quad \text{rank} \leq \alpha + \beta$$

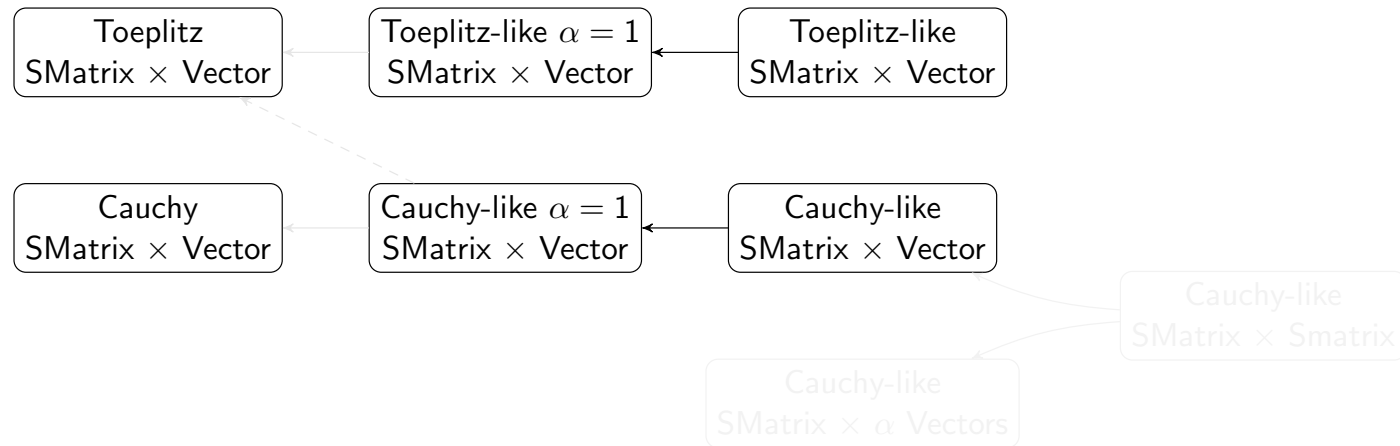
Note: Requires (structured matrix) \times (dense matrix) product



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 ?

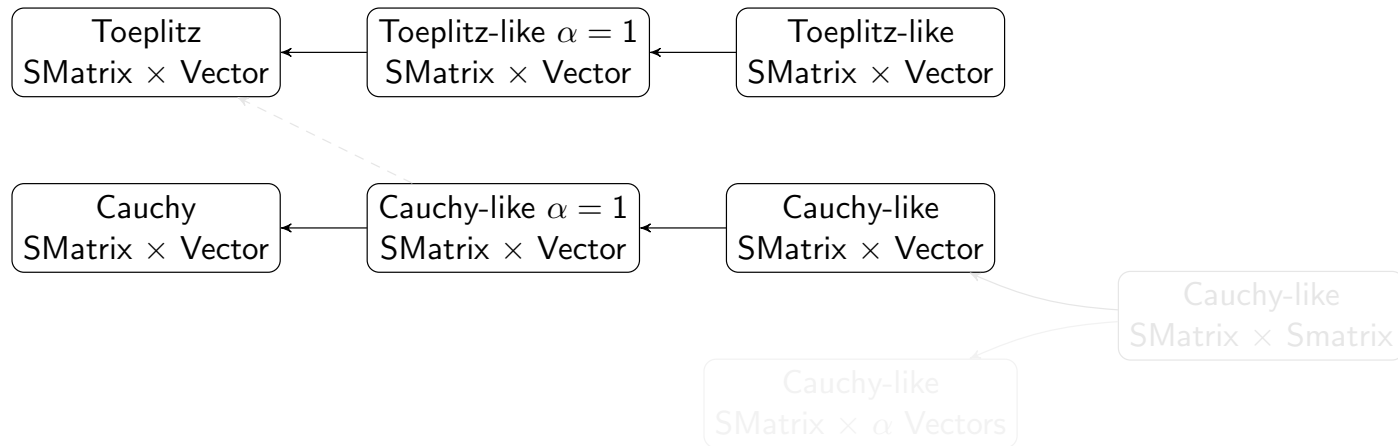


Remark. A is the sum of displacement rank 1 matrices

Decompose $G \cdot H^t = \underbrace{G_1 \cdot H_1^t}_{\text{rank 1}} + \dots + \underbrace{G_\alpha \cdot H_\alpha^t}_{\text{rank 1}}$ where G_i, H_i are the i -th column

Then A is the sum of displacement rank 1 matrices:

$$A = \nabla^{-1}(G \cdot H^t) = \underbrace{\nabla^{-1}(G_1 \cdot H_1^t)}_{\text{displacement rank 1}} + \dots + \underbrace{\nabla^{-1}(G_\alpha \cdot H_\alpha^t)}_{\text{displacement rank 1}}$$



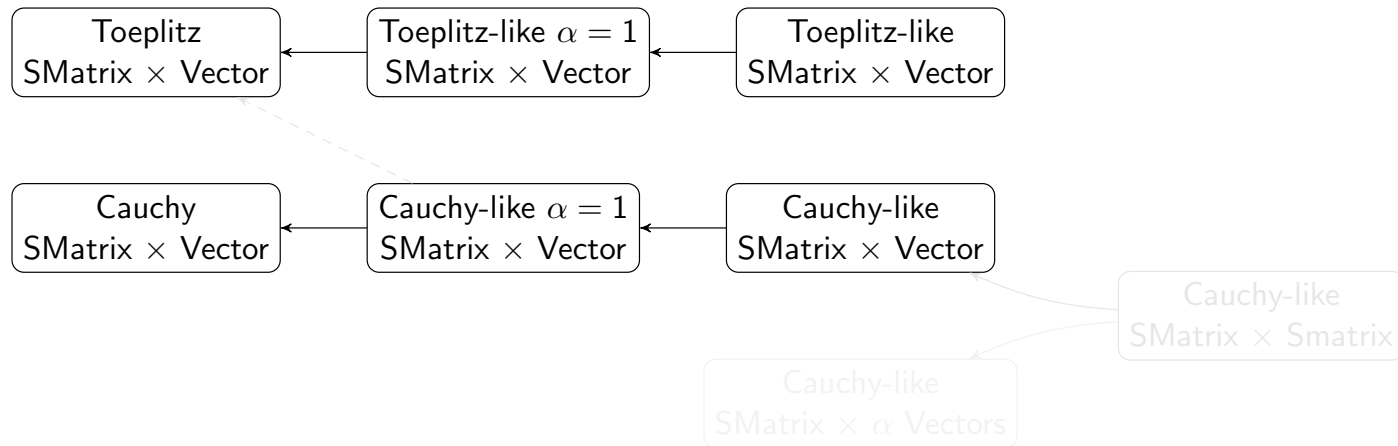
Toeplitz-like $\alpha = 1$

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

$$A = Z_1^{-1} \cdot \text{Krylov}(Z_1^{-1}, g) \cdot \text{Krylov}(Z, h)^t = \underbrace{\begin{bmatrix} g_1 & g_n & \ddots & g_2 \\ g_2 & g_1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & g_n \\ g_n & g_{n-1} & \ddots & g_1 \end{bmatrix}}_{\text{Toeplitz}} \cdot \underbrace{\begin{bmatrix} h_n & 0 & \dots & 0 \\ \vdots & h_n & \ddots & \vdots \\ h_2 & \ddots & \ddots & 0 \\ h_1 & h_2 & \dots & h_n \end{bmatrix}}_{\text{Toeplitz}}$$

So reduction to Toeplitz matrix-vector product.

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



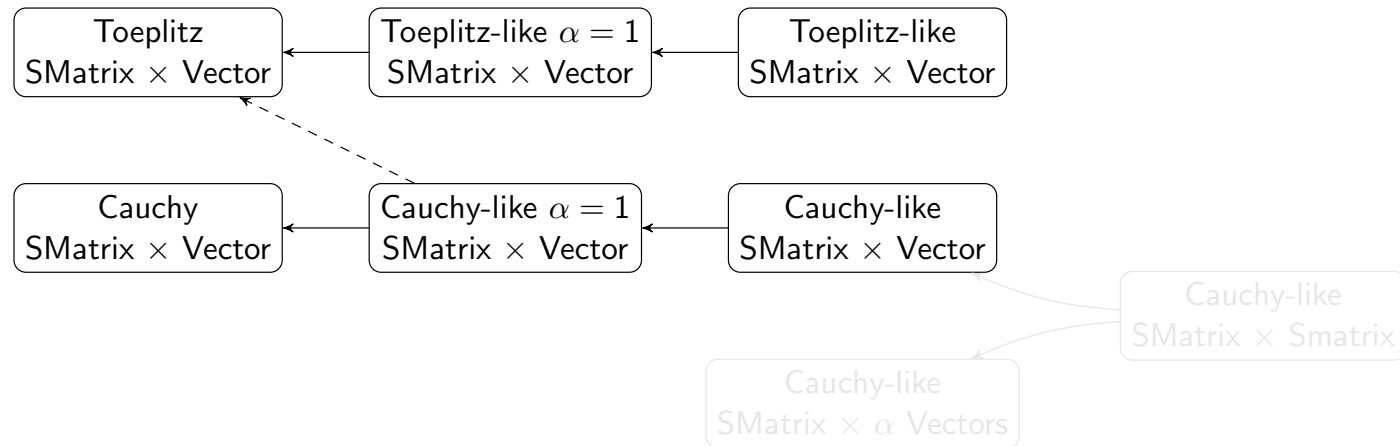
Cauchy-like $\alpha = 1$

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

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

So reduction to Cauchy matrix-vector product.

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



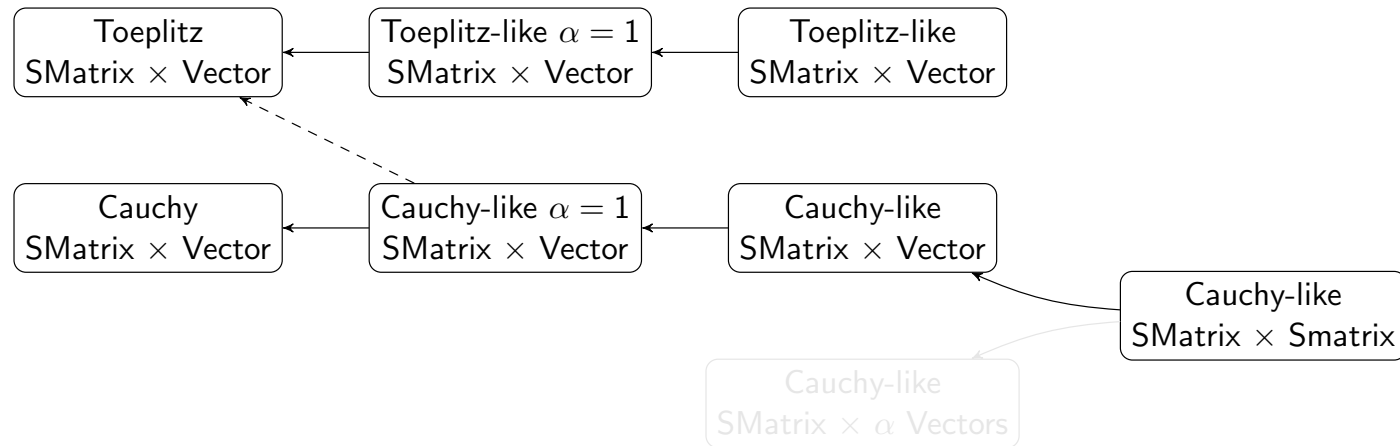
Cauchy-like $\alpha = 1$ – Trick for geometric sequences with same ratio

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

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

Reduce to Toeplitz matrix vector-product

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



Consequence to product of structured matrices

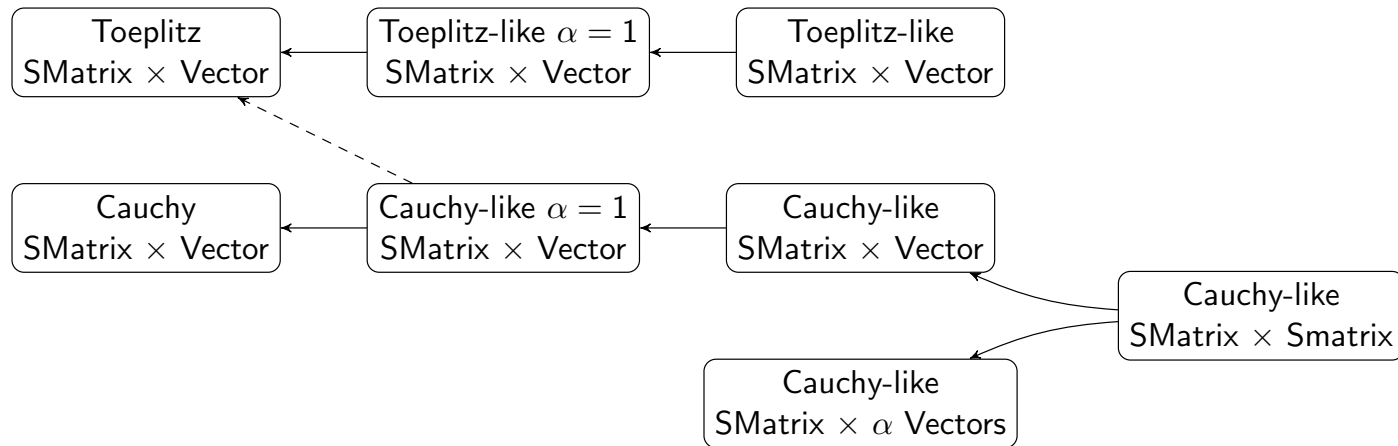
If $\nabla_{M,N}(A) = G \cdot H^t$, $\nabla_{N,P}(B) = Y \cdot Z^t$ with $G, H, Y, Z \in \mathbb{K}^{n \times \alpha}$ then

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

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

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

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



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

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

$A Y$ 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 $\tilde{O}(\alpha^{\omega-1} n)$

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

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

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

[CARDINAL '99], [JEANNEROD, MOUILLERON '10]

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

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

Facts.

1. $S^{(0)} = A, S^{(n)} = A^{-1}$

2. We have

$$\text{PartialInversion}_i = \text{PartialInversion}_1 \circ \dots \circ \text{PartialInversion}_1$$

which implies that

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

How can we compute PartialInversion with structured matrices ?

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

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} .

Algorithm PartialInversion_i-Structured

Input: $0 \leq i \leq n$

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

G'_0, H'_0, u_0, v_0 s.t. $\nabla_{v_0, u_0}(A_{00}^{-1}) = G'_0 \cdot H_0'^t$ where $A_{00} \in \mathbb{K}^{i \times i}$

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

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

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

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** $n \leq \text{threshold}$ **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:

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

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** $n \leq \text{threshold}$ **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

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** $n \leq \text{threshold}$ **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: $0 \leq \beta \leq \alpha$ 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: G, H, u, v such that $\nabla_{u,v}(A) = G \cdot H^t$ and step size β ($1 \leq \beta \leq \alpha$)

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

1. **for** ($i = 0; i < n; i = i + \beta$)

a. $S^{(i\beta)} \xrightarrow{\text{PartialInversion}_\beta} S^{((i+1)\beta)}$

 Compute generators Y, Z of $S^{((i+1)\beta)}$ using PartialInversion-Dense

2. **return** Y, Z

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

Total cost: $n / \alpha \times \mathcal{O}(\alpha^{\omega-1} n) = \mathcal{O}(\alpha^{\omega-2} n^2)$

Algorithm Struct-inverse-iter

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

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

1. **for** ($i = 0; i < n; i = i + \beta$)

a. $S^{(i\beta)} \xrightarrow{\text{PartialInversion}_\beta} S^{((i+1)\beta)}$

 Compute generators Y, Z of $S^{((i+1)\beta)}$ 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

[PAN '90] 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.

Transformation

If T is Toeplitz-like of displacement rank α then

$C = V_u T W_v$ is Cauchy-like with displacement rank $\leq \alpha + 2$

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

$$A' = D_x C_{a,u} A C_{v,b} D_y$$

[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 $\alpha \geq 10$

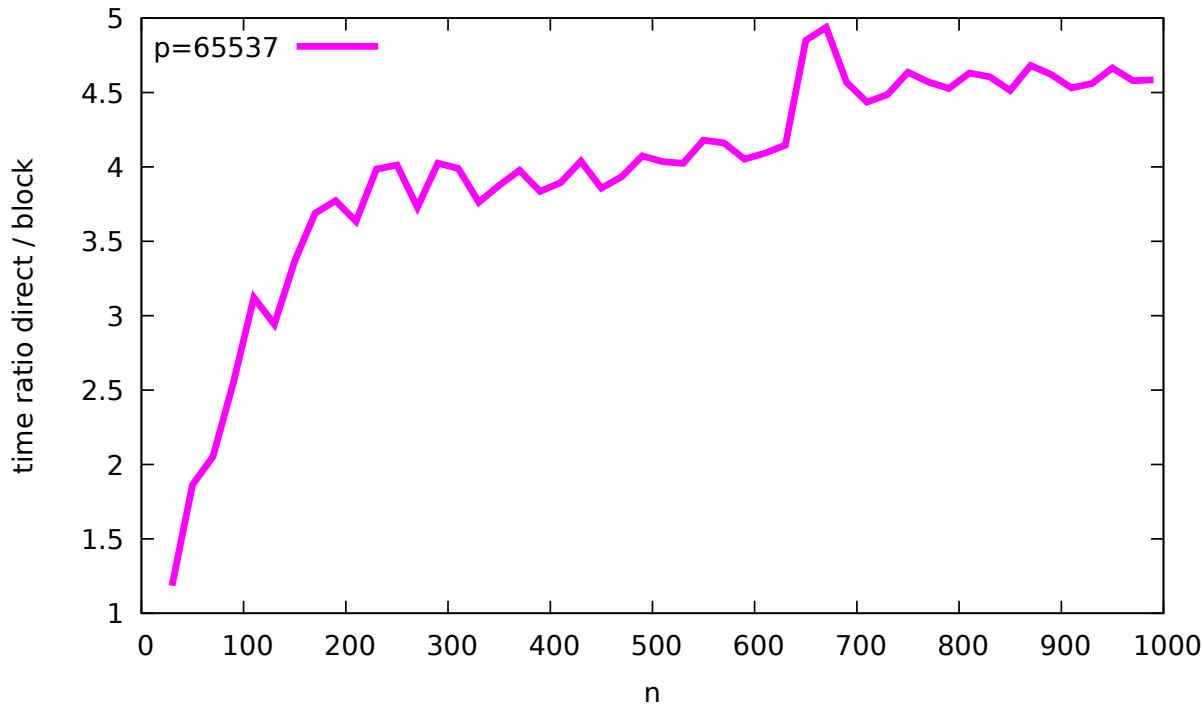
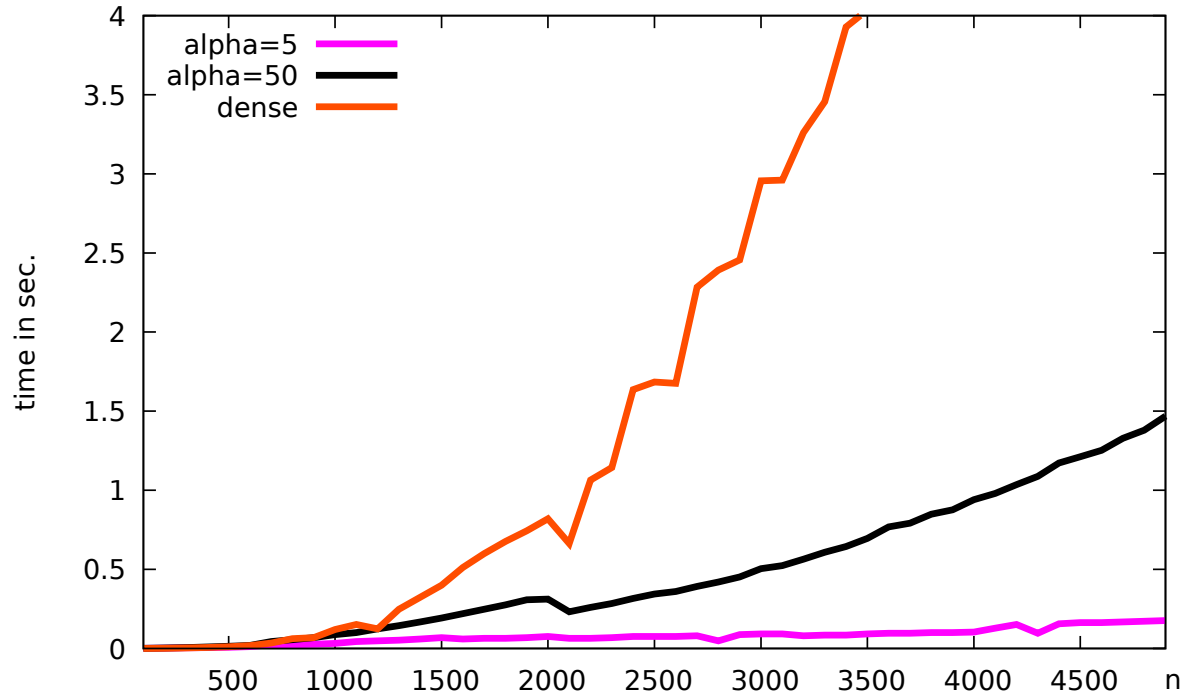


Figure. Time ratio “ $(\alpha n^2) / (\alpha^{\omega-2} n^2)$ ” for $\alpha = 30$

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 $\alpha \leq 0.2n$



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

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

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

Reduce the problem:

1. Cauchy matrices:

$$C = V_u T W_v \quad \text{and} \quad Cy = 0 \Leftrightarrow T(W_v y) = 0$$

2. Kernel to linear system solving:

$$\ker C = \left\{ \left[\begin{array}{c} C_{00}^{-1} C_{01} r \\ -r \end{array} \right]_r \right\} \quad \text{where } C_{00} \text{ maximal minor}$$

3. Modular techniques:

Solve modulo N , then apply rational reconstruction

Current Goal – Modular Cauchy system solving

Given C invertible Cauchy-like matrix, solve

$$Cx = b \pmod{N}$$

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, \dots, p^t
- Bit complexity: $\mathcal{O}(\alpha^{\omega-1} \mathcal{M}(n) \mathcal{I}(t))$

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: $\mathcal{O}(\alpha \mathcal{M}(n) \mathcal{I}(t) \log(t))$

Chinese Remainder Theorem

$$N = p_1 \cdots p_t$$

- Solve $Cx = b$ modulo t primes p_1, \dots, p_t of the same magnitude
- Bit complexity: $\mathcal{O}(\alpha^{\omega-1} \mathcal{M}(n) \log(n) t + n \mathcal{I}(t) \log(t))$

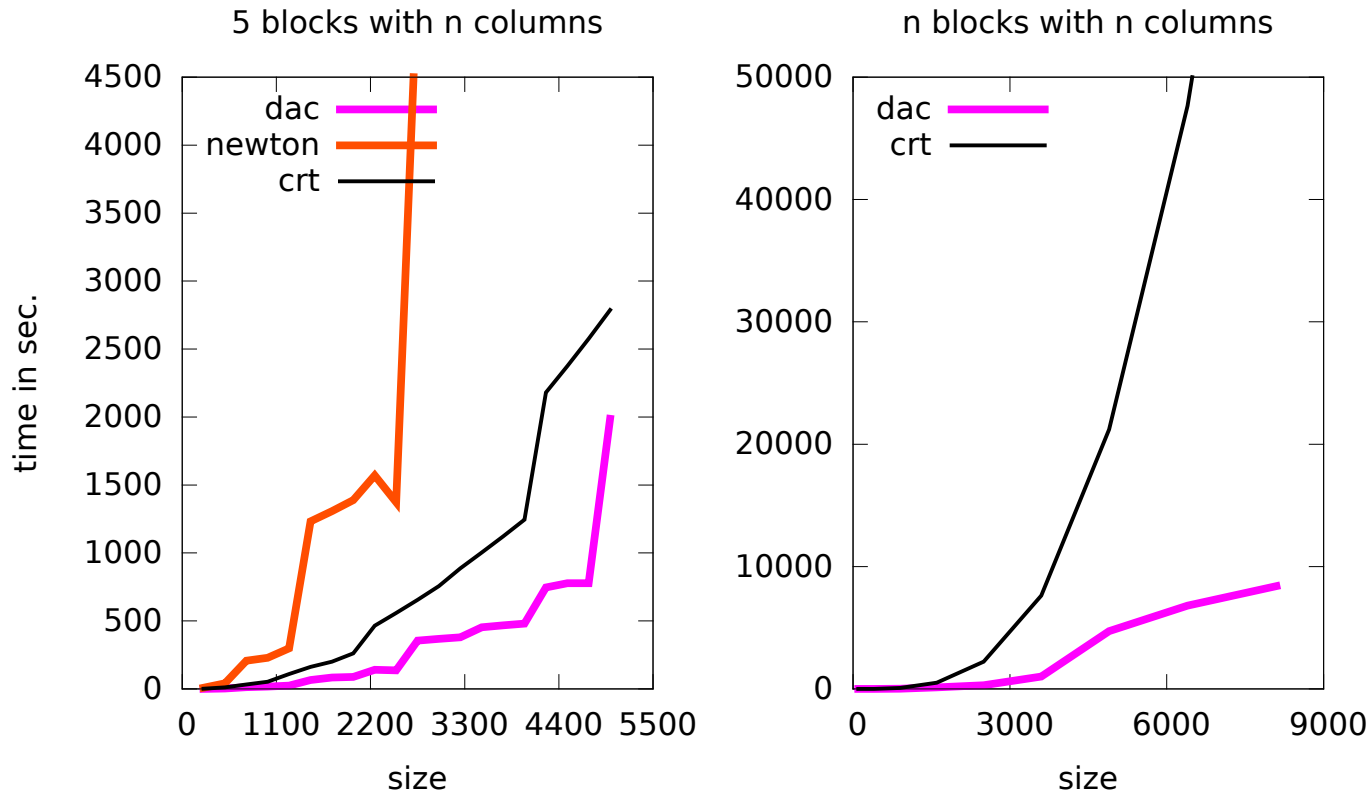


Figure. Pade-Hermite approximants over \mathbb{Q} with degree bounds $\underbrace{(n, n, n, n, n)}_{5 \text{ times}}$ and $\underbrace{(n, \dots, n)}_{n \text{ times}}$

Bottleneck in DAC solving are matrix-vector products

Special case of algebraic approximants

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

Cost \mathcal{C}_T of matrix-vector product $T r$

1. Structured algorithms: $\alpha = s, n = \mathcal{O}(d s)$ $\mathcal{C}_T = \mathcal{O}(\alpha \mathcal{M}(n)) = \tilde{\mathcal{O}}(s^2 d)$

2. Bivariate modular composition: [BRENT, KUNG '78], [NÜSKEN, ZIEGLER '04]

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

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

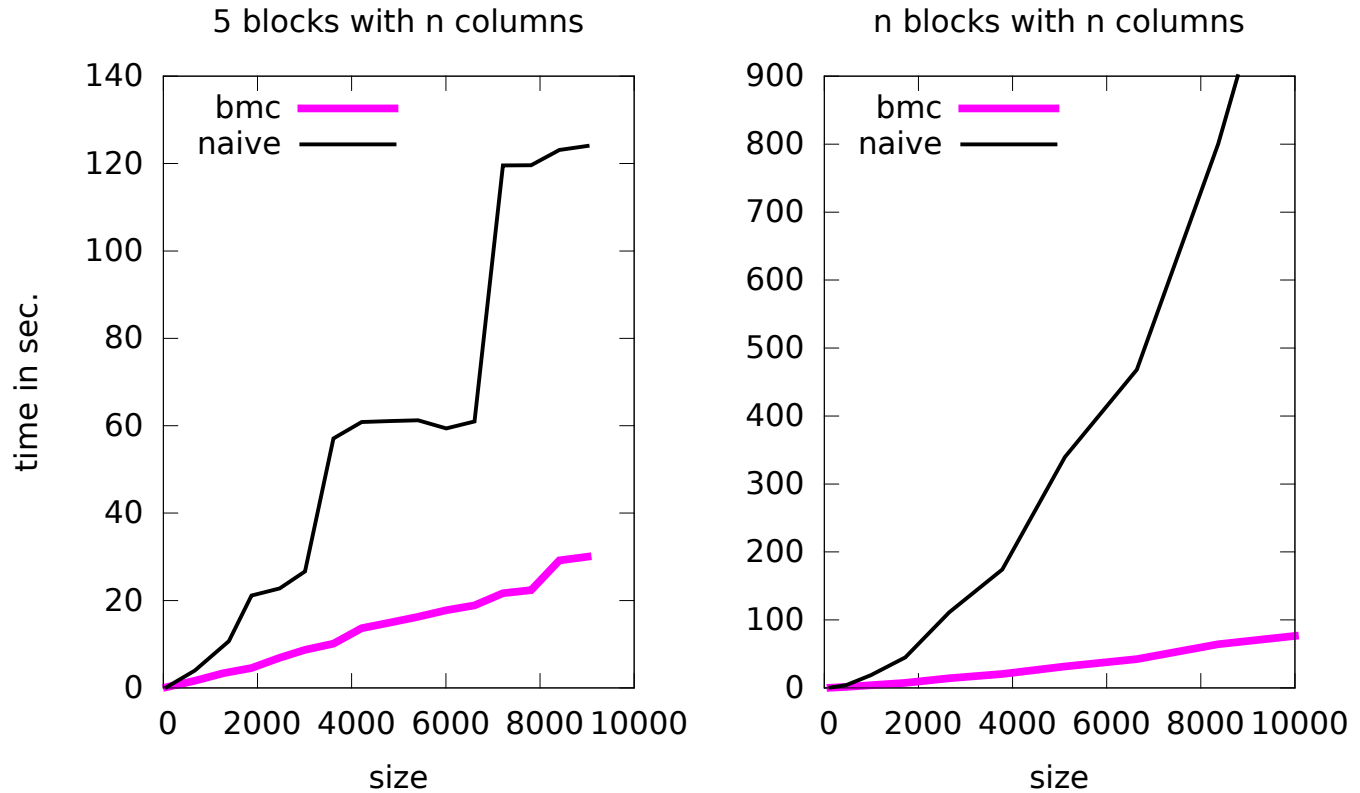


Figure. Algebraic approximants over \mathbb{Q} with degree bounds $\underbrace{(n, n, n, n, n)}_{5 \text{ times}}$ and $\underbrace{(n, \dots, n)}_{n \text{ times}}$

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'', \dots)

Bivariate modular composition in this context [BOSTAN, SCHOST '09]

Thank you for your attention ;-)