Algorithms for Structured Linear Systems Solving and their Implementation

S. G. Hyun, Éric Schost

University of Waterloo

Romain Lebreton

Université de Montpellier

Séminaire Aromath – Inria Sophia Antipolis

April 25th 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

Motivation

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

1. **Structured matrices**
   a. Toeplitz, Hankel, Vandermonde, Cauchy
   b. Displacement operators
   c. Pan's motto: Compress, operate, decompress
   d. Structured matrices multiplication

2. Structured matrices inversion
   a. Cauchy-like inversion: iterative & divide-and-conquer algorithm
   b. Toeplitz-like inversion

3. Implementations

Toeplitz matrices

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

Remarks: Described by $O(n)$ elements, Hermite-Padé matrix was block Toeplitz

Fast algorithm for linear system solving – Arithmetic cost ($\mathbb{K} = \mathbb{F}_q$)

1. $\mathcal{O}(n^2)$ [Levinson '47, Trench '64]
2. $\tilde{\mathcal{O}}(n)$ [Brent, Gustavson, Yun '80]

Fast matrix vector product $\mathsf{T} \cdot \mathsf{b}$: [Morf '80], [Bitmead, Anderson '80]

$$\mathsf{T} \cdot \mathsf{b} = \begin{bmatrix} t_0 & \cdots & t_{-(n-1)} \\ \vdots & \ddots & \vdots \\ t_{n-1} & \cdots & t_0 \end{bmatrix} \begin{bmatrix} b_0 \\ \vdots \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} b_0\, t_0 + b_1\, t_{-1} + \cdots + b_{n-1}\, t_{-(n-1)} \\ \vdots \\ b_0\, t_{n-1} + b_1\, t_{n-2} + \cdots + b_{n-1}\, t_0 \end{bmatrix}$$

computed using $P_{\mathsf{T}}(x)\, P_{\mathsf{b}}(x)$ in time $\sim \mathcal{M}(n)$ where $P_{\mathsf{T}}(x) = \sum t_{-(n-1)+i}\, x^i$.

Hankel, Vandermonde and Cauchy matrices

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

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

Structured matrix - vector product

Hankel

Polynomial multiplication in $\sim\mathcal{M}(n)$

$$\begin{bmatrix} h_0 & \cdots & h_{n-1} \\ \vdots & \iddots & \vdots \\ h_{n-1} & \cdots & h_{2n-2} \end{bmatrix} \begin{bmatrix} b_{n-1} \\ \vdots \\ b_0 \end{bmatrix} = \begin{bmatrix} b_{n-1}\,h_0 + \cdots + b_0\,h_{n-1} \\ \vdots \\ b_{n-1}\,h_{n-1} + \cdots + b_0\,h_{2n-2} \end{bmatrix}$$

Vandermonde

Multipoint evaluation of $\sum b_i X^i$ at $u_1, ..., u_n$ in $\mathcal{O}(\mathcal{M}(n)\log(n))$

$$\begin{bmatrix} 1 & u_1 & \cdots & u_1^{n-1} \\ 1 & \vdots & & \vdots \\ 1 & u_n & \cdots & u_n^{n-1} \end{bmatrix} \begin{bmatrix} b_0 \\ \vdots \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} b_0 + b_1\,u_1 + \cdots + b_{n-1}\,u_1^{n-1} \\ \vdots \\ b_0 + b_1\,u_n + \cdots + b_{n-1}\,u_n^{n-1} \end{bmatrix}$$

Cauchy

Multipoint evaluation of $\sum_j \frac{b_j}{X - v_j}$ at $u_1, ..., u_n$ in $\mathcal{O}(\mathcal{M}(n)\log(n))$

$$\begin{bmatrix} \frac{1}{u_1 - v_1} & \cdots & \frac{1}{u_1 - v_n} \\ \vdots & & \vdots \\ \frac{1}{u_n - v_1} & \cdots & \frac{1}{u_n - v_n} \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} \frac{b_1}{u_1 - v_1} + \cdots + \frac{b_n}{u_1 - v_n} \\ \vdots \\ \frac{b_1}{u_n - v_1} + \cdots + \frac{b_n}{u_n - v_n} \end{bmatrix}$$

Note: Cost drops from $\mathcal{O}(\mathcal{M}(n)\log(n))$ to $\mathcal{O}(\mathcal{M}(n))$ if (u_i) and (v_j) are geometric sequences.

What is a structured matrix ?

Toeplitz, Hankel, Cauchy, Vandermonde frameworks have similarities

Can we find one unified framework ?

What we could want from this framework:

1. Incorporate block Toeplitz
2. Requirements to have a divide-and-conquer algorithm for inversion like [Strassen '69]
   a. Stability by sum and product
   b. Stability by inverse
   c. Quasi-linear product of structured matrices

Strassen formula: Set $K = A_{11} - A_{10} A_{00}^{-1} A_{01}$,

$$A^{-1} = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix}^{-1} = \begin{bmatrix} \mathrm{Id} & -A_{00}^{-1} A_{01} \\ 0 & \mathrm{Id} \end{bmatrix} \begin{bmatrix} A_{00}^{-1} & 0 \\ 0 & K^{-1} \end{bmatrix} \begin{bmatrix} \mathrm{Id} & 0 \\ -A_{10} A_{00}^{-1} & \mathrm{Id} \end{bmatrix}$$

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

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

So 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]

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

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

Example – Hankel

Consider $\nabla_{\mathsf{Z},\mathsf{Z}^t}(\mathsf{H}) = \mathsf{Z}\,\mathsf{H} - \mathsf{H}\,\mathsf{Z}^t = (\mathsf{H}\downarrow) - (\mathsf{H}\rightarrow)$, then

$$\nabla_{\mathsf{Z},\mathsf{Z}^t}\left(\begin{bmatrix} h_0 & \cdots & h_{n-1} \\ \vdots & \iddots & \vdots \\ h_{n-1} & \cdots & h_{2n-2} \end{bmatrix}\right) = \begin{bmatrix} 0 & h_1 & \cdots & h_{n-1} \\ h_1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ h_{n-1} & 0 & \cdots & 0 \end{bmatrix}$$

So H has $\nabla_{\mathsf{Z},\mathsf{Z}^t}$-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: $\operatorname{rank}(\nabla(A))$

Example – Vandermonde

Consider $\nabla_{D_u,Z}(V) = D_u V - V Z$, $D_u = \operatorname{Diag}(u_1, ..., u_n)$, then

$$\nabla_{D_u,Z}\left(\begin{bmatrix} 1 & u_1 & \cdots & u_1^{n-1} \\ 1 & \vdots & & \vdots \\ 1 & u_n & \cdots & u_n^{n-1} \end{bmatrix}\right) = \begin{bmatrix} 0 & \cdots & 0 & u_1^n \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & u_n^n \end{bmatrix}$$

So V has $\nabla_{D_u,Z}$-displacement rank $\alpha = 1$.

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: $\operatorname{rank}(\nabla(A))$

Example – Cauchy

Consider $\nabla_{D_u,D_v}(C) = D_u C - C D_v$, then

$$\nabla_{D_u,D_v}\left(\begin{bmatrix} \frac{1}{u_1-v_1} & \cdots & \frac{1}{u_1-v_n} \\ \vdots & & \vdots \\ \frac{1}{u_n-v_1} & \cdots & \frac{1}{u_n-v_n} \end{bmatrix}\right) = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

So C has ∇_{D_u,D_v}-displacement rank $\alpha = 1$.

Historical note:

Initially, the displacement rank approach was intended for more restricted use.

[KKM '79] introduced the concept to a measure of how 'close' to Toeplitz a given matrix is.

Displacement operators

Two main families of displacement operators

1. **Sylvester displacement operator** $\nabla_{M,N}(A) = MA - AN$
2. Stein displacement operator $\Delta_{M,N}(A) = A - MAN$

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

There exists 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]

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

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

Pan's motto: Compress, Operate, Decompress

1. Compression: Store A with $\ll n^2$ elements
2. Operate: Sum, product of structured matrices, matrix-vector product, inversion...
3. Decompression: Recover A

Pan's motto: Compress, Decompress

Compression

If $\nabla(\mathsf{A})$ has rank α, A is stored as $\nabla(\mathsf{A}) = \mathsf{G} \cdot \mathsf{H}^t$ for $\mathsf{G}, \mathsf{H} \in \mathbb{K}^{n \times \alpha}$

Size: $2\,\alpha\,n \ll n^2$

Cost: $\mathcal{O}(\alpha^{\omega-2}\,n^2)$ Echelonize $\nabla(\mathsf{A})$ (e.g. PLUQ)

Decompression – Inversibility

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

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

Decompression – Reconstruction formulae – Cauchy

If $\mathsf{A} = (a_{i,j})_{i,j}$ then $\nabla_{\mathsf{D_u},\mathsf{D_v}}(\mathsf{A}) = (a_{i,j}\,(u_i - v_j))_{i,j}$

Cost: $\mathcal{O}(\alpha^{\omega-2}\,n^2)$ Compute $\mathsf{G} \cdot \mathsf{H}^t$

Decompression – Reconstruction formulae – Vandermonde

If M invertible and N nilpotent :

$$\begin{aligned}
M^n A &= M^{n-1}(MA - AN) + M^{n-1}AN \\
&= M^{n-1}(MA - AN) + M^{n-2}(MA - AN)N + M^{n-2}AN^2 \\
&= \sum_{i=1}^{n} M^{i-1}(MA - AN)N^{n-i} + \underbrace{AN^n}_{0} \\
&= \sum_{i=1}^{n} M^{i-1}GH^tN^{n-i} \\
&= \left[\, G \,\middle|\, MG \,\middle|\, \cdots \,\middle|\, M^{n-1}G \,\right] \cdot \left[\, (N^t)^{n-1}H \,\middle|\, \cdots \,\middle|\, N^tH \,\middle|\, H \,\right]^t
\end{aligned}$$

Set $\mathrm{Krylov}(M, G) = \left[\, M^{n-1}G \,\middle|\, \cdots \,\middle|\, G \,\right]$, you get

$$\begin{aligned}
A &= M^{-1} \cdot \left[\, M^{-(n-1)}G \,\middle|\, \cdots \,\middle|\, G \,\right] \cdot \left[\, (N^t)^{n-1}H \,\middle|\, \cdots \,\middle|\, N^tH \,\middle|\, H \,\right]^t \\
&= M^{-1} \cdot \mathrm{Krylov}(M^{-1}, G) \cdot \mathrm{Krylov}(N^t, H)^t
\end{aligned}$$

Can be applied to Vandermonde-like since $M = D_u$ and $N = Z$

Decompression – Reconstruction formulae – Toeplitz, Hankel

Tricks for $\nabla_{Z,Z}$:

- Consider $\nabla_{Z_1,Z}$: $Z_1 = Z + e_{1,n}$

  $\nabla_{Z_1,Z}$ is invertible since Z_1 is invertible and Z nilpotent

  $|\alpha_{\nabla_{Z_1,Z}} - \alpha_{\nabla_{Z,Z}}| \leqslant 1$ since $\nabla_{Z_1,Z}(A) - \nabla_{Z,Z}(A) = \underbrace{(Z_1 - Z)}_{e_{1,n}} A$

- Or consider **Stein operator:** $\Delta_{Z^t,Z}(A) = A - Z^t A Z$, who is invertible.

Stability of structured matrices

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 structured matrices

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 structured matrices

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 structured matrices

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

Structured matrix – vector product

Do we still have quasi-linear structured matrix – vector multiplication ?

Remark: A is the sum of α structured matrices of displacement rank 1

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

where $\mathsf{G}_i, \mathsf{H}_i$ is the i-th column

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 Cauchy-like matrix-vector product reduces to Cauchy matrix-vector product.

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

Cauchy-like

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

Structured matrix – vector product

Do we still have quasi-linear structured matrix – vector multiplication ?

Remark: A is the sum of α structured matrices of displacement rank 1

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

where $\mathsf{G}_i, \mathsf{H}_i$ is the i-th column

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

Suppose $u_i = \tau^{i-1}\, u_1$, $v_i = \tau^{i-1}\, v_1$.

Classical approach. $\sum_j \frac{b_j}{X - v_j}$ via interpolation on geometric sequence $\sim 2\,\mathcal{M}(n)$

\+ Evaluation on geometric sequence $u_1, \ldots, u_n$ $\sim \mathcal{M}(n)$

Approach with trick.

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

Reduce to Toeplitz matrix vector-product $\sim \mathcal{M}(n)$ (vs. $\sim 3\,\mathcal{M}(n)$)

Toeplitz-like $\alpha = 1$

When $\operatorname{rank} \nabla_{Z_1,Z}(A) = 1$ and $\nabla_{Z_1,Z}(A) = g \cdot h^t$,

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

So reduction to Toeplitz matrix-vector product.

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

Toeplitz-like

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

Same for Hankel-like, Vandermonde-like.

Structured matrix – dense matrix product

Structured matrix – dense matrix product

Algorithm 1. Decompose into many structured matrix – vector product
Each cost $\tilde{\mathcal{O}}(\alpha n)$

Algorithm 2. [Bostan, Jeannerod, Schost '08], [Bostan, Jeannerod, Mouilleron, Schost '17]
Decompose into many structured matrix – α vectors product
Each cost $\tilde{\mathcal{O}}(\alpha^{\omega-1} n)$ (instead of $\tilde{\mathcal{O}}(\alpha^2 n)$ for Algo 1)

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

Cost: $\mathcal{O}(\alpha^{\omega-1} M(n))$ or $\mathcal{O}(\alpha^{\omega-1} M(n) \log(n))$

Outline

1. Structured matrices
    a. Toeplitz, Hankel, Vandermonde, Cauchy
    b. Displacement operators
    c. Pan's motto: Compress, operate, decompress
    d. Structured matrices multiplication

2. **Structured matrices inversion**
    a. Cauchy-like inversion: iterative & divide-and-conquer algorithm
    b. Toeplitz-like inversion

3. Implementations

Intermediate inverse matrices $S^{(i)}$ [Cardinal '99], [Jeannerod, Mouilleron '10]

For $0 \leqslant i \leqslant n$, 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

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

Note:

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

$A_{11} - A_{10}A_{00}^{-1}A_{01}$ is the Schur complement

Where do these intermediate inverse matrices come from ?

Intermediate inverse matrices $S^{(i)}$ [Cardinal '99], [Jeannerod, Mouilleron '10]

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

Where do these intermediate inverse matrices come from ?

From $R^{(0)} = \begin{bmatrix} A \\ \mathrm{Id} \end{bmatrix}$, perform column elimination to transform A to Id, you get $R^{(n)} = \begin{bmatrix} \mathrm{Id} \\ A^{-1} \end{bmatrix}$.

Partial column elimination:

ColRed_i : Column reduction to transform $\begin{bmatrix} A_{00} & A_{01} \end{bmatrix}$ to $\begin{bmatrix} \mathrm{Id}_i & 0 \end{bmatrix}$ using A_{00} as pivot

$$R^{(i)} := \mathrm{ColRed}_i\left(\begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \\ \mathrm{Id}_i & 0 \\ 0 & \mathrm{Id}_{n-i} \end{bmatrix}\right) = \left[\begin{array}{c|c} \mathrm{Id}_i & 0 \\ \hline A_{10}A_{00}^{-1} & A_{11} - A_{10}A_{00}^{-1}A_{01} \\ \hline A_{00}^{-1} & -A_{00}^{-1}A_{01} \\ \hline 0 & \mathrm{Id}_{n-i} \end{array}\right].$$

$S^{(i)}$ appears in $R^{(i)}$.

Shift columns so that we always reduce top left corner $\rightsquigarrow \mathrm{ShColRed}_i$

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

Intermediate inverse matrices $\mathsf{S}^{(i)}$ [Cardinal '99], [Jeannerod, Mouilleron '10]

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

Iterative construction:

$$S^{(0)} \xrightarrow{\text{ShColRed}_1} S^{(1)} \xrightarrow{\text{ShColRed}_1} S^{(2)} \longrightarrow \cdots \xrightarrow{\text{ShColRed}_1} S^{(i)} \longrightarrow \cdots \xrightarrow{\text{ShColRed}_1} S^{(n)}$$

and

$$\text{ShColRed}_j(S^{(i)}) = (\text{ShColRed}_1 \circ \cdots \circ \text{ShColRed}_1)(S^{(i)}) = S^{(i+j)}$$

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

How to compute $\mathrm{ShColRed}_i$ with structured matrices ?

$$\mathrm{ShColRed}_i\left(\begin{bmatrix} \mathsf{A}_{00} & \mathsf{A}_{01} \\ \mathsf{A}_{10} & \mathsf{A}_{11} \end{bmatrix}\right) = \begin{bmatrix} \mathsf{A}_{11} - \mathsf{A}_{10}\mathsf{A}_{00}^{-1}\mathsf{A}_{01} & \mathsf{A}_{10}\mathsf{A}_{00}^{-1} \\ -\mathsf{A}_{00}^{-1}\mathsf{A}_{01} & \mathsf{A}_{00}^{-1} \end{bmatrix}$$

Cauchy magic – Submatrices

If $\nabla_{\mathsf{u},\mathsf{v}}(\mathsf{A}) = \mathsf{G}\cdot\mathsf{H}^t$ (rank α) then $\nabla_{\mathsf{u}_i,\mathsf{v}_j}(\mathsf{A}_{ij}) = \mathsf{G}_i\cdot\mathsf{H}_j^t$ (rank α) where

$\mathsf{u} = [\mathsf{u}_0, \mathsf{u}_1]$, $\mathsf{v} = [\mathsf{v}_0, \mathsf{v}_1]$, $\mathsf{G} = \begin{bmatrix} \mathsf{G}_0 \\ \mathsf{G}_1 \end{bmatrix}$, $\mathsf{H} = \begin{bmatrix} \mathsf{H}_0 \\ \mathsf{H}_1 \end{bmatrix}$ and $\mathsf{u}_0, \mathsf{v}_0 \in \mathbb{K}^{1\times i}$ and $\mathsf{G}_0, \mathsf{H}_0 \in \mathbb{K}^{i\times\alpha}$.

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

$$\nabla_{[\mathsf{u}_1,\mathsf{v}_0],[\mathsf{v}_1,\mathsf{u}_0]}(\mathsf{S}^{(i)}) = \begin{bmatrix} -\mathsf{A}_{10}\mathsf{G}_0' + \mathsf{G}_1 \\ \mathsf{G}_0' \end{bmatrix} \cdot \begin{bmatrix} -\mathsf{A}_{01}^t\mathsf{H}_0' + \mathsf{H}_1 \\ \mathsf{H}_0' \end{bmatrix}^t$$

where $\mathsf{G}_0', \mathsf{H}_0'$ are $\nabla_{\mathsf{v}_0,\mathsf{u}_0}$-generators of A_{00}^{-1}.

(more on Cardinal's formula in appendix slide ?)

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. Cut G, H, u, v in two halves $(i = \lceil n/2 \rceil)$

   $G'_0, H'_0 = \text{Struct-inverse-DAC}(G_0, H_0, u_0, v_0)$

   $$G = \begin{bmatrix} -A_{10} G'_0 + G_1 \\ G'_0 \end{bmatrix}, H = \begin{bmatrix} -A_{01}^t H'_0 + H_1 \\ H'_0 \end{bmatrix}, u = [u_1, v_0], v = [v_1, u_0]$$
3. Cut G, H, u, v in two halves $(i = n - \lceil n/2 \rceil)$

   $G'_0, H'_0 = \text{Struct-inverse-DAC}(G_0, H_0, u_0, v_0)$

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

Idea: $S^{(0)} = A \longrightarrow S^{(\lceil n/2 \rceil)} \longrightarrow S^{(n)} = A^{-1}$

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. Cut G, H, u, v in two halves $(i = \lceil n/2 \rceil)$

   $G'_0, H'_0 = \text{Struct-inverse-DAC}(G_0, H_0, u_0, v_0)$

   $$G = \begin{bmatrix} -A_{10} G'_0 + G_1 \\ G'_0 \end{bmatrix}, H = \begin{bmatrix} -A_{01}^t H'_0 + H_1 \\ H'_0 \end{bmatrix}, u = [u_1, v_0], v = [v_1, u_0]$$
3. Cut G, H, u, v in two halves $(i = n - \lceil n/2 \rceil)$

   $G'_0, H'_0 = \text{Struct-inverse-DAC}(G_0, H_0, u_0, v_0)$

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

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. Cut G, H, u, v in two halves $(i = \lceil n/2 \rceil)$

   $G'_0, H'_0 = \text{Struct-inverse-DAC}(G_0, H_0, u_0, v_0)$

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

3. Cut G, H, u, v in two halves $(i = n - \lceil n/2 \rceil)$

   $G'_0, H'_0 = \text{Struct-inverse-DAC}(G_0, H_0, u_0, v_0)$

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

4. **return** G, H

Cost analysis

Non recursive cost : $A_{10} G'_0, A_{01}^t H'_0$ $\simeq$ Cauchy-like matrix $\times$ α vectors

Total cost: $\mathcal{O}((\text{Non recursive cost}) \times \log(n)) = \mathcal{O}(\alpha^{\omega-1} \mathcal{M}(n) \log^2(n))$

Algorithm Struct-inverse-DAC

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

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

1. **if** $n \leqslant \text{threshold}$ **then return** Struct-inverse-iter$(\mathsf{G}, \mathsf{H}, \mathsf{u}, \mathsf{v})$
2. Cut $\mathsf{G}, \mathsf{H}, \mathsf{u}, \mathsf{v}$ in two halves $(i = \lceil n/2 \rceil)$

   $\mathsf{G}'_0, \mathsf{H}'_0 = \text{Struct-inverse-DAC}(\mathsf{G}_0, \mathsf{H}_0, \mathsf{u}_0, \mathsf{v}_0)$

   $$\mathsf{G} = \begin{bmatrix} -\mathsf{A}_{10}\,\mathsf{G}'_0 + \mathsf{G}_1 \\ \mathsf{G}'_0 \end{bmatrix}, \mathsf{H} = \begin{bmatrix} -\mathsf{A}_{01}^t \mathsf{H}'_0 + \mathsf{H}_1 \\ \mathsf{H}'_0 \end{bmatrix}, \mathsf{u} = [\mathsf{u}_1, \mathsf{v}_0], \mathsf{v} = [\mathsf{v}_1, \mathsf{u}_0]$$
3. Cut $\mathsf{G}, \mathsf{H}, \mathsf{u}, \mathsf{v}$ in two halves $(i = n - \lceil n/2 \rceil)$

   $\mathsf{G}'_0, \mathsf{H}'_0 = \text{Struct-inverse-DAC}(\mathsf{G}_0, \mathsf{H}_0, \mathsf{u}_0, \mathsf{v}_0)$

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

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

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

Check the generic rank profile property

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. Cut $\mathsf{G}, \mathsf{H}, \mathsf{u}, \mathsf{v}$ in size β and $n - \beta$
   b. Decompress $\mathsf{A}_{00} = \nabla^{-1}_{\mathsf{u}_0,\mathsf{v}_0}(\mathsf{G}_0 \cdot \mathsf{H}_0^t)$
   c. Invert A_{00} — Dense inversion
   d. Compress $\mathsf{A}_{00}^{-1} \rightsquigarrow \mathsf{G}'_0, \mathsf{H}'_0$
   e. Decompress $\mathsf{A}_{01} = \nabla^{-1}_{\mathsf{u}_0,\mathsf{v}_1}(\mathsf{G}_0 \cdot \mathsf{H}_1^t)$ and $\mathsf{A}_{10} = \nabla^{-1}_{\mathsf{u}_1,\mathsf{v}_0}(\mathsf{G}_1 \cdot \mathsf{H}_0^t)$
   f. Compute $\mathsf{A}_{10}\,\mathsf{G}'_0$ and $\mathsf{A}_{01}^t \mathsf{H}'_0$ — Dense matrix multiplication
   g. $\mathsf{G} = \begin{bmatrix} -\mathsf{A}_{10}\,\mathsf{G}'_0 + \mathsf{G}_1 \\ \mathsf{G}'_0 \end{bmatrix}, \mathsf{H} = \begin{bmatrix} -\mathsf{A}_{01}^t \mathsf{H}'_0 + \mathsf{H}_1 \\ \mathsf{H}'_0 \end{bmatrix}, \mathsf{u} = [\mathsf{u}_1, \mathsf{v}_0], \mathsf{v} = [\mathsf{v}_1, \mathsf{u}_0]$
2. **return** G, H

Idea: $\mathsf{S}^{(0)} = \mathsf{A} \longrightarrow \mathsf{S}^{(\beta)} \longrightarrow \mathsf{S}^{(2\beta)} \longrightarrow \cdots \longrightarrow \mathsf{S}^{(k\beta)} \longrightarrow \cdots \longrightarrow \mathsf{S}^{(n)} = \mathsf{A}^{-1}$

Algorithm Struct-inverse-iter

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

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

1. **for** $(i = 0; i < n; i = i + \beta)$
   a. Cut G, H, u, v in size β and $n - \beta$
   b. Decompress $A_{00} = \nabla^{-1}_{u_0,v_0}(G_0 \cdot H_0^t)$
   c. Invert A_{00} — Dense inversion
   d. Compress $A_{00}^{-1} \rightsquigarrow G'_0, H'_0$
   e. Decompress $A_{01} = \nabla^{-1}_{u_0,v_1}(G_0 \cdot H_1^t)$ and $A_{10} = \nabla^{-1}_{u_1,v_0}(G_1 \cdot H_0^t)$
   f. Compute $A_{10} G'_0$ and $A_{01}^t H'_0$ — Dense matrix multiplication
   g. $G = \begin{bmatrix} -A_{10} G'_0 + G_1 \\ G'_0 \end{bmatrix}, H = \begin{bmatrix} -A_{01}^t H'_0 + H_1 \\ H'_0 \end{bmatrix}, u = [u_1, v_0], v = [v_1, u_0]$
2. **return** G, H

Why dense matrices ?

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

So what is the benefit of dealing with structured matrices ?

Only compute compressed form of large matrix $(A_{11} - A_{10} A_{00}^{-1} A_{01})$

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. Cut $\mathsf{G}, \mathsf{H}, \mathsf{u}, \mathsf{v}$ in size β and $n - \beta$
   b. Decompress $\mathsf{A}_{00} = \nabla^{-1}_{\mathsf{u}_0,\mathsf{v}_0}(\mathsf{G}_0 \cdot \mathsf{H}_0^t)$
   c. Invert A_{00} — Dense inversion
   d. Compress $\mathsf{A}_{00}^{-1} \rightsquigarrow \mathsf{G}'_0, \mathsf{H}'_0$
   e. Decompress $\mathsf{A}_{01} = \nabla^{-1}_{\mathsf{u}_0,\mathsf{v}_1}(\mathsf{G}_0 \cdot \mathsf{H}_1^t)$ and $\mathsf{A}_{10} = \nabla^{-1}_{\mathsf{u}_1,\mathsf{v}_0}(\mathsf{G}_1 \cdot \mathsf{H}_0^t)$
   f. Compute $\mathsf{A}_{10}\,\mathsf{G}'_0$ and $\mathsf{A}_{01}^t \mathsf{H}'_0$ — Dense matrix multiplication
   g. $\mathsf{G} = \begin{bmatrix} -\mathsf{A}_{10}\,\mathsf{G}'_0 + \mathsf{G}_1 \\ \mathsf{G}'_0 \end{bmatrix}, \mathsf{H} = \begin{bmatrix} -\mathsf{A}_{01}^t \mathsf{H}'_0 + \mathsf{H}_1 \\ \mathsf{H}'_0 \end{bmatrix}, \mathsf{u} = [\mathsf{u}_1, \mathsf{v}_0], \mathsf{v} = [\mathsf{v}_1, \mathsf{u}_0]$
2. **return** G, H

Cost analysis for one iteration when $\beta = \alpha$:

1. Decompress, invert A_{00} and compress A_{00}^{-1} in $\mathcal{O}(\alpha^\omega)$

2. Decompress $\mathsf{A}_{01}, \mathsf{A}_{10}$ ($\simeq$ compute $\mathsf{G}_0 \cdot \mathsf{H}_1^t$), multiply $\mathsf{A}_{10} \cdot \mathsf{G}'_0$ in $\mathcal{O}(\alpha^{\omega-1} n)$

Total cost: n/α iterations $\mathcal{O}(\alpha^{\omega-2} n^2)$

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. Cut $\mathsf{G}, \mathsf{H}, \mathsf{u}, \mathsf{v}$ in size β and $n - \beta$
   b. Decompress $\mathsf{A}_{00} = \nabla^{-1}_{\mathsf{u}_0,\mathsf{v}_0}(\mathsf{G}_0 \cdot \mathsf{H}_0^t)$
   c. Invert A_{00} — Dense inversion
   d. Compress $\mathsf{A}_{00}^{-1} \rightsquigarrow \mathsf{G}'_0, \mathsf{H}'_0$
   e. Decompress $\mathsf{A}_{01} = \nabla^{-1}_{\mathsf{u}_0,\mathsf{v}_1}(\mathsf{G}_0 \cdot \mathsf{H}_1^t)$ and $\mathsf{A}_{10} = \nabla^{-1}_{\mathsf{u}_1,\mathsf{v}_0}(\mathsf{G}_1 \cdot \mathsf{H}_0^t)$
   f. Compute $\mathsf{A}_{10}\,\mathsf{G}'_0$ and $\mathsf{A}_{01}^t \mathsf{H}'_0$ — Dense matrix multiplication
   g. $\mathsf{G} = \begin{bmatrix} -\mathsf{A}_{10}\,\mathsf{G}'_0 + \mathsf{G}_1 \\ \mathsf{G}'_0 \end{bmatrix}, \mathsf{H} = \begin{bmatrix} -\mathsf{A}_{01}^t \mathsf{H}'_0 + \mathsf{H}_1 \\ \mathsf{H}'_0 \end{bmatrix}, \mathsf{u} = [\mathsf{u}_1, \mathsf{v}_0], \mathsf{v} = [\mathsf{v}_1, \mathsf{u}_0]$
2. **return** G, H

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. The generated C will have generic rank profile
3. Can choose u, v geometric sequences with same ratio $\rightsquigarrow$ Gain factor 3

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 ?

1. $\nabla_{M,P}(A\,B) = \nabla_{M,N}(A)\,B + A\,\nabla_{N,P}(B)$
2. V_u Vandermonde: So $\nabla_{D_u,Z}(V_u)$ has rank 1
3. $\nabla_{Z,Z}(T)$ has rank α
4. W_v reversed-Vandermonde so that $\nabla_{Z,D_v}(W_v)$ has rank 1

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 a generic u, v in $\mathbb{K}$

(see sketch of the proof in appendix slide ?)

Implementations

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

Timings:

1. Iterative Cauchy-like inversion: step size $\beta = 1$ vs $\beta = \alpha$
2. Cauchy-like matrix multiplication: $\mathcal{O}(\alpha^2 \mathcal{M}(n))$ vs $\mathcal{O}(\alpha^{\omega-1} \mathcal{M}(n))$ [BJMS '17]
3. DAC Cauchy-like inversion: dense vs structured algorithms

Implementation – Iterative Cauchy-like inversion

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

Notes:
- For moderate α, $\omega = 3$ and same complexity but better constant in $\mathcal{O}$
- Crossover for $\alpha \simeq 10$

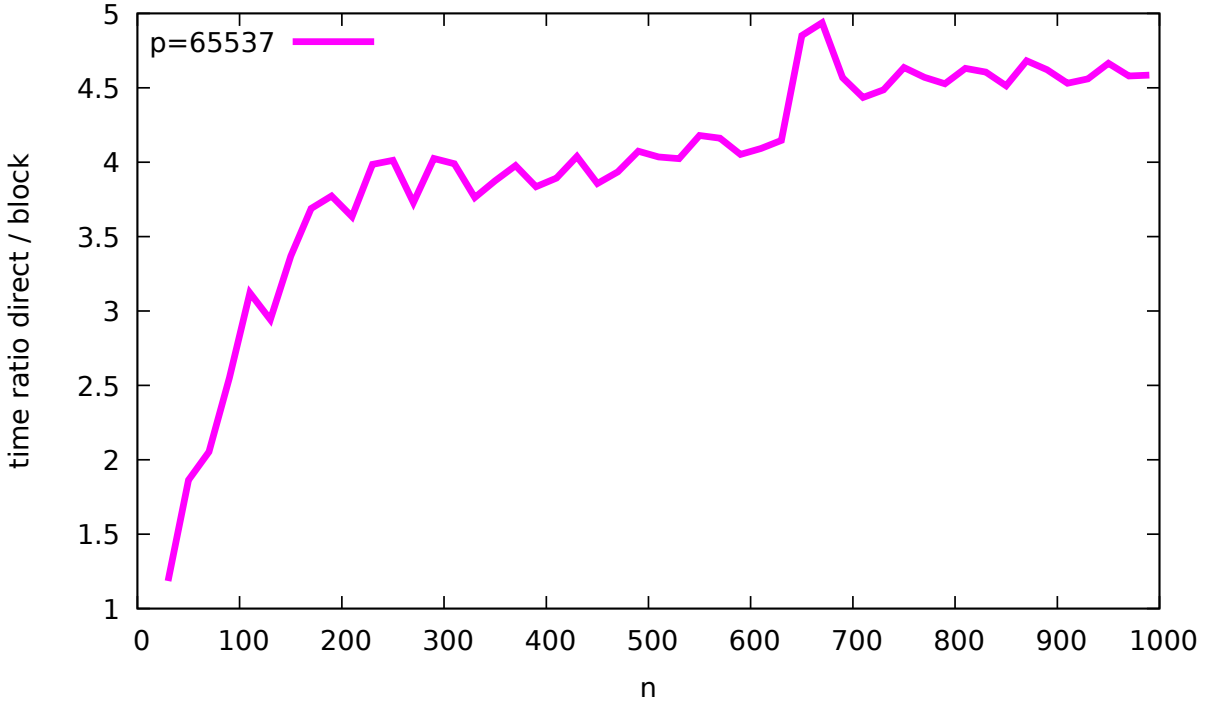


Figure. Time ratio $(\alpha\, n^2)\,/\,(\alpha^{\omega-2}\, n^2)$ for $\alpha = 30$

Implementation – Cauchy-like matrix multiplication

Cauchy-like matrix multiplication: $\mathcal{O}(\alpha^2\,\mathcal{M}(n))$ vs. $\mathcal{O}(\alpha^{\omega-1}\,\mathcal{M}(n))$ [BJMS '17]

Notes:
- For moderate α, $\omega = 3$ and same complexity but better constant in $\mathcal{O}$
- Crossover for $30 \leqslant \alpha \leqslant 55$
- u, v are geometric sequences with same ratio $\rightsquigarrow$ gain factor 3 in $\mathcal{O}(\alpha^2\,\mathcal{M}(n))$

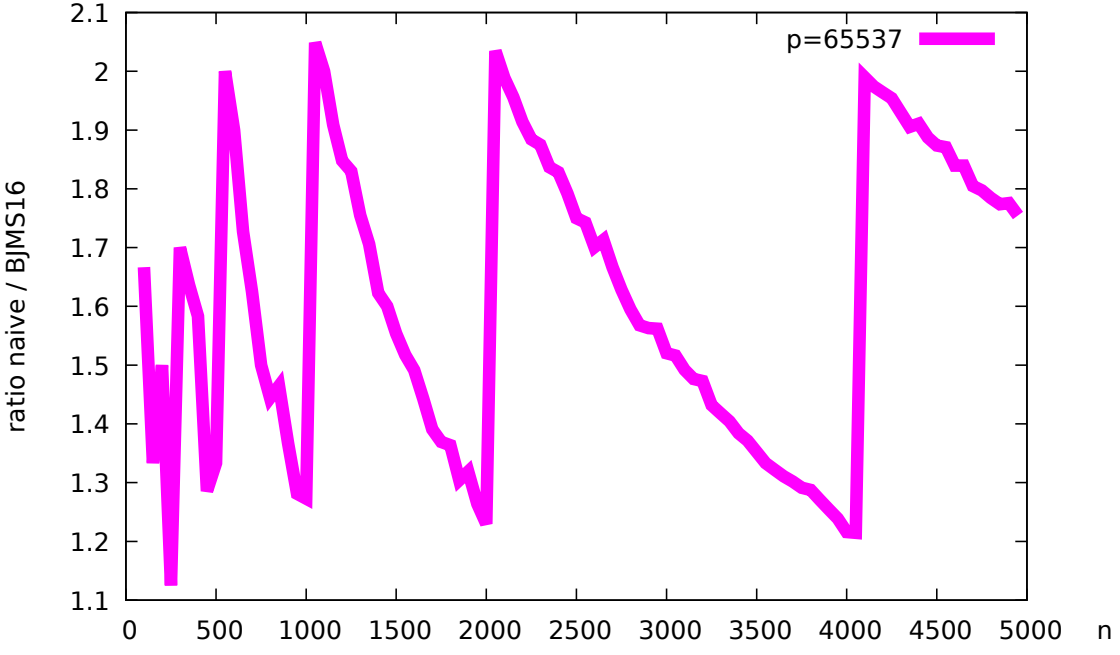
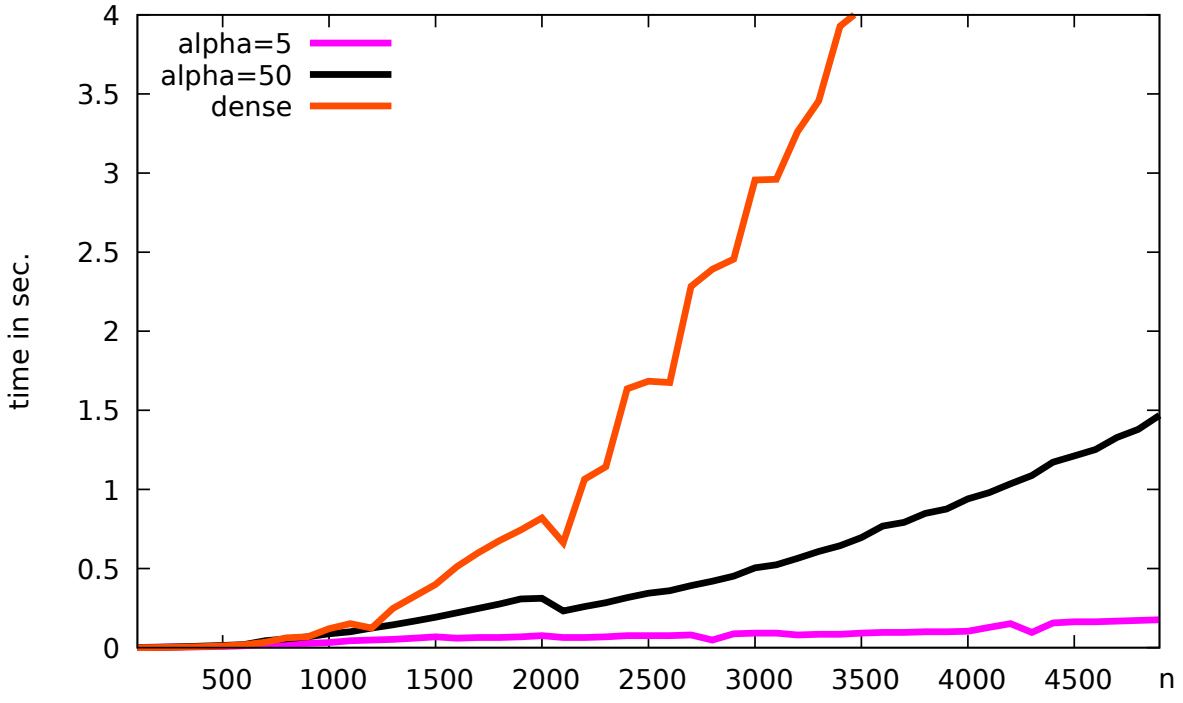


Figure. Time ratio $(\alpha^2\,\mathcal{M}(n))\,/\,(\alpha^{\omega-1}\,\mathcal{M}(n))$ for $\alpha = 60$

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 gains as soon as $\alpha \leqslant 0.2\,n$



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

Conclusion

Summary:

- Implemented many different algorithms using efficient libraries
- Improvements to iterative algorithm with practical benefits
- New class of Cauchy-like matrices with faster matrix-vector product
- New regularization process

Other aspects of [Hyun, L., Schost '17] not covered here:

- Same problems over $\mathbb{Q}$ (instead of $\mathbb{F}_p$)
- DAC and Newton lifting algorithm
- Exploited the additional structure of the algebraic approximants

Thank you for your attention ;-)

Recall

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

It is surprising that $S^{(i)}$ has displacement rank α :

$A_{00}^{-1} A_{01}$ could be of rank $2\,\alpha$, $A_{11} - A_{10}A_{00}^{-1}A_{01}$ of rank $4\,\alpha$, $S^{(i)}$ of rank $6\,\alpha$

Beginning of explanation:

1. $A_{00}^{-1} A_{01}$ is a submatrix of $\begin{bmatrix} A_{00}^{-1} & -A_{00}^{-1} A_{01} \\ 0 & \mathrm{Id}_{n-i} \end{bmatrix} = \begin{bmatrix} A_{00} & A_{01} \\ 0 & \mathrm{Id}_{n-i} \end{bmatrix}^{-1}$ $\Rightarrow$ rank α
2. $(A_{11} - A_{10}A_{00}^{-1}A_{01})^{-1} = (A^{-1})_{11}$ $\Rightarrow$ rank α
3. Idea for $S^{(i)}$? Use generators of A_{00}^{-1}, A_{11}^{-1} and

$$\begin{bmatrix} A_{11} - A_{10} A_{00}^{-1} A_{01} & A_{10} A_{00}^{-1} \\ -A_{00}^{-1} A_{01} & A_{00}^{-1} \end{bmatrix} = \begin{bmatrix} A_{11}^{-1} & -A_{11}^{-1} A_{10} \\ A_{01}A_{11}^{-1} & A_{00} - A_{01} A_{11}^{-1} A_{10} \end{bmatrix}^{-1}$$

Back to slide ?

Theorem. ([Hyun, L., Schost '17])

$\mathsf{A} = \mathsf{V_u}\,\mathsf{T}\,\mathsf{W_v}$ *has generic rank profile for a generic* u, v *in* $\mathbb{K}$

Sketch of proof. If entries of u, v are indeterminates, let's show that A has generic rank profile.

Use Cauchy-Binet formulae: Let $I = \{1, ..., i\}$ for $i \leqslant \mathrm{rank}(\mathsf{A})$. Then

$$\det(\mathsf{V_u}\,\mathsf{T}\,\mathsf{W_v})_{I,I} = \sum_{\substack{|J|=i \\ |K|=i}} \det(\mathsf{V_u})_{I,J} \det(\mathsf{A})_{J,K} \det(\mathsf{W_v})_{K,I}$$

Show that it is not the zero polynomial.

The leading monomial of $\det(\mathsf{V_u})_{I,J}\,\det(\mathsf{W_v})_{K,I}$ is uniquely determined by J, K (for some monomial ordering).

Take $J^{\uparrow}, K^{\uparrow}$ so that $\det(\mathsf{V_u})_{I,J} \det(\mathsf{W_v})_{K,I}$ is maximal amongst $\{J, K \text{ s.t. } \det(\mathsf{A})_{J,K} \neq 0\}$.

$\mathrm{LM}(\det(\mathsf{V_u}\,\mathsf{T}\,\mathsf{W_v})_{I,I}) = \mathrm{LM}(\det(\mathsf{V_u})_{I,J^{\uparrow}} \det(\mathsf{W_v})_{K^{\uparrow},I})$ and the polynomial is not zero.

Note: No proof when u, v are geometric sequences.

Back to slide ?