Correcting errors in a matrix inverse
(or product)

Daniel S. Roche

Computer Science Department
United States Naval Academy
Annapolis, Maryland, U.S.A.

Laboratoire Jean Kuntzmann
Université Grenoble Alpes
Grenoble, France

24 January 2018
Journées Nationales de Calcul Formel
Luminy, France

Correcting Errors

General Framework

- Some problem is slow to solve locally.
- So we outsource the computation and get back the result.
- If the result is wrong, we should detect that.
- If the result is "close", we should correct it.

Sources of error

- Noisy communication channel
- Faults in computation
- Malicious servers?

Example: Sudoku

Sent to server

	7		8		2			
								7
1	4	8	3	5				
		5			8	3		
	6							9
3						7		
5						1	7	
		4		9				
		7	4					2

Response

9	7	6	8	1	2	4	5	3
2	5	3	9	6	4	8	1	7
1	4	5	3	8	7	9	2	6
7	9	8	6	2	5	3	4	1
4	6	1	5	7	3	2	8	9
3	8	2	1	4	9	7	6	5
5	3	9	2	9	6	1	7	4
6	2	4	7	9	1	5	3	8
8	1	7	4	3	8	6	9	2

Source: `https://www.nytimes.com/crosswords/game/sudoku/hard`

Example: Sudoku

Sent to server

	7		8		2			
								7
1	4	8	3	5				
		5			8	3		
	6							9
3						7		
5						1	7	
		4		9				
		7	4					2

Response

9	7	6	8	1	2	4	5	3
2	5	3	9	6	4	8	1	7
1	4	5	3	8	7	9	2	6
7	9	8	6	2	5	3	4	1
4	6	1	5	7	3	2	8	9
3	8	2	1	4	9	7	6	5
5	3	9	2	9	6	1	7	4
6	2	4	7	9	1	5	3	8
8	1	7	4	3	8	6	9	2

Source: `https://www.nytimes.com/crosswords/game/sudoku/hard`

First Problem: Matrix Multiplication

Matrix Multiplication with Errors

Input: $A, B, C \in \mathsf{F}^{n\times n}$, where $C \approx AB$

Output: $E \in \mathsf{F}^{n\times n}$ s.t. $AB = C - E$

Parameters:

- Dimension n
- Input size $t = \#A + \#B + \#C$
- Error count $k = \#E$

First Problem: Matrix Multiplication

Matrix Multiplication with Errors

Input: $A, B, C \in \mathsf{F}^{n \times n}$, where $C \approx AB$

Output: $E \in \mathsf{F}^{n \times n}$ s.t. $AB = C - E$

Parameters:

- Dimension n
- Input size $t = \#A + \#B + \#C$
- Error count $k = \#E$

- Naïve recomputation: $O(n^\omega)$
- Gasieniec, Levcopoulos, Lingas, Pagh, Tokuyama, *Algorithmica* 2017: $\widetilde{O}\left(n^2 + kn\right)$

First Problem: Matrix Multiplication

Matrix Multiplication with Errors

Input: $A, B, C \in \mathsf{F}^{n\times n}$, where $C \approx AB$
Output: $E \in \mathsf{F}^{n\times n}$ s.t. $AB = C - E$
Parameters:

- Dimension n
- Input size $t = \#A + \#B + \#C$
- Error count $k = \#E$

- Naïve recomputation: $O(n^\omega)$
- Gasieniec, Levcopoulos, Lingas, Pagh, Tokuyama, *Algorithmica* 2017: $\widetilde{O}\left(n^2 + kn\right)$
- Ours: $\widetilde{O}(t + kn)$ worst-case; $\widetilde{O}\left(t + k^{0.686}n\right)$ best case

Related Work

Monte Carlo **verification** in Linear Algebra

- Matrix multiplication: Freivalds ’79
- Positive semidefiniteness: Kaltofen, Nehring, Saunders ’11
- Rank, characteristic polynomial,. . . : Dumas & Kaltofen ’14
- Rank profile, triangular forms: Dumas, Lucas, Pernet ’17

Error correction in symbolic computation

Complexity of sparse matrix multiplication

Related Work

Monte Carlo **verification** in Linear Algebra

Error correction in symbolic computation

- Chinese remaindering: Goldreich, Ron, Sudan ’99
- Rational reconstruction: Khonji, Pernet, Roch, Roche, Stalinski ’10
- Sparse interpolation: Comer, Kaltofen, Pernet ’12

Complexity of sparse matrix multiplication

Related Work

Monte Carlo **verification** in Linear Algebra

Error correction in symbolic computation

Complexity of sparse matrix multiplication

k = number of nonzeros

- $\widetilde{O}\left(n^2 + k^{0.7}n^{1.2}\right)$: Yuster & Zwick ’04
- $\widetilde{O}\left(k^{1.34}\right)$ output sensitive: Amossen & Pagh ’09
- $\widetilde{O}\left(k^{0.188}n^2\right)$: Lingas ’10
- $\widetilde{O}\left(n^2 + kn\right)$ output sensitive: Pagh ’13

Outline

- Introduction
- Matrix Multiplication with Errors
  Crucial tools: Nonzero row identification, Sparse interpolation
- Matrix Inverse with Errors
  Additional tools: Commutativity, Sparse low-rank linear algebra
- Open Problems

Correcting errors in matrix multiplication

Input: $A, B, C \in \mathsf{F}^{n\times n}$
Output: $E \in \mathsf{F}^{n\times n}$ s.t. $AB = C - E$

Algorithm overview:

1. Formulate as $E = C - AB$
2. Find nonzero rows
   Choose random $v \in \mathsf{F}^n$ and compute Ev
3. Evaluate row polynomials
   For each evaluation θ, compute $E[1, \theta, \ldots, \theta^{n-1}]^T$
4. Interpolate row polynomials
   Sparse interpolation to recover at least half of the nonero rows
5. Repeat $O(\log k)$ times until all rows are found

Finding nonzero rows

For unknown $M \in \mathsf{F}^{n \times n}$, which rows are nonzero?

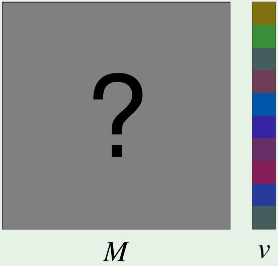
Example

M

Finding nonzero rows

For unknown $M \in \mathsf{F}^{n \times n}$, which rows are nonzero?

Example

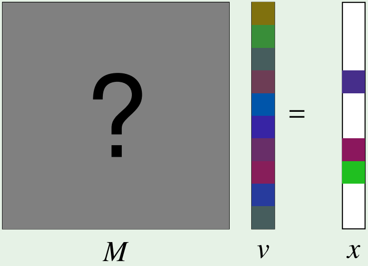


1. Choose random vector $v \in \mathsf{F}^n$

Finding nonzero rows

For unknown $M \in \mathsf{F}^{n \times n}$, which rows are nonzero?

Example

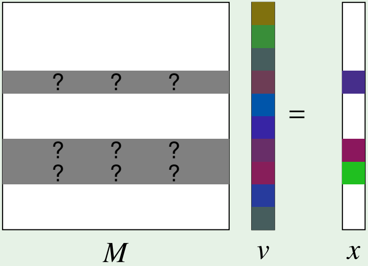


1. Choose random vector $v \in \mathsf{F}^n$
2. Apply $Mv = x$

Finding nonzero rows

For unknown $M \in \mathsf{F}^{n \times n}$, which rows are nonzero?

Example

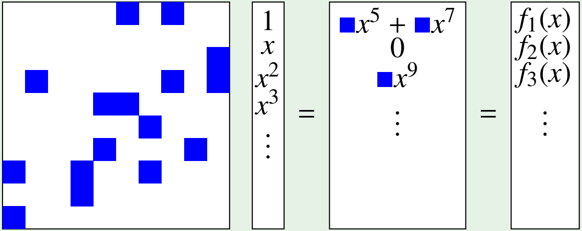


1. Choose random vector $v \in \mathsf{F}^n$
2. Apply $Mv = x$
3. Nonzero rows of M are nonzero entries in x w.h.p.

Row Polynomials

Treat each row of $M \in \mathsf{F}^{n \times n}$ as a degree-$(n-1)$ polynomial

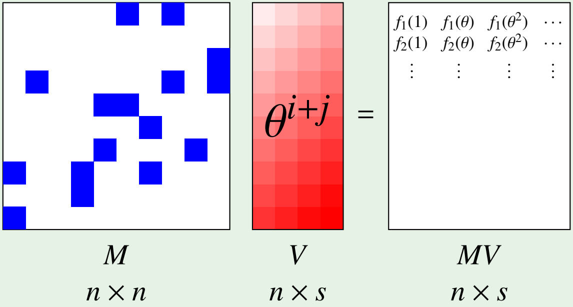
Example



Row Polynomial Evaluation

- Need to evaluate each row polynomial $f_i(x)$ at $1, \theta, \theta^2, \ldots$
- This is equivalent to multiplying by a (truncated) DFT matrix.

Example



Important: Takes $\widetilde{O}(sn + t)$ field ops, where $t = \#M$

Oligonomial Polynomial Interpolation

Algorithm (Kaltofen, Lakshman, Wiley 1990)

Unknown polynomial $f \in \mathsf{F}[x]$ with t nonzero terms
is uniquely determined by $f(1), f(\theta), f(\theta^2), \ldots, f(\theta^{2s-1})$ iff:

- f has at most s nonzero terms; and
- $\mathrm{order}(\theta) \geq \deg f$

Problem over GF(q):

Oligonomial Polynomial Interpolation

Algorithm (Kaltofen, Lakshman, Wiley 1990)

Unknown polynomial $f \in \mathsf{F}[x]$ with t nonzero terms
is uniquely determined by $f(1), f(\theta), f(\theta^2), \ldots, f(\theta^{2s-1})$ iff:

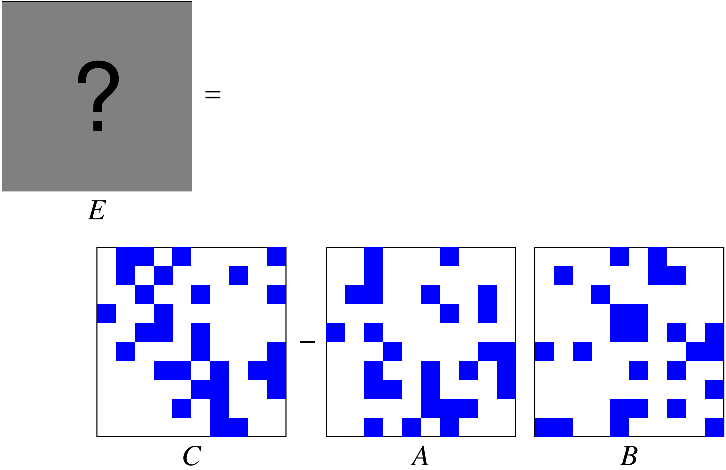
- f has at most s nonzero terms; and
- $\operatorname{order}(\theta) \geq \deg f$

Problem over GF(q):
Needs t discrete logarithms to discover the exponents

Solution: Pre-compute first $\deg f = n$ powers of θ

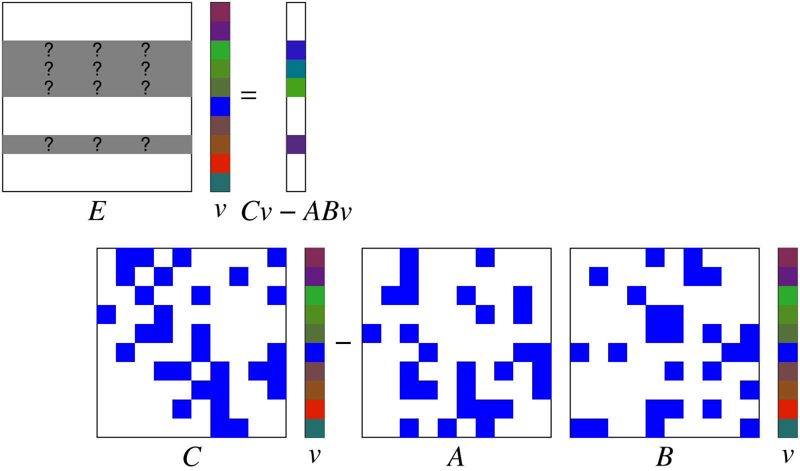
Multiplication Algorithm Overview

Given A, B, C, we want to compute $E = C - AB$



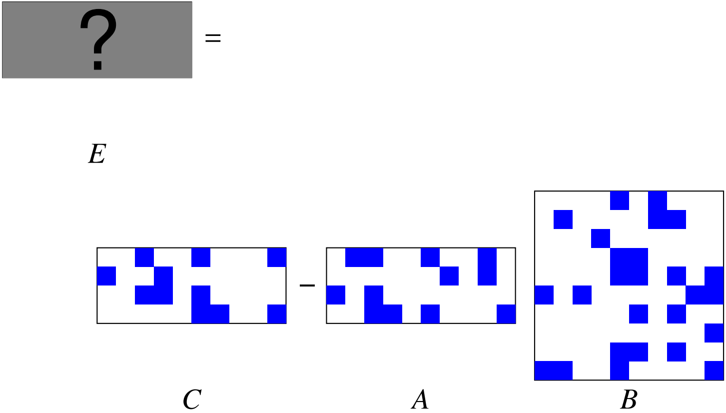
Multiplication Algorithm Overview

1. Find zero rows of E



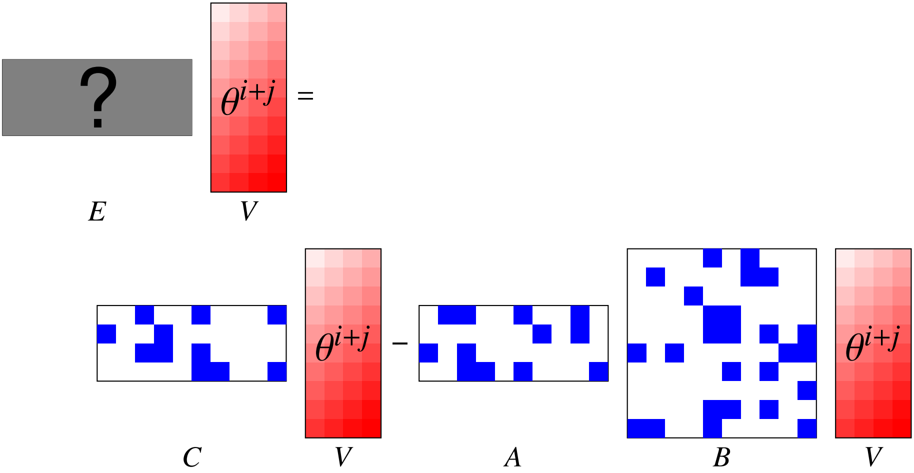
Multiplication Algorithm Overview

2 Remove corresponding rows from C and A



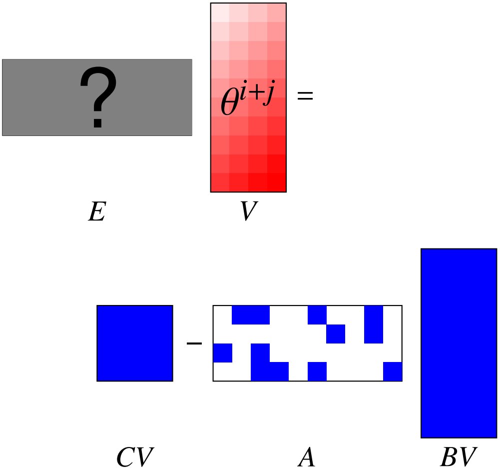
Multiplication Algorithm Overview

3 Choose $s \leq n$ and evaluate row polynomials at $2s$ powers of θ



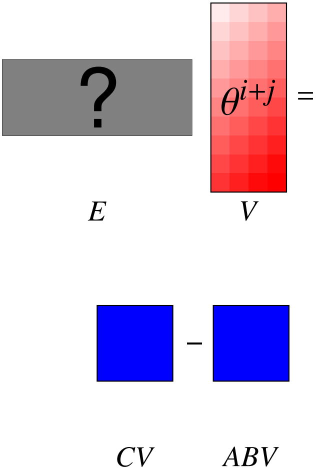
Multiplication Algorithm Overview

3 Choose $s \leq n$ and evaluate row polynomials at $2s$ powers of θ



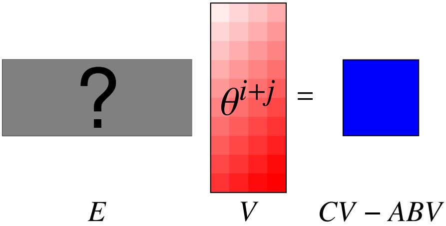
Multiplication Algorithm Overview

3. Choose $s \leq n$ and evaluate row polynomials at $2s$ powers of θ



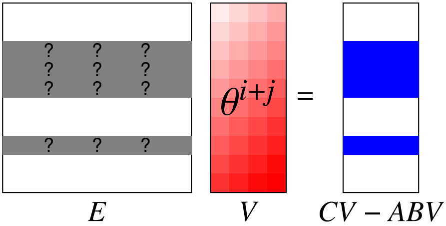
Multiplication Algorithm Overview

3. Choose $s \leq n$ and evaluate row polynomials at $2s$ powers of θ



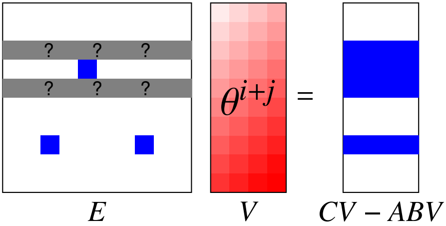
Multiplication Algorithm Overview

3. Choose $s \leq n$ and evaluate row polynomials at $2s$ powers of θ



Multiplication Algorithm Overview

4. Perform sparse interpolation to recover s-sparse rows of E



Complexity Analysis

- How large should s (number of evaluations) be?
  If $s \geq 2k/r$, then we recover at least half of the nonzero rows.
- We can transpose as necessary
  Make the number of nonzero rows in E minimal.

Multiplication Complexity

Total cost in field operations is

$$\widetilde{O}\left(t + \frac{kn}{\min(r, \frac{k}{r})^{3-\omega}}\right)$$

- n = dimension
- t = number of nonzeros in A, B, C
- k = number of errors (nonzeros in E)
- r = number of nonzero rows or columns in E, whichever is smaller
- $\omega < 2.373$, exponent of matrix multiplication

Multiplication Complexity

Total cost in field operations is $\widetilde{O}\left(t + kn/\min(r, \frac{k}{r})^{3-\omega}\right)$

- n = dimension
- t = number of nonzeros in A, B, C
- k = number of errors (nonzeros in E)
- r = number of nonzero rows or columns in E, whichever is smaller
- $\omega < 2.373$, exponent of matrix multiplication

Very sparse	"Compact" errors	Worst case	
$t \ll nk^{\omega/2}$	$r \approx \sqrt{k}$	$k \le n$	$k \ge n$
$tk^{0.5}$	$t + k^{0.69}n$ $t + k^{(\omega-1)/2}n$	$t + kn$	$t + k^{0.38}n^{1.63}$ $t + k^{\omega-2}n^{4-\omega}$

Second Problem: Matrix Inverse

Matrix Inverse with Errors

Input: $A, B \in \mathsf{F}^{n \times n}$

Output: $E \in \mathsf{F}^{n \times n}$ s.t. $A^{-1} = B + E$

Parameters:

- Dimension n
- Input size $t = \#A + \#B$
- Error count $k = \#E$

- Naïve recomputation: $O(n^\omega)$
- Ours: $\widetilde{O}(t + kn + k^\omega)$ worst-case; $\widetilde{O}\left(t + k^{0.69} n\right)$ best case

Problem Formulation

Rewriting the problem definition:

$$\begin{aligned} EA &= I - BA \\ AE &= I - AB \end{aligned}$$

Problem Formulation

Rewriting the problem definition:
$$EA = I - BA$$
$$AE = I - AB$$

Crucial steps in multiplication algorithm:

- Computing nonzero rows of E
  Evaluate $EAv = v - A(Bv)$ for random vector v

Problem Formulation

Rewriting the problem definition:
$$EA = I - BA$$
$$AE = I - AB$$

Crucial steps in multiplication algorithm:

- Computing nonzero rows of E
  Evaluate $EAv = v - A(Bv)$ for random vector v
- Evaluating row polynomials at $2s$ powers of θ
  (Coming up next. . .)

Low-Rank Linear Algebra

Algorithm (Chung, Kwok, Lau '13)

Input: Matrix M with rank r and t nonzeros
Output: Indices of r linearly independent rows
Cost: $\widetilde{O}(t + r^{\omega})$

Low-Rank Linear Algebra

Algorithm (Chung, Kwok, Lau '13)

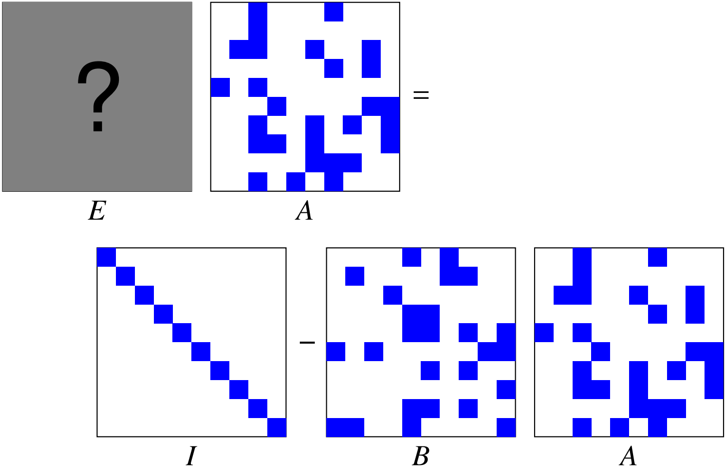
Input: Matrix M with rank r and t nonzeros
Output: Indices of r linearly independent rows
Cost: $\widetilde{O}(t + r^\omega)$

How we use this:

1. Find zero rows of E
2. Remove corresponding **columns** of A
3. Resulting $(r \times n)$ matrix has rank r
4. Remove linearly dependent rows
5. Compute inverse of resulting $r \times r$ matrix

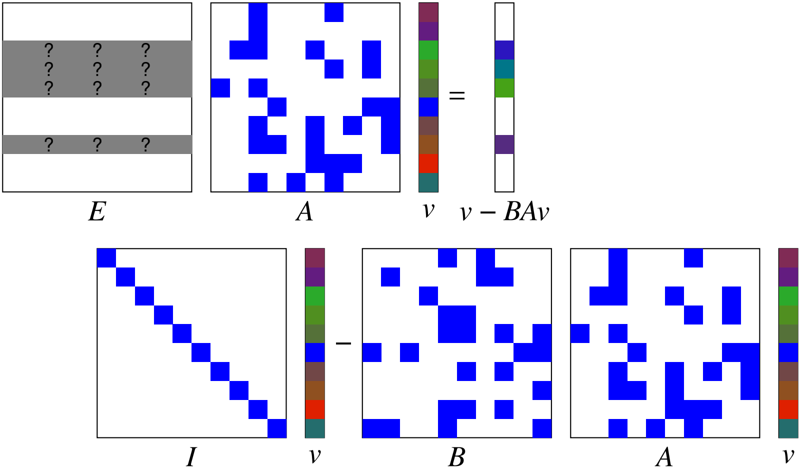
Inverse Algorithm Overview

Write $EA = I - BA$



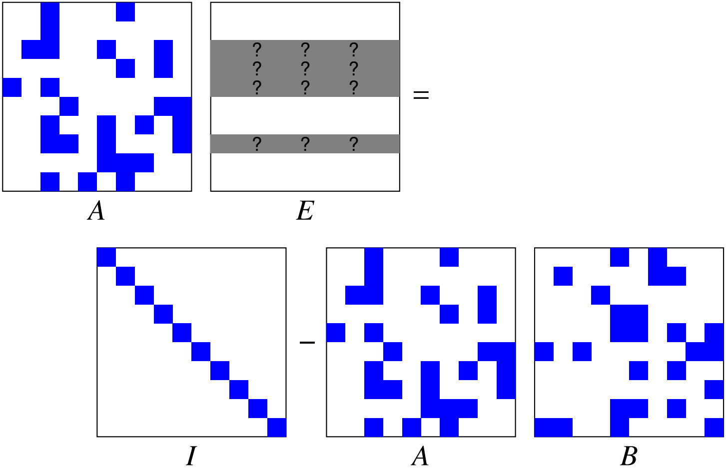
Inverse Algorithm Overview

Find nonzero rows of E



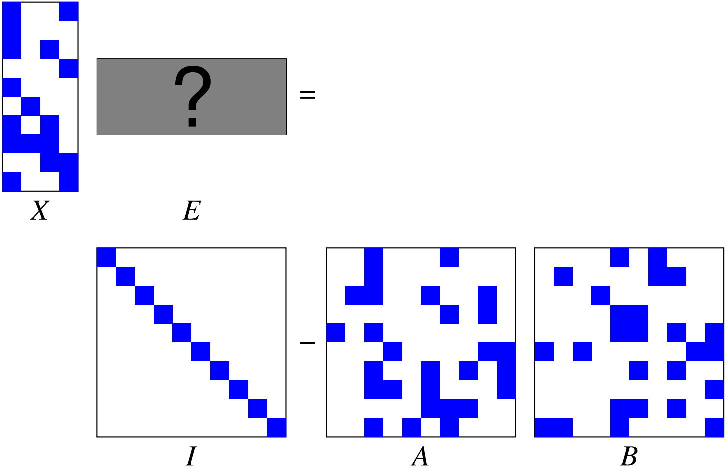
Inverse Algorithm Overview

Write $AE = I - AB$



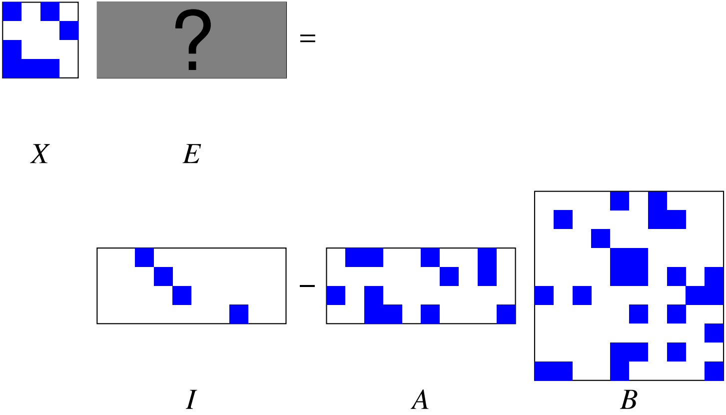
Inverse Algorithm Overview

Remove zero rows of E and corresp. columns from A



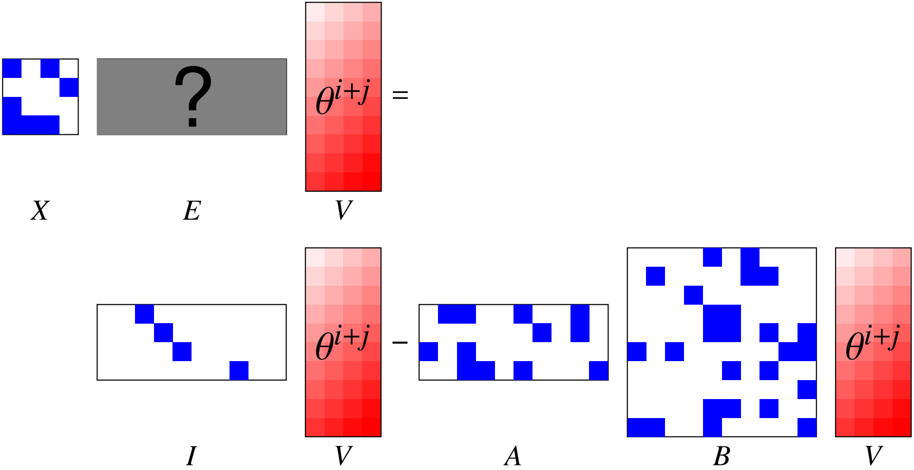
Inverse Algorithm Overview

Find linear independent rows of X, remove others from both sides



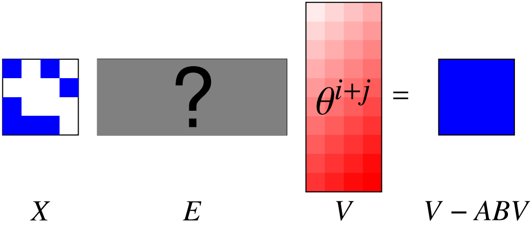
Inverse Algorithm Overview

Evaluate row polynomials at first $2s$ powers of θ



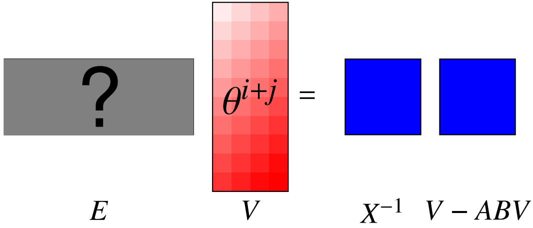
Inverse Algorithm Overview

Evaluate row polynomials at first $2s$ powers of θ



Inverse Algorithm Overview

Compute X^{-1} and apply to both sides



Inverse Complexity

Total cost in field operations is $\widetilde{O}\left(t + r^\omega + kn/\min(r, \frac{k}{r})^{3-\omega}\right)$

- n = dimension
- t = number of nonzeros in A, B, C
- k = number of errors (nonzeros in E)
- r = number of nonzero rows or columns in E, whichever is smaller
- $\omega < 2.373$, exponent of matrix multiplication

"Compact" errors	Worst case		
$r \approx \sqrt{k}$	$k \le n^{0.73}$	$n^{0.74} \le k \le n$	$k \ge n$
$t + k^{0.69}n$ $t + k^{(\omega-1)/2}n$	$t + kn$	$t + k^{2.38}$ $t + k^{\omega}$	$n^{2.38}$ n^{ω}

(Potential) Applications

- Fault-tolerant computing
- Sparse and output-sensitive computation
- Simultaneous Chinese remaindering with different sizes

Challenge: LU Factorization

Problem

Input: $A, L, U \in \mathsf{F}^{n \times n}$
Output: $E_L, E_U \in \mathsf{F}^{n \times n}$ s.t. $A = (L + E_L)(U + E_U)$
(where L, E_L are lower triangular and U, E_U are upper triangular)

We can apply the inverse with errors algorithm except that:

- Only L **OR** U can have errors
- No commutativity, so can't choose min(nonzero rows, nonzero columns)

Next Steps

- **Faster**
  Can you achieve the “best-case” cost w.h.p.?
- **Further**
  LU factorization and other interesting problems. . .
  (Matrix inverse is kind of bullshit.)
- **Code**
  What is the *practical* number of errors k
  before recomputation is faster?


Merci !