Matrix Multiplication Based Computations of the Characteristic Polynomial

Clément PERNET,
joint work with Arne Storjohann

Symbolic Computation Group
University of Waterloo

Joint Lab Meeting ORCCA-SCG,
February 9, 2007

Introduction

Dense Linear Algebra over a Field:

- one of the usual models for complexity in linear algebra
- applied to
  - $\mathbb{R}$: floating point linear algebra
  - $GF(q), Z_p$ and $\mathbb{Z}$ (using CRT)

Introduction

Dense Linear Algebra over a Field:

- one of the usual models for complexity in linear algebra
- applied to
  - $\mathbb{R}$: floating point linear algebra
  - $GF(q), Z_p$ and $\mathbb{Z}$ (using CRT)

Applications in exact computation:

Cryptography :	number field sieves
Representation theory :	null space basis
Topology :	Smith normal forms
Graph theory :	characteristic polynomial

...

Approach

Problem

Compute the characteristic polynomial of a dense matrix over a field

- Deterministic or Las Vegas randomized algorithmns

Approach

Problem

Compute the characteristic polynomial of a dense matrix over a field

- Deterministic or Las Vegas randomized algorithmns
- Asymptotic time complexity...
- ... and practical algorithms

Approach

Problem

Compute the characteristic polynomial of a dense matrix over a field

- Deterministic or Las Vegas randomized algorithmns
- Asymptotic time complexity...
- ... and practical algorithms
  $\Rightarrow$Balance between asymptotic complexity and practical efficiency considerations

Approach

Problem

Compute the characteristic polynomial of a dense matrix over a field

- Deterministic or Las Vegas randomized algorithmns
- Asymptotic time complexity...
- ... and practical algorithms
  $\Rightarrow$Balance between asymptotic complexity and practical efficiency considerations
- space complexity

Outline

1. Matrix multiplication based linear algebra
   - Matrix multiplication: a building block
   - Reductions to matrix multiplication
2. Computing the characteristic polynomial
   - State of the art
   - A new algorithm
   - Algorithm into practice

Outline

Outline

1. Matrix multiplication based linear algebra
   - Matrix multiplication: a building block
   - Reductions to matrix multiplication

2. Computing the characteristic polynomial
   - State of the art
   - A new algorithm
   - Algorithm into practice

Asymptotic complexity

Matrix multiplication:

Folklore:	$2n^3 - n^2$
Strassen 1969:	$7n^{2.807} + o(n^{2.807})$
Winograd 1971:	$6n^{2.807} + o(n^{2.807})$
...	
Coppersmith Winograd 1990:	$\mathcal{O}\left(n^{2.376}\right)$

$\Rightarrow \mathcal{O}\left(n^{\omega}\right)$, where ω denotes an admissible exponent

Efficiency in practice

The most efficient routine in linear algebra.
Several reasons:

- dedicated processor instruction `fused-mac`: $z \leftarrow xy + z$

Efficiency in practice

The most efficient routine in linear algebra.
Several reasons:

- dedicated processor instruction `fused-mac`: $z \leftarrow xy + z$
- simple structure of the dot-product (pipelining is easy)

Efficiency in practice

The most efficient routine in linear algebra.
Several reasons:

- dedicated processor instruction `fused-mac`: $z \leftarrow xy + z$
- simple structure of the dot-product (pipelining is easy)
- enables better memory management

Efficiency in practice

The most efficient routine in linear algebra.
Several reasons:

- dedicated processor instruction `fused-mac`: $z \leftarrow xy + z$
- simple structure of the dot-product (pipelining is easy)
- enables better memory management
- sub-cubic algorithm
  - used to be considered as not practicable
  - beware of unstability with floating point numbers
  - but improves efficiency over finite fields

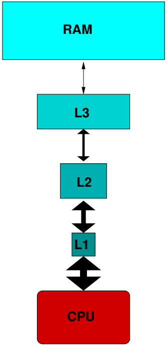
Efficiency in practice

The most efficient routine in linear algebra.
Several reasons:

- dedicated processor instruction `fused-mac`: $z \leftarrow xy + z$
- simple structure of the dot-product (pipelining is easy)
- enables better memory management
- sub-cubic algorithm
  - used to be considered as not practicable
  - beware of unstability with floating point numbers
  - but improves efficiency over finite fields

Memory management considerations

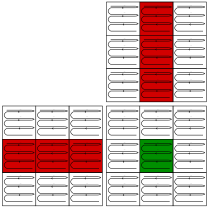
CPU-Memory communication: bandwidth gap
$\Rightarrow$Hierarchy of several cache memory levels



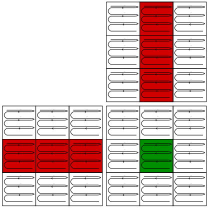
Memory management considerations

CPU-Memory communication: bandwidth gap
⇒Hierarchy of several cache memory levels

Imposes a structure for algorithms: operations must be blocked to increase data locality and fit in the cache



Memory management considerations



CPU-Memory communication: bandwidth gap

⇒Hierarchy of several cache memory levels

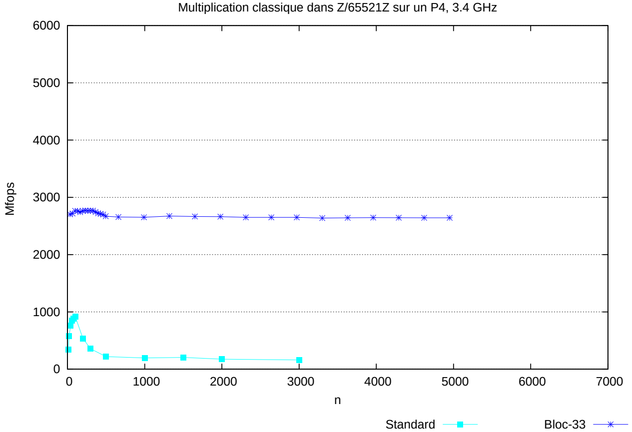
Imposes a structure for algorithms: operations must be blocked to increase data locality and fit in the cache

Reuse of the data

- `Work` $\gg$ `Data` to amortize memory transfer
  ⇒reach the peak performance of the CPU
- Matrix multiplication: $n^3 \gg n^2$
  ⇒well suited for block design

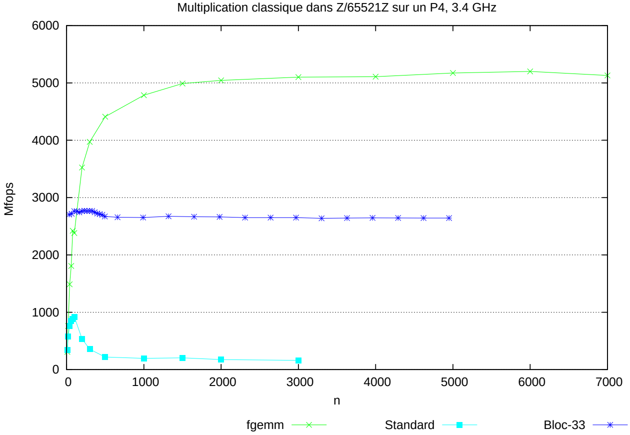
Practical implementation over finite fields

Matrix multiplication



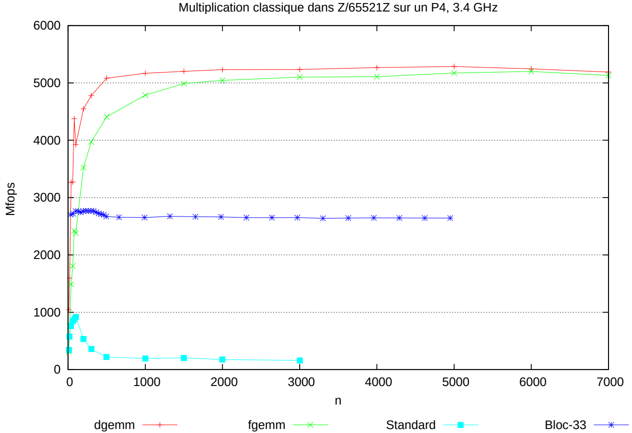
Practical implementation over finite fields

Matrix multiplication



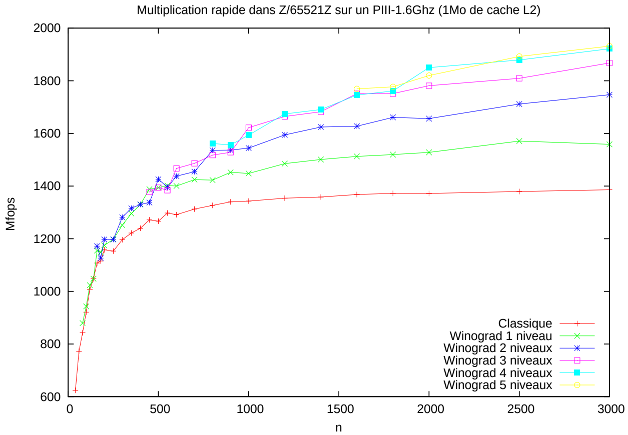
Practical implementation over finite fields

Matrix multiplication



Practical implementation over finite fields

Matrix multiplication



Outline

Other linear algebra problems

Asymptotic complexity:

- Used to be in $\mathcal{O}\left(n^3\right)$
- Room for improvement: $\mathcal{O}\left(n^\omega\right)$ for everyone ?

Other linear algebra problems

Asymptotic complexity:

- Used to be in $\mathcal{O}\left(n^3\right)$
- Room for improvement: $\mathcal{O}\left(n^\omega\right)$ for everyone ?

Practical efficiency: reuse the efficient matrix multiplication kernel

Reductions to matrix multiplication

Matrix Inversion [Strassen 69]

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & -B \\ & I \end{bmatrix} \begin{bmatrix} I & \\ & (D - CA^{-1}B)^{-1} \end{bmatrix} \begin{bmatrix} I & \\ CA^{-1} & I \end{bmatrix}$$

Reductions to matrix multiplication

Matrix Inversion [Strassen 69]

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & -B \\ & I \end{bmatrix} \begin{bmatrix} I & \\ & (D - CA^{-1}B)^{-1} \end{bmatrix} \begin{bmatrix} I & \\ CA^{-1} & I \end{bmatrix}$$

1: Compute $E = A^{-1}$ (Recursive call)
2: Compute $F = D - CEB$ (MM)
3: Compute $G = F^{-1}$ (Recursive call)
4: Compute $H = -EB$ (MM)
5: Compute $J = HG$ (MM)
6: Compute $K = CE$ (MM)
7: Compute $L = E + JK$ (MM)
8: Compute $M = GK$ (MM)
9: Return $\begin{bmatrix} E & J \\ M & G \end{bmatrix}$

Reductions to matrix multiplication

TRSM: Multiple triangular system solving

$$\begin{bmatrix} A & B \\ & C \end{bmatrix}^{-1} \begin{bmatrix} D \\ E \end{bmatrix} = \begin{bmatrix} A^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & -B \\ & I \end{bmatrix} \begin{bmatrix} I & \\ & C^{-1} \end{bmatrix} \begin{bmatrix} D \\ E \end{bmatrix}$$

Reductions to matrix multiplication

TRSM: Multiple triangular system solving

$$\begin{bmatrix} A & B \\ & C \end{bmatrix}^{-1} \begin{bmatrix} D \\ E \end{bmatrix} = \begin{bmatrix} A^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & -B \\ & I \end{bmatrix} \begin{bmatrix} I & \\ & C^{-1} \end{bmatrix} \begin{bmatrix} D \\ E \end{bmatrix}$$

1: Compute $F = C^{-1}E$ (Recursive call)
2: Compute $G = D - BF$ (MM)
3: Compute $H = A^{-1}G$ (Recursive call)
4: Return $\begin{bmatrix} H \\ F \end{bmatrix}$

Reductions to matrix multiplication

LU decomposition

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} L_A & \\ CU_A^{-1} & L_E \end{bmatrix} \begin{bmatrix} U_A & L_A^{-1}B \\ & U_E \end{bmatrix}$$

where $E = D - CA^{-1}B$

Reductions to matrix multiplication

LU decomposition

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} L_A & \\ CU_A^{-1} & L_E \end{bmatrix} \begin{bmatrix} U_A & L_A^{-1}B \\ & U_E \end{bmatrix}$$

where $E = D - CA^{-1}B$

1: Compute $A = L_A U_A$ (Recursive call)
2: Compute $F = CU_A^{-1}$ (TRSM)
3: Compute $G = L_A^{-1}B$ (TRSM)
4: Compute $E = D - FG$ (MM)
5: Compute $E = L_E U_E$ (Recursive call)
6: Return $\left(\begin{bmatrix} L_A & \\ F & L_E \end{bmatrix}, \begin{bmatrix} U_A & G \\ & U_E \end{bmatrix} \right)$

Reductions to matrix multiplication

Divide and conquer approach:

⇒ involves computations with dimensions $\frac{n}{2^i}$ for $i = 1 \dots \log_2 n$

⇒ overall time complexity by geometric progression

$$\sum_{i=1}^{\log_2 n} 2^i \left(\frac{n}{2^i}\right)^\omega = \mathcal{O}\left(n^\omega\right)$$

Reductions to matrix multiplication

Divide and conquer approach:

$\Rightarrow$ involves computations with dimensions $\frac{n}{2^i}$ for $i = 1 \dots \log_2 n$

$\Rightarrow$ overall time complexity by geometric progression

$$\sum_{i=1}^{\log_2 n} 2^i \left(\frac{n}{2^i}\right)^\omega = \mathcal{O}\left(n^\omega\right)$$

These reductions reduce to in $\mathcal{O}\left(n^\omega\right)$ the following problems

- det, rank, rank profile,
- echelon form, inverse, system solving.

Reductions to matrix multiplication

Divide and conquer approach:

$\Rightarrow$ involves computations with dimensions $\frac{n}{2^i}$ for $i = 1 \ldots \log_2 n$

$\Rightarrow$ overall time complexity by geometric progression

$$\sum_{i=1}^{\log_2 n} 2^i \left(\frac{n}{2^i}\right)^\omega = \mathcal{O}\left(n^\omega\right)$$

These reductions reduce to in $\mathcal{O}\left(n^\omega\right)$ the following problems

- det, rank, rank profile,
- echelon form, inverse, system solving.

What about the characteristic polynomial ?

Outline

Outline

Pre-Strassen age

Leverrier 1840: trace of powers of A, and Newton's formula

- improved/rediscovered by Souriau, Faddeev, Frame and Csanky
- $\mathcal{O}\left(n^4\right)$, based on Matrix multiplication
- Suited for parallel computation model

Pre-Strassen age

Leverrier 1840: trace of powers of A, and Newton's formula

- improved/rediscovered by Souriau, Faddeev, Frame and Csanky
- $\mathcal{O}\left(n^4\right)$, based on Matrix multiplication
- Suited for parallel computation model

Danilevskii 1937: elementary row/column operations

$\Rightarrow \mathcal{O}\left(n^3\right)$

Pre-Strassen age

Leverrier 1840: trace of powers of A, and Newton's formula

- improved/rediscovered by Souriau, Faddeev, Frame and Csanky
- $\mathcal{O}\left(n^4\right)$, based on Matrix multiplication
- Suited for parallel computation model

Danilevskii 1937: elementary row/column operations
$\Rightarrow \mathcal{O}\left(n^3\right)$

Hessenberg 1942: transformation to quasi-upper triangular and determinant expansion formula.
$\Rightarrow \mathcal{O}\left(n^3\right)$

Pre-Strassen age

Leverrier 1840: trace of powers of A, and Newton's formula

- improved/rediscovered by Souriau, Faddeev, Frame and Csanky
- $\mathcal{O}\left(n^4\right)$, based on Matrix multiplication
- Suited for parallel computation model

Danilevskii 1937: elementary row/column operations
$\Rightarrow \mathcal{O}\left(n^3\right)$

Hessenberg 1942: transformation to quasi-upper triangular and determinant expansion formula.
$\Rightarrow \mathcal{O}\left(n^3\right)$

But no trivial translation into a block algorithm with $\mathcal{O}\left(n^\omega\right)$ complexity.

Post-Strassen age

Preparata & Sarwate 1978: Update Csanky with fast matrix multiplication

$\Rightarrow \mathcal{O}\left(n^{\omega+1}\right)$

Post-Strassen age

Preparata & Sarwate 1978: Update Csanky with fast matrix multiplication
$\Rightarrow \mathcal{O}\left(n^{\omega+1}\right)$

Keller-Gehrig 1985, alg.1: computes $(A^{2^i})_{i=1\dots\log_2 n}$ to form a Krylov basis.

- $\mathcal{O}\left(n^{\omega}\log n\right)$
- the best complexity up to now

Post-Strassen age

Preparata & Sarwate 1978: Update Csanky with fast matrix multiplication
$\Rightarrow \mathcal{O}\left(n^{\omega+1}\right)$

Keller-Gehrig 1985, alg.1: computes $(A^{2^i})_{i=1\dots\log_2 n}$ to form a Krylov basis.

- $\mathcal{O}\left(n^{\omega}\log n\right)$
- the best complexity up to now

Keller-Gehrig 1985, alg.2: inspired by Danilevskii, block operations

- $\mathcal{O}\left(n^{\omega}\right)$
- Only valid with generic matrices

Outline

1. Matrix multiplication based linear algebra
   - Matrix multiplication: a building block
   - Reductions to matrix multiplication
2. Computing the characteristic polynomial
   - State of the art
   - A new algorithm
   - Algorithm into practice

Statement

Theorem

If A is a $n \times n$ matrix over a field having more than $2n^2$ elements, the characteristic polynomial of A can be computed in $\mathcal{O}(n^\omega)$ field operations by a Las Vegas randomized algorithm.

Definition (degree d Krylov matrix of one vector v)

$$K = \begin{bmatrix} v & Av & \dots & A^{d-1}v \end{bmatrix}$$

Property

$$A \times K = K \times \begin{bmatrix} 0 & & & * \\ 1 & & & * \\ & \ddots & & * \\ & & 1 & * \end{bmatrix}$$

Definition (degree d Krylov matrix of one vector v)

$$K = \begin{bmatrix} v & Av & \dots & A^{d-1}v \end{bmatrix}$$

Property

$$A \times K = K \times \begin{bmatrix} 0 & & & * \\ 1 & & & * \\ & \ddots & & * \\ & & 1 & * \end{bmatrix}$$

$\Rightarrow$ if $d = n$, $K^{-1}AK = C_{P^A_{car}}$

Definition (degree d Krylov matrix of one vector v)

$$K = \begin{bmatrix} v & Av & \dots & A^{d-1}v \end{bmatrix}$$

Property

$$A \times K = K \times \begin{bmatrix} 0 & & & * \\ 1 & & & * \\ & \ddots & & * \\ & & 1 & * \end{bmatrix}$$

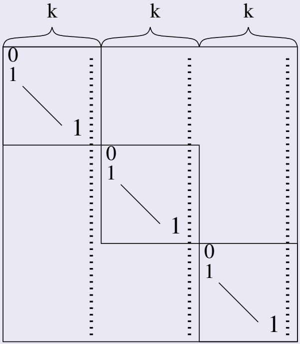
$\Rightarrow$ if $d = n$, $K^{-1}AK = C_{P^A_{car}}$
$\Rightarrow$ [Keller-Gehrig, alg. 2] computes K in $\mathcal{O}(n^\omega)$

Definition (degree k Krylov matrix of several vectors v_i)

$$K = \left[\begin{array}{ccc|ccc|c|ccc} v_1 & \dots & A^{k-1}v_1 & v_2 & \dots & A^{k-1}v_2 & \dots & v_l & \dots & A^{k-1}v_l \end{array} \right]$$

Property

$A \times K = K \times$



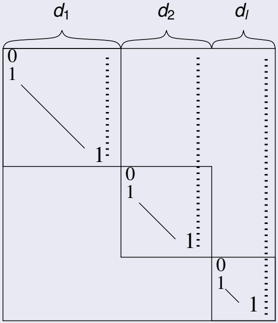
Fact

If $(d_1, \dots d_l)$ is lexicographically maximal such that

$$K = \left[\begin{array}{ccc|c|ccc} v_1 & \dots & A^{d_1-1}v_1 & \dots & v_l & \dots & A^{d_l-1}v_l \end{array} \right]$$

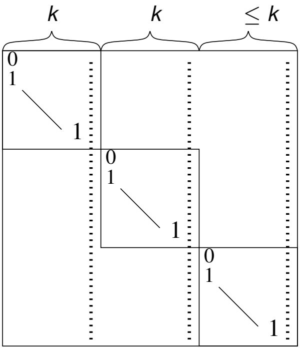
is non-singular, then

$K^{-1}AK =$



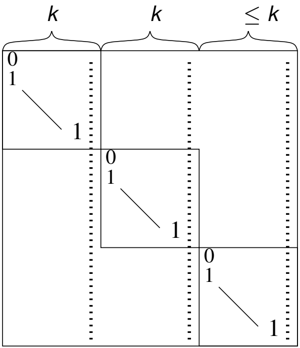
Principle

k-shifted form:



Principle

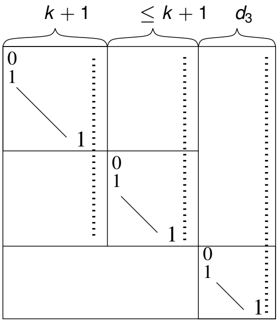
k-shifted form:



- try to inflate each slice by one ...

Principle

$k+1$-shifted form:



- try to inflate each slice by one ...
- ... to obtain the $k+1$-shifted form

Principle

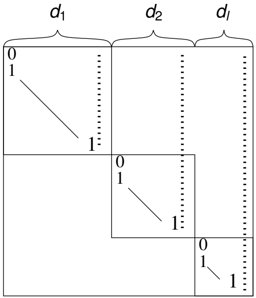
- Compute iteratively from 1-shifted form to d_1-shifted form

Principle

- Compute iteratively from 1-shifted form to d_1-shifted form
- each diagonal block appears in the increasing degree

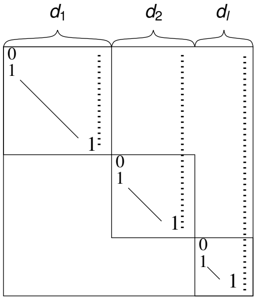
Principle

- Compute iteratively from 1-shifted form to d_1-shifted form
- each diagonal block appears in the increasing degree
- until the shifted Hessenberg form is obtained:



Principle

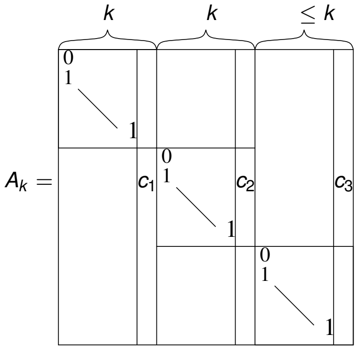
- Compute iteratively from 1-shifted form to d_1-shifted form
- each diagonal block appears in the increasing degree
- until the shifted Hessenberg form is obtained:



How to transform from k to $k+1$-shifted form ?

Krylov normal extension

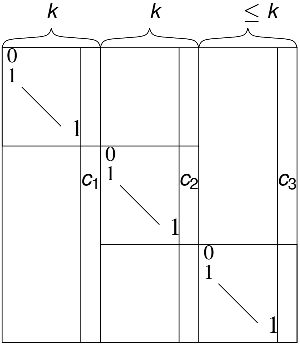
for any k-shifted form



Krylov normal extension

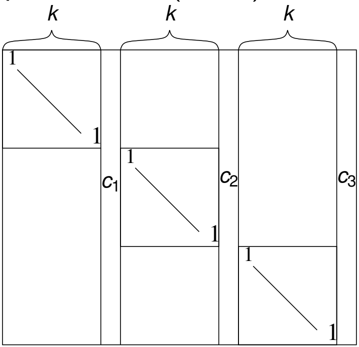
for any k-shifted form

$A_k =$



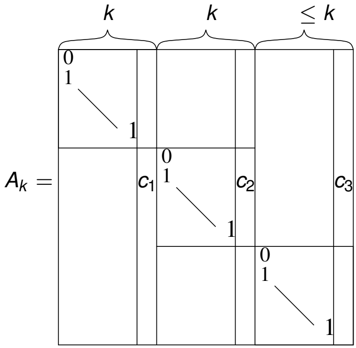
compute the $n \times (n+k)$ matrix

$\overline{K} =$

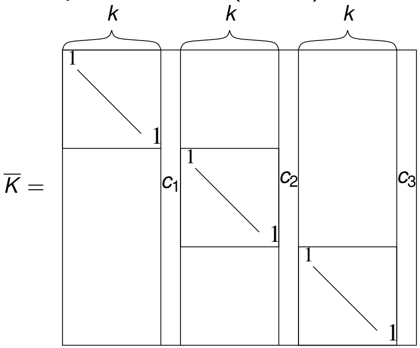


Krylov normal extension

for any k-shifted form



compute the $n \times (n+k)$ matrix



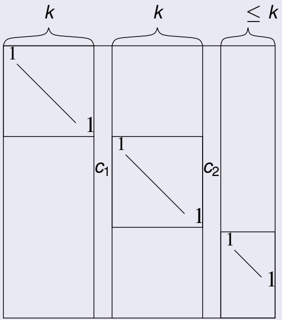
and form K by picking its first linearly independent columns.

Krylov normal extension

Lemma

If $\#F > 2n^2$, with high probability, the matrix K will have the form

$K =$



and $A_{k+1} = K^{-1} A_k K$ will be in $k+1$ shifted form

The algorithm

- Form $\overline{K}$: just copy the columns of A_k

The algorithm

- Form $\overline{K}$: just copy the columns of A_k
- Compute K: rank profile of $\overline{K}$

The algorithm

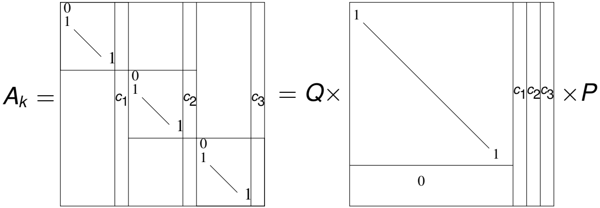
- Form $\overline{K}$: just copy the columns of A_k
- Compute K: rank profile of $\overline{K}$
- Apply the similarity transformation $K^{-1}A_kK$

The algorithm

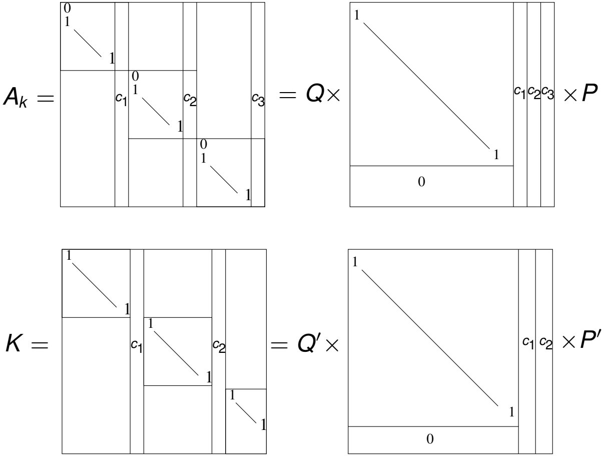
- Form $\overline{K}$: just copy the columns of A_k
- Compute K: rank profile of $\overline{K}$
- Apply the similarity transformation $K^{-1}A_kK$

How to use matrix multiplication knowing the structure ?

Permutations: compressing the dense columns



Permutations: compressing the dense columns



Reduction to Matrix multiplication

Rank profile: derived from LQUP

$\Rightarrow \mathcal{O}\left(k\left(\frac{n}{k}\right)^{\omega}\right)$

Reduction to Matrix multiplication

Rank profile: derived from LQUP
$\Rightarrow \mathcal{O}\left(k\left(\frac{n}{k}\right)^{\omega}\right)$

Similarity transformation: parenthesing

$$K^{-1}AK = Q'^{T}\begin{bmatrix} I & * \\ 0 & * \end{bmatrix} P'^{T} Q \begin{bmatrix} I & * \\ 0 & * \end{bmatrix} PQ' \begin{bmatrix} I & * \\ 0 & * \end{bmatrix} P'$$

Reduction to Matrix multiplication

Rank profile: derived from LQUP

$\Rightarrow \mathcal{O}\left(k\left(\frac{n}{k}\right)^{\omega}\right)$

Similarity transformation: parenthesing

$$K^{-1}AK = Q'^{T}\left(\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\left(P'^{T}Q\left(\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\left(PQ'\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\right)\right)\right)\right)P'$$

Reduction to Matrix multiplication

Rank profile: derived from LQUP

$\Rightarrow \mathcal{O}\left(k\left(\frac{n}{k}\right)^{\omega}\right)$

Similarity transformation: parenthesing

$$K^{-1}AK = Q'^{T}\left(\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\left(P'^{T}Q\left(\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\left(PQ'\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\right)\right)\right)\right)P'$$

$\Rightarrow \mathcal{O}\left(k\left(\frac{n}{k}\right)^{\omega}\right)$

Reduction to Matrix multiplication

Rank profile: derived from LQUP

$\Rightarrow \mathcal{O}\left(k\left(\frac{n}{k}\right)^{\omega}\right)$

Similarity transformation: parenthesing

$$K^{-1}AK = Q'^{T}\left(\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\left(P'^{T}Q\left(\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\left(PQ'\begin{bmatrix} I & * \\ 0 & * \end{bmatrix}\right)\right)\right)\right)P'$$

$\Rightarrow \mathcal{O}\left(k\left(\frac{n}{k}\right)^{\omega}\right)$

Overall complexity: summing for each iteration:

$$\sum_{k=1}^{n} k\left(\frac{n}{k}\right)^{\omega} = n^{\omega}\sum_{k=1}^{n}\left(\frac{1}{k}\right)^{\omega-1} = \mathcal{O}\left(n^{\omega}\right)$$

Outline

Heuristic improvement

The randomization:

the iterate vectors at the first iteration must be random vectors.
or equivalently
the matrix has to be preconditioned: $M^{-1}AM$ for a random matrix M.

$\Rightarrow$as expensive as the rest of the algorithm

A block Krylov preconditoner

1: Pick n/c random vectors $U = \begin{bmatrix} u_1 & \dots & u_{n/c} \end{bmatrix}$.
2: $M = \begin{bmatrix} U & AU & \dots & A^{c-1} \end{bmatrix}$

A block Krylov preconditoner

1: Pick n/c random vectors $U = \begin{bmatrix} u_1 & \dots & u_{n/c} \end{bmatrix}$.
2: $M = \begin{bmatrix} U & AU & \dots & A^{c-1} \end{bmatrix}$
3: **if** M is non singular **then**
4: $\quad M^{-1}AM = H_c$ is in c-shifted form.

A block Krylov preconditoner

Pick n/c random vectors $U = \begin{bmatrix} u_1 & \dots & u_{n/c} \end{bmatrix}$.
$M = \begin{bmatrix} U & AU & \dots & A^{c-1} \end{bmatrix}$
if M is non singular **then**
    $M^{-1}AM = H_c$ is in c-shifted form.
else
    complete M into a non singular matrix $\overline{M}$ by adding some columns at the end
    then $\overline{M}^{-1}A\overline{M} = \begin{bmatrix} H_c & * \\ & R \end{bmatrix}$
end if

Efficiency balancing parameter

- c small: full square matrix multiplications, but more ops
- c large: tends to matrix-vector products, but less ops

Efficiency balancing parameter

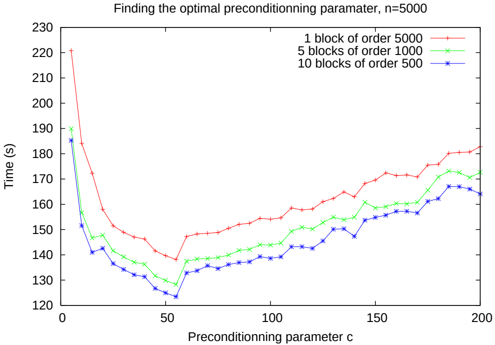
- c small: full square matrix multiplications, but more ops
- c large: tends to matrix-vector products, but less ops

$\Rightarrow$parameter c balances efficiency

Efficiency balancing parameter

- c small: full square matrix multiplications, but more ops
- c large: tends to matrix-vector products, but less ops

$\Rightarrow$ parameter c balances efficiency

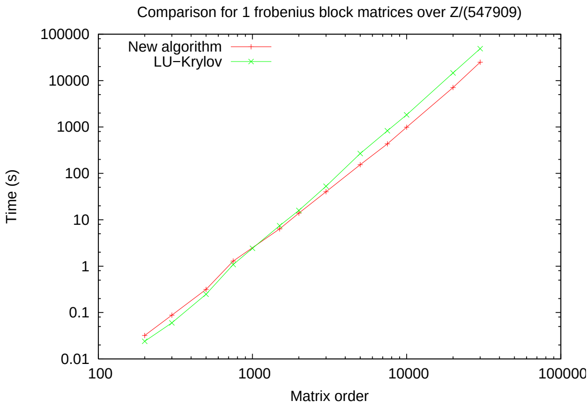


Experiments

n	`LU-Krylov`	`New algorithm`
200	0.024	0.032
300	0.06s	0.088s
500	0.248s	0.316s
750	1.084s	1.288s
1000	2.42s	2.296s
5000	267.6s	153.9s
10 000	1827s	991s
20 000	14 652s	7097s
30 000	48 887s	24 928s

Computation time for 1 Frobenius block matrices, Itanium2-64 1.3Ghz, 192Gb

Experiments



Timing comparison between the new algorithm and LU-Krylov, logarithmic scales, Itanium2-64 1.3Ghz, 192Gb

Comparison to Magma

n	magma-2.11	LU-Krylov	New algorithm
100	0.010s	0.005s	0.006s
300	0.830s	0.294s	0.105s
500	3.810s	1.316s	0.387s
800	15.64s	4.663s	1.387s
1000	29.96s	10.21s	2.755s
1500	102.1s	33.36s	7.696s
2000	238.0s	79.13s	17.91s
3000	802.0s	258.4s	61.09s
5000	3793s	1177s	273.4s
7500	MT	4209s	991.4s
10 000	MT	8847s	2080s

Computation time for 1 Frobenius block matrices, Athlon 2200, 1.8Ghz, 2Gb

MT: Memory thrashing

Conclusion and perspectives

Results:

- Las Vegas reduction to matrix multiplication,
- The Frobenius normal form is easily derivable in $\mathcal{O}(n^\omega)$...
- ...but no transformation matrix
- Adaptive combination with block Krylov in practice.

Conclusion and perspectives

Results:

- Las Vegas reduction to matrix multiplication,
- The Frobenius normal form is easily derivable in $\mathcal{O}(n^\omega)$...
- ...but no transformation matrix
- Adaptive combination with block Krylov in practice.

Still to be done:

- Condition on the size of the field is a limitation. Eberly's algorithm ?
- Ideally: derandomization ? (deterministic)