Online order basis algorithm and its application to block Wiedemann algorithm*

Pascal Giorgi
Université Montpellier II

Romain Lebreton
University of Waterloo
Université Montpellier II

ISSAC'14 - Kobe

July 25, 2014

*. This document has been written using the GNU TEXMACS text editor (see `www.texmacs.org`).

Sparse linear algebra

Sparse linear algebra:

big matrices with few non zero coefficients over a field $\mathbb{K}$

Real challenges need efficiency:

- integer factorization, discrete logarithm [CADO-NFS]
- algebraic K-theory [Dumas et. al 2007]

Problem :

How to provide efficient implementations ?

Review of Wiedemann's methods

Algorithm - Common part of Wiedemann's methods

Input: A sparse matrix $A \in \mathcal{M}_N(\mathbb{K})$

1. Choose random $u, v \in \mathbb{K}^{N \times 1}$
2. Compute the sequence $S = (S_i)_{i \in \mathbb{N}}$ of projections $S_i = {}^t u A^i v \in \mathbb{K}$
3. Return the *minimal generating polynomial* $\pi = \sum_{i=0}^{d} \pi_i x^i$ of S, *i.e.*

$$\forall j, \quad \sum_{i=0}^{d} \pi_i S_{i+j} = 0$$

Remark: Use Berlekamp-Massey or Padé approximants for Step 3.

[Berlekamp '67], [Massey '69], [Padé, 1892]

Applications: [Wiedemann '86], [Kaltofen, Saunders '91]

- sparse linear system solving $A y = b$, kernel $A y = 0$
- rank, determinant, minimal polynomial

Block Wiedemann algorithm

Algorithm - Common part of Block Wiedemann's methods

Input: A sparse matrix $A \in \mathcal{M}_N(\mathbb{K})$

1. Choose random blocks $U, V \in \mathbb{K}^{N \times m}$
2. Compute the sequence $S = (S_i)_{i \in \mathbb{N}}$ of block projections $S_i = U^t A^i V \in \mathbb{K}^{m \times m}$
3. Return the *minimal matrix generating polynomial* Π of S, *i.e.*

$$\forall j, \quad \sum_{i=0}^{d} \Pi_i S_{i+j} = 0^{m \times m} \quad \text{with } \Pi = \sum_{i=0}^{d} \Pi_i x^i$$

Remark: Many approaches for Step 3, *e.g.* order bases, matrix-Berlekamp Massey

[Beckermann, Labahn '94], [Coppersmith '94]

Applications: [Coppersmith '94], [Turner '06]

- sparse linear system solving $A y = b$, kernel $A y = 0$
- rank, determinant, minimal polynomial

Block Wiedemann algorithm

Algorithm - Common part of Block Wiedemann's methods

Input: A sparse matrix $A \in \mathcal{M}_N(\mathbb{K})$

1. Choose random blocks $U, V \in \mathbb{K}^{N \times m}$
2. Compute the sequence $S = (S_i)_{i \in \mathbb{N}}$ of block projections $S_i = U^t A^i V \in \mathbb{K}^{m \times m}$
3. Return the *minimal matrix generating polynomial* Π of S, *i.e.*

$$\forall j, \quad \sum_{i=0}^{d} \Pi_i S_{i+j} = 0^{m \times m} \quad \text{with } \Pi = \sum_{i=0}^{d} \Pi_i x^i$$

Advantages of block Wiedemann algorithm :

- Better probability of success if $\mathbb{K}$ is a small field
- Enable parallelization of the algorithm (step 2)

Block Wiedemann algorithm

Algorithm - Common part of Block Wiedemann's methods

Input: A sparse matrix $A \in \mathcal{M}_N(\mathbb{K})$

1. Choose random blocks $U, V \in \mathbb{K}^{N \times m}$
2. Compute the sequence $S = (S_i)_{i \in \mathbb{N}}$ of block projections $S_i = U^t A^i V \in \mathbb{K}^{m \times m}$
3. Return the *minimal matrix generating polynomial* Π of S, *i.e.*

$$\forall j, \quad \sum_{i=0}^{d} \Pi_i S_{i+j} = 0^{m \times m} \quad \text{with } \Pi = \sum_{i=0}^{d} \Pi_i x^i$$

How many terms $S_0, \ldots, S_\delta$ of the sequence $(S_i)_{i \in \mathbb{N}}$ do we need ?

1. Worst-case (bad choices of U, V):

   $\delta \leqslant 2N$ terms of S

2. Generic case (most choices of U, V), *a priori* bound:

   $\delta \leqslant 2N/m + O(1)$ terms of S [Coppersmith '94], [Kaltofen '95], [Villard '97]

3. Preconditioned A, generic U, V, tighter *a posteriori* bound:

   $\delta = 2r/m + O(1)$ terms of S where $r = \operatorname{rank} A$ [Kaltofen, Villard '04]

Our motivation : Sparse matrix rank computation

One of our motivation:

Rank computation of sparse matrices coming from algebraic K-theory [Dumas et. al 2007]

Matrix	#non zero entry	size $N \times M$	rank deficiency (ratio) $1 - \text{rank} / \min(N, M)$
GL7d16	$14\,M$	$955\,128 \times 460\,261$	29 %
GL7d17	$25\,M$	$1\,548\,650 \times 955\,128$	35%
GL7d18	$35\,M$	$1\,955\,309 \times 1\,548\,650$	40%
GL7d19	$37\,M$	$1\,911\,130 \times 1\,955\,309$	46%
GL7d20	$29\,M$	$1\,437\,547 \times 1\,911\,130$	39%

How many terms of $(S_i)_{i \in \mathbb{N}}$ do we need ?

$\delta = 2\,r/m + O(1)$ but we only know $\delta \leqslant 2\,N/m + O(1)$ *a priori*

$\Rightarrow$ **Use early termination to :**

- detect the required precision δ at runtime
- compute less elements in the projection sequence S (dominant cost)
- reduce costs of block Wiedemann

Early termination in block Wiedemann - Iterative algorithms

Two types of algorithms for minimum matrix generating polynomial (MMGP) Π :

- Type 1: Iterative algorithm

Algorithm - Block Wiedemann using iterative MMGP

1. Choose random $U, V \in \mathbb{K}^{N \times m}$
2. **for** $i = 0 \ldots 2\,N$
   a. Update S from $[S_0, \ldots, S_{i-1}]$ to $[S_0, \ldots, S_i]$
   b. Update MMGP Π of S from precision $i-1$ to i
   c. **if** StopCriteria(S, Π) **then** break
3. return Π

Stop Criteria:

- heuristic, test if Π is stable [Lobo '95], [Kaltofen, Lee '03]
- deterministic, using determinantal degree [Kaltofen, Yuhasz '13]

Early termination in block Wiedemann - Iterative algorithms

Two types of algorithms for minimum matrix generating polynomial (MMGP) Π :

- Type 1: Iterative algorithm

Algorithm - Block Wiedemann using iterative MMGP

1. Choose random $U, V \in \mathbb{K}^{N \times m}$
2. **for** $i = 0 \ldots 2\,N$
   a. Update S from $[S_0, \ldots, S_{i-1}]$ to $[S_0, \ldots, S_i]$
   b. Update MMGP Π of S from precision $i-1$ to i
   c. **if** StopCriteria(S, Π) **then** break
3. return Π

Pros and cons of block Wiedemann using iterative algorithm for MMGP:

Iterative
Early termination of MMGP computation Early termination on input S
Slow MMGP computation (Step 2.b non negligible)

Early termination in block Wiedemann - D-A-C algorithms

Two types of algorithms for minimum matrix generating polynomial (MMGP) Π :

- Type 2: Divide-and-conquer algorithm

Algorithm - Block Wiedemann using divide-and-conquer MMGP
1. Choose random $U, V \in \mathbb{K}^{N \times m}$
2. **for** $\ell = 0 \ldots \lceil \log_2(2N) \rceil$ //Assume $2N = 2^k$
a. Update S from $[S_0, \ldots, S_{2^{\ell-1}-1}]$ to $[S_0, \ldots, S_{2^\ell-1}]$
b. Update MMGP Π of S from precision $2^{\ell-1}$ to 2^ℓ
c. **if** StopCriteria(S, Π) **then** break
3. return Π

Pros and cons of block Wiedemann using divide-and-conquer algorithm for MMGP:

Divide-and-conquer

Restrictive early termination of MMGP computation
Restrictive early termination on input S

Fast MMGP computation (negligible)

Our contribution

Previous approaches for early termination in block Wiedemann :

Iterative	Divide-and-conquer
Early termination of MMGP computation Early termination on MMGP's input S	Restrictive early termination of computations Restrictive early termination of input
Slow MMGP	Fast MMGP

Our contribution :

Design fast *online* algorithms for MMGP

Online
Early termination of MMGP computation Early termination on input S
Fast MMGP

Order Basis - Definition

We use *order bases* to compute minimal matrix generating polynomial

Definition. *Let $\mathbb{K}$ be a field and $F \in \mathbb{K}[[x]]^{m \times n}$.*

Let (F, σ) be the $\mathbb{K}[x]$-module $\{v \in \mathbb{K}[[x]]^{1 \times m}$ such that $vF = 0 \bmod x^{\sigma}\}$.

An (F, σ) order basis P is a basis of (F, σ) of minimal row degree.

Example.

$$\underbrace{\begin{pmatrix} 1 & 0 & 1 & 1 \\ x & 1 & 1+x & 0 \\ 1 & x^2+x^3 & x & 0 \\ x^2 & 0 & x^3+x^4 & 0 \end{pmatrix}}_{\text{Output: } (F,8)-\text{order basis over } \mathbb{F}_2} \underbrace{\begin{pmatrix} x+x^2+x^3+x^4+x^5+x^6 \\ 1+x+x^5+x^6+x^7 \\ 1+x^2+x^4+x^5+x^6+x^7 \\ 1+x+x^3+x^7 \end{pmatrix}}_{\text{Input: } F \text{ in } \mathbb{F}_2[[x]]^{4 \times 1}} = 0^{4 \times 1} \bmod x^8$$

Online order basis algorithm

Definition: Online order basis algorithm $\simeq$ Minimal requirement on the input

An order basis algorithm is *online* if

1. it computes the order bases one after the others : $(F,1) \to (F,2) \to (F,3) \to \cdots$
2. it reads at most $F \bmod x^{\sigma}$ when computing a (F,σ)-order basis.

Remark:

- Iterative order basis algorithm is online but quadratic in σ
- Divide-and-conquer order basis algorithm is not online but is quasi-linear in σ

Our goal:

Design a quasi-linear online order basis algorithm

Towards an online order basis algorithm - First step

All algorithms reduce computation of (F, σ) to several $\left(F^{(k)}, 1\right)$ (base cases)

If base cases are $M^{(k)} \in \mathbb{K}[x]^{m \times n}_{\leqslant 1}$, then

$$(F, \sigma)\text{-order basis is } M^{(\sigma-1)} M^{(\sigma-2)} \cdots M^{(0)}$$

First idea: Store the (F, k)-order bases $M^{(k-1)} \cdots M^{(0)}$ as a *partially unevaluated product*.

Why? Because of the cost.

Remark: We discover $M^{(k)}$ during the computation of the $(F, k+1)$-order basis

Towards an online order basis algorithm - First step

All algorithms reduce computation of (F, σ) to several $\left(F^{(k)}, 1\right)$ (base cases)

If base cases are $M^{(k)} \in \mathbb{K}[x]^{m \times n}_{\leqslant 1}$, then

$$(F, \sigma)\text{-order basis is } M^{(\sigma-1)} M^{(\sigma-2)} \cdots M^{(0)}$$

First idea: Store the (F, k)-order bases $M^{(k-1)} \cdots M^{(0)}$ as a *partially unevaluated product*.

How do we compute the products $M^{(k-1)} \cdots M^{(0)}$?

Computation of $(F, 1)$

Output: $\left[M^{(0)}\right]$

$M^{(0)}$

Towards an online order basis algorithm - First step

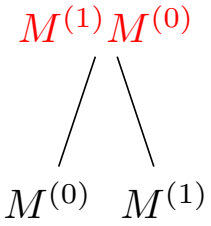
All algorithms reduce computation of (F, σ) to several $\left(F^{(k)}, 1\right)$ (base cases)

If base cases are $M^{(k)} \in \mathbb{K}[x]^{m \times n}_{\leqslant 1}$, then

$$(F, \sigma)\text{-order basis is } M^{(\sigma-1)} M^{(\sigma-2)} \cdots M^{(0)}$$

First idea: Store the (F, k)-order bases $M^{(k-1)} \cdots M^{(0)}$ as a *partially unevaluated product*.

How do we compute the products $M^{(k-1)} \cdots M^{(0)}$?



Computation of $(F, 2)$

Output: $\left[M^{(1)} \cdot M^{(0)}\right]$

Towards an online order basis algorithm - First step

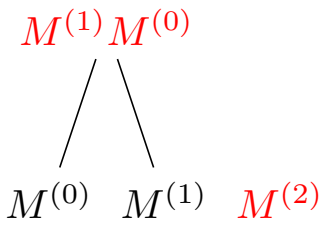
All algorithms reduce computation of (F, σ) to several $\left(F^{(k)}, 1\right)$ (base cases)

If base cases are $M^{(k)} \in \mathbb{K}[x]^{m \times n}_{\leqslant 1}$, then

$$(F, \sigma)\text{-order basis is } M^{(\sigma-1)} M^{(\sigma-2)} \cdots M^{(0)}$$

First idea: Store the (F, k)-order bases $M^{(k-1)} \cdots M^{(0)}$ as a *partially unevaluated product*.

How do we compute the products $M^{(k-1)} \cdots M^{(0)}$?



Computation of $(F, 3)$

Output: $\left[M^{(2)}, M^{(1)} \cdot M^{(0)}\right]$

Towards an online order basis algorithm - First step

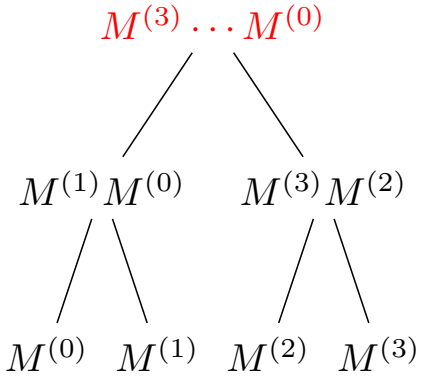
All algorithms reduce computation of (F, σ) to several $\left(F^{(k)}, 1\right)$ (base cases)

If base cases are $M^{(k)} \in \mathbb{K}[x]^{m \times n}_{\leqslant 1}$, then

$$(F, \sigma)\text{-order basis is } M^{(\sigma-1)} M^{(\sigma-2)} \cdots M^{(0)}$$

First idea: Store the (F, k)-order bases $M^{(k-1)} \cdots M^{(0)}$ as a *partially unevaluated product*.

How do we compute the products $M^{(k-1)} \cdots M^{(0)}$?



Computation of $(F, 4)$

Output: $\left[M^{(3)} \cdots M^{(0)}\right]$

Towards an online order basis algorithm - First step

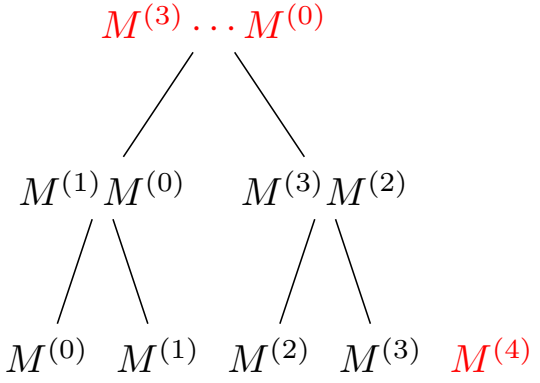
All algorithms reduce computation of (F, σ) to several $\left(F^{(k)}, 1\right)$ (base cases)

If base cases are $M^{(k)} \in \mathbb{K}[x]^{m \times n}_{\leqslant 1}$, then

$$(F, \sigma)\text{-order basis is } M^{(\sigma-1)} M^{(\sigma-2)} \cdots M^{(0)}$$

First idea: Store the (F, k)-order bases $M^{(k-1)} \cdots M^{(0)}$ as a *partially unevaluated product*.

How do we compute the products $M^{(k-1)} \cdots M^{(0)}$?



Computation of $(F, 5)$

Output: $\left[M^{(4)}, M^{(3)} \cdots M^{(0)}\right]$

Towards an online order basis algorithm - First step

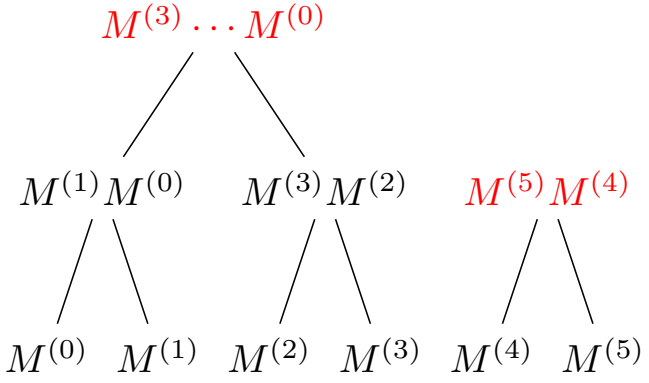
All algorithms reduce computation of (F, σ) to several $\left(F^{(k)}, 1\right)$ (base cases)

If base cases are $M^{(k)} \in \mathbb{K}[x]^{m \times n}_{\leqslant 1}$, then

$$(F, \sigma)\text{-order basis is } M^{(\sigma-1)} M^{(\sigma-2)} \cdots M^{(0)}$$

First idea: Store the (F, k)-order bases $M^{(k-1)} \cdots M^{(0)}$ as a *partially unevaluated product*.

How do we compute the products $M^{(k-1)} \cdots M^{(0)}$?



Computation of $(F, 6)$

Output: $\left[M^{(5)} \cdot M^{(4)}, M^{(3)} \cdots M^{(0)}\right]$

Towards an online order basis algorithm - First step

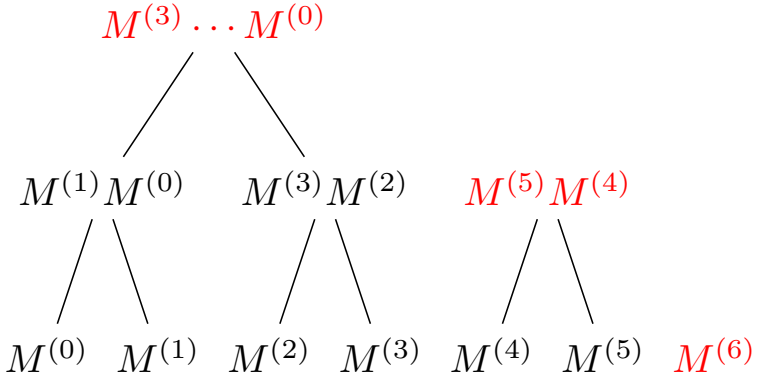
All algorithms reduce computation of (F, σ) to several $\left(F^{(k)}, 1\right)$ (base cases)

If base cases are $M^{(k)} \in \mathbb{K}[x]^{m \times n}_{\leqslant 1}$, then

$$(F, \sigma)\text{-order basis is } M^{(\sigma-1)} M^{(\sigma-2)} \cdots M^{(0)}$$

First idea: Store the (F, k)-order bases $M^{(k-1)} \cdots M^{(0)}$ as a *partially unevaluated product*.

How do we compute the products $M^{(k-1)} \cdots M^{(0)}$?



Computation of $(F, 7)$

Output: $\left[M^{(6)}, M^{(5)} \cdot M^{(4)}, M^{(3)} \cdots M^{(0)}\right]$

Towards an online order basis algorithm - First step

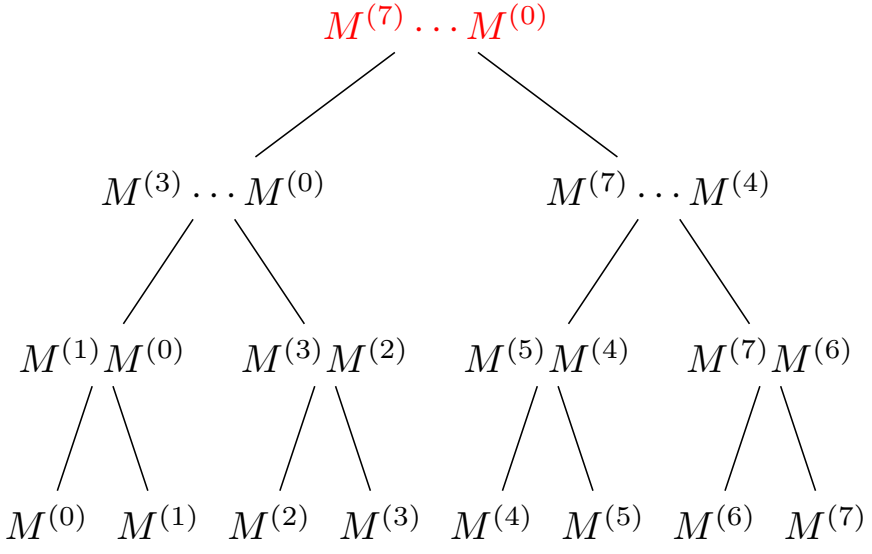
All algorithms reduce computation of (F, σ) to several $\left(F^{(k)}, 1\right)$ (base cases)

If base cases are $M^{(k)} \in \mathbb{K}[x]^{m \times n}_{\leqslant 1}$, then

$$(F, \sigma)\text{-order basis is } M^{(\sigma-1)} M^{(\sigma-2)} \cdots M^{(0)}$$

First idea: Store the (F, k)-order bases $M^{(k-1)} \cdots M^{(0)}$ as a *partially unevaluated product*.

How do we compute the products $M^{(k-1)} \cdots M^{(0)}$?



Computation of $(F, 8)$

Output: $\left[M^{(7)} \cdots M^{(0)}\right]$

Towards an online order basis algorithm - Second step

Overview of divide-and-conquer order basis algorithm PM-Basis:

Algorithm PM-Basis(F, σ)

1. **if** $\sigma = 1$ **then** Base Case
2. **else**
3. $\quad P_\ell := \text{PM-Basis}(F, \lfloor \sigma/2 \rfloor)$ //First recursive call
4. $\quad F' := \text{MiddleProduct}(P_\ell, F)$ //Update input
5. $\quad P_h := \text{PM-Basis}(F', \lceil \sigma/2 \rceil)$ //Second recursive call
6. **return** $P_h \cdot P_\ell$

Step 4 "Update input"

- anticipates computations $\Rightarrow$ good complexity of PM-Basis
- BUT reads too many coefficients of F $\Rightarrow$ PM-Basis is not online

Second idea: Use fast online arithmetic for the middle product of Step 4

$\Rightarrow$ Spread the computations of F' over time, and compute its coefficients just in time.

Shifted online middle product

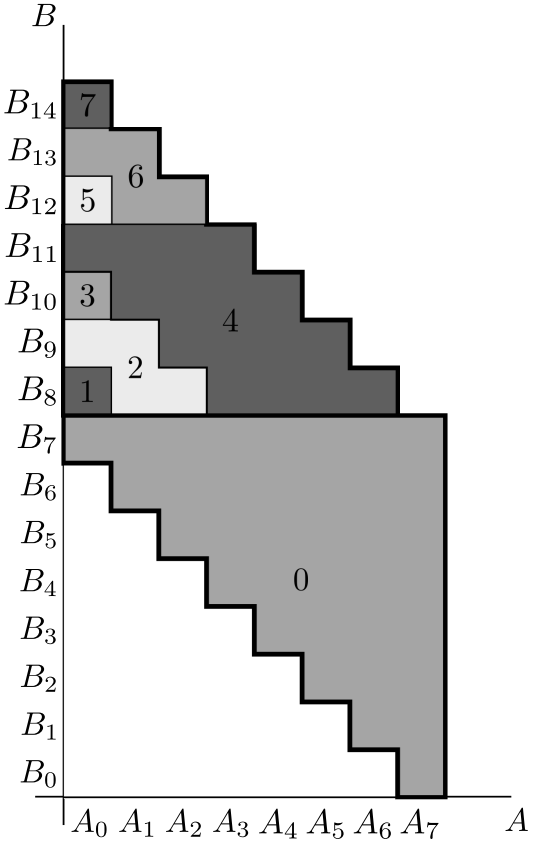
Example of online middle product: Compute

$$C = \left(\left(\sum_{i=0}^{7} A_i x^i\right) \cdot \left(\sum_{i=0}^{14} B_i x^i\right)\right)_{7\ldots14}$$

Shifted online algorithm OnlineMiddleProduct:

- Step i computes C_i and reads at most $B_0, \ldots, B_{i+7}$
- Online algorithm is build on top of "classical" middle product algorithm MidProd

Step	computes	Computation
0	$(A\,B)_7$	$C \quad = (A\,B_{0\ldots7})_{7\ldots14}$
1	$(A\,B)_8$	$C \mathrel{+}= x \;\; \mathrm{MidProd}(A_0, B_8, 0)$
2	$(A\,B)_9$	$C \mathrel{+}= x^2\,\mathrm{MidProd}(A_{0\ldots2}, B_{8\ldots9}, 1\ldots2)$
3	$(A\,B)_{10}$	$C \mathrel{+}= x^3\,\mathrm{MidProd}(A_0, B_{10}, 0)$
4	$(A\,B)_{11}$	$C \mathrel{+}= x^4\,\mathrm{MidProd}(A_{0\ldots6}, B_{8\ldots11}, 3\ldots6)$
5	$(A\,B)_{12}$	$C \mathrel{+}= x^5\,\mathrm{MidProd}(A_0, B_{12}, 0)$
6	$(A\,B)_{13}$	$C \mathrel{+}= x^6\,\mathrm{MidProd}(A_{0\ldots2}, B_{12\ldots13}, 1\ldots2)$
7	$(A\,B)_{14}$	$C \mathrel{+}= x^7\,\mathrm{MidProd}(A_0, B_{14}, 0)$



Timings - Online order basis algorithms

Using partially unevaluated product and online middle product,
we obtain a fast online order basis algorithm in $\tilde{\mathcal{O}}(m^\omega \sigma)$

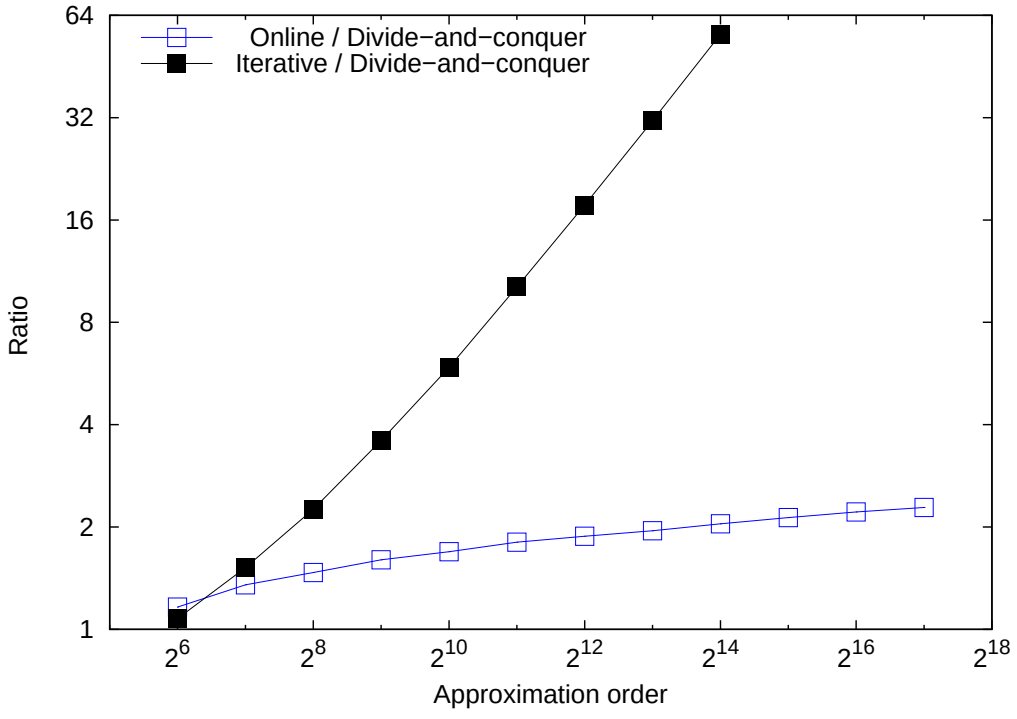


Figure. Relative performance of online order basis algorithms against offline algorithm PM-Basis. Matrices in $\mathbb{F}_p[[x]]^{16\times 16}$ with p a 23-bit FFT prime

Online order basis algorithm

Algorithm - Block Wiedemann using online MMGP

1. Choose random $U, V \in \mathbb{K}^{N \times m}$
2. **for** $i = 0 \ldots 2N$
   a. Update S from $[S_0, \ldots, S_{i-1}]$ to $[S_0, \ldots, S_i]$
   b. Update MMGP Π of S from precision $i-1$ to i using online order basis
   c. **if** StopCriteria(S, Π) **then** break
3. return Π

Pros and cons of block Wiedemann w.r.t. MMGP algorithm:

Iterative	Divide-and-conquer	Online order basis
Early termination Minimal knowledge on S	No early termination More knowledge on S	Early termination Minimal knowledge on S
Slow Step 2.b	Fast Step 2.b	Fast Step 2.b

Timings - Early termination in block Wiedemann

Example: $A \in \mathrm{GL}_{2^{17}}(\mathbb{F}_p)$ with 20 elements per row, $m = 16$, *a priori* bound $\delta \leqslant 16384$

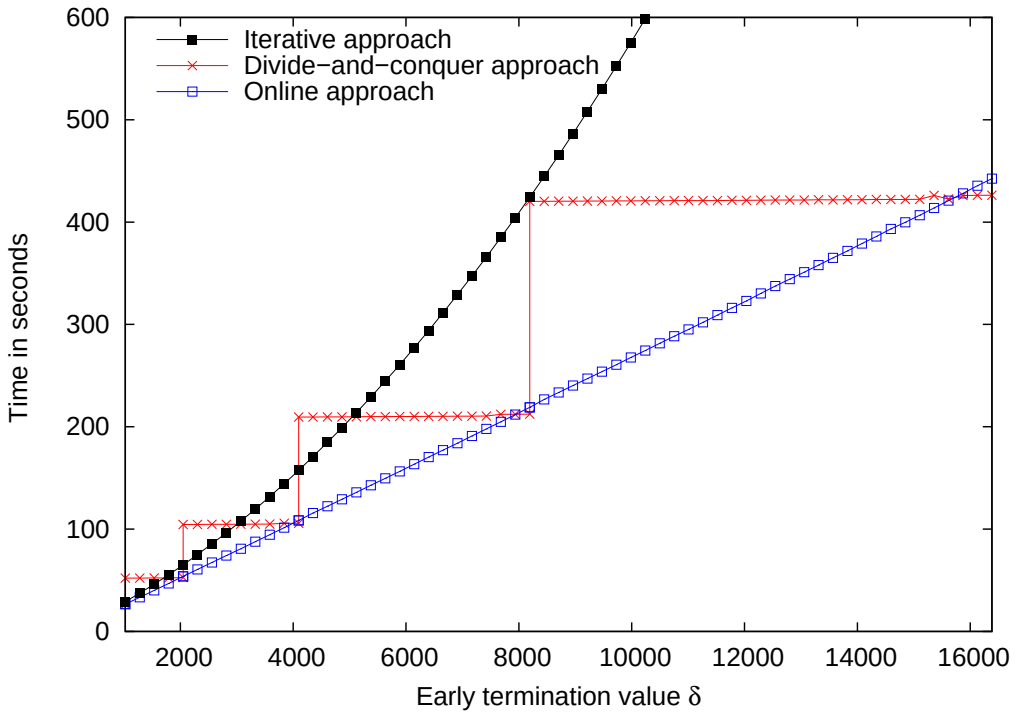


Figure. Computation times of early termination in block Wiedemann using order basis algorithm.

$\Rightarrow$ Online approach gains up to a factor 2

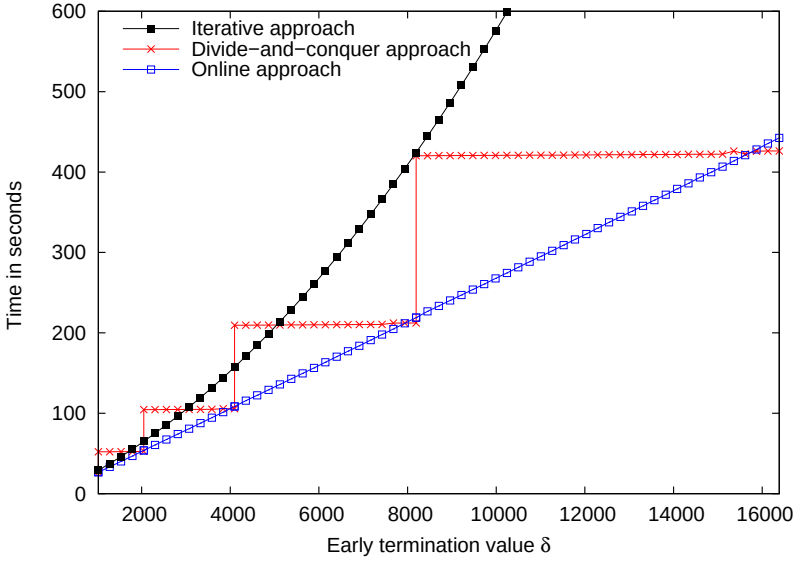
Conclusion - Future Work

Conclusion :

Online order basis

Early termination
Minimal knowledge on S

Fast : complexity $\tilde{\mathcal{O}}(m^\omega \sigma)$



Implementation will soon be released in [LinBox]

Future work :

- Apply online order basis algorithm to other contexts
- More generally, apply online algorithms to other domains

(linear algebra, polynomial equations, ...)