

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

PASCAL GIORGI

Université Montpellier II

ROMAIN LEBRETON

Université Montpellier II
University of Waterloo

JNCF'14 - Luminy

November 3rd, 2014

*. This document has been written using the GNU T_EX_L 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* $\pi(x) = \sum_{i=0}^d \pi_i x^i$ 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 $Ay = b$, kernel $Ay = 0$
- 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* $\Pi(x)$ of S , i.e.

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

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 $Ay = b$, kernel $Ay = 0$
- 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* $\Pi(x)$ of S , i.e.

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

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* $\Pi(x)$ of S , i.e.

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

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

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

$$\delta \leq 2N \text{ terms of } S$$

2. Generic case (most choices of U, V):

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

3. Preconditioned A , generic U, V :

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

Our motivation : Sparse matrix rank computation

One of our motivation:

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

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

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

Matrix	#non zero entries	size $N \times M$	rank / $\min(N, M)$
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%

\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** $i = 0 \dots 2N$
 - a. Update S from $[S_0, \dots, S_{i-1}]$ to $[S_0, \dots, S_i]$
 - 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** $i = 0 \dots 2N$
 - a. Update S from $[S_0, \dots, S_{i-1}]$ to $[S_0, \dots, S_i]$
 - 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** $\ell = 0 \dots \lceil \log_2(2N) \rceil$ // Assume $2N = 2^k$
 - a. Update S from $[S_0, \dots, S_{2^{\ell-1}-1}]$ to $[S_0, \dots, S_{2^\ell-1}]$
 - 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)

Early termination in block Wiedemann - Timings

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

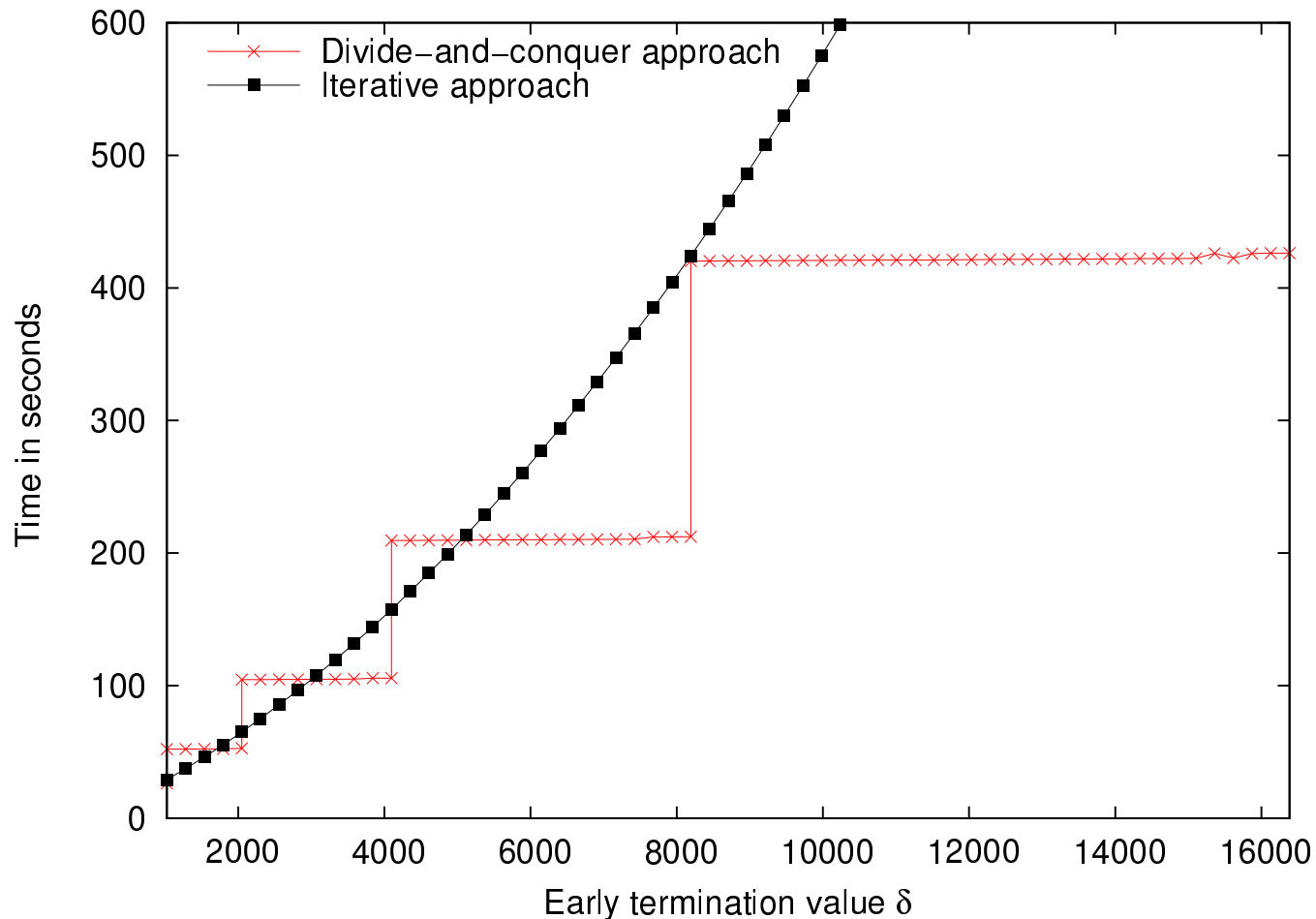


Figure. Computation times of early termination in block Wiedemann using order basis algorithm.

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

Our contribution - Timings

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

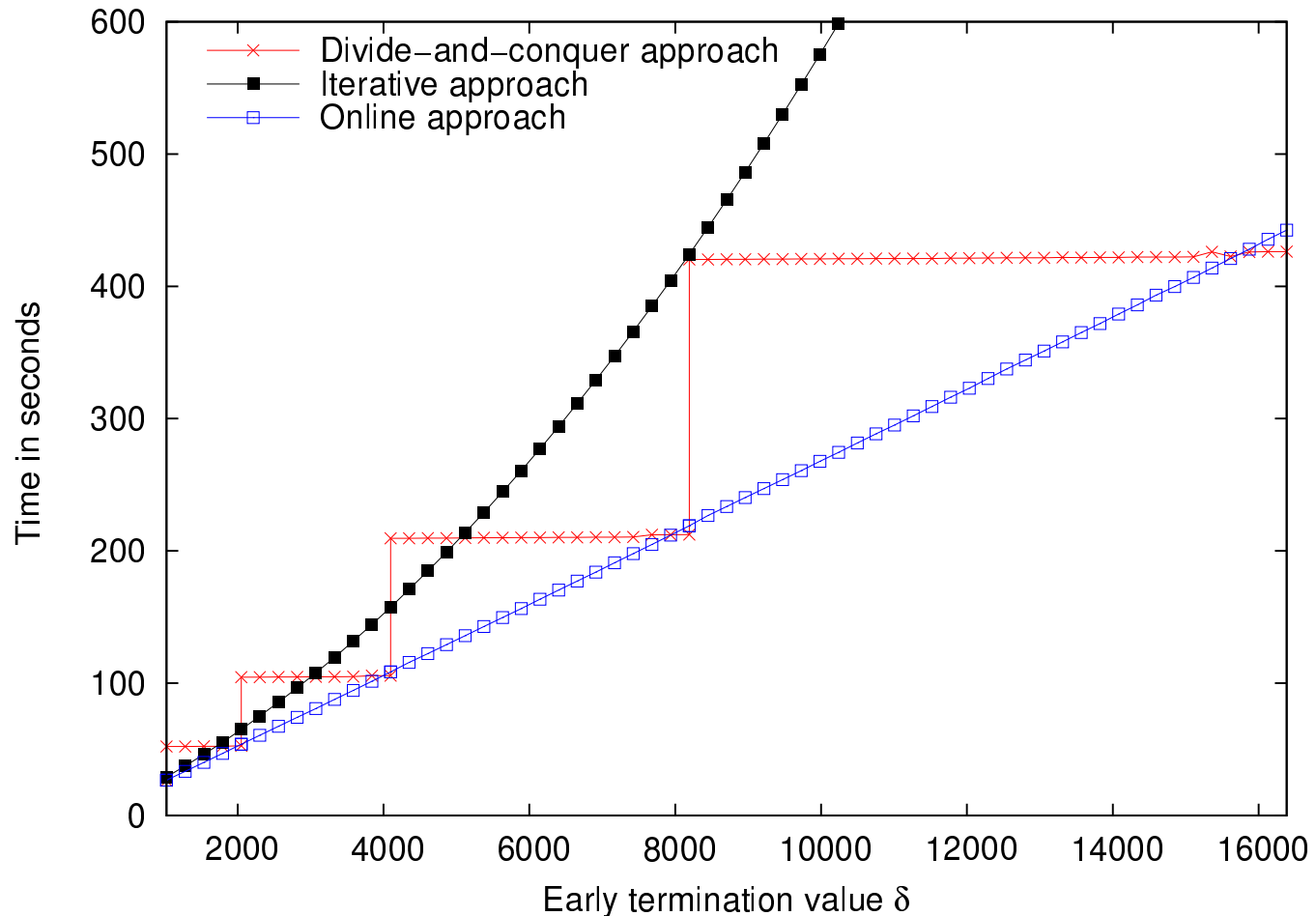


Figure. Computation times of early termination in block Wiedemann using order basis algorithm.

Order Basis - Definition

► We use *order bases* to compute MMGP

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

$$\{v \in \mathbb{K}[[x]]^{1 \times m} \text{ s.t. } vF = 0 \text{ mod } x^\sigma\}.$$

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

Example.

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

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 : $(F, 1) \rightarrow (F, 2) \rightarrow (F, 3) \rightarrow \dots$
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 base cases $(F^{(k)}, 1)$

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

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

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

Why? Because of the cost would be *quadratic* if we computed **all** (F, k) -order bases.

Towards an online order basis algorithm - First step

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

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

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

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

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

Computation of $(F, 1)$

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

$M^{(0)}$

Towards an online order basis algorithm - First step

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

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

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

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

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

$$\begin{array}{c} M^{(1)} M^{(0)} \\ \diagdown \quad \diagup \\ M^{(0)} \quad M^{(1)} \end{array}$$

Computation of $(F, 2)$

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

Towards an online order basis algorithm - First step

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

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

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

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

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

$$\begin{array}{c} M^{(1)} M^{(0)} \\ \diagdown \quad \diagup \\ M^{(0)} \quad M^{(1)} \quad M^{(2)} \end{array}$$

Computation of $(F, 3)$

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

Towards an online order basis algorithm - First step

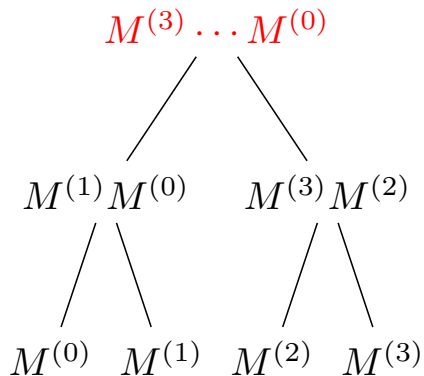
All algorithms reduce computation of (F, σ) to several base cases $(F^{(k)}, 1)$

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

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

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

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



Computation of $(F, 4)$

Output: $[M^{(3)} \dots M^{(0)}]$

Towards an online order basis algorithm - First step

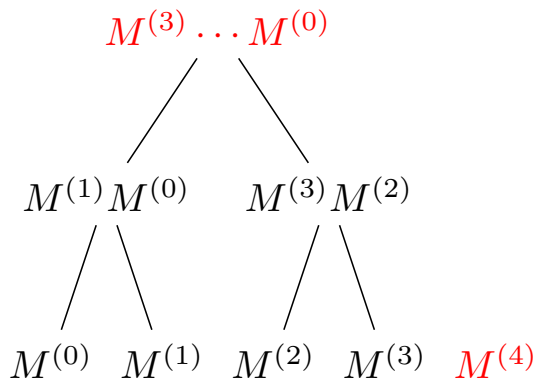
All algorithms reduce computation of (F, σ) to several base cases $(F^{(k)}, 1)$

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

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

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

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



Computation of $(F, 5)$

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

Towards an online order basis algorithm - First step

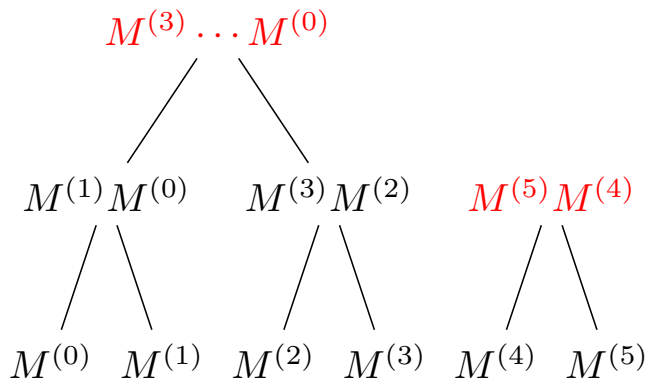
All algorithms reduce computation of (F, σ) to several base cases $(F^{(k)}, 1)$

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

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

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

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



Computation of $(F, 6)$

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

Towards an online order basis algorithm - First step

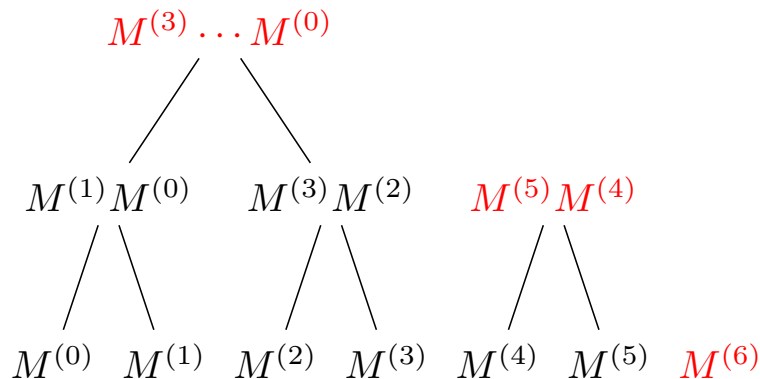
All algorithms reduce computation of (F, σ) to several base cases $(F^{(k)}, 1)$

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

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

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

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



Computation of $(F, 7)$

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

Towards an online order basis algorithm - First step

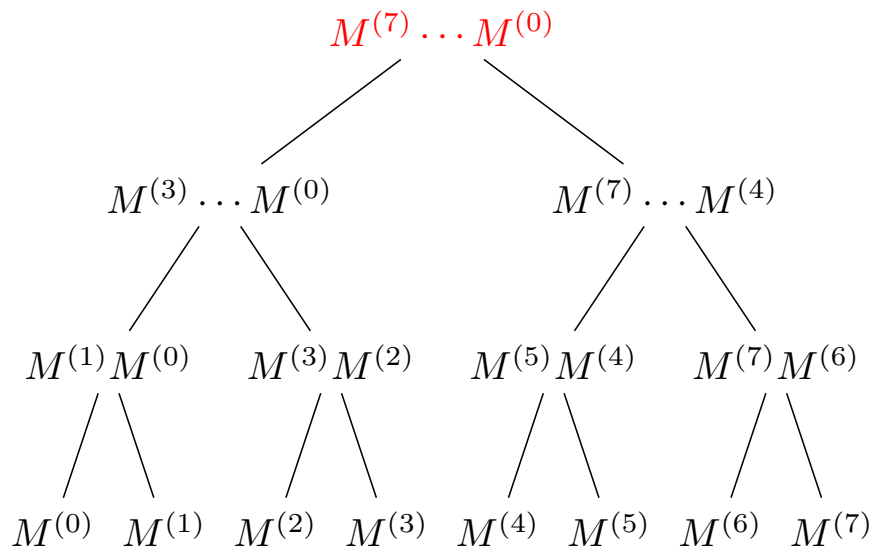
All algorithms reduce computation of (F, σ) to several base cases $(F^{(k)}, 1)$

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

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

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

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



Computation of $(F, 8)$

Output: $[M^{(7)} \dots M^{(0)}]$

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. $P_\ell := \text{PM-Basis}(F, \lfloor \sigma/2 \rfloor)$ //First recursive call
4. $F' := \text{MiddleProduct}(P_\ell, F)$ //Update input
5. $P_h := \text{PM-Basis}(F', \lceil \sigma/2 \rceil)$ //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

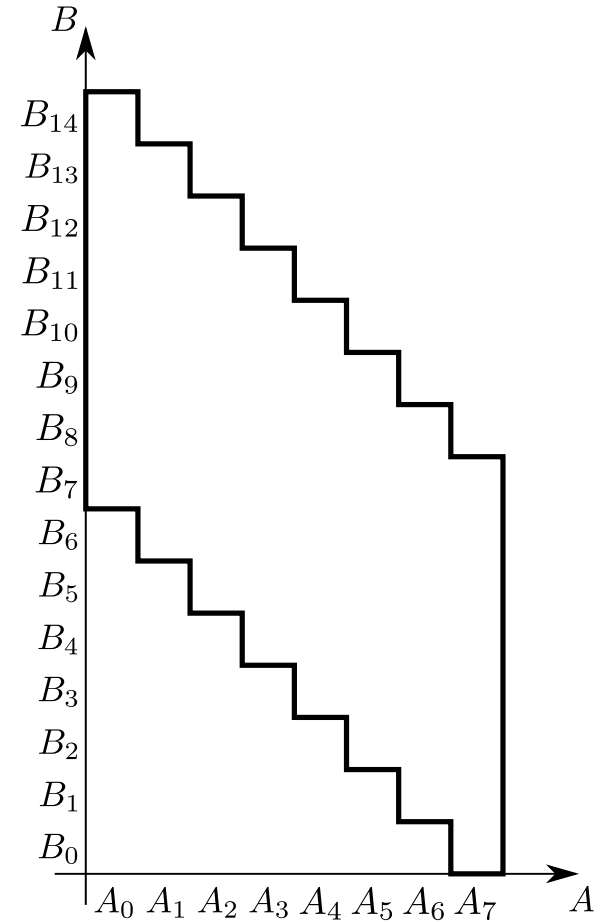
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation



Shifted online middle product

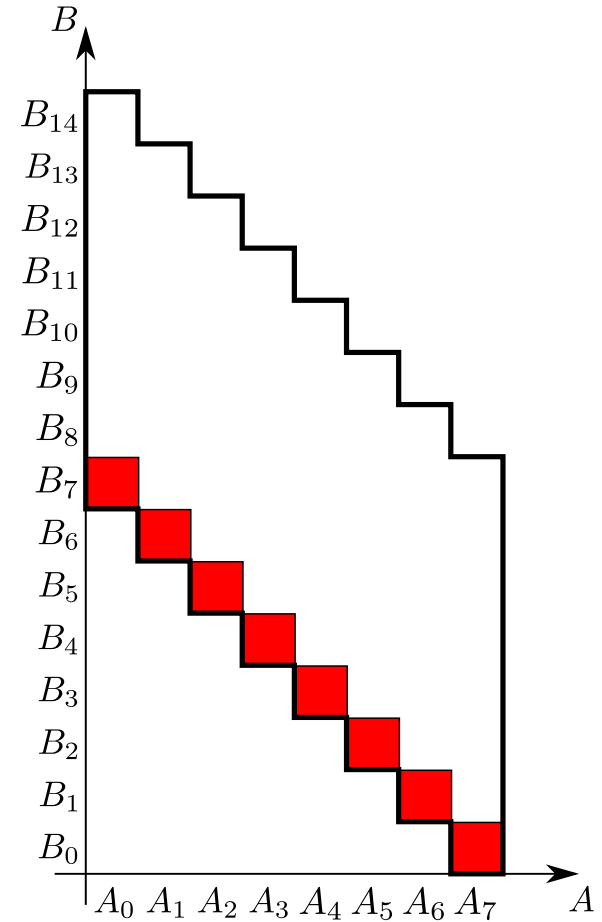
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	



Shifted online middle product

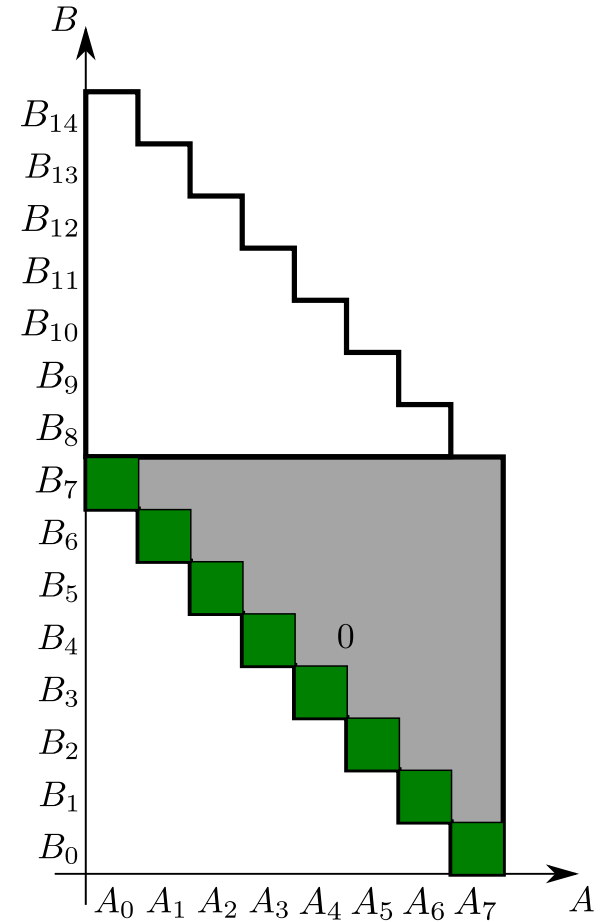
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0 \dots 7})_{7 \dots 14}$



Shifted online middle product

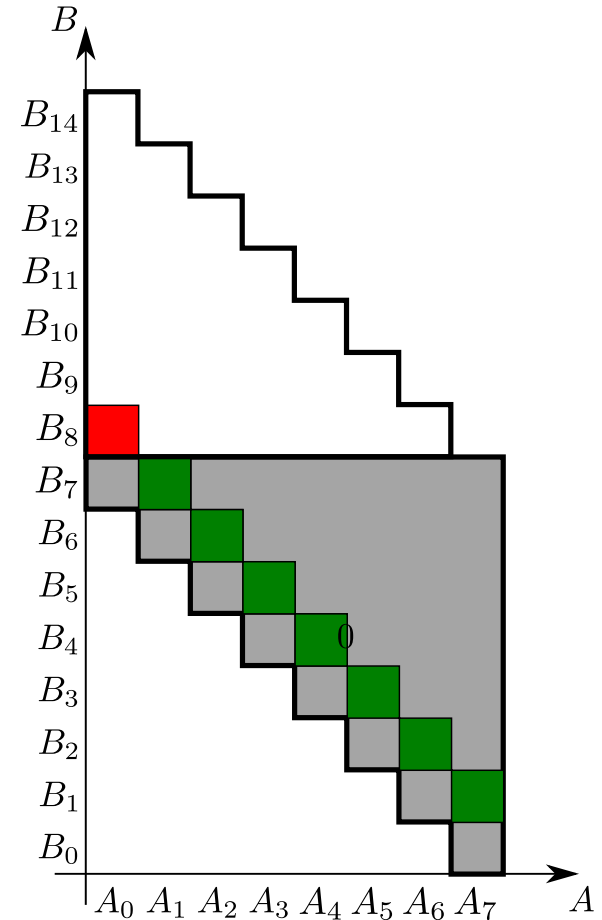
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0 \dots 7})_{7 \dots 14}$
1	$(AB)_8$	



Shifted online middle product

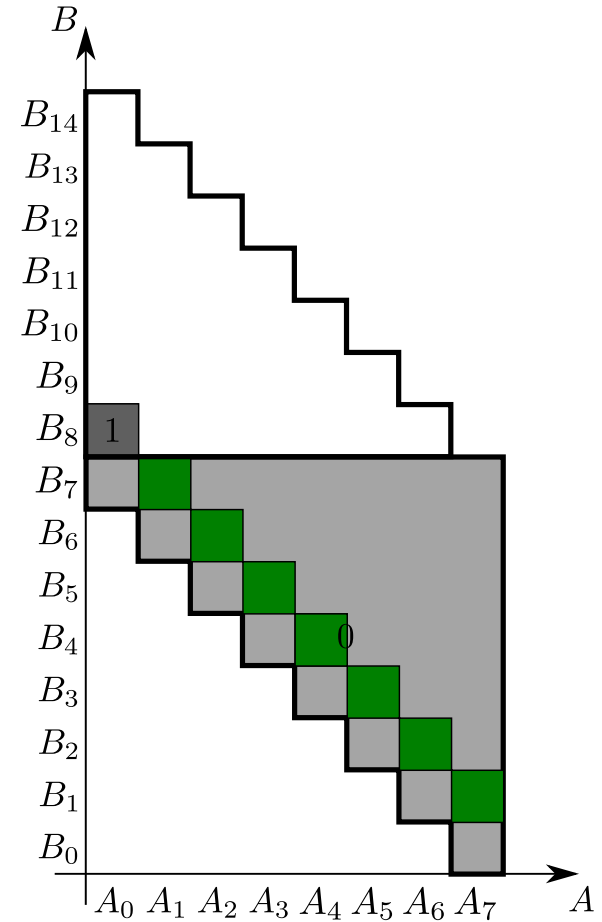
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0\dots 7})_{7\dots 14}$
1	$(AB)_8$	$C += x \text{ MidProd}(A_0, B_8, 0)$



Shifted online middle product

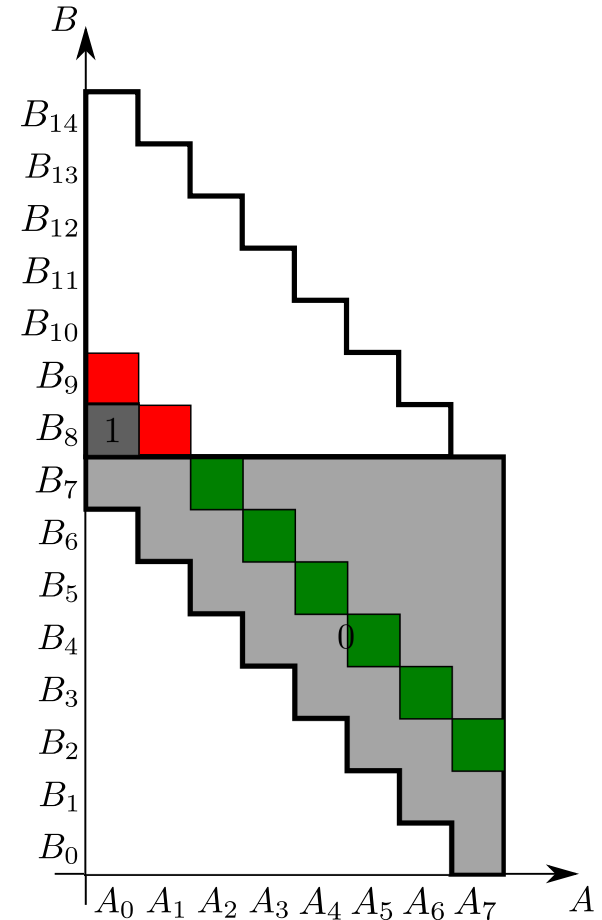
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0\dots7})_{7\dots14}$
1	$(AB)_8$	$C += x \text{ MidProd}(A_0, B_8, 0)$
2	$(AB)_9$	



Shifted online middle product

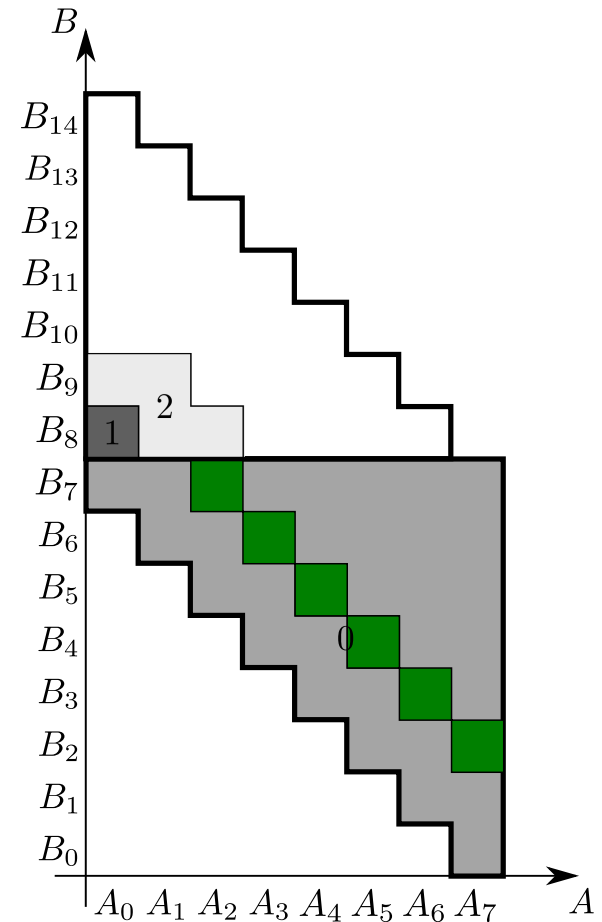
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0 \dots 7})_{7 \dots 14}$
1	$(AB)_8$	$C += x \text{ MidProd}(A_0, B_8, 0)$
2	$(AB)_9$	$C += x^2 \text{ MidProd}(A_{0 \dots 2}, B_{8 \dots 9}, 1 \dots 2)$



Shifted online middle product

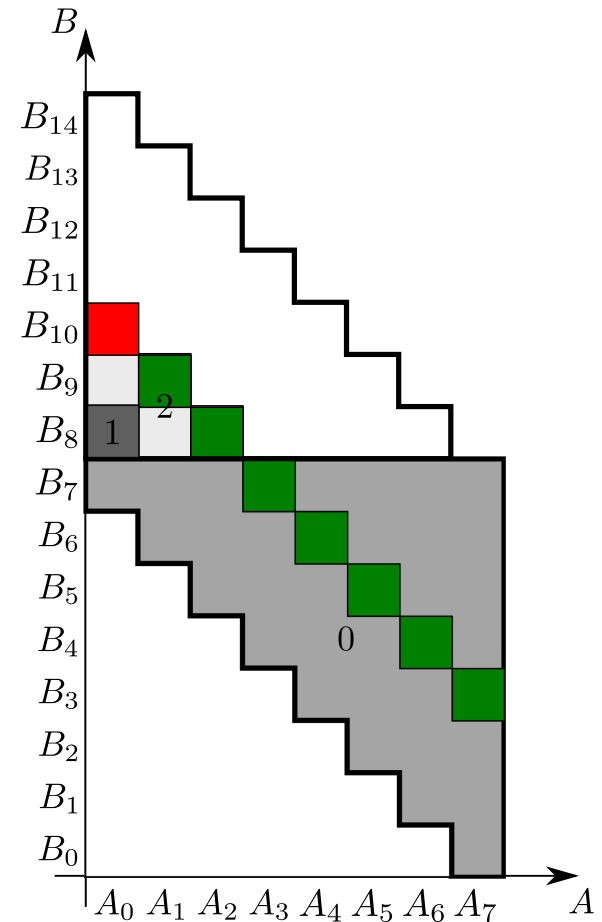
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0\dots 7})_{7\dots 14}$
1	$(AB)_8$	$C += x \text{ MidProd}(A_0, B_8, 0)$
2	$(AB)_9$	$C += x^2 \text{ MidProd}(A_{0\dots 2}, B_{8\dots 9}, 1\dots 2)$
3	$(AB)_{10}$	



Shifted online middle product

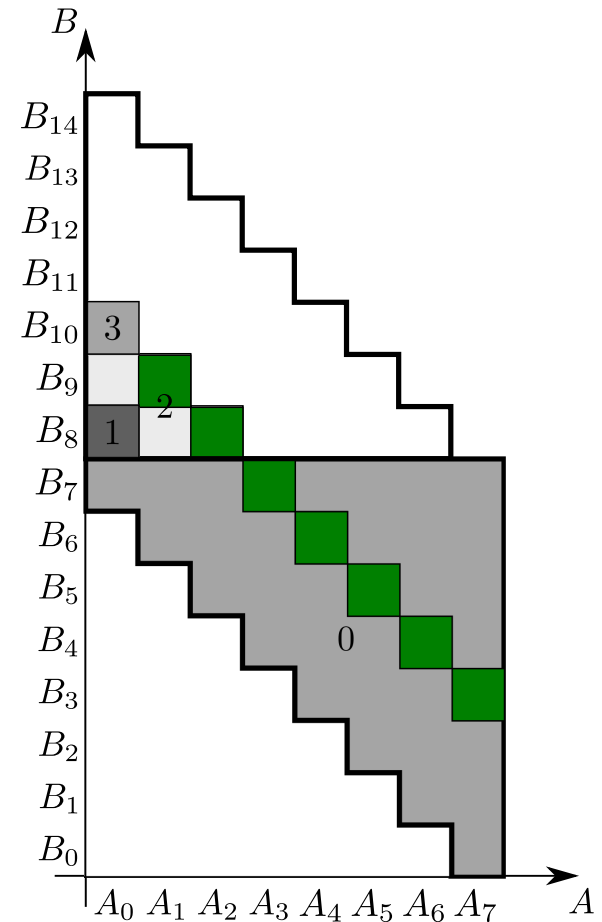
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0\dots 7})_{7\dots 14}$
1	$(AB)_8$	$C += x \text{ MidProd}(A_0, B_8, 0)$
2	$(AB)_9$	$C += x^2 \text{ MidProd}(A_{0\dots 2}, B_{8\dots 9}, 1\dots 2)$
3	$(AB)_{10}$	$C += x^3 \text{ MidProd}(A_0, B_{10}, 0)$



Shifted online middle product

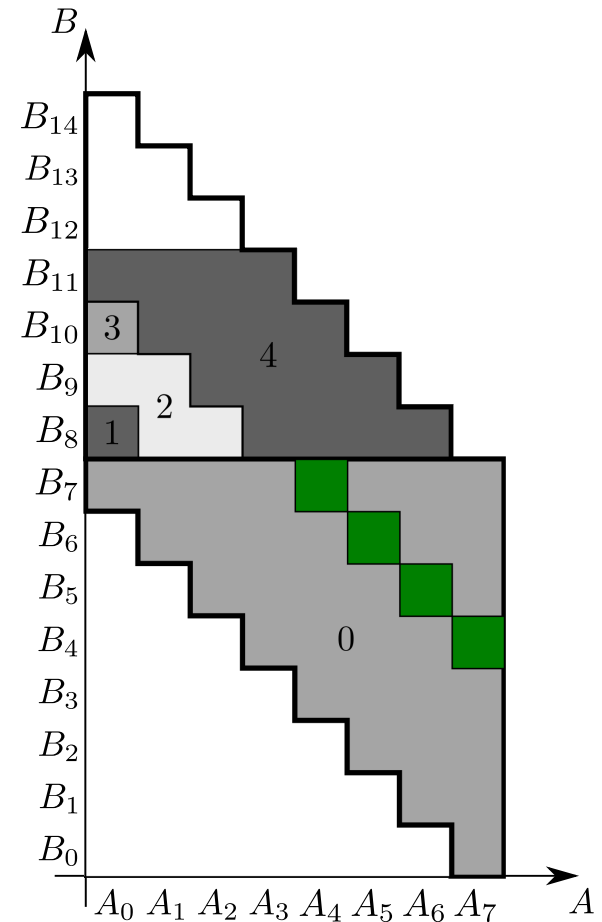
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0\dots 7})_{7\dots 14}$
1	$(AB)_8$	$C += x \text{ MidProd}(A_0, B_8, 0)$
2	$(AB)_9$	$C += x^2 \text{ MidProd}(A_{0\dots 2}, B_{8\dots 9}, 1\dots 2)$
3	$(AB)_{10}$	$C += x^3 \text{ MidProd}(A_0, B_{10}, 0)$
4	$(AB)_{11}$	$C += x^4 \text{ MidProd}(A_{0\dots 6}, B_{8\dots 11}, 3\dots 6)$



Shifted online middle product

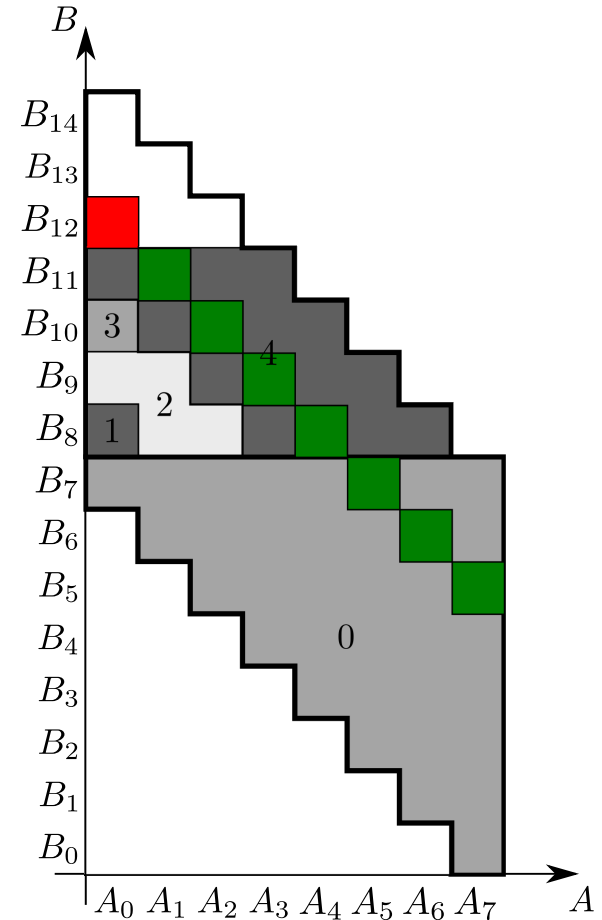
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0 \dots 7})_{7 \dots 14}$
1	$(AB)_8$	$C += x \text{ MidProd}(A_0, B_8, 0)$
2	$(AB)_9$	$C += x^2 \text{ MidProd}(A_{0 \dots 2}, B_{8 \dots 9}, 1 \dots 2)$
3	$(AB)_{10}$	$C += x^3 \text{ MidProd}(A_0, B_{10}, 0)$
4	$(AB)_{11}$	$C += x^4 \text{ MidProd}(A_{0 \dots 6}, B_{8 \dots 11}, 3 \dots 6)$
5	$(AB)_{12}$	



Shifted online middle product

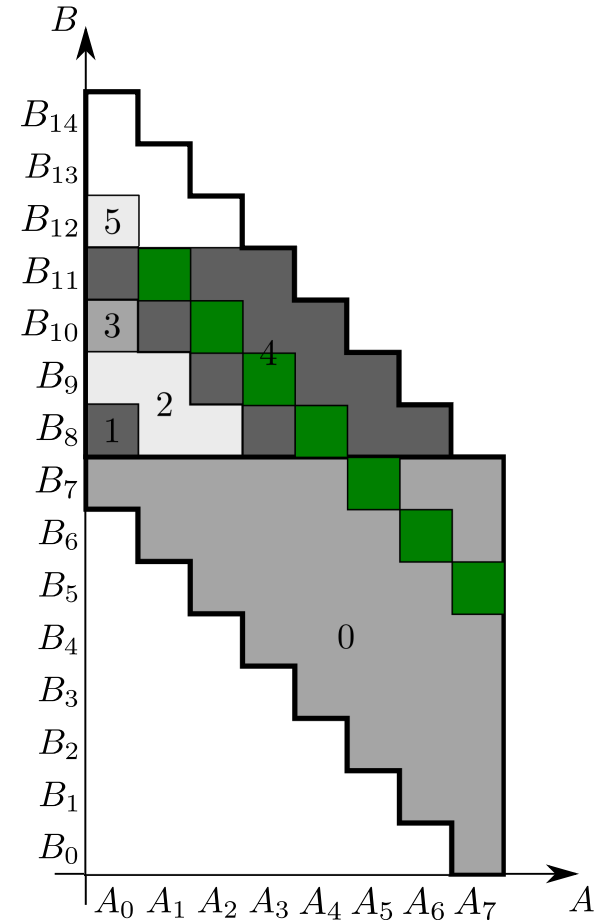
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0..7})_{7..14}$
1	$(AB)_8$	$C += x \text{ MidProd}(A_0, B_8, 0)$
2	$(AB)_9$	$C += x^2 \text{ MidProd}(A_{0..2}, B_{8..9}, 1..2)$
3	$(AB)_{10}$	$C += x^3 \text{ MidProd}(A_0, B_{10}, 0)$
4	$(AB)_{11}$	$C += x^4 \text{ MidProd}(A_{0..6}, B_{8..11}, 3..6)$
5	$(AB)_{12}$	$C += x^5 \text{ MidProd}(A_0, B_{12}, 0)$



Shifted online middle product

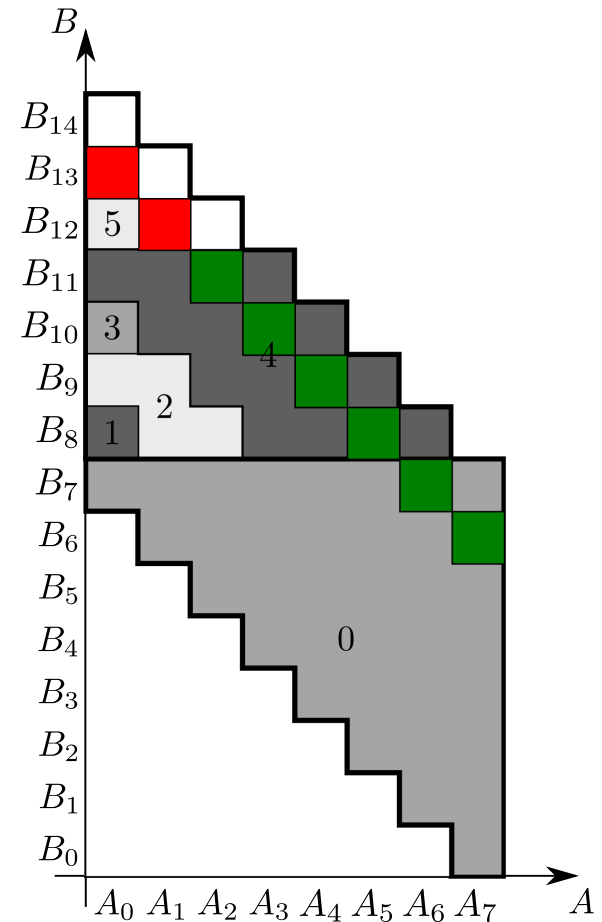
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0 \dots 7})_{7 \dots 14}$
1	$(AB)_8$	$C += x \text{ MidProd}(A_0, B_8, 0)$
2	$(AB)_9$	$C += x^2 \text{ MidProd}(A_{0 \dots 2}, B_{8 \dots 9}, 1 \dots 2)$
3	$(AB)_{10}$	$C += x^3 \text{ MidProd}(A_0, B_{10}, 0)$
4	$(AB)_{11}$	$C += x^4 \text{ MidProd}(A_{0 \dots 6}, B_{8 \dots 11}, 3 \dots 6)$
5	$(AB)_{12}$	$C += x^5 \text{ MidProd}(A_0, B_{12}, 0)$
6	$(AB)_{13}$	



Shifted online middle product

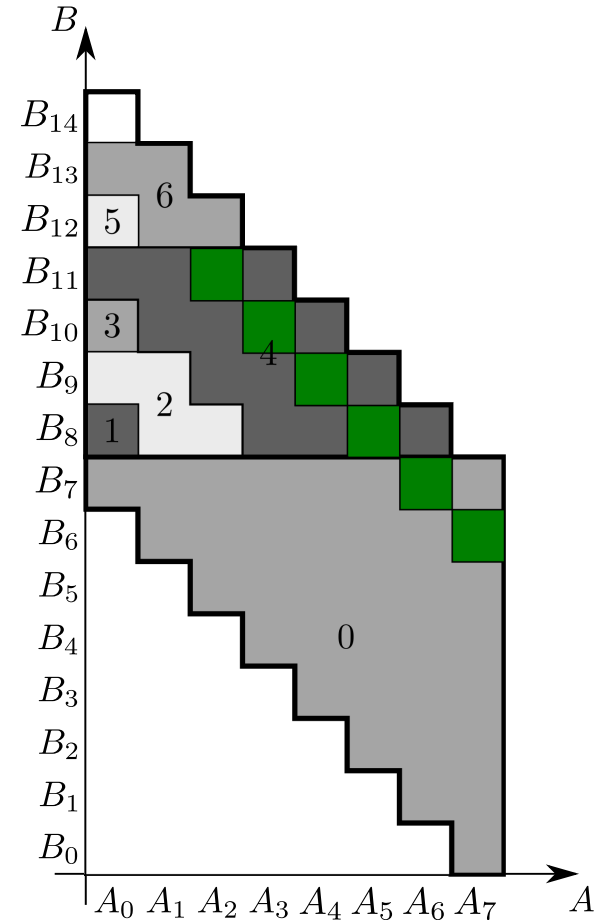
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0 \dots 7})_{7 \dots 14}$
1	$(AB)_8$	$C += x \text{ MidProd}(A_0, B_8, 0)$
2	$(AB)_9$	$C += x^2 \text{ MidProd}(A_{0 \dots 2}, B_{8 \dots 9}, 1 \dots 2)$
3	$(AB)_{10}$	$C += x^3 \text{ MidProd}(A_0, B_{10}, 0)$
4	$(AB)_{11}$	$C += x^4 \text{ MidProd}(A_{0 \dots 6}, B_{8 \dots 11}, 3 \dots 6)$
5	$(AB)_{12}$	$C += x^5 \text{ MidProd}(A_0, B_{12}, 0)$
6	$(AB)_{13}$	$C += x^6 \text{ MidProd}(A_{0 \dots 2}, B_{12 \dots 13}, 1 \dots 2)$



Shifted online middle product

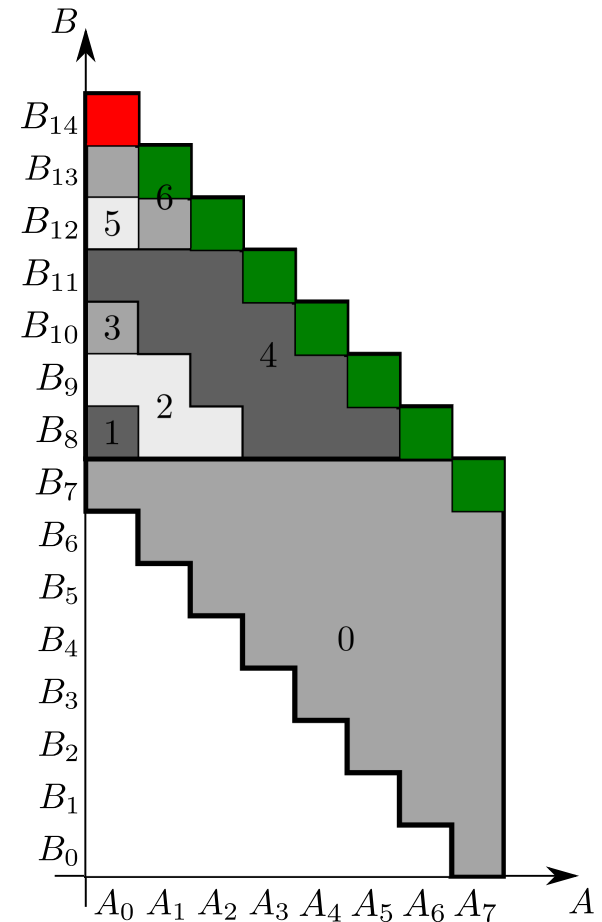
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0..7})_{7..14}$
1	$(AB)_8$	$C += x \text{ MidProd}(A_0, B_8, 0)$
2	$(AB)_9$	$C += x^2 \text{ MidProd}(A_{0..2}, B_{8..9}, 1..2)$
3	$(AB)_{10}$	$C += x^3 \text{ MidProd}(A_0, B_{10}, 0)$
4	$(AB)_{11}$	$C += x^4 \text{ MidProd}(A_{0..6}, B_{8..11}, 3..6)$
5	$(AB)_{12}$	$C += x^5 \text{ MidProd}(A_0, B_{12}, 0)$
6	$(AB)_{13}$	$C += x^6 \text{ MidProd}(A_{0..2}, B_{12..13}, 1..2)$
7	$(AB)_{14}$	



Shifted online middle product

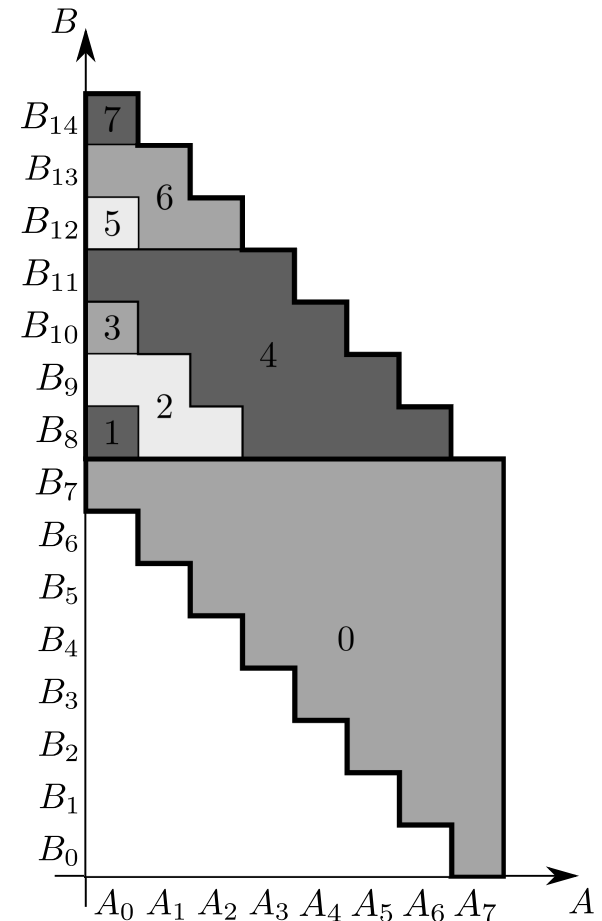
Example of online middle product: Compute

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

Shifted online algorithm OnlineMiddleProduct:

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

Step	computes	Computation
0	$(AB)_7$	$C = (AB_{0..7})_{7..14}$
1	$(AB)_8$	$C += x \text{ MidProd}(A_0, B_8, 0)$
2	$(AB)_9$	$C += x^2 \text{ MidProd}(A_{0..2}, B_{8..9}, 1..2)$
3	$(AB)_{10}$	$C += x^3 \text{ MidProd}(A_0, B_{10}, 0)$
4	$(AB)_{11}$	$C += x^4 \text{ MidProd}(A_{0..6}, B_{8..11}, 3..6)$
5	$(AB)_{12}$	$C += x^5 \text{ MidProd}(A_0, B_{12}, 0)$
6	$(AB)_{13}$	$C += x^6 \text{ MidProd}(A_{0..2}, B_{12..13}, 1..2)$
7	$(AB)_{14}$	$C += x^7 \text{ MidProd}(A_0, B_{14}, 0)$



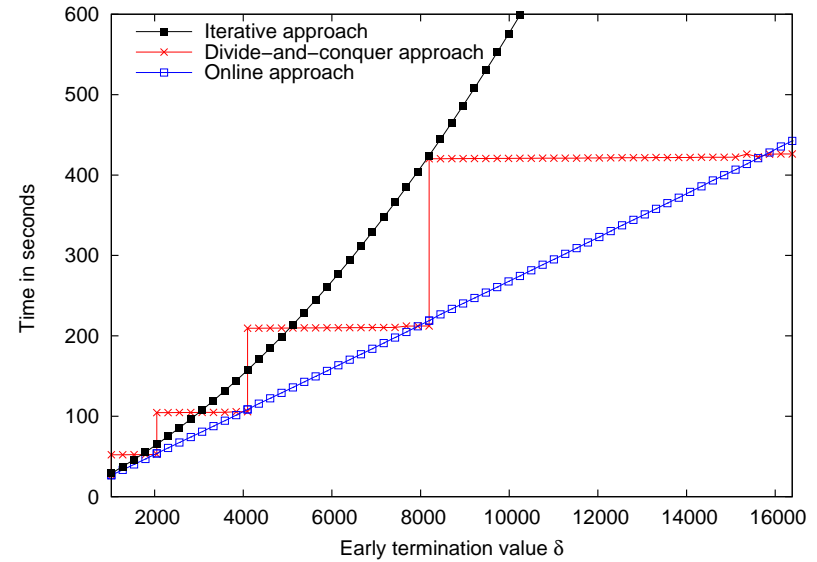
Conclusion - Future Work

Conclusion :

Online order basis

Early termination
Minimal knowledge on S

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



Implementation is available in [LinBox]

Future work :

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

(linear algebra, polynomial equations, ...)