

---

# A cyclic reduction approach for solving BABD linear systems

**Giuseppe Romanazzi**

Results from the PhD Thesis

“Numerical Solution of BABD linear systems arising from BVPs”

Department of Mathematics

Università degli Studi di Bari

# Outline

---

- BABD and ABD linear systems

# Outline

---

- BABD and ABD linear systems  
B ABD: Bordered Almost Block Diagonal

# Outline

---

- BABD and ABD linear systems
- BABD linear systems arising from BVPs

# Outline

---

- BABD and ABD linear systems
- BABD linear systems arising from BVPs
- BABD Cyclic Reduction algorithm

# Outline

---

- BABD and ABD linear systems
- BABD linear systems arising from BVPs
- BABD Cyclic Reduction algorithm
  - Comparisons
  - Application in the BVP solver, MIRKDC

# Outline

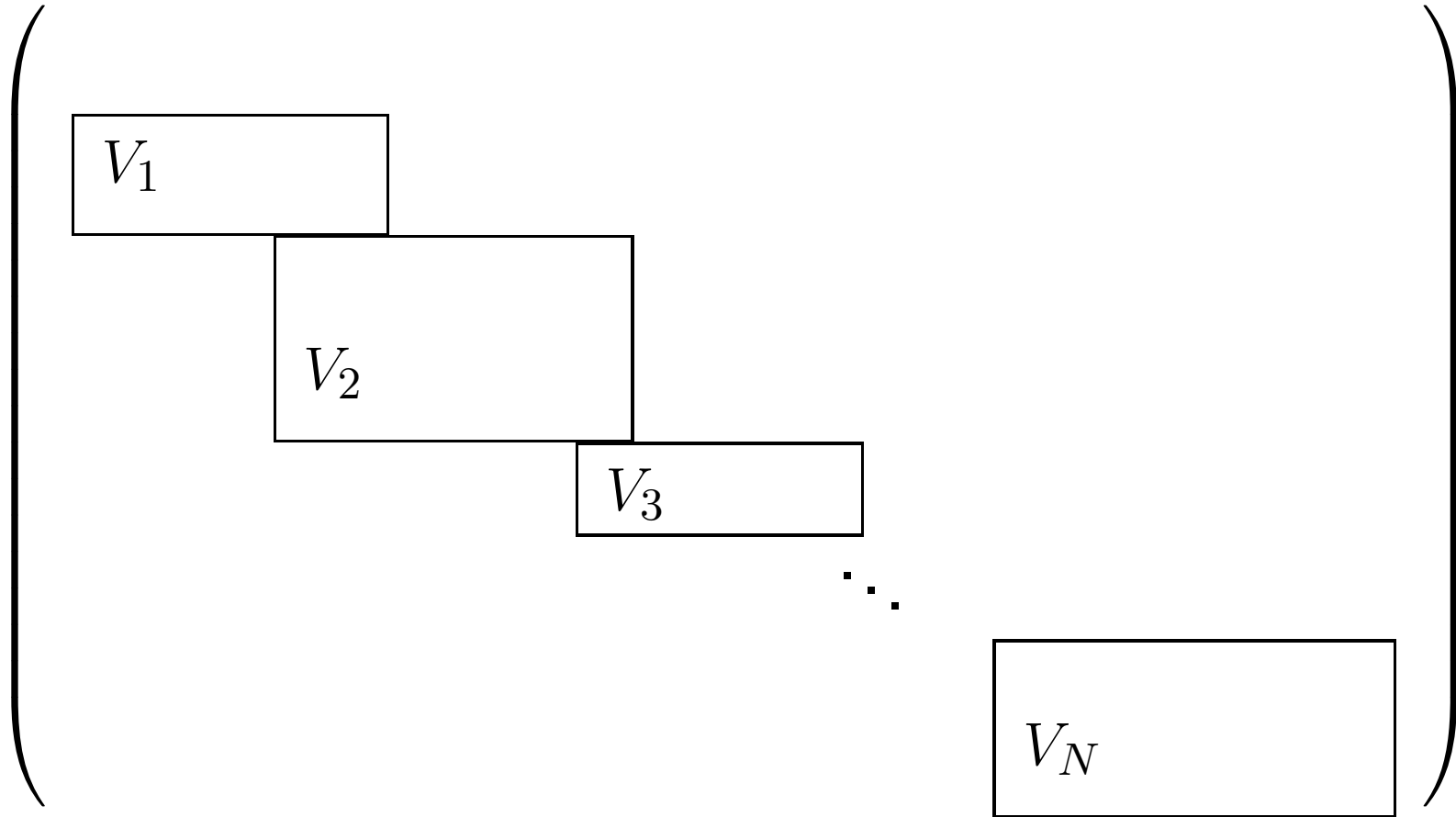
---

- BABD and ABD linear systems
- BABD linear systems arising from BVPs
- BABD Cyclic Reduction algorithm
  - Comparisons
  - Application in the BVP solver, MIRKDC
- Parallel implementation

# BABD linear systems

---

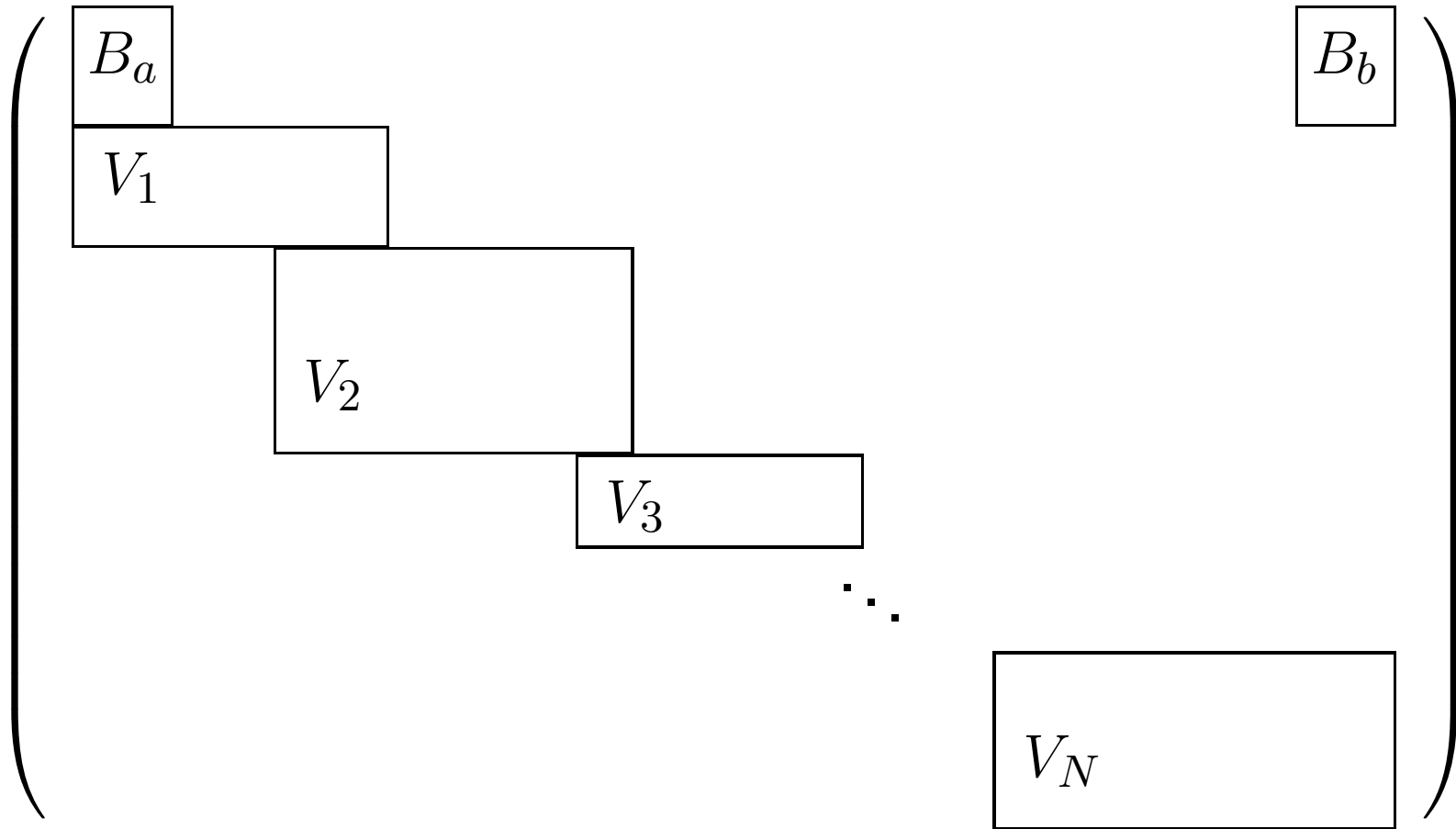
The General structure



# BABD linear systems

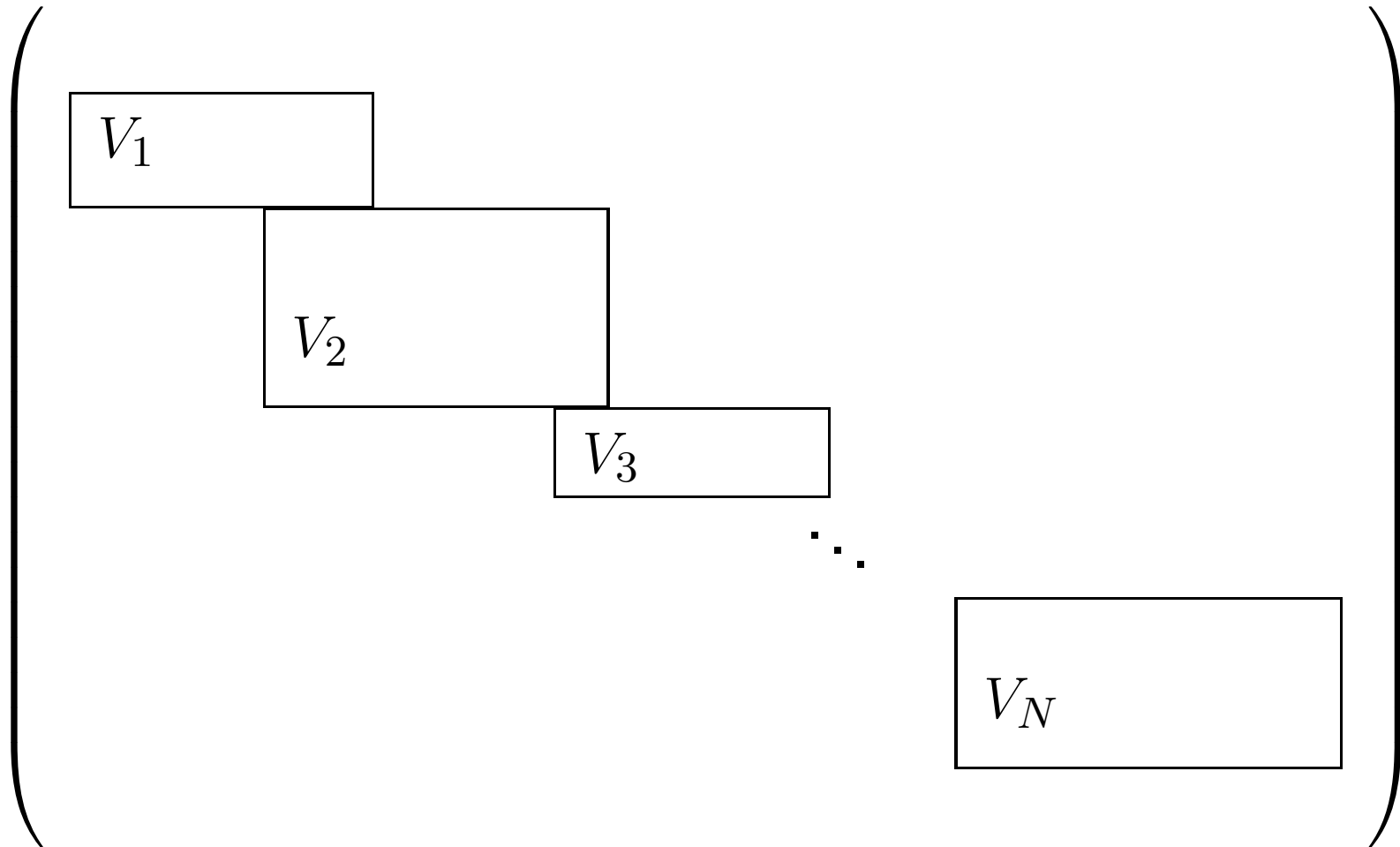
---

The General structure  
(Non separated boundary blocks)



# ABD linear systems

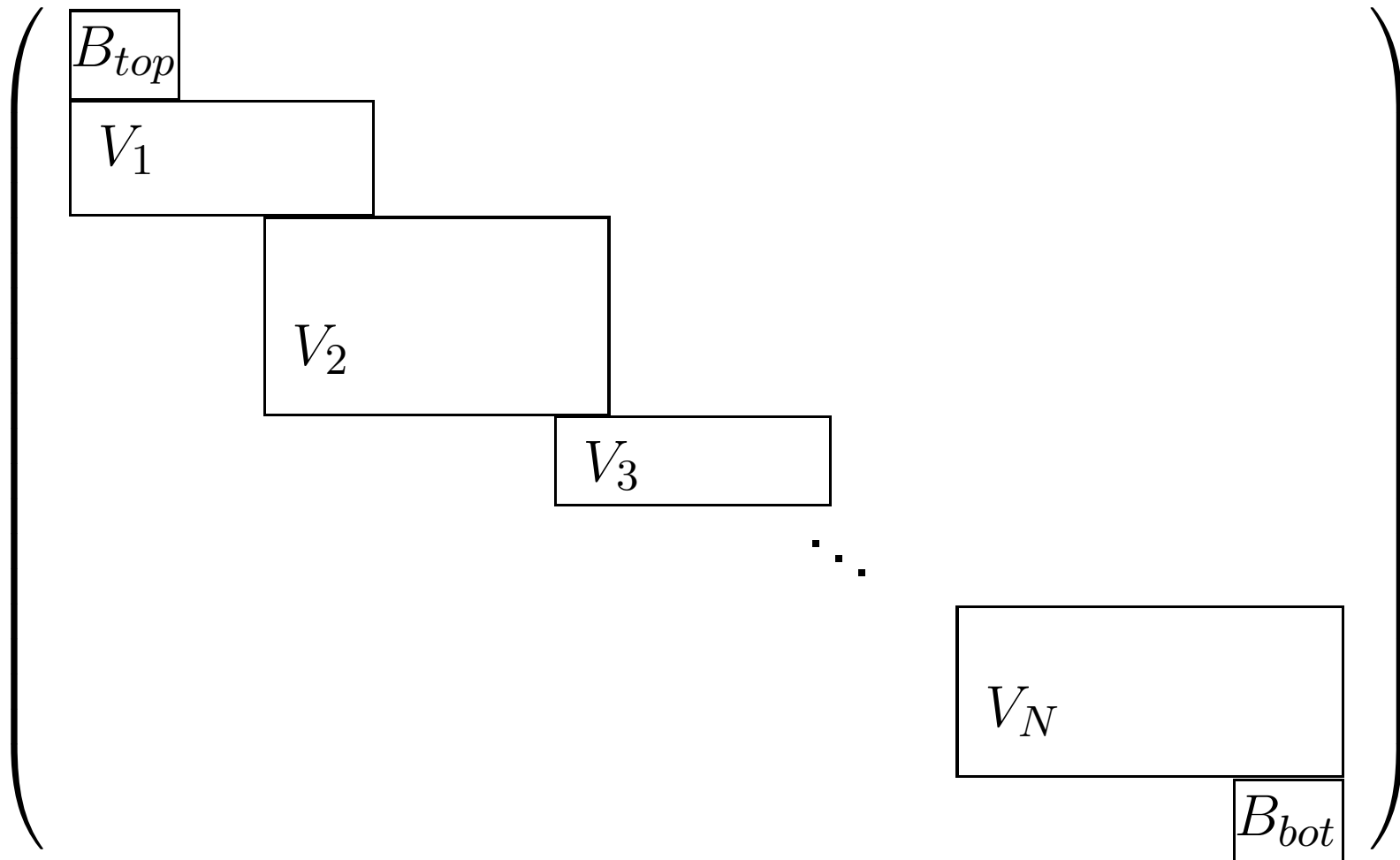
---



# ABD linear systems

---

(Separated boundary blocks)



# BABD systems arising from BVPs

---

Consider the  $m$ -th order BVP

$$(0) \quad D^m y(x) - \sum_{l=1}^m c_l(x) D^{l-1} y(x) = f(x),$$

$$x \in [a, b], y(x) \in \mathbb{R}$$

with **non-separated boundary conditions**

$$(0) \quad B_a \mathbf{z}(y(a)) + B_b \mathbf{z}(y(b)) = \mathbf{d},$$

where

$$\mathbf{z}(y(x)) = (y(x), Dy(x), D^2 y(x), \dots, D^{m-1} y(x))^T.$$

# BABD systems arising from BVPs

---

Orthogonal Spline Collocation applied to the BVP (??)-(??) with  $k$  Gaussian points,  $k \geq m$

- using B-spline basis, yields

$$\begin{pmatrix} D_a & & & & & & & & D_b \\ W_{11} & W_{12} & W_{13} & & & & & & \\ & & W_{21} & W_{22} & W_{23} & & & & \\ & & & & \ddots & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & \ddots & \\ & & & & & & & & \\ & & & & & & W_{N1} & W_{N2} & W_{N3} \end{pmatrix}$$

where  $W_{i1} \in \mathbb{R}^{k \times m}$ ,  $W_{i2} \in \mathbb{R}^{k \times (k-m)}$  and  $W_{i3} \in \mathbb{R}^{k \times m}$ .



# Common BABD linear system

---

$$\begin{pmatrix} B_a & & & & & B_b \\ S_0 & R_1 & & & & \\ & S_1 & R_2 & & & \\ & & \ddots & \ddots & & \\ & & & S_{N-1} & R_N & \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix} = \begin{pmatrix} \mathbf{d} \\ \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_N \end{pmatrix}$$

where  $S_i, R_i, B_a, B_b \in \mathbb{R}^{m \times m}$ . It arises from the discretization of the linear BVP  $y'(x) = A(x)y(x) + \mathbf{q}(x)$ ,  $B_a y(a) + B_b y(b) = \mathbf{d}$ ,  $\mathbf{y}(x) \in \mathbb{R}^m$ , using

- Trapezoidal rule, Midpoint Rule
- Multiple-shooting
- Mono-Implicit Runge Kutta methods **MIRK**

# BABDCR

---

$$\begin{pmatrix} B_a & & & & & & B_b \\ S_0 & R_1 & & & & & \\ & S_1 & R_2 & & & & \\ & & \ddots & \ddots & & & \\ & & & S_{N-1} & R_N & & \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix} = \begin{pmatrix} \mathbf{d} \\ \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_N \end{pmatrix}$$

The serial code **BABDCR** solves this system using a cyclic reduction approach.

In the next, we suppose that  $N = 2^p$  and the coefficient matrix is nonsingular.

# Cyclic Reduction

---

Each pair of block row equations, for  $i = 2j - 1$ ,  $j = 1, 2, \dots, N/2$ ,

$$\begin{pmatrix} S_{i-1} & R_i \\ & S_i & R_{i+1} \end{pmatrix} \begin{pmatrix} \mathbf{y}_{i-1} \\ \mathbf{y}_i \\ \mathbf{y}_{i+1} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_i \\ \mathbf{f}_{i+1} \end{pmatrix}$$

is reduced into one block row equation involving only the unknowns  $\mathbf{y}_{i-1}$  and  $\mathbf{y}_{i+1}$

$$S'_{i-1} \mathbf{y}_{i-1} + R'_{i+1} \mathbf{y}_{i+1} = \mathbf{f}'_{i+1}.$$

# Cyclic Reduction

---

In each step, we obtain a BABD linear system with approximately half of the previous dimension.



# Cyclic Reduction

---

In each step, we obtain a BABD linear system with approximately half of the previous dimension.

**First step:** we get

$$\begin{pmatrix} B_a & & & & & & B_b \\ S'_0 & R'_2 & & & & & \\ & S'_2 & R'_4 & & & & \\ & & S'_4 & R'_6 & & & \\ & & & \ddots & \ddots & & \\ & & & & S'_{N-2} & R'_N & \\ & & & & & & \end{pmatrix} \begin{pmatrix} y_0 \\ y_2 \\ \vdots \\ y_{N-2} \\ y_N \end{pmatrix} = \begin{pmatrix} \mathbf{d} \\ \mathbf{f}'_2 \\ \vdots \\ \mathbf{f}'_N \end{pmatrix}$$

$$S'_i, R'_i, B_a, B_b \in \mathbb{R}^{m \times m}$$

# Cyclic Reduction

---

In each step, we obtain a BABD linear system with approximately half of the previous dimension.

**Second step:** From

$$\begin{pmatrix} B_a & & & & & & B_b \\ S'_0 & R'_2 & & & & & \\ & S'_2 & R'_4 & & & & \\ & & \ddots & \ddots & & & \\ & & & S'_{N-2} & R'_N & & \end{pmatrix} \begin{pmatrix} y_0 \\ y_2 \\ \vdots \\ y_{N-2} \\ y_N \end{pmatrix} = \begin{pmatrix} \mathbf{d} \\ \mathbf{f}'_2 \\ \vdots \\ \mathbf{f}'_N \end{pmatrix}$$

# Cyclic Reduction

---

In each step, we obtain a BABD linear system with approximately half of the previous dimension.

**Second step:** we get

$$\begin{pmatrix} B_a & & & & B_b \\ S''_0 & R''_4 & & & \\ & & \ddots & & \\ & & & S''_{N-4} & R''_N \end{pmatrix} \begin{pmatrix} y_0 \\ y_4 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} \mathbf{d} \\ \mathbf{f}''_4 \\ \vdots \\ \mathbf{f}''_N \end{pmatrix}$$

$$S''_i, R''_i, B_a, B_b \in \mathbb{R}^{m \times m}$$

# Cyclic Reduction

---

In each step, we obtain a BABD linear system with approximately half of the previous dimension.

**Final step** ( $p$ -th step,  $N = 2^p$ ): From

$$\begin{pmatrix} B_a & & B_b \\ S_0^{(p-1)} & R_{\frac{N}{2}}^{(p-1)} & \\ & S_{\frac{N}{2}}^{(p-1)} & R_N^{(p-1)} \end{pmatrix} \begin{pmatrix} \mathbf{y}_0 \\ \mathbf{y}_{\frac{N}{2}} \\ \mathbf{y}_N \end{pmatrix} = \begin{pmatrix} \mathbf{d} \\ \mathbf{f}_{\frac{N}{2}}^{(p-1)} \\ \mathbf{f}_N^{(p-1)} \end{pmatrix}$$

# Cyclic Reduction

---

In each step, we obtain a BABD linear system with approximately half of the previous dimension.

**Final step** ( $p$ -th step,  $N = 2^p$ ): we get a full system

$$\begin{pmatrix} B_a & B_b \\ S_0^{(p)} & R_N^{(p)} \end{pmatrix} \begin{pmatrix} \mathbf{y}_0 \\ \mathbf{y}_N \end{pmatrix} = \begin{pmatrix} \mathbf{d} \\ \mathbf{f}_N^{(p)} \end{pmatrix}$$

$$S_0^{(p)}, R_N^{(p)}, B_a, B_b \in \mathbb{R}^{m \times m}$$

# Get $S'_{i-1}$ , $R'_{i+1}$ and $f'_{i+1}$

---

Since the overlapping columns  $\begin{pmatrix} R_i \\ S_i \end{pmatrix}$  are linearly independent, we can use a **partial pivoting LU factorization**

$$P_i \begin{pmatrix} R_i \\ S_i \end{pmatrix} = \begin{pmatrix} L_i \\ T_i \end{pmatrix} U_i = \begin{pmatrix} L_i & \\ T_i & I \end{pmatrix} \begin{pmatrix} U_i \\ 0 \end{pmatrix}.$$

Then, premultiplying the couple of block equations by the permutation matrix  $P_i$  and the inverse of  $\begin{pmatrix} L_i & \\ T_i & I \end{pmatrix}$  we get

$$\begin{pmatrix} L_i & \\ T_i & I \end{pmatrix}^{-1} P_i \begin{pmatrix} S_{i-1} & R_i \\ S_i & R_{i+1} \end{pmatrix} \equiv \begin{pmatrix} \tilde{S}_{i-1} & U_i & \tilde{R}_{i+1} \\ S'_{i-1} & & R'_{i+1} \end{pmatrix}$$

# Get $S'_{i-1}$ , $R'_{i+1}$ and $\mathbf{f}'_{i+1}$

---

The resulting blocks of the reduced equation  $S'_{i-1}\mathbf{y}_{i-1} + R'_{i+1}\mathbf{y}_{i+1} = \mathbf{f}'_{i+1}$  are

$$\bullet S'_{i-1} = P_{i2} \begin{pmatrix} S_{i-1} \\ 0 \end{pmatrix} - T_i L_i^{-1} P_{i1} \begin{pmatrix} S_{i-1} \\ 0 \end{pmatrix}$$

$$\bullet R'_{i+1} = P_{i2} \begin{pmatrix} 0 \\ R_{i+1} \end{pmatrix} - T_i L_i^{-1} P_{i1} \begin{pmatrix} 0 \\ R_{i+1} \end{pmatrix}$$

$$\bullet \mathbf{f}'_{i+1} = P_{i2} \begin{pmatrix} \mathbf{f}_i \\ \mathbf{f}_{i+1} \end{pmatrix} - T_i L_i^{-1} P_{i1} \begin{pmatrix} \mathbf{f}_i \\ \mathbf{f}_{i+1} \end{pmatrix}$$

where  $P_i = \begin{pmatrix} P_{i1} \\ P_{i2} \end{pmatrix}$  with  $P_{i1}, P_{i2} \in \mathbb{R}^{m \times m}$ .  $T_i L_i^{-1}$  is saved in order to reduce the right hand side.

---

# Back substitution phase and Fill-in

---

- After  $p$  back-substitution solve steps the initial linear system is solved.

# Back substitution phase and Fill-in

---

- After  $p$  back-substitution solve steps the initial linear system is solved.
- The cyclic reduction approach has an extra memory requirement (fill-in block:  $T_i L_i^{-1}$ ) of size  $m \times m$  for each of the  $N - 1$  reductions.

# MIRKDC

---

The code **MIRKDC** solves non linear BVPs

$$(1) \quad \mathbf{y}' = \mathbf{f}(t, \mathbf{y}(t)), \quad t \in [a, b], \mathbf{y}(t) \in \mathbb{R}^m$$

with separated BCs

$$(2) \quad \mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)) = \begin{pmatrix} \mathbf{g}_0(\mathbf{y}(a)) \\ \mathbf{g}_1(\mathbf{y}(b)) \end{pmatrix} = 0.$$

# MIRKDC

---

The code **MIRKDC** solves non linear BVPs

$$(0) \quad \mathbf{y}' = \mathbf{f}(t, \mathbf{y}(t)), \quad t \in [a, b], \mathbf{y}(t) \in \mathbb{R}^m$$

with separated BCs

$$(0) \quad \mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)) = \begin{pmatrix} \mathbf{g}_0(\mathbf{y}(a)) \\ \mathbf{g}_1(\mathbf{y}(b)) \end{pmatrix} = 0.$$

- It uses MIRK formulae, that applied to (1) on a given subdivision  $\{t_i\}_{i=0}^N$  of  $[a, b]$ , yields a nonlinear system  $\Phi(\mathbf{Y}) = 0$ ,  $\mathbf{Y} = (\mathbf{y}_0^T, \dots, \mathbf{y}_N^T)^T \in \mathbb{R}^{m(N+1)}$

# MIRKDC

---

The code **MIRKDC** solves non linear BVPs

$$(0) \quad \mathbf{y}' = \mathbf{f}(t, \mathbf{y}(t)), \quad t \in [a, b], \mathbf{y}(t) \in \mathbb{R}^m$$

with separated BCs

$$(0) \quad \mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)) = \begin{pmatrix} \mathbf{g}_0(\mathbf{y}(a)) \\ \mathbf{g}_1(\mathbf{y}(b)) \end{pmatrix} = 0.$$

- It uses MIRK formulae, that applied to (1) on a given subdivision  $\{t_i\}_{i=0}^N$  of  $[a, b]$ , yields a nonlinear system  $\Phi(\mathbf{Y}) = 0$ ,  $\mathbf{Y} = (\mathbf{y}_0^T, \dots, \mathbf{y}_N^T)^T \in \mathbb{R}^{m(N+1)}$
- A continuous solution approximation  $\mathbf{u}(t)$  is obtained using a Continuous MIRK (CMIRK) scheme, it provides defect control and mesh selection capabilities.

# MIRKDC (Cont.)

---

The nonlinear system

$$\Phi(\mathbf{Y}) = 0$$

is solved using the Newton iteration

$$\mathbf{Y}^{(q+1)} = \mathbf{Y}^{(q)} + \Delta\mathbf{Y}^{(q)}, \quad q = 0, 1, \dots$$

where

$$\left[ \frac{\partial \Phi(\mathbf{Y}^{(q)})}{\partial \mathbf{Y}} \right] \Delta\mathbf{Y}^{(q)} = -\Phi(\mathbf{Y}^{(q)}),$$

given  $\mathbf{Y}^{(0)}$ .

# MIRKDC (Cont.)

---

The coefficient matrix has the **ABD** structure

$$\frac{\partial \Phi(\mathbf{Y}^{(q)})}{\partial \mathbf{Y}} = \begin{pmatrix} B_{top} & & & & & \\ S_0 & R_1 & & & & \\ & S_1 & R_2 & & & \\ & & \ddots & \ddots & & \\ & & & & S_{N-1} & R_N \\ & & & & & B_{bot} \end{pmatrix}$$

with  $B_{top} = \frac{\partial \mathbf{g}_0(\mathbf{y}_0^{(q)})}{\partial y_0}$ ,  $B_{bot} = \frac{\partial \mathbf{g}_1(y_N^{(q)})}{\partial y_N}$  and

$$S_i = -I - h_i K_{i,i}, \quad R_{i+1} = I - h_i K_{i+1,i},$$

where blocks  $K_{i,j}$  depend on the used RK formulae.

---

# Linear solvers inside MIRKDC

---

- MIRKDC solves the resulting **ABD** linear systems with the algorithm **COLROW** based on an alternate row and column elimination procedure



# RSCALE and BABDCR

---

- The parallel version **PMIRKDC** uses the algorithm **RSCALE** based on an eigenvalue rescaling procedure.
- As for **BABDCR**, **RSCALE** can solve directly **BABD** linear systems.

# RSCALE and BABDCR

---

- The parallel version **PMIRKDC** uses the algorithm **RSCALE** based on an eigenvalue rescaling procedure.
- As for **BABDCR**, **RSCALE** can solve directly **BABD** linear systems.
- ABD linear system can be represented as BABD linear system with

$$B_a = \begin{pmatrix} B_{top} \\ 0 \end{pmatrix} \in \mathbb{R}^{m \times m}, \quad B_b = \begin{pmatrix} 0 \\ B_{bot} \end{pmatrix} \in \mathbb{R}^{m \times m}$$

# RSCALE and BABDCR

---

- The parallel version **PMIRKDC** uses the algorithm **RSCALE** based on an eigenvalue rescaling procedure.
- As for **BABDCR**, **RSCALE** can solve directly **BABD** linear systems.
- ABD linear system can be represented as BABD linear system with

$$B_a = \begin{pmatrix} B_{top} \\ 0 \end{pmatrix} \in \mathbb{R}^{m \times m}, \quad B_b = \begin{pmatrix} 0 \\ B_{bot} \end{pmatrix} \in \mathbb{R}^{m \times m}$$

We can compare both three codes on ABD and BABD linear systems

# Comparisons among linear solvers

---

- As first, we compare **BABDCR** with **RSCALE** and **COLROW** on **ABD** linear systems generated by **MIRKDC** applied to a linear problem

$$\mathbf{y}' = M\mathbf{y}, \quad \mathbf{y}(t) \in \mathbb{R}^m$$

with  $B_{top}\mathbf{y}(a) = \mathbf{d}_a \in \mathbb{R}^{m_0}$  and  $B_{bot}\mathbf{y}(b) = \mathbf{d}_b \in \mathbb{R}^{m-m_0}$ .

We fix  $m = 20$  and  $m_0 = 10$

# Comparisons among linear solvers

---

- As first, we compare **BABDCR** with **RSCALE** and **COLROW** on **ABD** linear systems generated by **MIRKDC** applied to a linear problem

$$\mathbf{y}' = M\mathbf{y}, \quad \mathbf{y}(t) \in \mathbb{R}^m$$

with  $B_{top}\mathbf{y}(a) = \mathbf{d}_a \in \mathbb{R}^{m_0}$  and  $B_{bot}\mathbf{y}(b) = \mathbf{d}_b \in \mathbb{R}^{m-m_0}$ .

We fix  $m = 20$  and  $m_0 = 10$

- We use the optimal 3 stages, 4-th order MIRK scheme

0	0	0	0	0
1	1	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{8}$	$-\frac{1}{8}$	0
<hr/>		$\frac{1}{6}$	$\frac{1}{6}$	$\frac{2}{3}$

and an associated 4-th order, 4 stages CMIRK scheme.

# Comparisons among linear solvers

---

	Time		
	N=256	N=512	N=1024
BABDCR	3.71e-02	7.12e-02	0.152
COLROW	1.46e-02	2.83e-02	6.34e-02
RSCALE	6.83e-02	0.136	0.274

---

- COLROW is more than 2 times faster than BABDCR and more than 4 times faster than RSCALE
- Results agree with the teoretical computational costs of the linear solvers
  - BABDCR :  $\frac{14}{3}m^3 N$
  - COLROW :  $(\frac{5}{3}m^3 + mm_0^2)N$
  - RSCALE :  $\frac{20}{3}m^3 N$

# Comparisons among the linear solvers

---

	Error		
	N=256	N=512	N=1024
BABDCR	1.65e-13	2.88e-13	3.46e-13
COLROW	1.55e-13	2.05e-13	7.08e-13
RSCALE	2.54e-12	6.02e-12	3.01e-11

- The errors for **COLROW** and **BABDCR** are similar
- **RSCALE** is less accurate

These results lead us to prefer **COLROW** with respect the other methods to solve **ABD** linear systems.

# Comparisons among the linear solvers

---

- In the next, we compare **BABDCR** with **RSCALE** and **COLROW** on **BABD** linear systems generated by **MIRKDC** applied to a linear problem

$$y' = My, \quad y(t) \in \mathbb{R}^m$$

with  $B_a y(a) + B_b y(b) = d \in \mathbb{R}^m$ . We fix  $m = 20$ .

# Comparisons among the linear solvers

---

	Time		
	N=256	N=512	N=1024
BABDCR	3.61e-02	7.03e-02	0.151
COLROW	0.102	0.224	0.464
RSCALE	6.83e-02	0.136	0.287

- **BABDCR** is approximately **3** times faster than **COLROW** and more than **1.5** times faster than **RSCALE**

# Comparisons among the linear solvers

---

- Timing associated to **COLROW** includes converting the linear system from the **BABD** structure to the **ABD** structure
- Results agree with the teoretical computational costs of the linear solvers
  - **BABDCR** :  $\frac{14}{3}m^3 N$
  - **COLROW** :  $\frac{46}{3}m^3 N$
  - **RSCALE** :  $\frac{20}{3}m^3 N$

# Comparisons among the linear solvers

---

	Error		
	N=256	N=512	N=1024
BABDCR	7.62e-13	1.22e-12	1.19e-12
COLROW	7.53e-13	4.10e-13	1.25e-12
RSCALE	5.17e-12	2.90e-11	6.02e-11

- The errors associated with **BABDCR** and **COLROW** are similar
- **RSCALE** is again the least accurate algorithm

# A variant of MIRKDC

---

We have written a variant of MIRKDC, called MIRKDC\_NS, which solves the system of BVPs with general non-separated boundary conditions. The algorithm uses the discretizations of MIRKDC resulting in linear systems that are solved using the BABD solver BABDCR. Therefore, MIRKDC\_NS essentially replaces COLROW with BABDCR.

# A variant of MIRKDC(Cont.)

---

In order to better emphasize the advantages of using **BABDCR** we give statistics for calls to **MIRKDC** and **MIRKDC\_NS** on a BVP

$$\mathbf{y}' = M\mathbf{y}(t)$$

with non-separated boundary conditions

$$B_a\mathbf{y}(a) + B_b\mathbf{y}(b) = \mathbf{d}.$$

We use initial mesh of 256 points and defect tolerance  $1e-07$

# A variant of MIRKDC(Cont.)

---

## Call to MIRKDC

MESH	#FACTs	time	#SOLVEs	time
256	1	0.11	2	0.16e-01
224	1	0.75e-01	2	0.12e-01
246	1	0.82e-01	2	0.14e-01
Total:	3	0.27	6	0.41e-01

---

---

Total monitored Linear Algebra time: 0.31 secs.

Total monitored Nonlinear Algebra time: 0.12 secs.

---

# A variant of MIRKDC(Cont.)

Call to **MIRKDC\_NS**

MESH	#FACTs	time	#SOLVEs	time
256	1	0.32e-01	2	0.98e-02
224	1	0.25e-01	2	0.78e-02
246	1	0.29e-01	2	0.78e-02
Total:	3	0.87e-01	6	0.25e-01

Total monitored Linear Algebra time: 0.11 secs.

Total monitored Nonlinear Algebra time: 0.12 secs.

- **BABDCR** in **MIRKDC\_NS** saves more than one half of the linear algebra time with respect to using *COLROW*.

# Parallel implementation

---

We consider a distributed memory parallel implementation of **BABDCR** on 16 processors. **Speed-up**:

# Parallel implementation

---

We consider a distributed memory parallel implementation of **BABDCR** on 16 processors. **Speed-up**:

●  $N = 512$

Problem	$P = 2$	$P = 4$	$P = 8$	$P = 16$
$m = 4$	1.864	2.592	3.684	2.661
$m = 16$	1.800	3.715	6.590	10.412
$m = 64$	1.839	3.649	6.944	11.752

# Parallel implementation

---

We consider a distributed memory parallel implementation of **BABDCR** on 16 processors. **Speed-up**:

●  $N = 512$

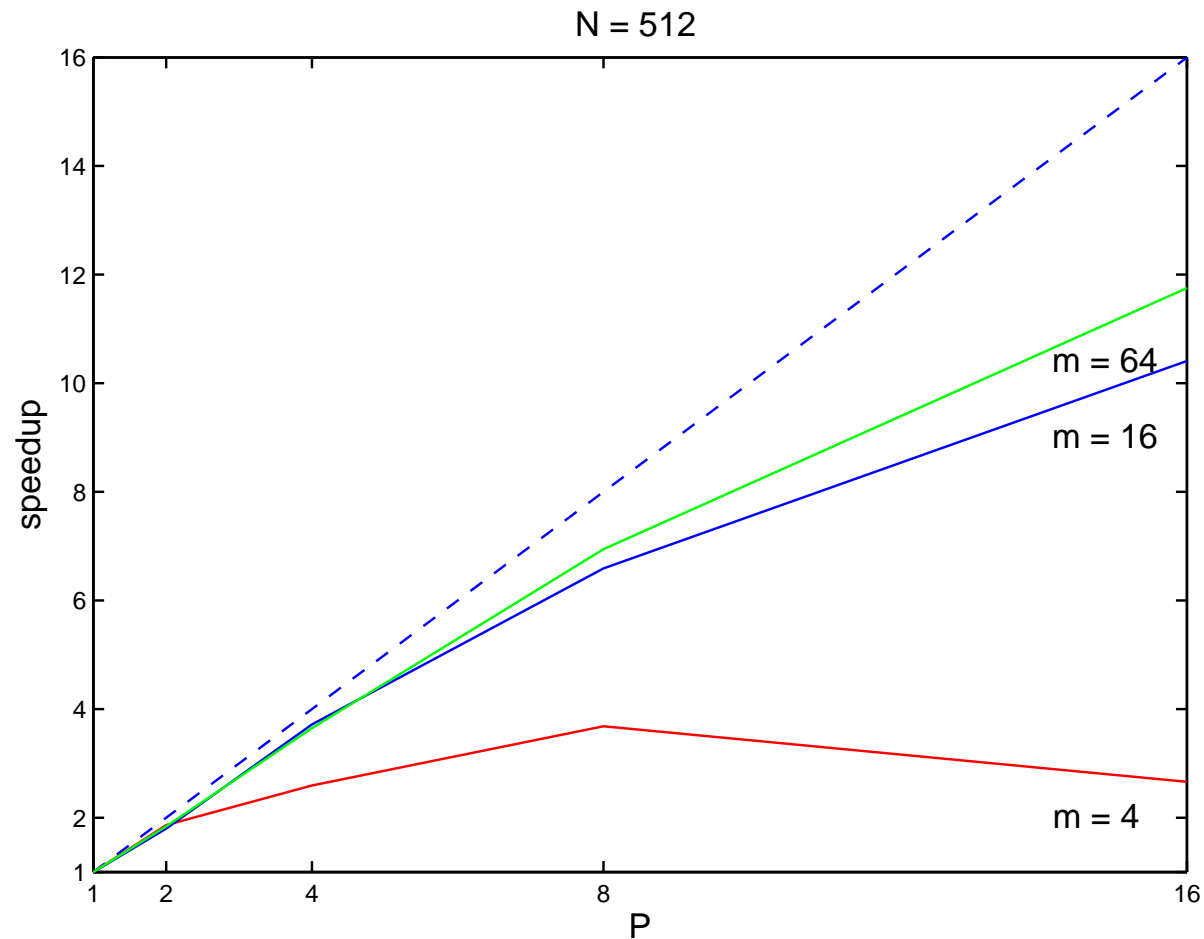
Problem	$P = 2$	$P = 4$	$P = 8$	$P = 16$
$m = 4$	1.864	2.592	3.684	2.661
$m = 16$	1.800	3.715	6.590	10.412
$m = 64$	1.839	3.649	6.944	11.752

●  $N = 1024$

Problem	$P = 2$	$P = 4$	$P = 8$	$P = 16$
$m = 4$	1.917	2.951	4.508	6.306
$m = 16$	1.950	3.615	7.578	12.605
$m = 64$	2.030	3.912	7.768	14.153

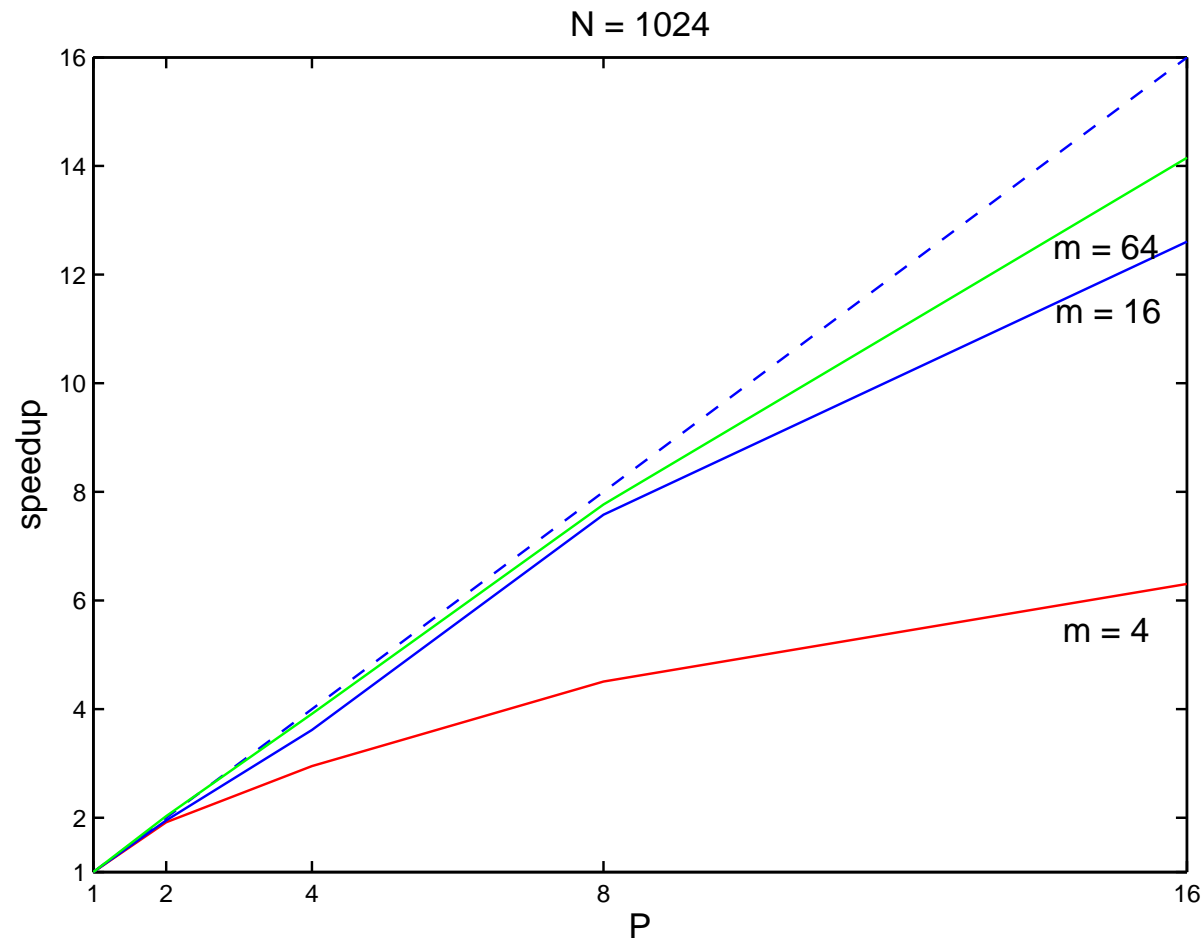
# Parallel implementation

- Speed-up for  $N = 512$



# Parallel implementation

- Speed-up for  $N = 1024$



# Conclusions

---

## BABDCR

- fast and accurate BABD algorithm
- slower than COLROW for ABD systems
- good speed-up for sufficiently large  $m$

# Conclusions

---

## BABDCR

- fast and accurate BABD algorithm
- slower than COLROW for ABD systems
- good speed-up for sufficiently large  $m$
- it has been also implemented for generalized BABD structure giving better time-results respect the other techniques, as COLROW