A massively-parallel algorithm for Bordered Almost Block Diagonal systems on GPUs

Sparse Days Meeting 2019, Toulouse

Monica Dessole, Fabio Marcuzzi

July 11, 2019

Università degli Studi di Padova

Università degli Studi di Padova

Outline

Introduction

BABD vs ABD systems

Boundary Value Problems for Ordinary Differential Equations (BVODEs)

$$y' = A(x)y(x) + q(x), \quad B_a y(a) + B_b y(b) = 0, \quad y, q \in \mathbb{R}^n,\ x \in [a, b].$$

BABD vs ABD systems

Boundary Value Problems for Ordinary Differential Equations (BVODEs)

$$y' = A(x)y(x) + q(x), \quad B_a y(a) + B_b y(b) = 0, \quad y, q \in \mathbb{R}^n,\ x \in [a, b].$$

yield Bordered Almost Block Diagonal (BABD) system

$$\begin{bmatrix} S_0 & T_0 & & & \\ & S_1 & T_1 & & \\ & & \ddots & \ddots & \\ & & & S_{N-1} & T_{N-1} \\ B_a & & & & B_b \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{N-1} \\ \mathbf{b}_N \end{bmatrix}$$

where S_i, T_i, B_a, B_b are square $n \times n$ blocks.

BABD vs ABD systems

Boundary Value Problems for Ordinary Differential Equations (BVODEs)

$$y' = A(x)y(x) + q(x), \quad B_a y(a) + B_b y(b) = 0, \quad y, q \in \mathbb{R}^n,\ x \in [a, b].$$

yield Bordered Almost Block Diagonal (BABD) system

$$\begin{bmatrix} S_0 & T_0 & & & \\ & S_1 & T_1 & & \\ & & \ddots & \ddots & \\ & & & S_{N-1} & T_{N-1} \\ B_a & & & & B_b \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{N-1} \\ \mathbf{b}_N \end{bmatrix}$$

where S_i, T_i, B_a, B_b are square $n \times n$ blocks. When BCs are separable, i.e.

$$B_a = \begin{bmatrix} \bar{B}_a \\ \mathbb{O} \end{bmatrix}, \quad B_b = \begin{bmatrix} \mathbb{O} \\ \bar{B}_b \end{bmatrix}, \quad \mathbf{b}_N = \begin{bmatrix} \mathbf{b}_a \\ \mathbf{b}_b \end{bmatrix}$$

BABD vs ABD systems

Boundary Value Problems for Ordinary Differential Equations (BVODEs)

$$y' = A(x)y(x) + q(x), \quad B_a y(a) + B_b y(b) = 0, \quad y, q \in \mathbb{R}^n, \; x \in [a, b].$$

yield Bordered Almost Block Diagonal (BABD) system

$$\begin{bmatrix} S_0 & T_0 & & & \\ & S_1 & T_1 & & \\ & & \ddots & \ddots & \\ & & & S_{N-1} & T_{N-1} \\ B_a & & & & B_b \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{N-1} \\ \mathbf{b}_N \end{bmatrix}$$

where S_i, T_i, B_a, B_b are square $n \times n$ blocks. When BCs are separable, i.e.

$$B_a = \begin{bmatrix} \bar{B}_a \\ \mathbb{O} \end{bmatrix}, \quad B_b = \begin{bmatrix} \mathbb{O} \\ \bar{B}_b \end{bmatrix}, \quad \mathbf{b}_N = \begin{bmatrix} \mathbf{b}_a \\ \mathbf{b}_b \end{bmatrix}$$

we obtain an Almost Block Diagonal (ABD) system

$$\begin{bmatrix} \bar{B}_a & & & \\ S_0 & T_0 & & \\ & \ddots & \ddots & \\ & & S_{N-1} & T_{N-1} \\ & & & \bar{B}_b \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} \mathbf{b}_a \\ \mathbf{b}_0 \\ \vdots \\ \mathbf{b}_{N-1} \\ \mathbf{b}_b \end{bmatrix}$$

Motivations and other applications

Numerical methods for nonlinear BVODEs

$$\begin{aligned} &y' = f(x, y(x)), && y, f \in \mathbb{R}^n,\ x \in [a, b] \\ &g(y(a), y(b)) = 0. \end{aligned}$$

require the solution of a sequence of BABD/ABD linear systems.

Motivations and other applications

Numerical methods for nonlinear BVODEs

$$y' = f(x, y(x)), \qquad\qquad y, f \in \mathbb{R}^n,\ x \in [a, b]$$
$$g(y(a), y(b)) = 0.$$

require the solution of a sequence of BABD/ABD linear systems.

- Model Predictive Control
- Markov chains modeling
- Quantum Monte Carlo simulations
- Parameter estimation with non-linear DAE models

Structured Orthogonal Factorization - SOF

Local Factorization [Wright 1992]

$$\left[\begin{array}{cccccc|c}
S_0 & T_0 & & & & & \mathbf{b}_0 \\
 & \ddots & & & & & \vdots \\
 & S_{k_1-1} & T_{k_1-1} & & & & \mathbf{b}_{k_1-1} \\
\hline
 & & \ddots & & & & \vdots \\
\hline
 & & S_{k_{P-1}} & T_{k_{P-1}} & & & \mathbf{b}_{k_{P-1}} \\
 & & & \ddots & & & \vdots \\
 & & & S_{N-1} & T_{N-1} & & \mathbf{b}_{N-1} \\
\hline
B_a & & & & B_b & & \mathbf{b}_N
\end{array}\right]$$

- Divide the BABD system into P slices with roughly the same number of block rows and assign each slice to one processor.

Local Factorization [Wright 1992]

$$\left[\begin{array}{cccccc|c} S_{k_p} & T_{k_p} & & & & & \mathbf{b}_{k_p} \\ & S_{k_p+1} & T_{k_p+1} & & & & \mathbf{b}_{k_p+1} \\ & & S_{k_p+2} & T_{k_p+2} & & & \mathbf{b}_{k_p+2} \\ & & & \ddots & & & \vdots \\ & & & & S_{k_{p+1}-1} & T_{k_{p+1}-1} & \mathbf{b}_{k_{p+1}-1} \end{array}\right]$$

- Divide the BABD system into P slices with roughly the same number of block rows and assign each slice to one processor.

Local Factorization [Wright 1992]

$$\left[\begin{array}{cccccc|c} S_{k_p} & T_{k_p} & & & & & \mathbf{b}_{k_p} \\ & S_{k_p+1} & T_{k_p+1} & & & & \mathbf{b}_{k_p+1} \\ & & S_{k_p+2} & T_{k_p+2} & & & \mathbf{b}_{k_p+2} \\ & & & \ddots & & & \vdots \\ & & & & S_{k_{p+1}-1} & T_{k_{p+1}-1} & \mathbf{b}_{k_{p+1}-1} \end{array}\right]$$

- Divide the BABD system into P slices with roughly the same number of block rows and assign each slice to one processor.
- Find Q_0 orthogonal such that

$$\begin{bmatrix} T_{k_p} \\ S_{k_p+1} \end{bmatrix} = Q_0 \begin{bmatrix} U_{k_p} \\ \mathbb{O} \end{bmatrix}.$$

Local Factorization [Wright 1992]

$$\begin{bmatrix} Q_0^T & \\ & \mathbb{I}_{(k-2)n} \end{bmatrix} \left[\begin{array}{cccccc|c} S_{k_p} & T_{k_p} & & & & & \mathbf{b}_{k_p} \\ & S_{k_p+1} & T_{k_p+1} & & & & \mathbf{b}_{k_p+1} \\ & & S_{k_p+2} & T_{k_p+2} & & & \mathbf{b}_{k_p+2} \\ & & & \ddots & & & \vdots \\ & & & & S_{k_{p+1}-1} & T_{k_{p+1}-1} & \mathbf{b}_{k_{p+1}-1} \end{array}\right]$$

- Divide the BABD system into P slices with roughly the same number of block rows and assign each slice to one processor.
- Find Q_0 orthogonal such that

$$\begin{bmatrix} T_{k_p} \\ S_{k_p+1} \end{bmatrix} = Q_0 \begin{bmatrix} U_{k_p} \\ \mathbb{O} \end{bmatrix}.$$

- Apply Q_0^T to update the system.

Local Factorization [Wright 1992]

$$
\left[\begin{array}{cccccc|c}
V_{k_p} & U_{k_p} & W_{k_p} & & & & \mathbf{f}_{k_p} \\
\overline{V}_{k_p+1} & & \overline{W}_{k_p+1} & & & & \overline{\mathbf{f}}_{k_p+1} \\
& & S_{k_p+2} & T_{k_p+2} & & & \mathbf{b}_{k_p+2} \\
& & & \ddots & \ddots & & \vdots \\
& & & & S_{k_{p+1}-1} & T_{k_{p+1}-1} & \mathbf{b}_{k_{p+1}-1}
\end{array}\right]
$$

- Divide the BABD system into P slices with roughly the same number of block rows and assign each slice to one processor.
- Find Q_0 orthogonal such that

$$
\begin{bmatrix} T_{k_p} \\ S_{k_p+1} \end{bmatrix} = Q_0 \begin{bmatrix} U_{k_p} \\ \mathbb{O} \end{bmatrix}.
$$

- Apply Q_0^T to update the system.

Local Factorization [Wright 1992]

$$\left[\begin{array}{cccccc|c} V_{k_p} & U_{k_p} & W_{k_p} & & & & \mathbf{f}_{k_p} \\ \overline{V}_{k_p+1} & & \overline{W}_{k_p+1} & & & & \overline{\mathbf{f}}_{k_p+1} \\ & & S_{k_p+2} & T_{k_p+2} & & & \mathbf{b}_{k_p+2} \\ & & & \ddots & \ddots & & \vdots \\ & & & & S_{k_{p+1}-1} & T_{k_{p+1}-1} & \mathbf{b}_{k_{p+1}-1} \end{array}\right]$$

- Divide the BABD system into P slices with roughly the same number of block rows and assign each slice to one processor.
- Find Q_0 orthogonal such that

$$\begin{bmatrix} T_{k_p} \\ S_{k_p+1} \end{bmatrix} = Q_0 \begin{bmatrix} U_{k_p} \\ \mathbb{O} \end{bmatrix}.$$

- Apply Q_0^T to update the system.
- Find Q_1 orthogonal such that

$$\begin{bmatrix} \overline{W}_{k_p+1} \\ S_{k_p+2} \end{bmatrix} = Q_1 \begin{bmatrix} U_{k_p+1} \\ \mathbb{O} \end{bmatrix}.$$

Local Factorization [Wright 1992]

$$\begin{bmatrix} \mathbb{I}_n & & \\ & Q_1^T & \\ & & \mathbb{I}_{(k-3)n} \end{bmatrix} \left[\begin{array}{cccccc|c} V_{k_p} & U_{k_p} & W_{k_p} & & & & \mathbf{f}_{k_p} \\ \overline{V}_{k_p+1} & & \overline{W}_{k_p+1} & & & & \bar{\mathbf{f}}_{k_p+1} \\ & & S_{k_p+2} & T_{k_p+2} & & & \mathbf{b}_{k_p+2} \\ & & & \ddots & \ddots & & \vdots \\ & & & & S_{k_{p+1}-1} & T_{k_{p+1}-1} & \mathbf{b}_{k_{p+1}-1} \end{array}\right]$$

- Divide the BABD system into P slices with roughly the same number of block rows and assign each slice to one processor.
- Find Q_0 orthogonal such that

$$\begin{bmatrix} T_{k_p} \\ S_{k_p+1} \end{bmatrix} = Q_0 \begin{bmatrix} U_{k_p} \\ \mathbb{O} \end{bmatrix}.$$

- Apply Q_0^T to update the system.
- Find Q_1 orthogonal such that

$$\begin{bmatrix} \overline{W}_{k_p+1} \\ S_{k_p+2} \end{bmatrix} = Q_1 \begin{bmatrix} U_{k_p+1} \\ \mathbb{O} \end{bmatrix}.$$

- Apply Q_1^T to update the system.

Local Factorization [Wright 1992]

$$
\left[\begin{array}{cccccc|c}
V_{k_p} & U_{k_p} & W_{k_p} & & & & \mathbf{f}_{k_p} \\
V_{k_p+1} & & U_{k_p+1} & W_{k_p+1} & & & \mathbf{f}_{k_p+1} \\
\overline{V}_{k_p+2} & & & \overline{W}_{k_p+2} & & & \overline{\mathbf{f}}_{k_p+2} \\
 & & & \ddots & \ddots & & \vdots \\
 & & & & S_{k_{p+1}-1} & T_{k_{p+1}-1} & \mathbf{b}_{k_{p+1}-1}
\end{array}\right]
$$

- Divide the BABD system into P slices with roughly the same number of block rows and assign each slice to one processor.
- Find Q_0 orthogonal such that

$$\begin{bmatrix} T_{k_p} \\ S_{k_p+1} \end{bmatrix} = Q_0 \begin{bmatrix} U_{k_p} \\ \mathbb{O} \end{bmatrix}.$$

- Apply Q_0^T to update the system.
- Find Q_1 orthogonal such that

$$\begin{bmatrix} \overline{W}_{k_p+1} \\ S_{k_p+2} \end{bmatrix} = Q_1 \begin{bmatrix} U_{k_p+1} \\ \mathbb{O} \end{bmatrix}.$$

- Apply Q_1^T to update the system.

Local Factorization [Wright 1992]

$$\left[\begin{array}{ccccccc|c} V_{k_p} & U_{k_p} & W_{k_p} & & & & & \mathbf{f}_{k_p} \\ V_{k_p+1} & & U_{k_p+1} & W_{k_p+1} & & & & \mathbf{f}_{k_p+1} \\ \vdots & & & \ddots & \ddots & & & \vdots \\ V_{k_{p+1}-2} & & & & U_{k_{p+1}-2} & W_{k_{p+1}-2} & & \mathbf{f}_{k_{p+1}-1} \\ S'_p & & & & & T'_p & & \mathbf{b}'_p \end{array}\right]$$

- Divide the BABD system into P slices with roughly the same number of block rows and assign each slice to one processor.
- Find Q_0 orthogonal such that

$$\begin{bmatrix} T_{k_p} \\ S_{k_p+1} \end{bmatrix} = Q_0 \begin{bmatrix} U_{k_p} \\ \mathbb{O} \end{bmatrix}.$$

- Apply Q_0^T to update the system.
- Find Q_1 orthogonal such that

$$\begin{bmatrix} \overline{W}_{k_p+1} \\ S_{k_p+2} \end{bmatrix} = Q_1 \begin{bmatrix} U_{k_p+1} \\ \mathbb{O} \end{bmatrix}.$$

- Apply Q_1^T to update the system.
- Repet ultil all rows have been processed.

Recursive procedure [Wright 1992]

By concatenating all local factorizations, we obtain the equivalent system

$$Q^T[A|b] = \underbrace{\left[\begin{array}{cccccc|c}
V_0 & U_0 & W_0 & & & & \mathbf{f}_0 \\
\vdots & \ddots & & & & & \vdots \\
S_0' & & T_0' & & & & \mathbf{b}_0' \\
\hline
 & & \ddots & & & & \vdots \\
\hline
 & & V_{k_{P-1}} & U_{k_{P-1}} & W_{k_{P-1}} & & \mathbf{f}_{k_{P-1}} \\
 & & \vdots & \ddots & & & \vdots \\
 & & S_{P-1}' & & T_{P-1}' & & \mathbf{b}_{N'-1}' \\
\hline
B_a & & & & B_b & & \mathbf{b}_N
\end{array}\right]}_{[\hat{A},\hat{b}]}$$

Recursive procedure [Wright 1992]

By concatenating all local factorizations, we obtain the equivalent system

$$Q^T[A|b] = \underbrace{\left[\begin{array}{cccccc|c}
V_0 & U_0 & W_0 & & & & \mathbf{f}_0 \\
\vdots & \ddots & & & & & \vdots \\
S'_0 & & T'_0 & & & & \mathbf{b}'_0 \\
\hline
& & \ddots & & & & \vdots \\
\hline
& & V_{k_{P-1}} & U_{k_{P-1}} & W_{k_{P-1}} & & \mathbf{f}_{k_{P-1}} \\
& & \vdots & \ddots & & & \vdots \\
& & S'_{P-1} & & T'_{P-1} & & \mathbf{b}'_{N'-1} \\
\hline
B_a & & & & B_b & & \mathbf{b}_N
\end{array}\right]}_{[\hat{A},\hat{b}]}$$

The solution of $Ax = b$ is decoupled as

Recursive procedure [Wright 1992]

By concatenating all local factorizations, we obtain the equivalent system

$$Q^T[A|b] = \underbrace{\left[\begin{array}{cccccc|c} V_0 & U_0 & W_0 & & & & \mathbf{f}_0 \\ \vdots & \ddots & & & & & \vdots \\ S_0' & & T_0' & & & & \mathbf{b}_0' \\ \hline & & \ddots & & & & \vdots \\ \hline & & V_{k_P-1} & U_{k_P-1} & W_{k_P-1} & & \mathbf{f}_{k_P-1} \\ & & \vdots & \ddots & & & \vdots \\ & & S_{P-1}' & & T_{P-1}' & & \mathbf{b}_{N'-1}' \\ \hline B_a & & & & B_b & & \mathbf{b}_N \end{array}\right]}_{[\hat{A}, \hat{b}]}$$

The solution of $Ax = b$ is decoupled as

1. obtain selected unknowns by solving the BABD system $\hat{A}$ of reduced size (recursion)

Recursive procedure [Wright 1992]

By concatenating all local factorizations, we obtain the equivalent system

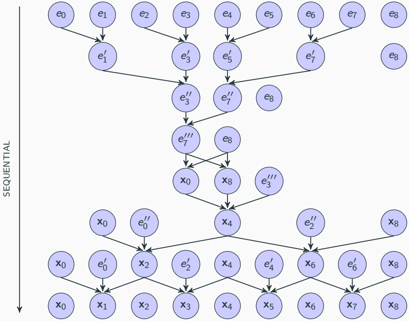
$$
Q^T[A|b] = \underbrace{\left[\begin{array}{cccccc|c}
V_0 & U_0 & W_0 & & & & \mathbf{f}_0 \\
\vdots & \ddots & & & & & \vdots \\
S'_0 & & T'_0 & & & & \mathbf{b}'_0 \\
\hline
 & & \ddots & & & & \vdots \\
\hline
 & & V_{k_P-1} & U_{k_P-1} & W_{k_P-1} & & \mathbf{f}_{k_P-1} \\
 & & \vdots & \ddots & & & \vdots \\
 & & S'_{P-1} & & T'_{P-1} & & \mathbf{b}'_{N'-1} \\
\hline
B_a & & & & B_b & & \mathbf{b}_N
\end{array}\right]}_{[\hat{A},\hat{b}]}
$$

The solution of $Ax = b$ is decoupled as

1. obtain selected unknowns by solving the BABD system $\hat{A}$ of reduced size (recursion)
2. retrieve the missing unknowns by back-substitution

SOF's workflow

Communication pattern in the case $N = 8$ ($9n$ unknowns) with $P = 4$ slices (and processors) of $k = 2$ block rows each, showing the dataflow between each block equation.



- $P \leq N/2$ processors needed
- $2log_2P$ sequential steps
- at each step half of the processors active at the previous step stays idle
- the amount of parallel work is likely not enough to fully exploit GPUs' potential

PARAllel Structured Orthogonal Factorization - PARASOF

Odd/Even SOF

Idea

Decouple odd/even unknowns in a parallel cyclic reduction fashion, i.e. i-th block row is coupled with both $i-1$-th and $i+1$-th block row.

Odd/Even SOF

Idea

Decouple odd/even unknowns in a parallel cyclic reduction fashion, i.e. i-th block row is coupled with both $i-1$-th and $i+1$-th block row.

Suppose $N+1$ is even. For even unknowns, apply SOF to

$$\left[\begin{array}{cccccc} S_0 & T_0 & & & & \\ & S_1 & T_1 & & & \\ \hline & & \ddots & \ddots & & \\ \hline & & & & S_{N-1} & T_{N-1} \\ B_a & & & & & B_b \end{array}\right]$$

Odd/Even SOF

Idea

Decouple odd/even unknowns in a parallel cyclic reduction fashion, i.e. i-th block row is coupled with both $i-1$-th and $i+1$-th block row.

Suppose $N+1$ is even. For even unknowns, apply SOF to

$$\left[\begin{array}{ccccc} S_0 & T_0 & & & \\ & S_1 & T_1 & & \\ \hline & & \ddots & \ddots & \\ \hline & & & S_{N-1} & T_{N-1} \\ B_a & & & & B_b \end{array}\right] \longrightarrow Q_e A x = Q_e b$$

Odd/Even SOF

Idea

Decouple odd/even unknowns in a parallel cyclic reduction fashion, i.e. i-th block row is coupled with both $i-1$-th and $i+1$-th block row.

Suppose $N+1$ is even. For even unknowns, apply SOF to

$$\left[\begin{array}{cccccc} S_0 & T_0 & & & & \\ & S_1 & T_1 & & & \\ \hline & & \ddots & \ddots & & \\ \hline & & & S_{N-1} & T_{N-1} \\ B_a & & & & B_b \end{array}\right] \longrightarrow Q_e A x = Q_e b \longrightarrow \begin{cases} A_e x_e = b_e \\ U_e x_o = b_o - A_o x_e \end{cases}$$

Odd/Even SOF

Idea

Decouple odd/even unknowns in a parallel cyclic reduction fashion, i.e. i-th block row is coupled with both $i-1$-th and $i+1$-th block row.

Suppose $N+1$ is even. For even unknowns, apply SOF to

$$\left[\begin{array}{cccccc} S_0 & T_0 & & & & \\ & S_1 & T_1 & & & \\ \hline & & \ddots & \ddots & & \\ \hline & & & & S_{N-1} & T_{N-1} \\ B_a & & & & & B_b \end{array}\right] \longrightarrow Q_e A x = Q_e b \longrightarrow \begin{cases} A_e x_e = b_e \\ U_e x_o = b_o - A_o x_e \end{cases}$$

For odd unknowns, apply SOF to

$$\left[\begin{array}{cccccc} & S_1 & T_1 & & & \\ & & S_2 & T_2 & & \\ \hline & & & \ddots & \ddots & \\ \hline S_0 & T_0 & & & & \\ B_a & & & & & B_b \end{array}\right]$$

Odd/Even SOF

Idea

Decouple odd/even unknowns in a parallel cyclic reduction fashion, i.e. i-th block row is coupled with both $i-1$-th and $i+1$-th block row.

Suppose $N+1$ is even. For even unknowns, apply SOF to

$$\left[\begin{array}{cccccc} S_0 & T_0 & & & & \\ & S_1 & T_1 & & & \\ \hline & & \ddots & \ddots & & \\ \hline & & & & S_{N-1} & T_{N-1} \\ B_a & & & & & B_b \end{array}\right] \longrightarrow Q_e A x = Q_e b \longrightarrow \begin{cases} A_e x_e = b_e \\ U_e x_o = b_o - A_o x_e \end{cases}$$

For odd unknowns, apply SOF to

$$\left[\begin{array}{cccccc} & S_1 & T_1 & & & \\ & & S_2 & T_2 & & \\ \hline & & & \ddots & \ddots & \\ \hline S_0 & T_0 & & & & \\ B_a & & & & & B_b \end{array}\right] \longrightarrow Q_o A x = Q_o b$$

Odd/Even SOF

Idea

Decouple odd/even unknowns in a parallel cyclic reduction fashion, i.e. i-th block row is coupled with both $i-1$-th and $i+1$-th block row.

Suppose $N+1$ is even. For even unknowns, apply SOF to

$$\left[\begin{array}{cccccc} S_0 & T_0 & & & & \\ & S_1 & T_1 & & & \\ \hline & & \ddots & \ddots & & \\ \hline & & & & S_{N-1} & T_{N-1} \\ B_a & & & & & B_b \end{array}\right] \longrightarrow Q_e A x = Q_e b \longrightarrow \begin{cases} A_e x_e = b_e \\ U_e x_o = b_o - A_o x_e \end{cases}$$

For odd unknowns, apply SOF to

$$\left[\begin{array}{cccccc} & S_1 & T_1 & & & \\ & & S_2 & T_2 & & \\ \hline & & & \ddots & \ddots & \\ \hline S_0 & T_0 & & & & \\ B_a & & & & & B_b \end{array}\right] \longrightarrow Q_o A x = Q_o b \longrightarrow \begin{cases} A_o x_o = b_o \\ U_o x_e = b_e - A_e x_e \end{cases}$$

Odd/Even SOF

Idea

Decouple odd/even unknowns in a parallel cyclic reduction fashion, i.e. i-th block row is coupled with both $i-1$-th and $i+1$-th block row.

Suppose $N+1$ is even. For even unknowns, apply SOF to

$$\left[\begin{array}{ccccc} S_0 & T_0 & & & \\ & S_1 & T_1 & & \\ \hline & & \ddots & \ddots & \\ \hline & & & S_{N-1} & T_{N-1} \\ B_a & & & & B_b \end{array}\right] \longrightarrow Q_e A x = Q_e b \longrightarrow \begin{cases} A_e x_e = b_e \\ U_e x_o = b_o - A_o x_e \end{cases}$$

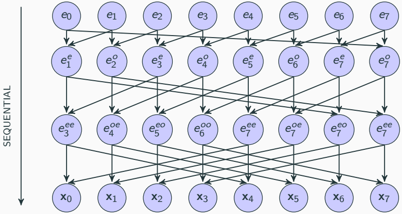
For odd unknowns, apply SOF to

$$\left[\begin{array}{cccccc} & S_1 & T_1 & & & \\ & & S_2 & T_2 & & \\ \hline & & & \ddots & \ddots & \\ \hline S_0 & T_0 & & & & \\ B_a & & & & & B_b \end{array}\right] \longrightarrow Q_o A x = Q_o b \longrightarrow \begin{cases} A_o x_o = b_o \\ U_o x_e = b_e - A_e x_e \end{cases}$$

These two orthogonal transformation can be performed in parallel and recursively.

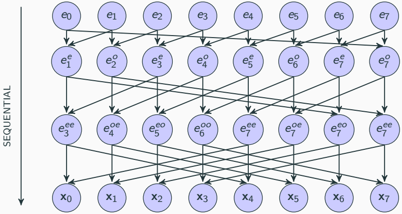
Odd/Even SOF's workflow

Communication pattern in the case $N = 7$ ($8n$ unknowns) with $P = 8$ slices (and processors) of $k = 2$ block rows each, showing the dataflow between each block equation.



- $P = N$ processors needed
- roughly $\log_2 N$ algorithmic steps
- no processor stays idle
- all steps contain the same amount of work

Communication pattern in the case $N = 7$ ($8n$ unknowns) with $P = 8$ slices (and processors) of $k = 2$ block rows each, showing the dataflow between each block equation.



- $P = N$ processors needed
- roughly $\log_2 N$ algorithmic steps
- no processor stays idle
- all steps contain the same amount of work

Observation

In a real application we often have $N \gg P$, thus the available parallel work is somehow serialized in chunks even on massively parallel architectures.

1. Apply one step of SOF's forward reduction phase of obtain a reduced $P \times P$ block system.

PARASOF

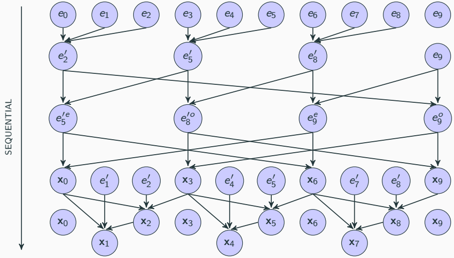
1. Apply one step of SOF's forward reduction phase of obtain a reduced $P \times P$ block system.
2. Solve the reduced intermediate system with the odd/even SOF algorithm

PARASOF

1. Apply one step of SOF's forward reduction phase of obtain a reduced $P \times P$ block system.
2. Solve the reduced intermediate system with the odd/even SOF algorithm
3. Retrieve the missing unknowns using one step of backward substitution

PARASOF

1. Apply one step of SOF's forward reduction phase of obtain a reduced $P \times P$ block system.
2. Solve the reduced intermediate system with the odd/even SOF algorithm
3. Retrieve the missing unknowns using one step of backward substitution



- arbitrary number of processors P
- $log_2 P + 2$ sequential steps
- no idle processors
- minimal amount of serialized work

Numerical Experiments

Theoretical speed-ups

- Local QR is computed with Householder reflectors
- $P =$ number of processors
- $P_c =$ number of coarse grained processors (Streaming Multiprocessors)
- $P_f =$ number of fine grained processors (warps)

Theoretical speed-ups

- Local QR is computed with Householder reflectors
- P = number of processors
- P_c = number of coarse grained processors (Streaming Multiprocessors)
- P_f = number of fine grained processors (warps)

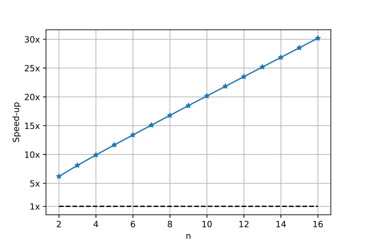
Algorithm	# steps	Factorization	Memory
SOF	$2\log_2(P)$	$\frac{46}{3}n^3\left(\frac{N}{P}+L-1\right)$	$4n^2\left(N+P\ L\right)+n(N+P)$
PARASOF	$\log_2(P_c)$	$\frac{42}{3}\frac{n^3}{P_f}L_r+\frac{46}{3}\frac{n^3}{P_f}\left(\frac{N}{P_c}\right)$	$2n^2P_c+4n^2\left(N+P_c\right)+n(N+P_c)$

Table 1: Complexity comparison of algorithms in terms of algorithmic steps and operation count.

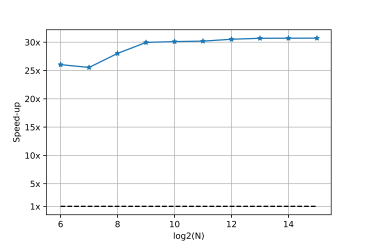
Theoretical speed-ups

- Local QR is computed with Householder reflectors
- P = number of processors
- P_c = number of coarse grained processors (Streaming Multiprocessors)
- P_f = number of fine grained processors (warps)

Setting: $P = 16,\ P_c = 10,\ P_f = 32.$



(a) Theoretical speed-up (PARASOF vs SOF), $N = 2^{11}$.



(b) Theoretical speed-up (PARASOF vs SOF), $n = 16$.

Figure 1: Theoretical speedup in function of the size n (left) and the number N (right) of internal blocks.

Experiments' setting

- C/CUDA language
- randomly generated linear systems (dense blocks, worst case)
- umfpack (Davis 2004), a well optimized CPU multifrontal LU factorization

Experiments' setting

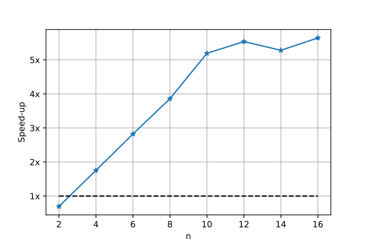
- C/CUDA language
- randomly generated linear systems (dense blocks, worst case)
- umfpack (Davis 2004), a well optimized CPU multifrontal LU factorization

We test its performance by running the algorithm on two different workstations:

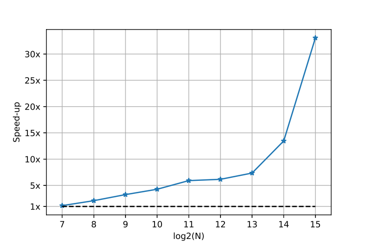
1. **dellcuda1**, with two 1.80GHz Intel(R) Xeon(R) CPU E5-2630L v3 CPU and a Nvidia TITAN Xp graphic card;
2. **gpu01**, with a 3.50GHz Intel(R) Core(TM) i7-2700K CPU and a Nvidia GeForce GTX1060 graphic card.

Speed-up on gpu01

- Nr = size of reduced system that is solved with odd/even SOF.



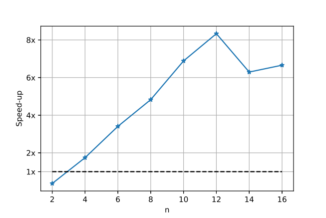
(a) Speed-up , $N = 2^{11}$ and $N_r = 63$.



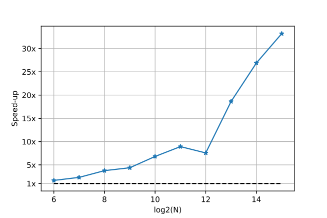
(b) Speed-up, $n = 16$ and $N_r = 63$.

Figure 2: PARASOF speed-up over `spsolve` on `gpu01`.

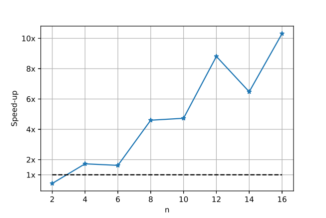
Speed-up on dellcuda1



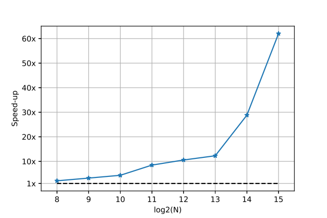
(a) Speed-up , $N = 2^{11}$ and $N_r = 63$



(b) Speed-up, $n = 16$ and $N_r = 63$.



(c) Speed-up , $N = 2^{11}$ and $N_r = 127$.



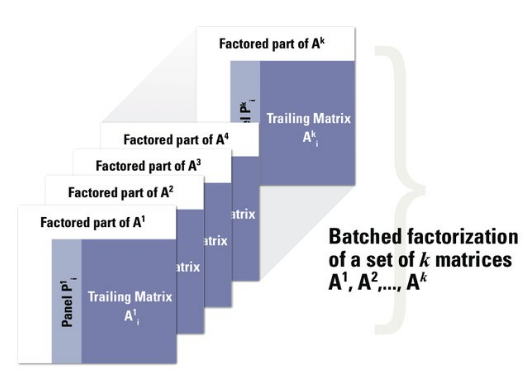
(d) Speed-up, $n = 16$ and $N_r = 127$.

Figure 3: PARASOF speed-up over `spsolve` on `dellcuda1`.

Conclusions and Future work

Fast Givens Transformations on GPUs

Batched routines computes multiple and independent linear algebra operations on small-sized matrices and/or vectors in a single routine call.



- Batched Givens QR can improve speed-ups exploiting sparsity of block rows whenever possible

Conclusions

- New stable parallel algorithm for solving of BABD systems has been proposed
- Same technique can be extended to the parallel solution of ABD systems with minor changes
- Speed-up up to 60x can be achieved in comparison to optimized CPU methods
- Timings are architecture dependent
- In particular, further optimization can be achieved with Givens rotations

Thank you for your attention!

References

- Amodio, P. et al. (2000). "Almost block diagonal linear systems: sequential and parallel solution techniques, and applications". In: *Numerical Linear Algebra with Applications* 7.5, pp. 275–317.
- Davis, T. A. (2004). "Algorithm 832: UMFPACK V4.3—an Unsymmetric-pattern Multifrontal Method". In: *ACM Trans. Math. Softw.* 30.2, pp. 196–199. ISSN: 0098-3500. DOI: 10.1145/992200.992206. URL: http://doi.acm.org/10.1145/992200.992206.
- R.W. Hockney, C. J. (1983). *Parallel Computers*.
- Wright, S. J. (1992). "Stable Parallel Algorithms For Two-Point Boundary Value Problems". In: *SIAM J. Sci. Statist. Comput* 13, pp. 742–764.

Instability phenomena Wright 1992

Consider the linear BVODE

$$y' = \begin{pmatrix} -1/6 & 1 \\ 1 & -1/6 \end{pmatrix} y + \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad y_a + y_b = 0, \quad x \in [0, 60].$$

This problem is well conditioned in the Hadamard sense.

Instability phenomena Wright 1992

Consider the linear BVODE

$$y' = \begin{pmatrix} -1/6 & 1 \\ 1 & -1/6 \end{pmatrix} y + \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad y_a + y_b = 0, \quad x \in [0, 60].$$

This problem is well conditioned in the Hadamard sense.
Standard discretization leads the following BABD matrix

$$A = \begin{bmatrix} \mathbb{I} & & & & \mathbb{I} \\ -B & \mathbb{I} & & & \\ & -B & \mathbb{I} & & \\ & & \ddots & \ddots & \\ & & & -B & \mathbb{I} \end{bmatrix}$$

Instability phenomena Wright 1992

Consider the linear BVODE

$$y' = \begin{pmatrix} -1/6 & 1 \\ 1 & -1/6 \end{pmatrix} y + \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad y_a + y_b = 0, \quad x \in [0, 60].$$

This problem is well conditioned in the Hadamard sense.
Standard discretization leads the following BABD matrix

$$A = \begin{bmatrix} \mathbb{I} & & & & \mathbb{I} \\ -B & \mathbb{I} & & & \\ & -B & \mathbb{I} & & \\ & & \ddots & \ddots & \\ & & & -B & \mathbb{I} \end{bmatrix}$$

Using Gaussian elimination with partial pivoting, no interchanges are required, and

$$A = LU = \begin{bmatrix} \mathbb{I} & & & & \\ -B & \mathbb{I} & & & \\ & -B & \mathbb{I} & & \\ & & \ddots & \ddots & \\ & & & -B & \bar{L} \end{bmatrix} \begin{bmatrix} \mathbb{I} & & & & \mathbb{I} \\ & \mathbb{I} & & & B \\ & & \ddots & & \vdots \\ & & & \mathbb{I} & B^{N-1} \\ & & & & \bar{U} \end{bmatrix}$$

The elements in the last column of U grow exponentially with N.