Parallel implementation of block circulant type preconditioner for all-at-once systems of linear time-depdent PDEs

EAGSTIM24 (April 3-5, 2024, Pisa, Italy)

†Bergische Universität Wuppertal, Germany

Ryo Yoda†, Matthias Bolten†

BERGISCHE UNIVERSITÄT WUPPERTAL

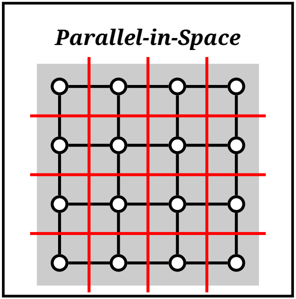
Outline

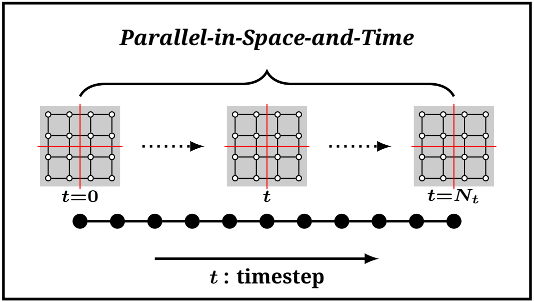
Outline

Background

- **Parallel solvers for time-dependent PDEs on modern systems**
  - **Spatial parallelism is exhausted on massively parallel envirs.**
  - **Extracts temporal parallelism by solving all time steps at once.**

    $\Rightarrow$ ***Parallel-in-Time approaches / All-at-once approaches***





Existing works

- **Some powerful parallel-in-time methods**
  - **Comprehensive survey papers** [Gander, 2015] [Ong and Schroder, 2020]
  - **Multilevel-base: Parareal** [Lions *et al.*, 2001] / **MGRIT** [Falgout *et al.*, 2014]
    - + **Good parallel performance for parabolic problems.**
    - - **Difficulty in selecting an appropriate coarse-grid operator for hyperoblic problems.**
  - **Diagonalization-base: Block ciruclant (BC) prec.** [McDonalds *et al.*, 2016]
    Block ϵ-ciruclant (BEC) prec. [Lin *et al.*, 2021]
    - + **Independent convergence for spatial mesh-size and time-step width, introducing a weight parameter ϵ.**
    - - **Assumption to use the same linear time integrator for all steps.**

Overview of this work

- **Motivation and Aim**
  - **Still not many parallel evaluations of BEC (Parallel results for the same context ParaDiag [Gander *et al.*, 2021][Caklovic *et al.*, 2023])**
  - **What is the parallel performance compared to classical time stepping or multigrid-based?**

    $\Rightarrow$ **This work investigates parallel performance of BEC prec.**

- **Novelty and Originality**
  - **Memory-distributed space-time parallelization of BEC prec. with `FFTW`,`Trilinos/Epetra`,`AztecOO`,`ML` libraries**
  - **Evaluate three types of implementations of FFT parts.**
  - **Compared with the sequential time-stepping method and multigrid reduction in time (MGRIT).**

Outline

All-at-once systems for time-dependent PDEs

- **Time-dependent PDEs / time-stepping**

$$u_t(x,t) = a\Delta u(x,t) + f(x,t) \quad (x,t) \in \Omega \times (0,T]$$
$$M\left(\frac{u_i - u_{i-1}}{\Delta t}\right) + K u_i = f_i \quad (i = 1, \dots, n_t)$$

n_x : **Spatial DoFs**
$M \in \mathbb{R}^{n_x \times n_x}$: **Mass matrix**
$K \in \mathbb{R}^{n_x \times n_x}$: **Stiffness matrix**
n_t : **Number of time steps**
$\Delta t \in \mathbb{R}$: **Time-step width**
Backward Euler scheme

- **All-at-once system with** $A_0 = M + \Delta t K$ **and** $A_1 = -M$

$$\mathcal{A}\mathbf{u} := \begin{bmatrix} A_0 & & & \\ A_1 & A_0 & & \\ & \ddots & \ddots & \\ & & A_1 & A_0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n_t} \end{bmatrix} = \begin{bmatrix} \Delta t f_1 - A_1 u_0 \\ \Delta t f_2 \\ \vdots \\ \Delta t f_{n_t} \end{bmatrix} =: \mathbf{f} \in \mathbb{R}^{n_t n_x}$$

$\Rightarrow$ $\mathcal{A}$ **has a block Toeplitz structure with the same time integrator**

Block circulant type preconditioners

- **Block circulant (BC) preconditioner** [McDonalds *et al.*, 2016]

$$\mathcal{P}_{\mathbf{BC}} \in \mathbb{R}^{n_t n_x \times n_t n_x} = \begin{bmatrix} A_0 & & & A_1 \\ A_1 & A_0 & & \\ & \ddots & \ddots & \\ & & A_1 & A_0 \end{bmatrix}$$

  - $\mathcal{P}_{\mathbf{BC}}$ **has a block circulant structure.**
    $\Rightarrow \mathcal{P}_{\mathbf{BC}}^{-1}\mathcal{A}$ **is diagonalizable when** $\mathcal{A}$ **is not.**
  - **Independent convergence for the number of time steps.**

- **Block** ϵ**-circulant (BEC) preconditioner** [Lin *et al.*, 2021]

$$\mathcal{P}_{\mathbf{BEC}} \in \mathbb{R}^{n_t n_x \times n_t n_x} = \begin{bmatrix} A_0 & & & \epsilon A_1 \\ A_1 & A_0 & & \\ & \ddots & \ddots & \\ & & A_1 & A_0 \end{bmatrix}$$

  - **Introduces a parameter** ϵ.
  - **Independent convergence for the number of time steps and spatial mesh size with sufficiently small** ϵ**.**

Three-step procedure of BEC precond.

$$R_{\textbf{BEC}} = \begin{bmatrix} r_0 & & \epsilon r_p & \cdots & \epsilon r_2 & \epsilon r_1 \\ r_1 & r_0 & & \ddots & \ddots & \epsilon r_2 \\ \vdots & \ddots & \ddots & & \ddots & \vdots \\ r_p & \ddots & \ddots & \ddots & & \epsilon r_p \\ & \ddots & \ddots & r_1 & r_0 & \\ & & r_p & \ddots & r_1 & r_0 \end{bmatrix} \in \mathbb{R}^{n_t \times n_t}$$

$$R_{\textbf{BEC}} = D_\epsilon^{-1} \mathcal{F}_{n_t}^* \Lambda_\epsilon \mathcal{F}_{n_t} D_\epsilon$$

$$\mathcal{F}_{n_t} := n_t\textbf{-sized DFT matrix}$$

$$D_\epsilon := \textbf{diag}\left(\epsilon^{\frac{0}{n_t}}, \epsilon^{\frac{1}{n_t}}, \dots, \epsilon^{\frac{n_t-1}{n_t}}\right)$$

$$\Lambda_\epsilon := \textbf{diag}\left(\lambda_0^{(\epsilon)}, \lambda_1^{(\epsilon)}, \dots, \lambda_{n_t-1}^{(\epsilon)}\right)$$

$$\mathcal{P}_{\textbf{BEC}} = R_{\textbf{BEC}} \otimes M + \Delta t I_{n_t} \otimes K$$

$$= \left[(D_\epsilon^{-1} \mathcal{F}_{n_t}^*) \otimes I_{n_x}\right] \textbf{blockdiag}(B_0, B_1, \dots, B_{n_t-1}) \left[(\mathcal{F}_{n_t} D_\epsilon) \otimes I_{n_x}\right],$$

where $B_k = \lambda_k^{(\epsilon)} M + \Delta t K \in \mathbb{C}^{n_x \times n_x}, \quad (k = 0, 1, \dots, n_t - 1)$

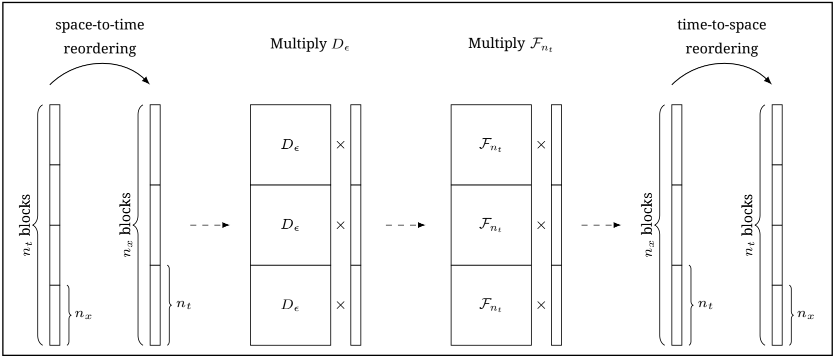
Three-step procedure of BEC precond. $\mathbf{z} = \mathcal{P}_{\textbf{BEC}}^{-1} \mathbf{y}$

1. **Compute** $\tilde{\mathbf{y}} = [(\mathcal{F}_{n_t} D_\epsilon) \otimes I_{n_x}]\, \mathbf{y}$ — **FFT part**
2. **Solve** $B_k \tilde{z}_k = \tilde{y}_k$ **for** $\tilde{z}_k \quad (k = 0, 1, \dots, n_t - 1)$ — **Complex solver part**
3. **Compute** $\mathbf{z} = \left[(D_\epsilon^{-1} \mathcal{F}_{n_t}^*) \otimes I_{n_x}\right] \tilde{\mathbf{z}}$ — **FFT part**

Implementation of FFT parts (1/4)

Three-step procedure of BEC precond. $\mathbf{z} = \mathcal{P}_{\mathbf{BEC}}^{-1}\mathbf{y}$

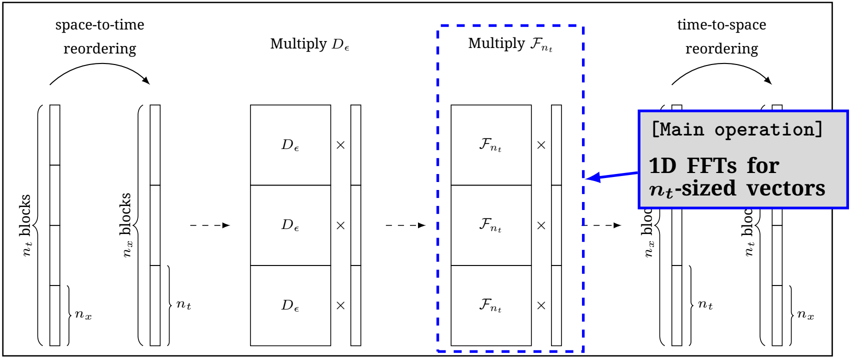
1. **Compute** $\tilde{\mathbf{y}} = [(\mathcal{F}_{n_t} D_\epsilon) \otimes I_{n_x}]\,\mathbf{y}$ **FFT part**
2. **Solve** $B_k \tilde{z}_k = \tilde{y}_k$ **for** $\tilde{z}_k$ $(k = 0, 1, \ldots, n_t - 1)$ **Complex solver part**
3. **Compute** $\mathbf{z} = \left[(D_\epsilon^{-1} \mathcal{F}_{n_t}^{*}) \otimes I_{n_x}\right] \tilde{\mathbf{z}}$ **FFT part**



Implementation of FFT parts (1/4)

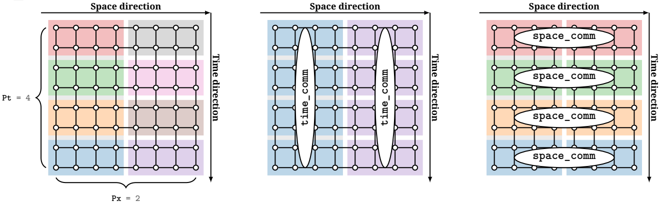
Three-step procedure of BEC precond. $\mathbf{z} = \mathcal{P}_{\mathbf{BEC}}^{-1}\mathbf{y}$

1. **Compute** $\tilde{\mathbf{y}} = [(\mathcal{F}_{n_t} D_\epsilon) \otimes I_{n_x}]\,\mathbf{y}$ **FFT part**
2. **Solve** $B_k \tilde{z}_k = \tilde{y}_k$ **for** $\tilde{z}_k \quad (k = 0, 1, \ldots, n_t - 1)$ **Complex solver part**
3. **Compute** $\mathbf{z} = \left[(D_\epsilon^{-1}\mathcal{F}_{n_t}^*) \otimes I_{n_x}\right]\tilde{\mathbf{z}}$ **FFT part**

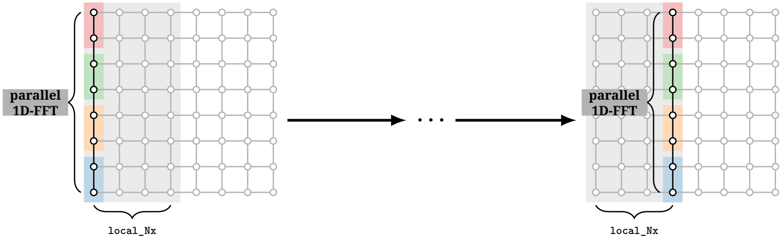


Implementation of FFT parts (2/4)

- **Straightforward parallel 1D-FFTs**
  - **Space-time parallel data distribution**
  - **Space and time communicators**

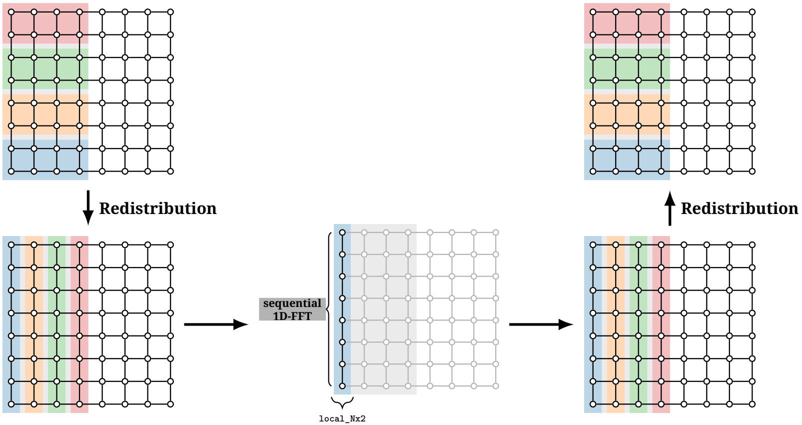


- **Parallel 1D-FFTs with `local_Nx`: `FFTW` [Frigo, 2005] / `FFTE` [Takahashi, 2001]**
  - `all-to-all` **communication on** `time_comm` **for each parallel 1D-FFT**



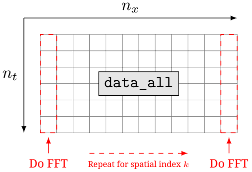
Implementation of FFT parts (3/4)

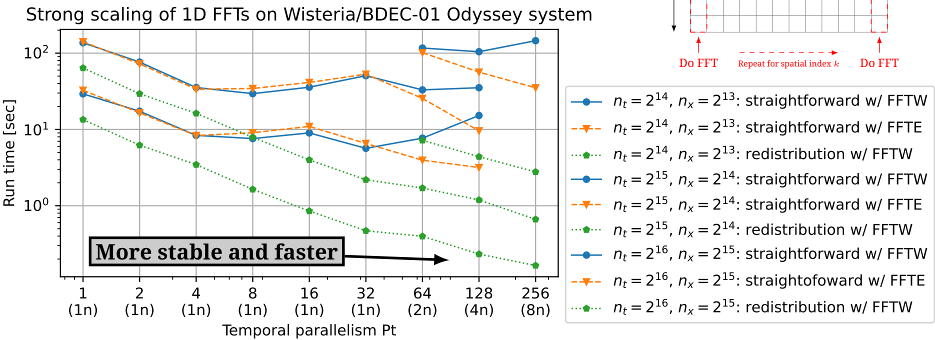
- **Redistributed sequential FFTs**
  - **Reassign temporal parallelism** $(\frac{N_t}{P_t}, \frac{N_x}{P_x}) \to (N_t, \frac{N_x}{P_x P_t})$
  - `all-to-all` **communication at once on** `time_comm`



Implementation of FFT parts (4/4)

- **Parallel tests for the (n_t, n_x) data array**
  - **Same operation in the precond. step**





- **Straightforward parallel 1D-FFTs with `FFTW`**
- **Straightforward parallel 1D-FFTs with `FFTE`**
- **Redistributed sequential FFTs with `FFTW` (used in this work)**

Implementation of complex solver parts (1/1)

Three-step procedure of BEC precond. $\mathbf{z} = \mathcal{P}_{\mathbf{BEC}}^{-1}\mathbf{y}$

1. **Compute** $\tilde{\mathbf{y}} = [(\mathcal{F}_{n_t} D_\epsilon) \otimes I_{n_x}]\,\mathbf{y}$ **FFT part**
2. **Solve** $B_k \tilde{z}_k = \tilde{y}_k$ **for** $\tilde{z}_k \quad (k = 0, 1, \ldots, n_t - 1)$ **Complex solver part**
3. **Compute** $\mathbf{z} = \left[(D_\epsilon^{-1} \mathcal{F}_{n_t}^*) \otimes I_{n_x}\right] \tilde{\mathbf{z}}$ **FFT part**

- **Spatial-sized complex system with** $B_k = \lambda_k^{(\epsilon)} M + \Delta t K$
  - **Solves complex-valued systems (future work)**
  - **Solves equivalent 2×2 real-valued systems** [Day and Heroux *et al.*, 2001]

Complex-valued system → **Split** → **Equivalent real-valued system**

$$B_k \tilde{z}_k = \tilde{y}_k$$

$$B_k = B_k^{(R)} + iB_k^{(I)}$$
$$\tilde{z}_k = \tilde{z}_k^{(R)} + i\tilde{z}_k^{(I)}$$
$$\tilde{y}_k = \tilde{y}_k^{(R)} + i\tilde{y}_k^{(I)}$$

$$\begin{bmatrix} +B_k^{(R)} & -B_k^{(I)} \\ +B_k^{(I)} & +B_k^{(R)} \end{bmatrix} \begin{bmatrix} \tilde{z}_k^{(R)} \\ \tilde{z}_k^{(I)} \end{bmatrix} = \begin{bmatrix} \tilde{y}_k^{(R)} \\ \tilde{y}_k^{(I)} \end{bmatrix}$$

▷ **SA-AMG preconditioned GMRES solver w/** `ML`, `AztecOO` **in** `Trilinos`.

BEC-GMRES: BEC preconditioned GMRES

GMRES for $\mathcal{A}\mathbf{u} = \mathbf{f}$

Preconditioning $\mathbf{z}_j = \mathcal{P}_{\mathbf{BEC}}^{-1}\mathbf{v}_j$

(1) $\tilde{\mathbf{y}} = [(\mathcal{F}_{n_t} D_\epsilon) \otimes I_{n_x}]\,\mathbf{y}$

`Redistributed FFTs w/ FFTW`

(2) $B_k \tilde{z}_k = \tilde{y}_k$ **for** $\tilde{z}_k \qquad (k = 0, 1, \ldots, n_t - 1)$

`AMG-GMRES w/ ML and AztecOO in Trilinos`

(3) $\mathbf{z} = \left[(D_\epsilon^{-1} \mathcal{F}_{n_t}^*) \otimes I_{n_x}\right] \tilde{\mathbf{z}}$

`Redistributed inverse FFTs w/ FFTW`

SpMV for $\mathbf{v}_{j+1} = \mathcal{A}\mathbf{z}_j$

Orthogonalization for $\mathbf{v}_{j+1}$

Outline

Numerical experiments

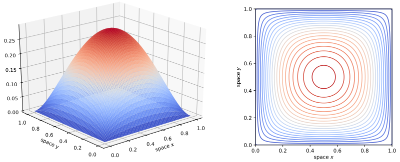
- **Aim: To investigate strong-scaling peformance**
- **Problem settings**
  - **These refer to Examples 1 and 3 in [Lin *et al.*, 2021]**
  - **Discretization matrices are generated with `IFISS` [Elman *et al.*, 2014]**

(1) 2D diffusion

$$u_t(x,y,t) = 10^{-5}\Delta u(x,y,t)$$

$$u(x,y,t) = 0 \quad \textbf{on} \quad \partial\Omega$$

$$u(x,y,0) = x(x-1)y(y-1)$$

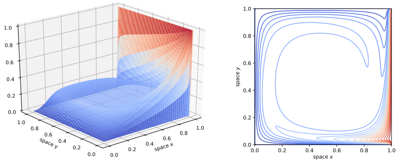


(2) 2D convection-diffusion

$$u_t(x,y,t) = \frac{1}{200}\Delta u - \overrightarrow{w}\cdot u$$

$$u(x,y,t) = (1-\exp(-10t))\phi(x,y) \ \textbf{on} \ \partial\Omega$$

$$u(x,y,0) = 0$$



Numerical experiments

- **Measurement environment**
  - Wisteria/BDEC-01 Odyssey system: Fujitsu A64FX cluster
  - Flat MPI mode: Fujitsu MPI and Compiler v4.9.0 with

    `-std=c++17 -Nclang -Ofast -mcpu=a64fx -march=armv8-a -fPIC`
  - `FFTW3 v3.3.9` and `Epetra`, `AztecOO`, `ML` in `Trilinos v14.5`

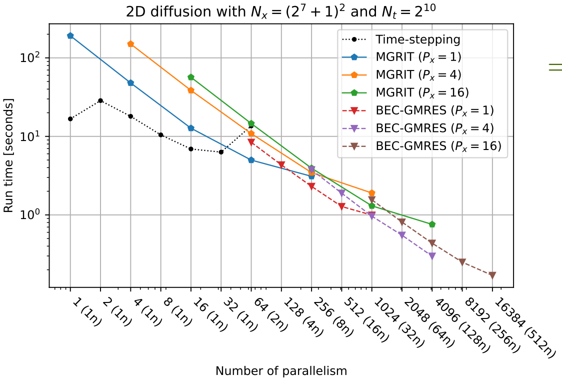
- **Solver settings**
  - ▷ Time-stepping: Sequential time-stepping method
  - ▷ MGRIT: multigrid reduction in time
    - A coarsening factor $m = 4$ and number of levels $L = 3$
  - ▷ BEC-GMRES: BEC preconditioned GMRES solver
    - A weighted parameter $\epsilon = \min(0.5, 0.5\Delta t)$

Parallel result: (1) diffusion

- **Problem settings:** $N_x \times N_t = (2^7+1)^2 \times 2^{10} \mid \Omega = [0,1]^2$ **and** $T = 1$

$\Rightarrow$ **Time-stepping stagnates at 32 parallelism.**

$\Rightarrow$ **MGRIT and BEC-GMRES achieve good scaling.**



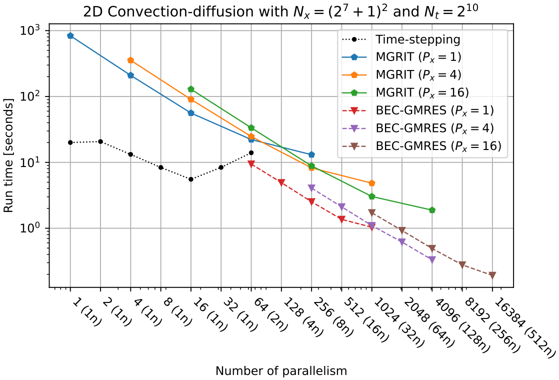
$\Rightarrow$ **BEC-GMRES is better than MGRIT in high parallelism, albeit slightly.**

Parallel result: (2) convection-diffusion

- **Problem settings:** $N_x \times N_t = (2^7+1)^2 \times 2^{10} \mid \Omega = [-1,1]^2$ **and** $T = 1$

$\Rightarrow$ **Increased time due to increased MGRIT iterations.**

$\Rightarrow$ **BEC-GMRES still has good convergence and scaling.**



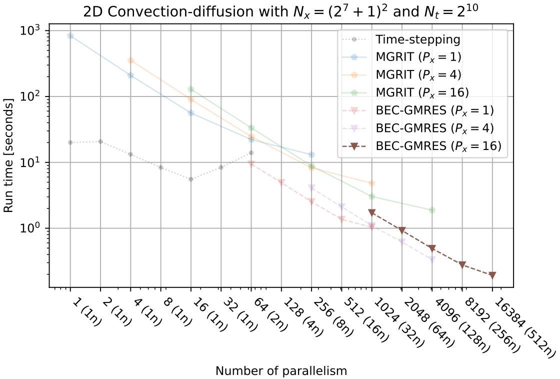
- For BEC-GMRES with $P_x = 16$
- Much of the time is spent on complex solver parts.

Parallel result: (2) convection-diffusion

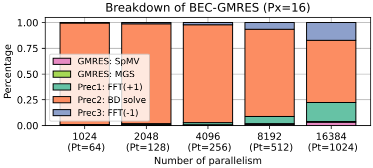
- **Problem settings:** $N_x \times N_t = (2^7+1)^2 \times 2^{10} \mid \Omega = [-1,1]^2$ **and** $T = 1$

$\Rightarrow$ **Increased time due to increased MGRIT iterations.**

$\Rightarrow$ **BEC-GMRES still has good convergence and scaling.**



- **For BEC-GMRES with $P_x = 16$**
- **Much of the time is spent on complex solver parts.**



Outline

Conlusion

- **BEC-GMRES achieves good temporal-parallelism scaling performance up to maximum temporal parallelism.**
  - **Redistributed sequential FFTs is stable and faster.**
  - **This solver may be a more promising solver than the multigrid-based methods, especially for hyperbolic problems.**
  - **The dominant runtime is for solving 2×2 real-valued equivalent equations.**

- **Future works**
  - **Purely complex solvers with `Tpetra,Belos,MueLu` stacks**
  - **Application to explicit time integration schemes**
  - **Towards more complicated problems / nonlinear problems**