University of Stuttgart
Germany

SAND2020-13743C

FROSch Preconditioners for Land Ice Simulations of Greenland and Antarctica

Alexander Heinlein[1] Mauro Perego[2] Sivasankaran Rajamanickam[2]

26th International Domain Decomposition Conference, December 9, 2020

[1] *University of Stuttgart, University of Cologne*

[2] Sandia National Laboratories

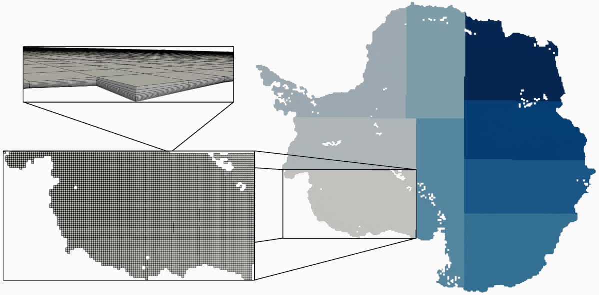
Land Ice Simulations of Greenland and Antarctica

Greenland and Antarctic ice sheets

- store most of the fresh water on earth and
- mass loss from these ice sheets significantly contributes to sea-level rise.

The simulation of temperature and velocity of the ice sheets gives rise to **large highly nonlinear systems of equations** with a **strong coupling** of the variables.

Taken from https://unsplash.com.



The simulations are also characterized by:

- The **mesh structure**:
  - Volume mesh is obtained by **extrusion** of the surface mesh $\Rightarrow$ **2D domain decomposition**.
  - **Highly anisotropic**.
- Specific combination of Dirichlet, Neumann, and Robin **boundary conditions**.

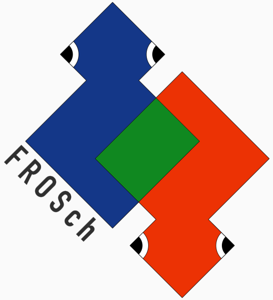
Domain Decomposition Methods in Trilinos

By Sandia National Laboratories

- Teko: **Block preconditioners** for multi-physics problems
- Ifpack/Ifpack2: **One-level overlapping Schwarz preconditioners**
  $\rightarrow$ **Algebraic** but **not scalable**
- ShyLU/BDDC: **BDDC** (Balancing Domain Decomposition by Constraints) **preconditioner**
  $\rightarrow$ **Scalable** but **less algebraic**

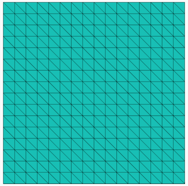
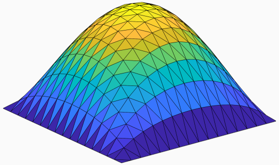
FROSch (**F**ast and **R**obust **O**verlapping **Sch**warz)

- **Schwarz preconditioners** with **algebraic coarse spaces** based on extension operators, e.g., **GDSW** (Generalized–Dryja–Smith–Widlund) coarse spaces
  $\rightarrow$ **Algebraic** and **scalable**
- Part of the package ShyLU:
  (Joint work with the Scalable Algorithms group of the **Sandia National Laboratories (SNL)**, Albuquerque, USA)
- Implementation based on Xpetra
  $\rightarrow$ Can be used with Epetra and Tpetra (linear algebra packages)
  Extension to current architectures, e.g., **GPUs**, *using the* Kokkos *programming model*



Easy access to FROSch through unified Trilinos solver interface Thyra.

Model Problem & Domain Decomposition

Consider a **Poisson model problem** on $[0,1]^2$:

$$
\begin{aligned}
-\Delta u &= f \quad \text{in } \Omega, \\
u &= 0 \quad \text{on } \partial\Omega.
\end{aligned}
$$

Discretize (e.g., using finite elements)

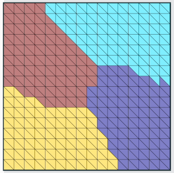
$$Kx = b.$$

$\Rightarrow$ Construct a **parallel scalable preconditioner** M^{-1} using **overlapping Schwarz domain decomposition methods**.

Overlapping domain decomposition

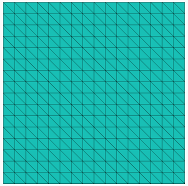
Overlapping Schwarz methods are based on **overlapping decompositions** of the computational domain Ω.

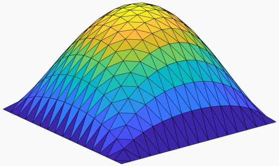
Overlapping subdomains $\Omega'_1, \dots, \Omega'_N$ can be constructed by **recursively adding layers of elements** to nonoverlapping subdomains $\Omega_1, \dots, \Omega_N$.



Nonoverlap. DD

Model Problem & Domain Decomposition





Consider a **Poisson model problem** on $[0,1]^2$:

$$-\Delta u = f \quad \text{in } \Omega,$$
$$u = 0 \quad \text{on } \partial\Omega.$$

Discretize (e.g., using finite elements)

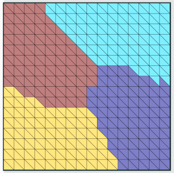
$$Kx = b.$$

$\Rightarrow$ Construct a **parallel scalable preconditioner** M^{-1} using **overlapping Schwarz domain decomposition methods**.

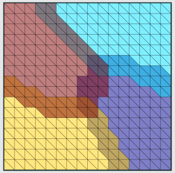
Overlapping domain decomposition

Overlapping Schwarz methods are based on **overlapping decompositions** of the computational domain Ω.

Overlapping subdomains $\Omega_1', \ldots, \Omega_N'$ can be constructed by **recursively adding layers of elements** to nonoverlapping subdomains $\Omega_1, \ldots, \Omega_N$.

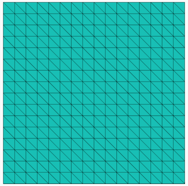
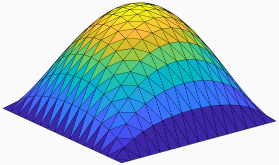


Nonoverlap. DD



Overlap $\delta = 1h$

Model Problem & Domain Decomposition

Consider a **Poisson model problem** on $[0,1]^2$:

$$
\begin{aligned}
-\Delta u &= f \quad \text{in } \Omega, \\
u &= 0 \quad \text{on } \partial\Omega.
\end{aligned}
$$

Discretize (e.g., using finite elements)

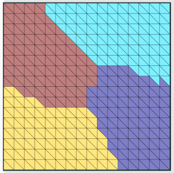
$$Kx = b.$$

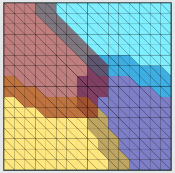
$\Rightarrow$ Construct a **parallel scalable preconditioner** M^{-1} using **overlapping Schwarz domain decomposition methods**.

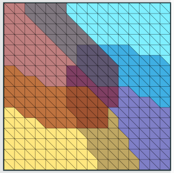
Overlapping domain decomposition

Overlapping Schwarz methods are based on **overlapping decompositions** of the computational domain Ω.

Overlapping subdomains $\Omega'_1, \ldots, \Omega'_N$ can be constructed by **recursively adding layers of elements** to nonoverlapping subdomains $\Omega_1, \ldots, \Omega_N$.

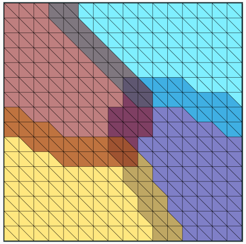

Nonoverlap. DD


Overlap $\delta = 1h$

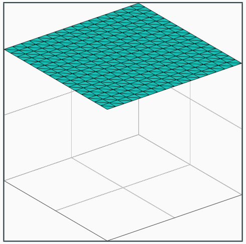

Overlap $\delta = 2h$

One-Level Schwarz Preconditioners

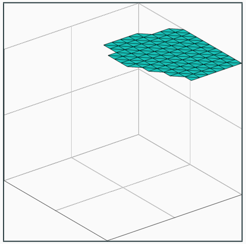
Overlapping domain decomposition



Function on Ω



Restriction to Ω_i'



Based on an **overlapping domain decomposition**, we define **local restriction operators** $R_i : V^h(\Omega) \rightarrow V_i := V^h(\Omega_i')$, for $i = 1, \ldots, N$, and obtain the **additive one-level Schwarz preconditioner**

$$M_{\mathrm{OS}-1}^{-1} = \underbrace{\sum\nolimits_{i=1}^{N} R_i^T K_i^{-1} R_i}_{\text{local}},$$

where $K_i := R_i K R_i^T$.

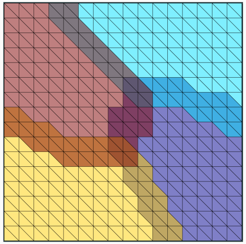
Condition number estimate:

$$\kappa \left(M_{\mathrm{OS}-1}^{-1} K \right) \leq C \left(1 + \frac{1}{H\delta} \right)$$

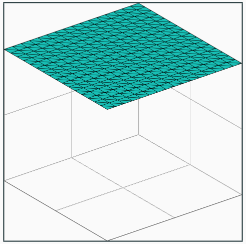
with the typical subdomain size H and the width of the overlap δ.

One-Level Schwarz Preconditioners

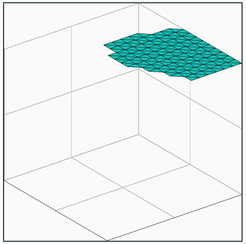
Overlapping domain decomposition



Function on Ω



Restriction to Ω_i'



Based on an **overlapping domain decomposition**, we define **local restriction operators** $R_i : V^h(\Omega) \rightarrow V_i := V^h(\Omega_i')$, for $i = 1, \ldots, N$, and obtain the **additive one-level Schwarz preconditioner**

$$M_{\mathrm{OS-1}}^{-1} = \underbrace{\sum\nolimits_{i=1}^{N} R_i^T K_i^{-1} R_i,}_{\text{local}}$$

where $K_i := R_i K R_i^T$.

Condition number estimate:

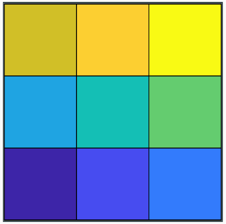
$$\kappa \left(M_{\mathrm{OS-1}}^{-1} K \right) \leq C \left(1 + \frac{1}{H\delta} \right)$$

with the typical subdomain size H and the width of the overlap δ.

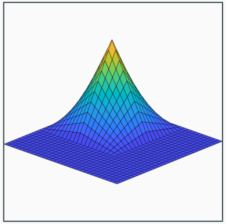
$\rightarrow$ **Algebraic! but not scalable...**

Two-Level Schwarz Preconditioners – Lagrangian Coarse Space

Coarse triangulation



Nodal bilinear basis function



The **additive two-level Schwarz preconditioner** reads

$$M_{\mathrm{OS}-2}^{-1} = \underbrace{\Phi K_0^{-1} \Phi^T}_{\text{coarse level – global}} + \underbrace{\sum_{i=1}^{N} R_i^T K_i^{-1} R_i}_{\text{first level – local}},$$

where Φ contains the coarse basis functions and $K_0 := \Phi^T K \Phi$; cf., e.g., **Toselli, Widlund (2005)**.

In the **classical Lagrangian coarse space**, the coarse basis functions are a **nodal finite element basis on the coarse triangulation**. Their construction **relies on geometric information and cannot be performed algebraically**.

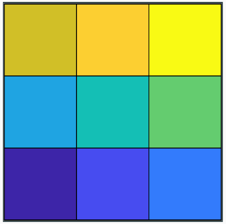
The **condition number of the two-level Schwarz operator with classical Lagrangian coarse space** is bounded by

$$\kappa \left(M_{\mathrm{OS}-2}^{-1} K \right) \leq C \left(1 + \frac{H}{\delta} \right);$$

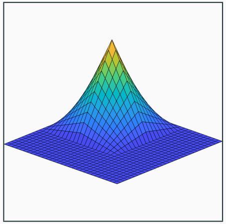
cf., e.g., **Toselli, Widlund (2005)**. The constant C is **independent of h, δ, and H**.

Two-Level Schwarz Preconditioners – Lagrangian Coarse Space

Coarse triangulation



Nodal bilinear basis function



The **additive two-level Schwarz preconditioner** reads

$$M_{\mathrm{OS}-2}^{-1} = \underbrace{\Phi K_0^{-1} \Phi^T}_{\text{coarse level – global}} + \underbrace{\sum\nolimits_{i=1}^{N} R_i^T K_i^{-1} R_i}_{\text{first level – local}},$$

where Φ contains the coarse basis functions and $K_0 := \Phi^T K \Phi$; cf., e.g., **Toselli, Widlund (2005)**.

In the **classical Lagrangian coarse space**, the coarse basis functions are a **nodal finite element basis on the coarse triangulation**. Their construction **relies on geometric information and cannot be performed algebraically**.

The **condition number of the two-level Schwarz operator with classical Lagrangian coarse space** is bounded by

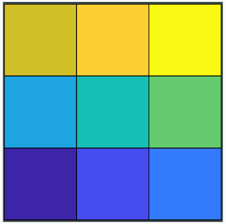
$$\kappa\left(M_{\mathrm{OS}-2}^{-1} K\right) \leq C \left(1 + \frac{H}{\delta}\right);$$

cf., e.g., **Toselli, Widlund (2005)**. The constant C is **independent of h, δ, and H**.

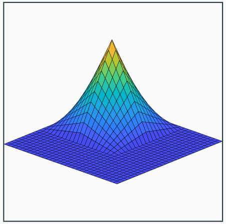
$\rightarrow$ **Scalable! But not algebraic...**

Two-Level Schwarz Preconditioners – Lagrangian Coarse Space

Coarse triangulation



Nodal bilinear basis function



The **additive two-level Schwarz preconditioner** reads

$$M_{\mathrm{OS}-2}^{-1} = \underbrace{\Phi K_0^{-1} \Phi^T}_{\text{coarse level – global}} + \underbrace{\sum\nolimits_{i=1}^{N} R_i^T K_i^{-1} R_i}_{\text{first level – local}},$$

where Φ contains the coarse basis functions and $K_0 := \Phi^T K \Phi$; cf., e.g., **Toselli, Widlund (2005)**.

In the **classical Lagrangian coarse space**, the coarse basis functions are a **nodal finite element basis on the coarse triangulation**. Their construction **relies on geometric information and cannot be performed algebraically**.

The **condition number of the two-level Schwarz operator with classical Lagrangian coarse space** is bounded by

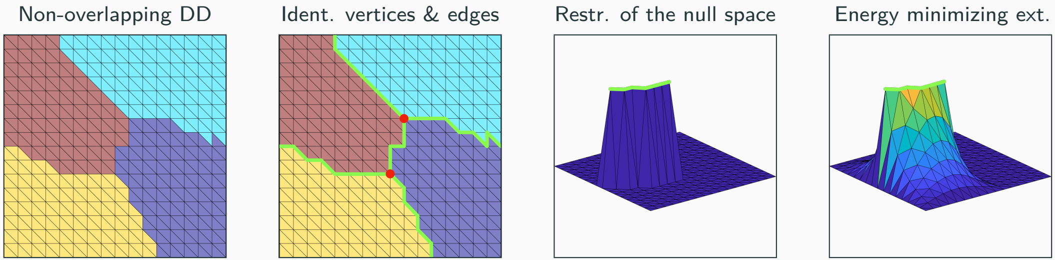
$$\kappa \left(M_{\mathrm{OS}-2}^{-1} K \right) \leq C \left(1 + \frac{H}{\delta} \right);$$

cf., e.g., **Toselli, Widlund (2005)**. The constant C is **independent of** h**,** δ**, and** H.

$\rightarrow$ **Scalable! But not algebraic...**

Can we construct a coarse space **algebraically**? $\rightarrow$ **GDSW coarse spaces**

Two-Level Schwarz Preconditioners – GDSW Coarse Space



In **GDSW (Generalized–Dryja–Smith–Widlund) coarse spaces**, the coarse basis functions are chosen as **energy minimizing extensions** of functions Φ_Γ that are defined on the interface Γ:

$$\Phi = \begin{bmatrix} -K_{II}^{-1} K_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix} = \begin{bmatrix} \Phi_I \\ \Phi_\Gamma \end{bmatrix}$$

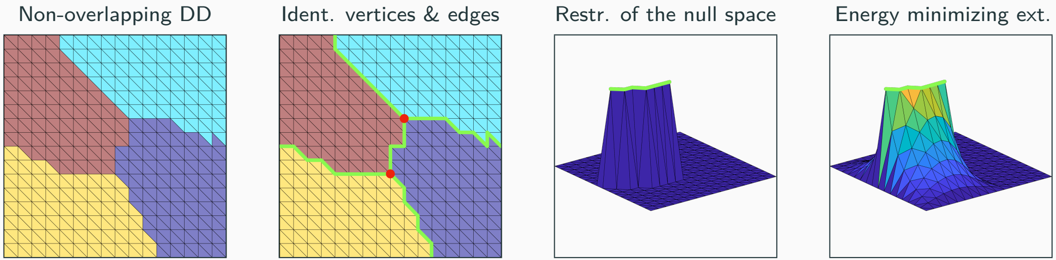
The functions Φ_Γ are **restrictions of the null space of global Neumann matrix** to the **edges, vertices, and, in 3D, faces (partition of unity)** of the non-overlapping decomposition.

The **condition number of the GDSW operator** is bounded by

$$\kappa\left(M_{\mathrm{GDSW}}^{-1} K\right) \leq C \left(1 + \frac{H}{\delta}\right) \left(1 + \log\left(\frac{H}{h}\right)\right)^2;$$

cf. **Dohrmann, Klawonn, Widlund (2008)**, **Dohrmann, Widlund (2009, 2010, 2012)**.

Two-Level Schwarz Preconditioners – GDSW Coarse Space



In **GDSW (Generalized–Dryja–Smith–Widlund) coarse spaces**, the coarse basis functions are chosen as **energy minimizing extensions** of functions Φ_Γ that are defined on the interface Γ:

$$\Phi = \begin{bmatrix} -K_{II}^{-1} K_{\Gamma I}^{T} \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix} = \begin{bmatrix} \Phi_I \\ \Phi_\Gamma \end{bmatrix}$$

The functions Φ_Γ are **restrictions of the null space of global Neumann matrix** to the **edges, vertices, and, in 3D, faces (partition of unity)** of the non-overlapping decomposition.

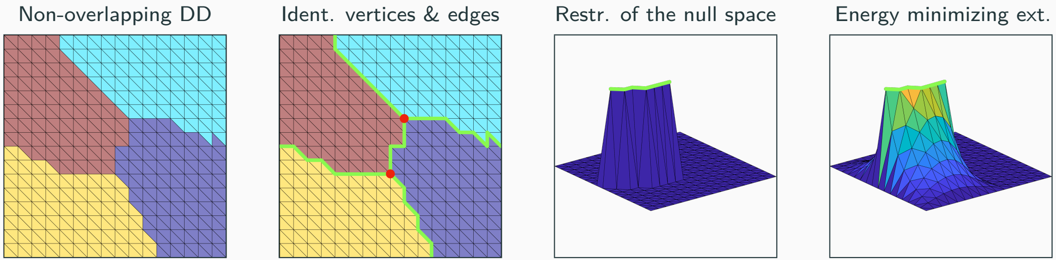
The **condition number of the GDSW operator** is bounded by

$$\kappa\left(M_{\mathrm{GDSW}}^{-1} K\right) \leq C \left(1 + \frac{H}{\delta}\right) \left(1 + \log\left(\frac{H}{h}\right)\right)^2;$$

cf. **Dohrmann, Klawonn, Widlund (2008)**, **Dohrmann, Widlund (2009, 2010, 2012)**.

$\rightarrow$ We only obtain the exponent 2 for very irregular subdomains.

Two-Level Schwarz Preconditioners – GDSW Coarse Space



In **GDSW (Generalized–Dryja–Smith–Widlund) coarse spaces**, the coarse basis functions are chosen as **energy minimizing extensions** of functions Φ_Γ that are defined on the interface Γ:

$$\Phi = \begin{bmatrix} -K_{II}^{-1} K_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix} = \begin{bmatrix} \Phi_I \\ \Phi_\Gamma \end{bmatrix}$$

The functions Φ_Γ are **restrictions of the null space of global Neumann matrix** to the **edges, vertices, and, in 3D, faces (partition of unity)** of the non-overlapping decomposition.

The **condition number of the GDSW operator** is bounded by

$$\kappa\left(M_{\mathrm{GDSW}}^{-1} K\right) \leq C \left(1 + \frac{H}{\delta}\right) \left(1 + \log\left(\frac{H}{h}\right)\right)^2;$$

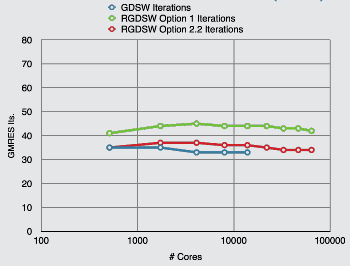
cf. **Dohrmann, Klawonn, Widlund (2008)**, **Dohrmann, Widlund (2009, 2010, 2012)**.

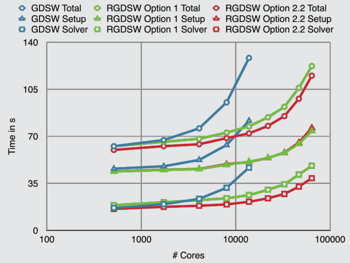
$\rightarrow$ **Scalable and algebraic!**

Weak Scalability up to $64\,\mathrm{k}$ MPI Ranks / $1.7\,\mathrm{b}$ Unknowns (3D Poisson; Juqueen)

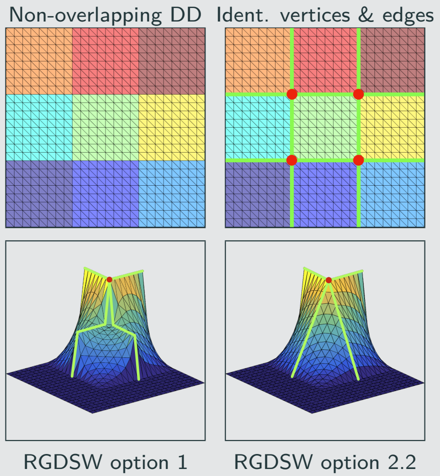
GDSW vs RGDSW (reduced dimension)

Heinlein, Klawonn, Rheinbach, Widlund (2019)





RGDSW (Reduced dimension GDSW)

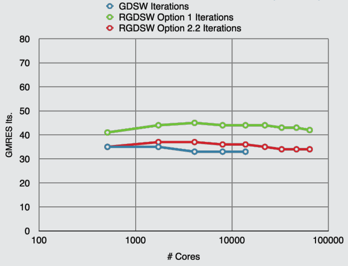


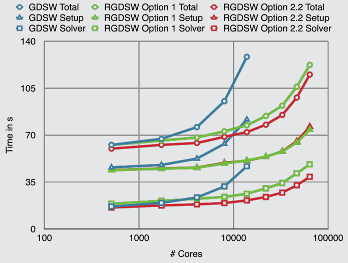
Reduced dimension GDSW coarse spaces are constructed from **nodal interface functions (different partition of unity)**; cf. Dohrmann, Widlund (2017).

Weak Scalability up to $64\,\mathrm{k}$ MPI Ranks / $1.7\,\mathrm{b}$ Unknowns (3D Poisson; Juqueen)

GDSW vs RGDSW (reduced dimension)

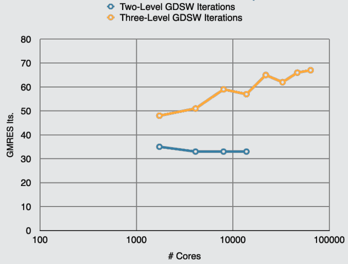
Heinlein, Klawonn, Rheinbach, Widlund (2019)

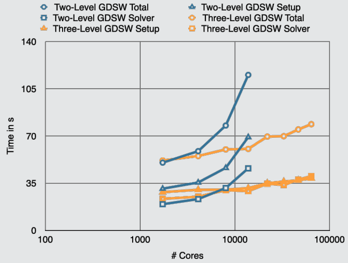




Two-level vs Three-level GDSW

Heinlein, Klawonn, Rheinbach, Röver (2019, 2020)





→ Talk by **F. Röver** in **MS11-01 (earlier today)**

Software Framework

https://github.com/trilinos/Trilinos

https://github.com/SNLComputation/Albany

Part of Trilinos.
$\rightarrow$ MKL Pardiso as the subdomain and coarse solver.

Software Framework

https://github.com/trilinos/Trilinos

https://github.com/SNLComputation/Albany

Part of `Trilinos`.
→ `MKL Pardiso` as the subdomain and coarse solver.

Hardware

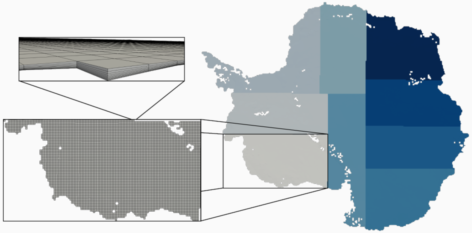
All simulations performed on Cori supercomputer (NERSC).

Velocity Problem

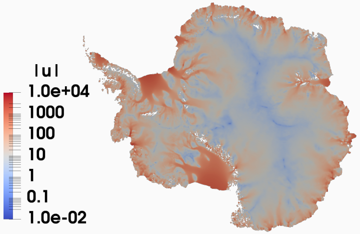
We use the so called **first-order (or Blatter-Pattyn) approximation of the Stokes equations**

$$\begin{cases} -\nabla \cdot (2\mu\, \dot{\epsilon}_1) &= -\rho_i\, |\boldsymbol{g}|\, \partial_x s, \\ -\nabla \cdot (2\mu\, \dot{\epsilon}_2) &= -\rho_i\, |\boldsymbol{g}|\, \partial_y s, \end{cases}$$

with the ρ_i the ice density, the ice surface elevation $s(x, y)$, the gravity acceleration $\boldsymbol{g}$, and strain rates $\dot{\epsilon}_1$ and $\dot{\epsilon}_2$; cf. **Blatter (1995)** and **Pattyn (2003)**.



Antarctica mesh & domain decomposition.



Velocity u solution

Nonlinear viscosity model

The **ice viscosity** μ is modeled using **Glen's law**

$$\mu = \frac{1}{2} A(T)^{-\frac{1}{n}}\, \dot{\epsilon}_e^{\frac{1-n}{n}},$$

where $A(T) = \alpha_1 e^{\alpha_2 T}$ is a temperature-dependent rate factor, $n = 3$ is the power-law exponent, and the effective strain rate $\dot{\epsilon}$.

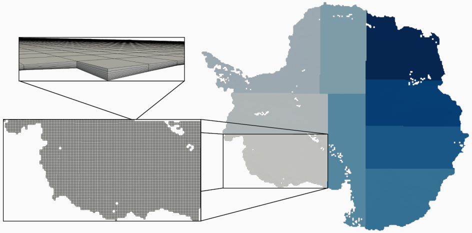
See **Perego, Gunzburger, Burkardt (2012)** and **Tezaur, Perego, Salinger, Tuminaro, Price (2015)** for more details.

Velocity Problem

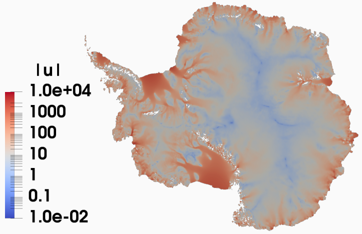
We use the so called **first-order (or Blatter-Pattyn) approximation of the Stokes equations**

$$\begin{cases} -\nabla \cdot (2\mu\, \dot{\epsilon}_1) &= -\rho_i\, |\boldsymbol{g}|\, \partial_x s, \\ -\nabla \cdot (2\mu\, \dot{\epsilon}_2) &= -\rho_i\, |\boldsymbol{g}|\, \partial_y s, \end{cases}$$

with the ρ_i the ice density, the ice surface elevation $s(x, y)$, the gravity acceleration $\boldsymbol{g}$, and strain rates $\dot{\epsilon}_1$ and $\dot{\epsilon}_2$; cf. **Blatter (1995)** and **Pattyn (2003)**.



Antarctica mesh & domain decomposition.



Velocity u solution

Boundary conditions

- *Upper surface*: $\dot{\epsilon}_j = 0$, $j = 1, 2$
  (**stress-free Neumann condition**)
- *Lower surface*: $2\mu_e \dot{\epsilon}_j \cdot \boldsymbol{n} + \beta u = 0$, $j = 1, 2$
  (**sliding Robin condition** with friction coefficient β)
- *Lateral boundary*: $2\mu\dot{\epsilon}_j \cdot \boldsymbol{n} = \frac{1}{2} gH \left(\rho_i - \rho_w r^2\right) n_1$, $j = 1, 2$
  (**open-ocean Neumann condition** with density of ocean water ρ_w and ratio of submerged ice thickness r)

See **Perego, Gunzburger, Burkardt (2012)** and **Tezaur, Perego, Salinger, Tuminaro, Price (2015)** for more details.

Antarctica Velocity Problem – Coarse Spaces

Without rotational coarse basis functions (2 rigid body modes)								
	GDSW				**RGDSW**			
MPI ranks	dim V_0	avg. its (nl its)	avg. setup	avg. solve	dim V_0	avg. its (nl its)	avg. setup	avg. solve
512	4 598	**40.8** (11)	15.36 s	**12.38 s**	1 834	42.6 (11)	**14.99 s**	12.50 s
1 024	9 306	**43.3** (11)	5.80 s	6.27 s	3 740	44.5 (11)	**5.65 s**	**6.08 s**
2 048	18 634	**41.7** (11)	3.27 s	2.91 s	7 586	42.7 (11)	**3.11 s**	**2.79 s**
4 096	37 184	**41.4** (11)	2.59 s	2.07 s	15 324	42.5 (11)	**1.07 s**	**1.54 s**
8 192	72 964	**39.5** (11)	1.51 s	1.84 s	30 620	42.0 (11)	**1.20 s**	**1.16 s**

With rotational coarse basis functions (3 rigid body modes)								
	GDSW				**RGDSW**			
MPI ranks	dim V_0	avg. its (nl its)	avg. setup	avg. solve	dim V_0	avg. its (nl its)	avg. setup	avg. solve
512	6 897	**35.5** (11)	15.77 s	**11.21 s**	2 751	40.7 (11)	**15.23 s**	12.22 s
1 024	13 959	**35.6** (11)	6.16 s	**5.78 s**	5 610	42.9 (11)	**5.65 s**	6.04 s
2 048	27 951	**33.5** (11)	3.78 s	3.45 s	11 379	42.2 (11)	**3.17 s**	**2.81 s**
4 096	55 776	**31.8** (11)	2.21 s	3.80 s	22 986	44.3 (11)	**1.95 s**	**2.70 s**
8 192	109 446	**29.3** (11)	2.49 s	5.33 s	45 930	40.8 (11)	**1.19 s**	**3.13 s**

Problem:	Velocity	**Mesh:**	Antarctica, 4 km hor. resolution 20 vert. layers	**Size:**	35.3 m degrees of freedom (P1 FE)

Antarctica Velocity Problem – Reuse

We employ different **reuse strategies** to **reduce the setup costs** of the two-level preconditioner

$$M_{\mathrm{OS}-2}^{-1} = \Phi \mathbf{K_0}^{-1} \Phi^T + \sum_{i=1}^{N} \mathbf{R_i}^T K_i^{-1} \mathbf{R_i}.$$

MPI ranks	restriction operators + symbolic fact. (1st level)			+ coarse basis + symbolic fact. (2nd level)			+ coarse matrix		
	avg. its (nl its)	avg. setup	avg. solve	avg. its (nl its)	avg. setup	avg. solve	avg. its (nl its)	avg. setup	avg. solve
512	**41.9** (11)	25.10 s	**12.29 s**	42.6 (11)	14.99 s	12.50 s	46.7 (11)	**14.94 s**	13.81 s
1 024	**43.3** (11)	9.18 s	**5.85 s**	44.5 (11)	**5.65 s**	6.08 s	49.2 (11)	5.75 s	6.78 s
2 048	**41.4** (11)	4.15 s	**2.63 s**	42.7 (11)	3.11 s	2.79 s	47.7 (11)	**2.92 s**	3.10 s
4 096	**41.2** (11)	1.66 s	**1.49 s**	42.5 (11)	1.07 s	1.54 s	48.9 (11)	**0.95 s**	1.75 s
8 192	**40.2** (11)	1.26 s	**1.06 s**	42.0 (11)	1.20 s	1.16 s	50.1 (11)	**0.63 s**	1.35 s

Problem: Velocity **Mesh:** Antarctica, 4 km hor. resolution 20 vert. layers **Size:** 35.3 m degrees of freedom (P1 FE) **Coarse space:** RGDSW

Antarctica Velocity Problem – OpenMP VS MPI Parallelization

We can make use of **OpenMP parallelization**:

- `Tpetra` linear algebra stack in `FROSch` and `Albany` $\Rightarrow$ **OpenMP parallelization of the linear algebra operations**.
- **OpenMP parallelization** of the **subdomain and coarse solver** `Pardiso MKL` used in `FROSch`.

	OpenMP parallelization (512 MPI ranks)				MPI parallelization			
cores	OpenMP threads	avg. its (nl its)	avg. setup	avg. solve	MPI ranks	avg. its (nl its)	avg. setup	avg. its solve
512	1	**42.6** (11)	**14.99 s**	**12.50 s**	512	**42.6** (11)	**14.99 s**	**12.50 s**
1 024	2	**42.6** (11)	9.43 s	6.80 s	1 024	44.5 (11)	**5.65 s**	**6.08 s**
2 048	4	**42.6** (11)	5.50 s	4.02 s	2 048	42.7 (11)	**3.11 s**	**2.79 s**
4 096	8	42.6 (11)	3.65 s	2.71 s	4 096	**42.5** (11)	**1.07 s**	**1.54 s**
8 192	16	42.6 (11)	2.56 s	2.32 s	8 192	**42.0** (11)	**1.20 s**	**1.16 s**

Problem:	Velocity	**Mesh:**	Antarctica, 4 km hor. resolution 20 vert. layers	**Size:**	35.3 m degrees of freedom (P1 FE)	**Coarse space:**	RGDSW

Antarctica Velocity Problem – OpenMP VS MPI Parallelization

We can make use of **OpenMP parallelization**:

- Tpetra linear algebra stack in FROSch and Albany ⇒ **OpenMP parallelization of the linear algebra operations**.
- **OpenMP parallelization** of the **subdomain and coarse solver** Pardiso MKL used in FROSch.

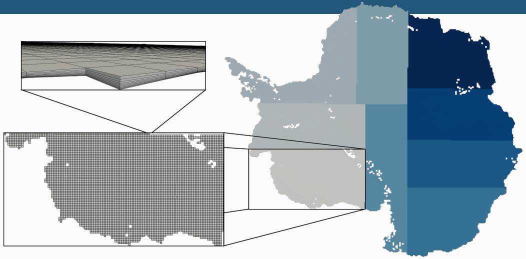
	OpenMP parallelization (512 MPI ranks)				MPI parallelization			
cores	OpenMP threads	avg. its (nl its)	avg. setup	avg. solve	MPI ranks	avg. its (nl its)	avg. setup	avg. its solve
512	1	**42.6** (11)	**14.99 s**	**12.50 s**	512	**42.6** (11)	**14.99 s**	**12.50 s**
1 024	2	**42.6** (11)	9.43 s	6.80 s	1 024	44.5 (11)	**5.65 s**	**6.08 s**
2 048	4	**42.6** (11)	5.50 s	4.02 s	2 048	42.7 (11)	**3.11 s**	**2.79 s**
4 096	8	42.6 (11)	3.65 s	2.71 s	4 096	**42.5** (11)	**1.07 s**	**1.54 s**
8 192	16	42.6 (11)	2.56 s	2.32 s	8 192	**42.0** (11)	**1.20 s**	**1.16 s**

Problem:	Velocity	**Mesh:**	Antarctica, 4 km hor. resolution 20 vert. layers	**Size:**	35.3 m degrees of freedom (P1 FE)	**Coarse space:**	RGDSW

→ **MPI parallelization** is more efficient than **OpenMP parallelization**. However, **for large numbers of MPI ranks** and a **large dimension of the coarse problem**, **OpenMP parallelization may be useful**.

Antarctica Velocity Problem – Weak Scalability

- Weak scalability study for in increasing horizontal mesh resolution.
  - **1 OpenMP thread:** From 32 to 8 192 processor cores
  - **4 OpenMP threads:** From 128 to 32 768 processor cores
- The number of vertical layers is fixed to 20.
- P1 FEM spatial discretization.


Antarctica mesh & domain decomposition.

MPI ranks	mesh	# dofs	1 OpenMP thread avg. its (nl its)	avg. setup	avg. solve	4 OpenMP threads avg. its (nl its)	avg. setup	avg. solve
32	16 km	2.2 m	24.1 (11)	11.97 s	9.47 s	23.5 (11)	4.15 s	3.25 s
128	8 km	8.8 m	32.0 (10)	14.08 s	8.71 s	32.0 (10)	4.97 s	2.85 s
512	4 km	35.3 m	42.6 (11)	14.99 s	12.50 s	42.6 (11)	5.50 s	4.02 s
2 048	2 km	141.5 m	61.0 (11)	22.83 s	19.76 s	61.0 (11)	7.36 s	6.55 s
8 192	1 km	566.1 m	67.1 (14)	17.36 s	22.91 s	67.1 (14)	6.20 s	7.39 s

Problem: Velocity **Mesh:** Antarctica, 20 vert. layers **Coarse space:** RGDSW (P1 FE)

Temperature Problem

The **steady state enthalpy equation** reads

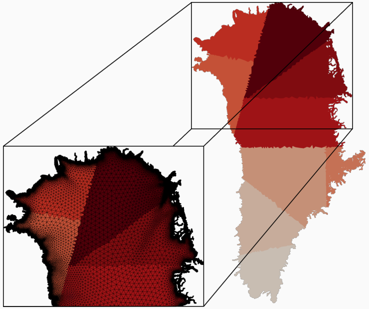
$$\nabla \cdot \boldsymbol{q}(h) + \boldsymbol{u} \cdot \nabla h = 4\mu\, \epsilon_e^2$$

with the enthalpy growing linearly with the water content ϕ

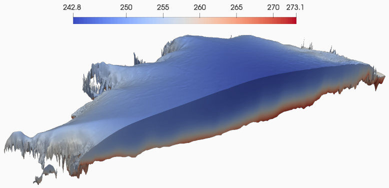
$$h = \begin{cases} \rho_i c\,(T - T_0), & \text{for cold ice } (h \leq h_m), \\ h_m + \rho_w L\,\phi, & \text{for temperate ice.} \end{cases}$$

the **melting enthalpy** $h_m := \rho_w c(T_m - T_0)$, the **uniform reference temperature** T_0, and the **enthalpy flux**

$$\boldsymbol{q}(h) = \begin{cases} \frac{k}{\rho_i c_i}\nabla h, & \text{for cold ice } (h \leq h_m), \\ \frac{k}{\rho_i c_i}\nabla h_m + \rho_w L\boldsymbol{j}(h), & \text{for temperate ice.} \end{cases}$$



Greenland mesh & domain decomposition.



Temperature T solution

Water flux term

The **water flux term**

$$\boldsymbol{j}(h) := \frac{1}{\eta_w}(\rho_w - \rho_i)k_0\phi^\gamma \boldsymbol{g}$$

describes the percolation of water driven by gravity; cf. **Schoof and Hewitt (2016, 2017)**.

See **Perego et al. (in preparation)** and **Heinlein et. al (in preparation)** for more details.

Temperature Problem

The **steady state enthalpy equation** reads

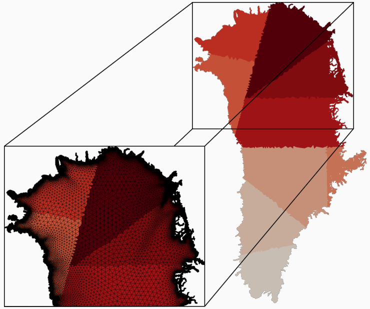
$$\nabla \cdot \boldsymbol{q}(h) + \boldsymbol{u} \cdot \nabla h = 4\mu\,\epsilon_e^2$$

with the enthalpy growing linearly with the water content ϕ

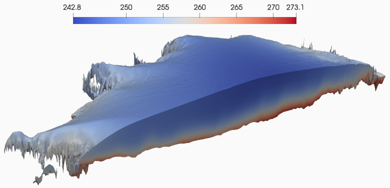
$$h = \begin{cases} \rho_i c\,(T - T_0), & \text{for cold ice } (h \leq h_m), \\ h_m + \rho_w L\,\phi, & \text{for temperate ice.} \end{cases}$$

the **melting enthalpy** $h_m := \rho_w c(T_m - T_0)$, the **uniform reference temperature** T_0, and the **enthalpy flux**

$$\boldsymbol{q}(h) = \begin{cases} \frac{k}{\rho_i c_i}\nabla h, & \text{for cold ice } (h \leq h_m), \\ \frac{k}{\rho_i c_i}\nabla h_m + \rho_w L \boldsymbol{j}(h), & \text{for temperate ice.} \end{cases}$$



Greenland mesh & domain decomposition.



Temperature T solution

Boundary conditions

- *Upper surface*: $h = \rho_i c(T_s - T_0)$
  (**Dirichlet boundary condition**)
- *Bed*: $m = G + \beta\sqrt{u^2 + v^2} - k\nabla T \cdot \boldsymbol{b}$,
  $m\,(T - T_m) = 0,\ T_m \leq 0.$
  (**Stefan boundary condition**)

See **Perego et al. (in preparation)** and **Heinlein et. al (in preparation)** for more details.

Greenland Temperature Problem – One-Level Schwarz VS Two-Level Schwarz

One-level Schwarz

MPI ranks	one layer of algebraic overlap: avg. its	avg. setup	avg. solve	two layers of algebraic overlap: avg. its	avg. setup	avg. solve
512	18.1 (11)	**0.42 s**	**0.35 s**	**17.1** (11)	0.51 s	0.40 s
1 024	23.7 (11)	**0.25 s**	**0.25 s**	**22.1** (11)	0.27 s	0.27 s
2 048	29.6 (11)	**0.16 s**	**0.17 s**	**27.6** (11)	0.23 s	0.20 s
4 096	39.8 (11)	**0.15 s**	**0.15 s**	**35.6** (11)	0.17 s	0.17 s

RGDSW

MPI ranks	one layer of algebraic overlap: avg. avg. its	avg. setup	avg. solve	two layers of algebraic overlap: avg. avg. its	avg. setup	avg. solve
512	19.5 (11)	**0.44 s**	**0.41 s**	**18.7** (11)	0.55 s	0.46 s
1 024	25.2 (11)	**0.28 s**	**0.29 s**	**23.9** (11)	0.35 s	0.33 s
2 048	31.5 (11)	0.26 s	**0.24 s**	**29.5** (11)	**0.25 s**	0.27 s
4 096	42.2 (11)	**0.25 s**	**0.27 s**	**38.2** (11)	**0.25 s**	0.29 s

Problem: Temperature **Mesh:** Greenland, 1-10 km hor. resolution 20 vert. layers **Size:** 1.9 m degrees of freedom (P1 FE)

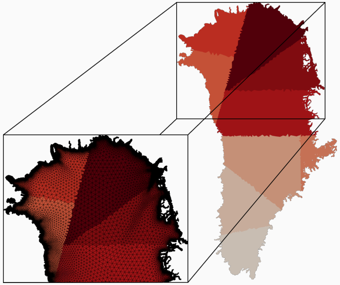
Coupled Problem

Couple the velocity and temperature problems. Therefore, compute the vertical velocity w using the incompressibility condition

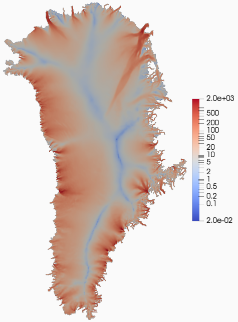
$$\partial_x u + \partial_y v + \partial_z w = 0,$$

with the **Dirichlet boundary condition at the ice lower surface**

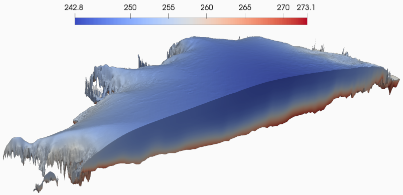
$$\boldsymbol{u} \cdot \boldsymbol{n} = \frac{m}{L(\rho_i - \rho_w \phi)}.$$



Greenland mesh & domain decomposition.



Velocity u solution



Temperature T solution

Then, the **tangent matrix** of the coupled problem has the structure

$$\begin{bmatrix} A_u & C_{uT} \\ C_{Tu} & A_T \end{bmatrix} \begin{bmatrix} x_u \\ x_T \end{bmatrix} = \begin{bmatrix} \tilde{r}_u \\ \tilde{r}_T \end{bmatrix}.$$

See **Perego et al. (in preparation)** and **Heinlein, Perego, Rajamanickam (in preparation)** for more details.

Monolithic (R)GDSW Preconditioners for CFD Simulations

Monolithic GDSW preconditioner

Consider the discrete saddle point problem

$$\mathcal{A}x = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} = b.$$

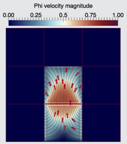
We construct a **monolithic GDSW preconditioner**

$$\mathcal{M}_{\mathrm{GDSW}}^{-1} = \phi \mathcal{A}_0^{-1} \phi^T + \sum_{i=1}^{N} \mathcal{R}_i^T \mathcal{A}_i^{-1} \mathcal{R}_i,$$

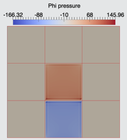
with block matrices $\mathcal{A}_0 = \phi^T \mathcal{A} \phi$, $\mathcal{A}_i = \mathcal{R}_i \mathcal{A} \mathcal{R}_i^T$, and

$$\mathcal{R}_i = \begin{bmatrix} \mathcal{R}_{u,i} & 0 \\ 0 & \mathcal{R}_{p,i} \end{bmatrix} \quad \text{and} \quad \phi = \begin{bmatrix} \Phi_{u,u_0} & \Phi_{u,p_0} \\ \Phi_{p,u_0} & \Phi_{p,p_0} \end{bmatrix}.$$

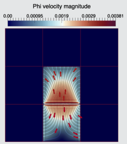
Using $\mathcal{A}$ to compute extensions: $\phi_I = -\mathcal{A}_{II}^{-1} \mathcal{A}_{I\Gamma} \phi_\Gamma$; cf. **Heinlein, Hochmuth, Klawonn (2019, 2020)**.



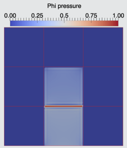
Φ_{u,u_0}



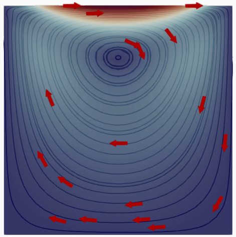
Φ_{p,u_0}

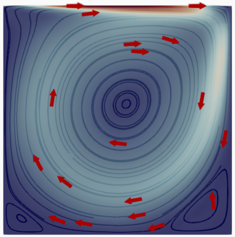


Φ_{u,p_0}



Φ_{p,p_0}


Stokes flow


Navier–Stokes flow

Related work:

- Original work on monolithic Schwarz preconditioners: **Klawonn and Pavarino (1998, 2000)**
- Other publications on monolithic Schwarz preconditioners: e.g., **Hwang and Cai (2006)**, **Barker and Cai (2010)**, **Wu and Cai (2014)**, and the presentation **Dohrmann (2010)** at the *Workshop on Adaptive Finite Elements and Domain Decomposition Methods* in Milan.

Monolithic (R)GDSW Preconditioners for CFD Simulations

Monolithic GDSW preconditioner

Consider the discrete saddle point problem

$$\mathcal{A}x = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} = b.$$

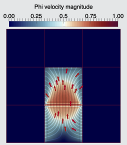
We construct a **monolithic GDSW preconditioner**

$$\mathcal{M}_{\mathrm{GDSW}}^{-1} = \phi \mathcal{A}_0^{-1} \phi^T + \sum_{i=1}^{N} \mathcal{R}_i^T \mathcal{A}_i^{-1} \mathcal{R}_i,$$

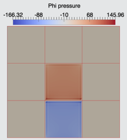
with block matrices $\mathcal{A}_0 = \phi^T \mathcal{A} \phi$, $\mathcal{A}_i = \mathcal{R}_i \mathcal{A} \mathcal{R}_i^T$, and

$$\mathcal{R}_i = \begin{bmatrix} \mathcal{R}_{u,i} & 0 \\ 0 & \mathcal{R}_{p,i} \end{bmatrix} \quad \text{and} \quad \phi = \begin{bmatrix} \Phi_{u,u_0} & \Phi_{u,p_0} \\ \Phi_{p,u_0} & \Phi_{p,p_0} \end{bmatrix}.$$

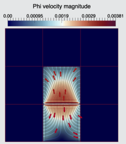
Using $\mathcal{A}$ to compute extensions: $\phi_I = -\mathcal{A}_{II}^{-1} \mathcal{A}_{I\Gamma} \phi_\Gamma$; cf. **Heinlein, Hochmuth, Klawonn (2019, 2020)**.



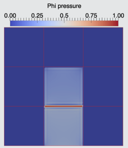
Φ_{u,u_0}



Φ_{p,u_0}

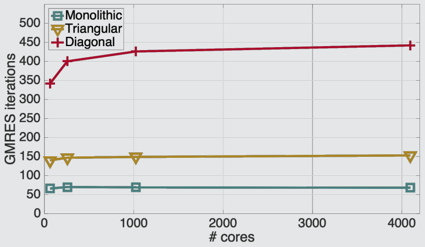


Φ_{u,p_0}



Φ_{p,p_0}

Monolithic vs Block Preconditioners



Prec.	MPI ranks	64	256	1 024	4 096
Monolithic	time	154.7s	170.0s	175.8s	188.7s
	effic.	**100 %**	**91 %**	**88 %**	**82 %**
Triangular	time	309.4s	329.1s	359.8s	396.7s
	effic.	50 %	47 %	43 %	39 %
Diagonal	time	736.7s	859.4s	966.9s	1105.0s
	effic.	21 %	18 %	16 %	14 %

Computations performed on magnitUDE, University Duisburg-Essen.

Monolithic (R)GDSW Preconditioners for CFD Simulations

Monolithic GDSW preconditioner

Consider the discrete saddle point problem

$$\mathcal{A}x = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} = b.$$

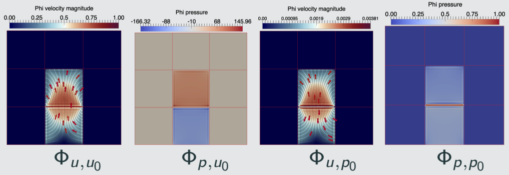
We construct a **monolithic GDSW preconditioner**

$$\mathcal{M}_{\mathrm{GDSW}}^{-1} = \phi \mathcal{A}_0^{-1} \phi^T + \sum\nolimits_{i=1}^{N} \mathcal{R}_i^T \mathcal{A}_i^{-1} \mathcal{R}_i,$$

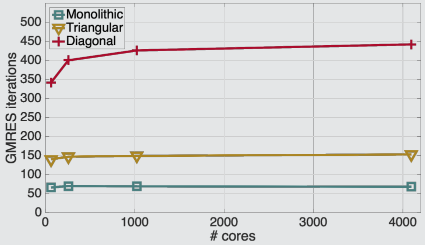
with block matrices $\mathcal{A}_0 = \phi^T \mathcal{A} \phi$, $\mathcal{A}_i = \mathcal{R}_i \mathcal{A} \mathcal{R}_i^T$, and

$$\mathcal{R}_i = \begin{bmatrix} \mathcal{R}_{u,i} & 0 \\ 0 & \mathcal{R}_{p,i} \end{bmatrix} \quad \text{and} \quad \phi = \begin{bmatrix} \Phi_{u,u_0} & \Phi_{u,p_0} \\ \Phi_{p,u_0} & \Phi_{p,p_0} \end{bmatrix}.$$

Using $\mathcal{A}$ to compute extensions: $\phi_I = -\mathcal{A}_{II}^{-1} \mathcal{A}_{I\Gamma} \phi_\Gamma$; cf. **Heinlein, Hochmuth, Klawonn (2019, 2020)**.



Monolithic vs Block Preconditioners



Prec.	MPI ranks	64	256	1 024	4 096
Monolithic	time	154.7s	170.0s	175.8s	188.7s
	effic.	**100 %**	**91 %**	**88 %**	**82 %**
Triangular	time	309.4s	329.1s	359.8s	396.7s
	effic.	50 %	47 %	43 %	39 %
Diagonal	time	736.7s	859.4s	966.9s	1105.0s
	effic.	21 %	18 %	16 %	14 %

Computations performed on magnitUDE, University Duisburg-Essen.

→ Talk by **C. Hochmuth** in **MS11-02 (right after this talk)** for more details.

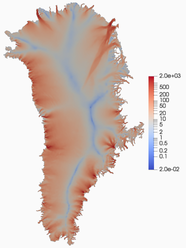
Monolithic (R)GDSW Preconditioners for Multiphysics Land Ice Simulations

We construct a **monolithic two-level GDSW preconditioner**

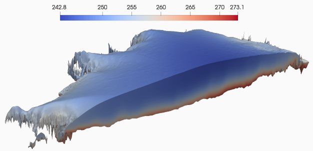
$$\mathcal{M}_{\mathrm{GDSW}}^{-1} = \phi \mathcal{A}_0^{-1} \phi^T + \sum\nolimits_{i=1}^{N} \mathcal{R}_i^T \mathcal{A}_i^{-1} \mathcal{R}_i,$$

for the tangent matrix of the coupled problem

$$\mathcal{A}x := \begin{bmatrix} A_u & C_{uT} \\ C_{Tu} & A_T \end{bmatrix} \begin{bmatrix} x_u \\ x_T \end{bmatrix} = \begin{bmatrix} \tilde{r}_u \\ \tilde{r}_T \end{bmatrix} =: r.$$



Velocity u solution



Temperature T solution

Null space

We use an **equal-order P1 finite element discretization in space** for all variables. Therefore, the null space in each finite element node is given by:

$$r_{u,1} := \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; r_{u,2} := \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \; r_{u,3} := \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \; r_{u,4} := \begin{bmatrix} y \\ -x \\ 0 \\ 0 \end{bmatrix}, \; \text{and } r_T := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

See **Heinlein, Perego, Rajamanickam (in preparation)** for more details.

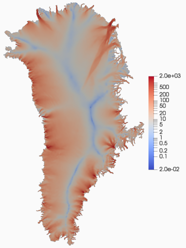
Monolithic (R)GDSW Preconditioners for Multiphysics Land Ice Simulations

We construct a **monolithic two-level GDSW preconditioner**

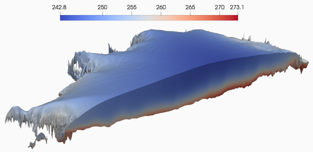
$$\mathcal{M}_{\text{GDSW}}^{-1} = \phi \mathcal{A}_0^{-1} \phi^T + \sum\nolimits_{i=1}^{N} \mathcal{R}_i^T \mathcal{A}_i^{-1} \mathcal{R}_i,$$

for the tangent matrix of the coupled problem

$$\mathcal{A}x := \begin{bmatrix} A_u & C_{uT} \\ C_{Tu} & A_T \end{bmatrix} \begin{bmatrix} x_u \\ x_T \end{bmatrix} = \begin{bmatrix} \tilde{r}_u \\ \tilde{r}_T \end{bmatrix} =: r.$$



Velocity u solution



Temperature T solution

Fully coupled extensions

We compute coarse basis function using extensions

$$\phi = \begin{bmatrix} -\mathcal{A}_{II}^{-1} \mathcal{A}_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix} = \begin{bmatrix} \phi_I \\ \phi_\Gamma \end{bmatrix}$$

based on the coupled matrix $\mathcal{A}$.

Decoupled extensions

We compute coarse basis function using extensions

$$\phi = \begin{bmatrix} -\tilde{\mathcal{A}}_{II}^{-1} \tilde{\mathcal{A}}_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix} = \begin{bmatrix} \phi_I \\ \phi_\Gamma \end{bmatrix}$$

based on the decoupled matrix

$$\tilde{\mathcal{A}} = \begin{bmatrix} A_u & 0 \\ 0 & A_T \end{bmatrix}.$$

See **Heinlein, Perego, Rajamanickam (in preparation)** for more details.

Greenland Coupled Problem – Coarse Spaces

fully coupled extensions

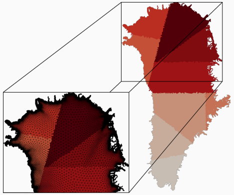
MPI ranks	dim V_0	no reuse: avg. its (nl its)	no reuse: avg. setup	no reuse: avg. solve	reuse coarse basis: avg. its (nl its)	reuse coarse basis: avg. setup	reuse coarse basis: avg. solve
256	1 400	100.1 (27)	4.10 s	6.40 s	**18.5** (70)	**2.28 s**	**1.07 s**
512	2 852	129.1 (28)	1.88 s	4.20 s	**24.6** (38)	**1.04 s**	**0.70 s**
1 024	6 036	191.2 (65)	1.21 s	4.76 s	**34.2** (32)	**0.66 s**	**0.70 s**
2 048	12 368	237.4 (30)	0.96 s	4.06 s	**37.3** (30)	**0.60 s**	**0.58 s**

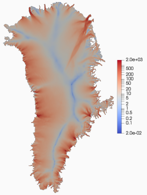
decoupled extensions

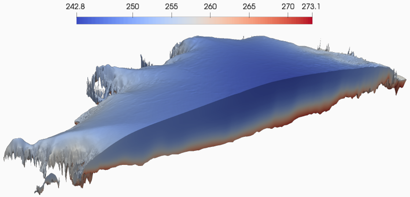
MPI ranks	dim V_0	no reuse: avg. its (nl its)	no reuse: avg. setup	no reuse: avg. solve	reuse coarse basis: avg. its (nl its)	reuse coarse basis: avg. setup	reuse coarse basis: avg. solve
256	1 400	23.6 (29)	3.90 s	1.32 s	**21.5** (34)	**2.23 s**	**1.18 s**
512	2 852	27.5 (30)	1.83 s	0.78 s	**26.4** (33)	**1.13 s**	**0.78 s**
1 024	6 036	30.1 (29)	1.19 s	**0.60 s**	**28.6** (43)	**0.66 s**	0.61 s
2 048	12 368	36.4 (30)	0.69 s	0.56 s	**31.2** (50)	**0.57 s**	**0.55 s**

Problem: Coupled **Mesh:** Greenland, 3-30 km hor. resolution 20 vert. layers **Size:** 7.5 m degrees of freedom (P1 FE) **Coarse space:** RGDSW

Greenland Coupled Problem – Large Problem







MPI ranks	decoupled (no reuse) avg. (nl its)	avg. setup	avg. solve	fully coupled (reuse coarse basis) avg. (nl its)	avg. setup	avg. solve	decoupled (reuse 1st level symb. fact. + coarse basis) avg. (nl its)	avg. setup	avg. solve
512	**41.3** (36)	18.78 s	**4.99** s	45.3 (32)	11.84 s	5.35 s	45.0 (35)	**10.53** s	5.36 s
1 024	53.0 (29)	8.68 s	4.22 s	**47.8** (37)	5.36 s	**3.82 s**	54.3 (32)	**4.59 s**	4.31 s
2 048	62.2 (86)	4.47 s	4.23 s	66.7 (38)	2.81 s	4.53 s	**59.1** (38)	**2.32 s**	**3.99 s**
4 096	**68.9** (40)	2.52 s	**2.86 s**	79.1 (36)	1.61 s	3.30 s	78.7 (38)	**1.37 s**	3.30 s

Problem: Coupled **Mesh:** Greenland, 1-10 km hor. resolution 20 vert. layers **Size:** 7.5 m degrees of freedom (P1 FE) **Coarse space:** RGDSW

Thank you for your attention!

Summary

- Scalable `FROSch` preconditioners
  - for the single physics **velocity and temperature problems**,
  - for the **coupled multi physics problem**. (**monolithic (R)GDSW preconditidoners**)

Outlook

- The solver time is dominated by the direct subdomain and coarse solvers
  $\rightarrow$ Speedup due to the use of **inexact solvers** & of **GPUs**.

Acknowledgements

- **Financial support:** DFG (KL2094/3-1, RH122/4-1), DOE (SciDAC projects FASTMath, ProSPect)
- **Computing ressources:** Cori (NERSC), JUQUEEN (JSC), magnitUDE (UDE), Piz Daint (CSCS)