Efficient multigrid solvers for strongly anisotropic PDEs in atmospheric modelling

Part I: Background, Algorithms & Performance on CPU clusters

Eike Hermann Müller\textsuperscript{(a)}, Robert Scheichl\textsuperscript{(a)}
Collaborators: Andreas Dedner\textsuperscript{(b)}, Markus Gross\textsuperscript{(c)}, Xu Guo\textsuperscript{(d)}, Benson Muite\textsuperscript{(e)}, Sinan Shi\textsuperscript{(d)}, Eero Vainikko\textsuperscript{(f)}

\textsuperscript{(a)}University of Bath, \textsuperscript{(b)}University of Warwick, \textsuperscript{(c)}Met Office, \textsuperscript{(d)}EPCC, \textsuperscript{(e)}KAUST, \textsuperscript{(f)}University of Tartu

CCP-ASEArch Study Group, Daresbury, Jan 21\textsuperscript{st}/22\textsuperscript{nd} 2014
Why fast elliptic solvers? Why parallel? Why manycore?

**Computational Challenges in Numerical Weather- and Climate Prediction**

- Substantial increase in resolution
  
  \[ \sim 25\text{km} \rightarrow \sim \text{few km} \quad \text{(global)} \]

  \[ \Rightarrow \gtrsim 10^{10} \text{ degrees of freedom per atmospheric variable} \]

- Model runtime \( \lesssim 1\text{hour} \) for 5 day forecast (Unified Model)

- **Exploit parallelism** (on different levels) over \( > 10^6 \) cores

- **Multi-/Manycore architectures** (CPU, GPU, Intel MIC)

**Key requirement**

Repeatedly solve **elliptic** PDE for pressure correction with \( \gtrsim 10^{10} \text{ dof in } \ll 1\text{second} \)

PDEs with similar structure in “flat” geometries arise in

- Ocean modelling
- Subsurface flow simulations
- Oil- and gas reservoir modelling
Solving sparse linear systems is challenging

**CHALLENGES FOR LINEAR SOLVERS**

3d elliptic PDE for pressure correction \( u(x, y, z) \)

\[
- \omega^2 \left\{ \xi_r(\partial_r u) + \frac{\partial_r (\alpha_r(\partial_r u)) + \nabla_S \cdot (\alpha_S(\nabla_S u))}{r^2} \right\} + \beta u
\]

→ **Very large sparse** linear algebra problem, \( n \gtrsim 10^{10} \)

\[
\begin{pmatrix}
A_{1,1} & A_{1,2} & \cdots \\
A_{2,1} & \ddots & \vdots \\
\vdots & \ddots & A_{n,n}
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2 \\
\vdots \\
u_n
\end{pmatrix} =
\begin{pmatrix}
b_1 \\
b_2 \\
\vdots \\
b_n
\end{pmatrix} \iff A u = b
\]

**Iterative solvers** (key components: SpMV, tridiagonal solve)

\#FLOPS \approx 2 \times -5 \times \#Memory references (matrix-free!)

\[
t(\text{Mem}) \approx 30 \times -40 \times t(\text{FLOP}) \quad \text{[Fermi M2090, Kepler GK110]}
\]

⇒ Application **memory bound**, especially on GPUs
Strategy

**Goal: minimise total solution time**

1. **Optimal iterative solver**
   - Krylov subspace methods and **geometric** multigrid

2. **Efficient implementation**
   - **Cache** efficiency $\Rightarrow$ data layout
   - **Matrix free** implementation

3. **Massively parallel implementation**
   - MPI implementation on 65536 HECToR CPUs

4. **Efficient single-GPU implementation**
   - **Global memory** access, **cache** reusage

5. **Scalable multi-GPU parallelisation**
   - GCL based implementation on 16384 GPUs
How could this be relevant for DL-MG?

Similar equations

**Linearised Poisson-Boltzmann**

\[-\lambda^2 \nabla (\varepsilon_r \nabla \phi) + \phi = \frac{4\pi}{\varepsilon_0} (\rho_{\text{free}} + \bar{\rho}_{\text{ion}})\]

Debye-length: \(\lambda = \sqrt{\frac{k_B T \varepsilon_0}{4\pi \sum_i c_i z_i^2}}\)

**Atmos. pressure correction**

\[-\omega^2 \left\{ \nabla (\alpha \nabla u) + \xi_r \partial_r u \right\} + \beta u = R\]

“Courant-length”

\(\omega = c_s \Delta t \approx 10\Delta x\)

- Range of interactions \(\sim \Lambda\)
- Only coarsen until \(\Delta x^{(\text{coarse})} \sim \Lambda\)

\(\Rightarrow\) Small number of multigrid levels sufficient

\(\Rightarrow\) Simple coarse grid solver (smoother)

\(\Rightarrow\) Improve/simplify parallelisation
How could this be relevant for DL_MG?

Matrix-free approach
Recalculate local stencil on-the-fly

\[ \varepsilon_{ijk} \rightarrow A_{ij}^{(0)}, A_{ij}^{(+x)}, A_{ij}^{(-x)}, A_{ij}^{(+y)}, A_{ij}^{(-y)}, A_{ij}^{(+z)}, A_{ij}^{(-z)} \]

- Significantly reduce memory transfer
- \( t(\text{Mem}) \approx 30 \times -40 \times t_{\text{FLOP}} \) on GPUs

Multi-GPU parallelisation
- Generic Communication Library (GCL) Mauro Bianco et al. (CSCS)
- Optimised C++ library for inter-GPU/CPU communications on regular grids
How could this be relevant for DL_MG?

What is specific to our problem?

- Strong anisotropy ("flat" domain)
- Global atmosphere
- Cell membrane?

(Approximate) Tensor-product structure of equations

\[
A = -\omega^2 \left( \mathbf{M}^{(r)} \otimes D^{(\text{horiz})}_h + D^{(r)} \otimes \mathbf{M}^{(S)}_h \right) + \mathbf{M}^{(r)} \otimes \mathbf{M}^{(S)}_h
\]

dominant ⇒ preconditioner \( P \), only vertical couplings
Overview

1. Elliptic PDEs in NWP and iterative solvers
   - Pressure correction equation in SISL time stepping
   - Iterative solvers
   - Tensor product multigrid

2. Algorithmic multigrid performance
   - Case study I: ENDGame UM dynamical core
   - Matrix-free DUNE implementation on quasi-uniform grids
   - Case study II: Idealised flow in spherical shell
   - Case study III: Aquaplanet

3. Parallel scalability tests
   - Massively parallel scaling on HECToR
SISL time stepping in NWP

Navier Stokes equations for atmospheric fields $\Phi = \{u, \theta, \pi, \rho\}$ in thin spherical shell

$$\frac{D\Phi}{Dt} = R\{\Phi\}$$

- $u$: Velocity (wind speed)
- $\theta$: Potential temperature
- $\pi$: (Exner-) pressure
- $\rho$: Density

Fast (acoustic) modes $\Rightarrow$ explicit schemes unstable for $\frac{c_s \Delta t}{\Delta x} \geq 1$

Semi-Implicit Semi-Lagrangian time stepping

$$\frac{\Phi(t+\Delta t)(x) - \Phi(t)(x_D)}{\Delta t} = \mu R\{\Phi(t+\Delta t)(x)\} + (1 - \mu) R\{\Phi(t)(x_D)\}$$

$\Rightarrow$ Nonlin. eqn. for $\Phi(t+\Delta t) = \{u(t+\Delta t), \theta(t+\Delta t), \pi(t+\Delta t), \rho(t+\Delta t)\}$
Non-linear iteration

Eliminate $u^{(t+\Delta t)}$, $\theta^{(t+\Delta t)}$, $\rho^{(t+\Delta t)}$ (Schur complement)

$\Rightarrow$ **Nonlinear equation** for (Exner-) pressure $\pi^{(t+\Delta t)}(\equiv \pi' + \pi^0)$

$\mathcal{N}\{\pi^{(t+\Delta t)}\} = \mathcal{R}$

Solve iteratively $\pi'_0 \mapsto \pi'_1 \mapsto \pi'_2 \mapsto \ldots \mapsto \pi'$ via

**Inexact Newton iteration** (“Outer iteration”)

Linearise around background/reference fields: $X^0 = X^{(t)}$

$$\mathcal{L}\pi'_{k+1} = R_k \equiv \mathcal{R} - \mathcal{L}\pi^0 - \delta\mathcal{N}\{\pi'_k + \pi^0\}, \quad \delta\mathcal{N} = \mathcal{N} - \mathcal{L}$$

Linear PDE

NB: $\mathcal{R}$, $\mathcal{N}$, $\mathcal{L}$ and $\pi^0$ vary with time step

This talk:

**Focus on linear solve** (= computationally most expensive part)
Linear PDE for pressure correction

**Elliptic PDE for pressure correction** $\pi'$

\[- \omega^2 \left\{ \Lambda^*0 \rho^0 (\partial_r \theta^0) (\partial_r \pi') + \frac{1}{r^2} \partial_r \left( r^2 \Lambda^*0 \rho^0 \theta^0 (\partial_r \pi') \right) \right\} + \frac{1}{r^2} \nabla_S \cdot \left( \rho^0 \theta^0 (\nabla_S \pi') \right) \right\} + \gamma \frac{\rho^0}{\pi^0} \pi' = \text{RHS} \]

- Vertical advection
- Vertical diffusion
- Horizontal diffusion
- Zero order term

Derivatives: $\partial_r \equiv \langle n, \nabla \rangle$, $\nabla_S \equiv r (\nabla - \langle n, \nabla \rangle n)$, Two sphere $S$

with $\omega = c_h \Delta t / R_{\text{earth}}$, $c_h \sim \text{speed of sound}$

Acoustic Courant number: $\omega/h = \frac{c_h \Delta t}{R_{\text{earth}} \cdot h} = \frac{c_h \Delta t}{\Delta x} = O(10-20)^\dagger$

$^\dagger$ Slow geostrophic modes with $c_{GS} \ll c_h$ still represented correctly
Model equation and grid structure

**Elliptic operator** \((\pi' \mapsto u \text{ from now on})\)

\[-\omega^2 \left\{ \xi_r (\partial_r u) + \frac{\partial_r (\alpha_r (\partial_r u)) + \nabla_S \cdot (\alpha_S (\nabla_S u))}{r^2} \right\} + \beta \pi' \]

Atmospheric state \(\{u^0, \theta^0, \pi^0, \rho^0\} \Rightarrow \text{profiles } \alpha, \beta, \xi\)

**Tensor product grid** (cubed sphere, icosahedral)

- Semi-structured **horizontal grid**: cubed sphere, icosahedral,\ldots
- Regular (graded) **1d vertical grid**, \(n_z = O(100)\)
- \(R_{\text{earth}}/H \approx 100 \Rightarrow \text{Grid-aligned anisotropy } \left(\frac{\Delta x}{\Delta z}\right)^2 \gg 1\)
Tensor product elliptic operator

NWP: **profiles factorise** approximately \((r \in [1, 1 + R_{\text{earth}}/H], \hat{r} \in S)\)

\[ \alpha_X(r, \hat{r}) \approx \alpha_X^r(r) \cdot \alpha_X^S(\hat{r}) \equiv \alpha_X^\otimes(r, \hat{r}) \]

Assume that \(\alpha_X = \alpha_X^\otimes, \beta = \beta^\otimes\), no vertical advection \((\xi_r = 0)\)

**Tensor product elliptic operator** \(D^{(\cdot)} \equiv 2^{\text{nd}} \text{ deriv.}, M^{(\cdot)} \equiv \text{mass term}\)

\[
A = -\omega^2 \left( \underbrace{M^{(r)} \otimes D_{h}^{(\text{horiz})}}_{\text{horizontal derivative}} + \underbrace{D^{(r)} \otimes M_{h}^{(S)}}_{\text{vertical derivative}} + \underbrace{M^{(r)} \otimes M_{h}^{(S)}}_{\text{zero order term}} \right)
\]

- **Exact factorisation only required** for multigrid convergence **theory, not implementation**
- In practise solvers work well if \(\alpha = \alpha^\otimes\) (see later results in this talk)
Iterative Solvers

**Krylov subspace methods** (CG, BiCGStab, GMRES)
- Popular in NWP: current UM uses GMRES/BiCGStab
- Strong vertical anisotropy: $\Rightarrow$ Preconditioner = block-SOR/-Jac.

$$y \leftarrow P^{-1}x$$

Tridiag. system in each vertical column (Thomas algorithm)

**Pros and Cons**
- 😊 Simple to implement
- 😊 Good calc/comm ratio $\Rightarrow$ easy to parallelise
- 😞 Lack of robustness
- 😞 Asymptotic cost $O(n \log n)$ $\Rightarrow$ Slow convergence
Iterative Solvers

**Multigrid** (geometric or AMG) Simultaneously reduce error on all scales

**Pros and Cons**

- 😊  ⇒ Asymptotic cost $O(n)$
  Algorithmically optimal
- 😊  Very robust
- 😞  Harder to implement (in parallel)
  [but that’s no excuse]
- 😞  Worse calc/comm ratio on coarse levels

In our experience:

Multigrid $2 \times -3\times$ faster than CG
Multigrid components

Multigrid components for anisotropic PDEs

- semi-structured grids $\Rightarrow$ geometric MG instead of AMG
- Vertical anisotropy: $\Rightarrow$ Smoother = line relaxation (block-SOR/-Jacobi)

$$y \leftarrow y + \rho_{\text{relax}} P^{-1} (b - Ax)$$

Solve via Thomas algorithm (CG preconditioner: $y \leftarrow P^{-1} x$)

- Coarsening strategy: horizontal-only semi-coarsening
  Counterintuitive, as that’s the weakly coupled direction! $\Rightarrow$ see next slide
Tensor product multigrid theory

Factorising 3d elliptic operator
\[ \alpha_X(r, \hat{r}) = \alpha_X^r(r) \cdot \alpha_X^S(\hat{r}) \]

\[ L^{(3d)} u(r, \hat{r}) = -\nabla_S \left( \alpha_S^r(r) \alpha_S^S(\hat{r}) \nabla_S u(r, \hat{r}) \right) - \partial_r \left( \alpha_r^r(r) \alpha_r^S(\hat{r}) \partial_r u(r, \hat{r}) \right) \]

**Tensor product multigrid (TPMG)**


- Vertical **line relaxation** (e.g. RB Gauss-Seidel)
- **Semi-coarsening** in horizontal direction only

⇒ Convergence rate bounded by convergence rate of “well behaved” & isotropic horizontal operator

\[ L_j^{(2d)} u(\hat{r}) = -\nabla_S \left( \alpha_S^S(\hat{r}) \nabla_S u(\hat{r}) \right) + \lambda_j \alpha_r^S(\hat{r}) u(\hat{r}) \quad \lambda_j \geq 0 \]

\[ \rho_{Vcycle} \left( L^{(3d)} \right) \leq \max_{j=1,\ldots,n_z} \rho_{Vcycle} \left( L_j^{(2d)} \right) \]
Overview

1. Elliptic PDEs in NWP and iterative solvers
   - Pressure correction equation in SISL time stepping
   - Iterative solvers
   - Tensor product multigrid

2. Algorithmic multigrid performance
   - Case study I: ENDGame UM dynamical core
   - Matrix-free DUNE implementation on quasi-uniform grids
   - Case study II: Idealised flow in spherical shell
   - Case study III: Aquaplanet

3. Parallel scalability tests
   - Massively parallel scaling on HECToR
**ENDGame solver** (new dynamical core of Unified Model)

1. **BiCGStab** with RB Block-SOR preconditioner (current solver)
2. **Tensor product geometric multigrid**

Latitude-longitude grid: additional horizontal anisotropy
⇒ conditional semi-coarsening


---

**Multigrid configuration**

- Standalone solver
- RB Block-SOR smoother
- BiCGStab coarse grid solver
- 6 multigrid levels

---

**Model setup**

- $2048 \times 1536 \times 85 = 2.7 \cdot 10^8$ dofs (exp. operational in 1-2 years)
- $48 \times 64 = 3072$ processor cores on MONSoN (IBM Power 7)
Numerical results

Number of iterations for double and single precision ($\epsilon = 10^{-3}$)

Not in asymptotic regime ($\epsilon = 10^{-3}$)

$\Rightarrow$ Convergence dominated by fine modes
Numerical results

**Number of iterations** for double and single precision ($\varepsilon = 10^{-4}$)

![Bar chart showing the number of iterations for BiCGStab and Multigrid for double and single precision.](chart.png)

- **BiCGStab**
  - 1st outer
  - 1st inner
  - 2nd outer
  - 1st inner
  - 2nd outer
  - 2nd inner

- **Multigrid**
  - 1st outer
  - 1st inner
  - 2nd outer
  - 1st inner
  - 2nd outer
  - 2nd inner

- **Double precision**
- **Single precision**
Numerical results

Breakdown of Non-linear solve (full UM model run)

Multigrid reduces error by extra order of magnitude at same cost!
Quasi-uniform grids

Next generation UM dynamical core GungHo! project

Goal

Develop (algorithmically & computationally) efficient and massively parallel elliptic solvers

Still to be decided

- Model equations
- Exact discretisation (probably minetic FEM)
- Transport scheme

Quasi-uniform grids

- $\Delta x_{\text{min}} / \Delta x_{\text{max}} > R$
- No pole problem
  $\Rightarrow$ No horizontal anisotropy
- Horizontal multigrid hierarchy
Quasi-uniform grids

**Coarse grid solver** (our PDE \( \approx \) linearised Poisson-Boltzmann)

\[
-\omega^2 \nabla_S (\alpha_S \nabla_S u) + \beta u \quad \text{Discretisation} \quad \mapsto \quad -\frac{\omega^2}{h^2} \Delta_S (\alpha_S \Delta_S u) + \beta u
\]

2nd order, \( O(\omega^2 h^{-2}) = O(\Lambda^2) \)

On multigrid level \( \ell \) (finest level: \( \ell = 0 \))

\[
-4^{-\ell} \Lambda^2 \Delta_S (\alpha_S \Delta_S u) + \beta u
\]

\( O(4^{-\ell} \Lambda^2) \)

\( O(1) \)

Diagonally dominant for \( \ell \geq \log_2 (\Lambda) \)

\( \Rightarrow \) **operator well conditioned on coarse grids**

- Small number of levels sufficient \( \Rightarrow \) improves parallelisation
- Coarse grid solve = small number of smoothing steps (\( \leq 5 \))
Factorising profiles

**Matrix-“free”**: assemble stencil on-the-fly

- *Non-factorising profiles*: store four 3d profiles in every grid cell
  \[ \Rightarrow O(n_{\text{horiz}} \cdot n_z) \] storage (& data movement!)

- *Factorising profiles*:
  \[ \Rightarrow O(n_{\text{horiz}} + n_z) \] storage (next slide)

**Loading data from memory is expensive** (esp. on GPUs)
\[ \Rightarrow \text{Can we save even more?} \]

1. Construct approximate factorisation of profiles
   \[ \alpha_X^\otimes(r, \hat{r}) = \alpha_X^L(r) \cdot \alpha_X^S(\hat{r}) \approx \alpha_X(r, \hat{r}) \ (\Rightarrow \text{matrix } A^\otimes \approx A) \]

2. Use TP operator with \( \alpha^\otimes \) for multigrid preconditioner
   Richardson iteration : \[ u \leftarrow u + \left( A^\otimes \right)^{-1} (b - Au) \]
Matrix-free algorithm

Tensor-product operator

\[ A = \omega^2 \left( M^{(r)} \otimes D_h^{(\text{horiz})} + \lambda^2 D^{(r)} \otimes M_h^{(\text{horiz})} \right) + M^{(r)} \otimes M_h^{(\text{horiz})} \]

**Matrix free implementation**

- Recompute local matrix stencil
  \[ \Rightarrow O(n_{\text{horiz}} + n_z) \ll O(n_{\text{horiz}} \cdot n_z) \text{ mem. ref.} \]
- Number of vertical levels \( n_z = 128 \gg 1 \)
  - Small overhead from calculation of horizontal coupling \( D_h^{(\text{horiz})}, M_h^{(\text{horiz})} \)
    (and indirect addressing on unstructured grids)
  - Precompute \( D^{(r)}, M^{(r)} \equiv 4 \) vectors of length \( n_z \)
    (keep in cache)
DUNE Implementation

Matrix-“free” DUNE implementation

- Distributed and Unified Numerics Environment (www.dune-project.org)
- C++ library for solving PDEs using grid-based methods
- Interfaces to parallel grids

Data structures:

- **3d Fields**: Store vector of length $n_z$ on each horizontal grid cell ⇒ cache efficiency
- **Matrix**: Construct on-the-fly from profiles $\alpha_S, \alpha_r, \xi_r, \beta$
  1. Non-factorising case:
     - 3d fields of size $O(n_{\text{horiz}} \cdot n_z)$
  2. Factorising case:
     - Horizontal components = 2d fields of size $O(n_{\text{horiz}})$
     - Vertical component = 1d field of size $O(n_z)$

**Indirect addressing** in horizontal only
⇒ “Hidden” by work in vertical as $n_z \gg 1$ [MacDonald et al. (2011)]
Iterative solvers

Iterative Solvers
1. Richardson iteration (1 × SpMV, 1 × preconditioner + BLAS)
2. Conjugate Gradient (1 × SpMV, 1 × preconditioner + BLAS)
3. BiCGStab (2 × SpMV, 2 × preconditioner + BLAS)

NB Always need SpMV with full matrix

Preconditioners $\text{SOR}_{\rho=1}(2, 2)$ V-cycle of TPMG
1. Full non-factorizing profiles ($\text{TPMG}^{(\text{full})}$)
2. Approximately factorized profiles ($\text{TPMG}^{\otimes}$)

$$\alpha(r, \hat{r}) \approx \alpha_r(r) \cdot \alpha_S(\hat{r}) \equiv \alpha^{\otimes}(r, \hat{r})$$

Tradeoff between
- Reduced time per iteration
  - faster matrix assembly, $O(n_z + n_{\text{horiz}})$ instead of $O(n_z \times n_{\text{horiz}})$ reads
- Increased number of iterations
Case Study II: Idealized flow

Idealised balanced atmospheric flow field
Zonal flow, mid-latitude jets with \( u_{\text{max}} = 100 \text{m/s} \)

Only non-factorising ingredient to \( \alpha, \beta, \xi \): Exner press. \( \pi \)

\[
\pi(r, \hat{r}) = \frac{\epsilon + E_r(r)E_S(\phi)}{1 + \epsilon}
\]

\[
\pi^\otimes(r, \hat{r}) \equiv \frac{\epsilon + E_r(r)}{1 + \epsilon} \cdot E_S(\phi)
\]

\( \epsilon = 0 \):
\( \Rightarrow \alpha = \alpha^\otimes, \beta = \beta^\otimes, \xi = \xi^\otimes \)

Non-factorisation: buoyancy frequency \( N = 0.018 \text{s}^{-1}, \ldots, 0.028 \text{s}^{-1} \)

\[
\epsilon = \left( \frac{N}{N^*} \right)^2 - 1, \quad N^* = \sqrt{\gamma \frac{c_s}{g}}
\]
Results: Time per iteration

Time per iteration (sequential) for ALUGrid and UGGrid

- V-Cycle (TPMG^{(full)})
- V-Cycle (TPMG⊗)
- SpMV
- Other [BLAS]

Richardson: 0.0
BiCGStab: 2.0

Eike Mueller
Efficient multigrid for anisotropic PDEs
Results: Number of iterations

Number of iterations (relative residual reduction $= 10^{-5}$)

- Richardson
- BiCGStab
- TPMG\textsuperscript{(full)}
- TPMG $\otimes$

$\epsilon$ = deviation from perfect factorization
Results: Total solution time

Total sequential solution time for $2.6 \cdot 10^6$ dof, $n_z = 128$

![Graph showing solution times for different methods and deviations from perfect factorization.](image)
Case Study III: Aquaplanet

Realistic atmospheric testcase
Profiles from “aquaplanet” run with full Unified Model

Preconditioners
- **TPMG\(^{(\text{full})}\):**
  still excellent preconditioner
- **TPMG\(^{\otimes}\):**
  Near-surface convection
    - \(\alpha_r\) hard to factorize
    - Solver diverges
- **TPMG\(^{[\otimes]}\):**
  - factorize \(\alpha_S, \xi_r\) and \(\beta\) only
  - Performance loss of \(\approx 6\%\) in \(t_{\text{iter}}\) relative to TPMG\(^{\otimes}\).
## Results

### Sequential performance, $2.6 \cdot 10^6$ unknowns

**Time per iteration [s]**

<table>
<thead>
<tr>
<th>Solver</th>
<th>$t_{\text{iter}}$</th>
<th>speedup</th>
<th>$t_{\text{iter}}$</th>
<th>speedup</th>
<th>$t_{\text{iter}}$</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Richardson</td>
<td>1.43</td>
<td>—</td>
<td>1.11</td>
<td>1.29×</td>
<td>1.05</td>
<td>1.36×</td>
</tr>
<tr>
<td>BiCGStab</td>
<td>2.88</td>
<td>—</td>
<td>2.26</td>
<td>1.27×</td>
<td>2.14</td>
<td>1.35×</td>
</tr>
</tbody>
</table>

**Number of iterations and total solution time**

| Solver       | # iterations ($||r||/||r_0||$) | total time [s] |
|--------------|---------------------------------|----------------|
| Richardson   | 5 ($9.1 \cdot 10^{-6}$)         | 7.94           |
|              | 7 ($4.8 \cdot 10^{-6}$)         | 7.28           |
| BiCGStab     | 3 ($5.1 \cdot 10^{-7}$)         | 8.81           |
|              | 3 ($5.2 \cdot 10^{-6}$)         | 6.94           |
Overview

1. Elliptic PDEs in NWP and iterative solvers
   - Pressure correction equation in SISL time stepping
   - Iterative solvers
   - Tensor product multigrid

2. Algorithmic multigrid performance
   - Case study I: ENDGame UM dynamical core
   - Matrix-free DUNE implementation on quasi-uniform grids
   - Case study II: Idealised flow in spherical shell
   - Case study III: Aquaplanet

3. Parallel scalability tests
   - Massively parallel scaling on HECToR
Parallel grids

Parallel grid implementations

Requirements: parallel 2d grid $\subset \mathbb{R}^3$ for unit sphere

- ALUGrid$\langle 2, 3 \rangle$
  - Does not scale!
- UGGrid$\langle 3, 3 \rangle$
  - 2d grid = thin shell
  - Horizontal refinement only

UGGrid originally not designed for more than $\sim 100$ processors
⇒ Fix bugs & plug several memory leaks [Oliver Sander]

Parallel partitioning

- Read macrogrid with $n_{\text{proc}}$ elements from .dgf
- Loadbalance $\Rightarrow$ one element / processor
- Refine (in horizontal only)
Massively parallel weak scaling

Weak scaling on HECToR: Case study I (balanced flow)

- Richardson
- BiCGStab
- TPMG (full)
- TPMG ⊗

- Number of processors: 20, 80, 320, 1280, 5120, 20480
- Time per iteration [s]: 1.0 \cdot 10^7, 1.1 \cdot 10^{10}

Eike Mueller
Efficient multigrid for anisotropic PDEs
Simplified model equation

Setup for parallel scaling runs

Simplified model equation

\[-\omega^2 \left[ \nabla_s \nabla_s + \lambda^2 \frac{1}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial}{\partial r} \right) \right] u + u = f \quad \omega \propto \Delta t \propto \Delta x\]

Compare different solvers

- AMG (DUNE, Hypre)
- Bespoke geometric TPMG †, ★
- CG with line relaxation †, ★

†: Matrix-free
★: Fortran 90 implementation

- Solve on one panel of cubed sphere
- Factorising profiles
Does the solver scale?

Weak scaling of total solution time on HECToR Cray XE6
(model equation, one panel of cubed sphere grid)

Solve PDE with $3.4 \cdot 10^{10}$ dof in 1 second
### Algorithmic performance (# iter) & parallel efficiency $E_S$

for different solvers (model equation, one panel of cubed sphere grid)

<table>
<thead>
<tr>
<th>cores</th>
<th># dof</th>
<th># iter</th>
<th>$t_{iter}$</th>
<th>$E_S$</th>
<th># iter</th>
<th>$t_{iter}$</th>
<th>$E_S$</th>
<th># iter</th>
<th>$t_{iter}$</th>
<th>$E_S$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>multilevel</td>
<td>AMG (DUNE)†</td>
<td></td>
<td></td>
<td></td>
<td>PCG</td>
<td></td>
<td></td>
<td>geometric MG</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>$8.3 \cdot 10^6$</td>
<td>12</td>
<td>0.56</td>
<td>—</td>
<td>44</td>
<td>0.109</td>
<td>—</td>
<td>6</td>
<td>0.143</td>
<td>—</td>
</tr>
<tr>
<td>64</td>
<td>$3.4 \cdot 10^7$</td>
<td>13</td>
<td>0.56</td>
<td>100%</td>
<td>43</td>
<td>0.113</td>
<td>96%</td>
<td>6</td>
<td>0.148</td>
<td>97%</td>
</tr>
<tr>
<td>256</td>
<td>$1.3 \cdot 10^8$</td>
<td>12</td>
<td>0.59</td>
<td>96%</td>
<td>41</td>
<td>0.114</td>
<td>96%</td>
<td>6</td>
<td>0.152</td>
<td>94%</td>
</tr>
<tr>
<td>1024</td>
<td>$5.4 \cdot 10^8$</td>
<td>14</td>
<td>0.60</td>
<td>95%</td>
<td>41</td>
<td>0.116</td>
<td>94%</td>
<td>6</td>
<td>0.155</td>
<td>93%</td>
</tr>
<tr>
<td>4096</td>
<td>$2.1 \cdot 10^9$</td>
<td>14</td>
<td>0.59</td>
<td>96%</td>
<td>41</td>
<td>0.117</td>
<td>93%</td>
<td>6</td>
<td>0.159</td>
<td>90%</td>
</tr>
<tr>
<td>16384</td>
<td>$8.6 \cdot 10^9$</td>
<td>14</td>
<td>0.60</td>
<td>94%</td>
<td>40</td>
<td>0.115</td>
<td>94%</td>
<td>6</td>
<td>0.161</td>
<td>89%</td>
</tr>
<tr>
<td>65536</td>
<td>$3.4 \cdot 10^{10}$</td>
<td>11</td>
<td>0.62</td>
<td>91%</td>
<td>40</td>
<td>0.115</td>
<td>94%</td>
<td>6</td>
<td>0.177</td>
<td>81%</td>
</tr>
</tbody>
</table>

† As preconditioner for CG
Robustness

Vary coefficients

\[ \omega^2 \mapsto f_{\omega^2} \times \omega^2, \quad \lambda^2 \mapsto f_{\lambda^2} \times \lambda^2 \]

in model equation

\[ -\omega^2 \left[ \nabla S \nabla S + \lambda^2 \frac{1}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial}{\partial r} \right) \right] u + u = f \quad \omega \propto \Delta t \propto \Delta x \]

Number of iterations (16384 cores, 8.6 \cdot 10^9 unknowns)

<table>
<thead>
<tr>
<th># of multigrid levels</th>
<th>13</th>
<th>7</th>
<th>4</th>
<th>1 (CG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{\omega^2} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( f_{\lambda^2} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>10^{2}</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>39</td>
</tr>
<tr>
<td>10^{-2}</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>37</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>6</td>
<td>14</td>
<td>131</td>
</tr>
<tr>
<td>100</td>
<td>6</td>
<td>12</td>
<td>129</td>
<td>438</td>
</tr>
</tbody>
</table>
Conclusion and Outlook

Summary

- Elliptic PDE for pressure correction in semi-implicit semi-Lagrangian time stepping \( \approx \) linearised PBE
- Tensor product grid and (anisotropic) operator
- Bespoke geometric multigrid
  - Algorithmically optimal (beats BiCGStab, CG)
  - Robust
  - Small number of multigrid levels
  - Matrix-free implementation
- Massively parallel scaling on HECToR

Outlook

- Hybrid MPI+OpenMP parallelisation
- Non-linear solve
- FEM discretisation
References

- **Andreas Dedner, EHM, RS**: *Efficient multigrid preconditioners for anisotropic elliptic PDEs in geophysical modelling* (in preparation)


- **Peter Bastian et al.**: *The Distributed and Unified Numerics Environment (DUNE)* [http://www.dune-project.org/](http://www.dune-project.org/)