Firedrake

Multigrid for numerical weather prediction

Lawrence Mitchell[1] Eike Müller[2]

20th May 2016

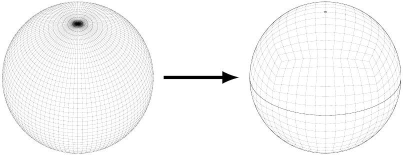
[1]Departments of Computing and Mathematics, Imperial College London

[2]Department of Mathematical Sciences, University of Bath

Imperial College London

Introduction

Weather forecasting centres moving from orthogonal lat-lon to non-orthogonal meshes (e.g. cubed sphere, or icosahedral sphere).



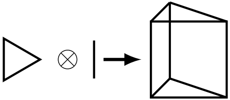
Can no longer use traditional staggered finite differences.

Compatible finite elements (FEEC) a plausible solution (Cotter and Shipton 2012).

Choose discrete spaces that match the properties of the continuous equations.

$$
\begin{array}{ccccccc}
H^1 & & H(\mathrm{curl}) & & H(\mathrm{div}) & & L^2 \\
\cup & & \cup & & \cup & & \cup \\
\mathbb{W}_0 & \xrightarrow{\mathrm{grad}} & \mathbb{W}_1 & \xrightarrow{\mathrm{curl}} & \mathbb{W}_2 & \xrightarrow{\mathrm{div}} & \mathbb{W}_3
\end{array}
$$

Tensor product spaces on wedges



Dynamical core needs to handle fast acoustic waves implicitly.

Need a solver that is *scalable* and *fast*.

5km horizontal, 100 vertical layers, around $2 \cdot 10^9$ dofs.

Large aspect ratio of the domain is challenging for black box solvers.

Formulation

Linear system for velocity $\boldsymbol{v}$, pressure p, and buoyancy b.

$$\frac{\partial \mathbf{v}}{\partial t} = \nabla p + b\hat{\mathbf{z}},$$
$$\frac{\partial p}{\partial t} = -c^2 \nabla \cdot \mathbf{v},$$
$$\frac{\partial b}{\partial t} = -N^2 \mathbf{v} \cdot \hat{\mathbf{z}}.$$

Enforce $\boldsymbol{v} \cdot \hat{\boldsymbol{n}} = 0$ at top and bottom.

$R \approx 6000\text{km}$, $H \approx 80\text{km}$.

After discretising in space and time we obtain a linear operator for $\boldsymbol{v} \in H(\mathrm{div})$, $p \in L^2$, b scalar, collocated with vertical part of $\boldsymbol{v}$ space.

$$\begin{pmatrix} M_v & -\frac{\Delta t}{2} D^T & -\frac{\Delta t}{2} Q \\ \frac{\Delta t}{2} c^2 D & M_p & 0 \\ \frac{\Delta t}{2} N^2 Q^T & 0 & M_b \end{pmatrix}$$

Eliminate buoyancy pointwise in preconditioner. Exact for spherical earth, approximate when mountainous.

Obtain a velocity-pressure system

$$\begin{pmatrix} M_v & -\frac{\Delta t}{2} D^T \\ \frac{\Delta t}{2} c^2 D & M_p \end{pmatrix}$$

Block diagonal

Using the appropriate function space inner products (Mardal and Winther 2011; Kirby 2010)

$$\begin{pmatrix} (I - \operatorname{grad}\operatorname{div})^{-1} & 0 \\ 0 & I \end{pmatrix}$$

- All operators sparse
- $H(\mathrm{div})$ multigrid (Arnold, Falk, and Winther 2000) is challenging

Schur complement

Block elimination and back substitution

$$\begin{pmatrix} M_v^{-1} & 0 \\ 0 & (M_p + \omega_c^2 D M_v^{-1} D^T)^{-1} \end{pmatrix}$$

- $S = M_p + \omega_c^2 D M_v^{-1} D^T$ is *dense*, but elliptic
- Can use similar methods as a *distributive* smoother.

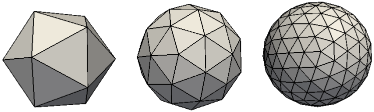
- Strong coupling in vertical direction, need to treat this exactly.
- Use a multigrid cycle with horizontal coarsening and vertical line relaxation.
- Exploit tensor-product structure to split horizontal and vertical components.

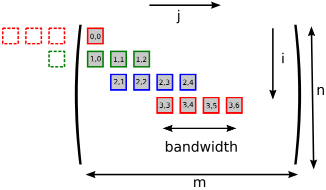
- Split horizontal and vertical $M_v^{-1} = (M_v^h)^{-1} \oplus (M_v^z)^{-1}$ (also $D = D^h \oplus D^z$)
- Diagonal approximation $M_v^{-1} \approx M_{v,\mathrm{inv}} = \delta_{ij}/(M_v)_{ii}$
- Ignore horizontal coupling, error $\mathcal{O}((\Delta z/\Delta x)^2)$

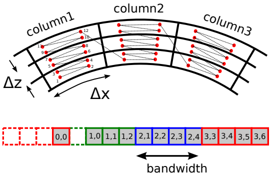
Final approximation

$$\tilde{S} = M_p + \omega_c^2 \mathrm{bdiag}\left[D^h M_{v,\mathrm{inv}}^h (D^h)^T\right] \oplus \frac{\omega_c^2}{1+\omega_N^2} D^z M_{v,\mathrm{inv}}^z (D^z)^T$$

Multigrid V-cycle using exact inverse of $\tilde{S}$ as smoother. Utilise column innermost numbering to perform fast banded matrix inverse







Imperial College London

Banded matrix algebra implemented as short PyOP2 kernels (Firedrake's *escape hatch*)

```
# C-kernel code for Matrix-vector product v = A.u in one vertical column
kernel_code = '''void ax(double **A, double **u, double **v) {
  for (int i = 0; i < n_row; ++i) {
    v[0][i] = 0.0;
    int j_m = (int)ceil((alpha*i - gamma_p)/beta);
    int j_p = (int)floor((alpha*i + gamma_m)/beta);
    for (int j = std::max(0, j_m); j < std::min(n_col, j_p+1); ++j)
      v[0][i] += A[0][bandwidth*i + (j - j_m)] * u[j];
  }'''
# Execute PyOP2 kernel over grid
kernel = op2.Kernel(kernel_code, 'ax', cpp=True)
op2.par_loop(kernel, hostmesh.cell_set,
             A(op2.READ, Vcell.cell_node_map()),
             u.dat(op2.READ, u.cell_node_map()),
             v.dat(op2.WRITE, u.cell_node_map()))
```

Implementation III

- Use PETSc to solve block system with fieldsplit preconditioner.
- Preconditioner for S a `PCSHELL` implementing custom multigrid.
- Allows comparison with purely algebraic approach.

Results

Compare three solvers to invert S

GMRES(30) + full schur complement factorisation tolerance 10^{-5}.

Single level

Two smoother iterations just on fine level (standard approach in Met Office Unified Model).

Custom MG

Bespoke multigrid, 5 levels. 1 smoother sweep per level. 2 smoother iterations on coarsest level.

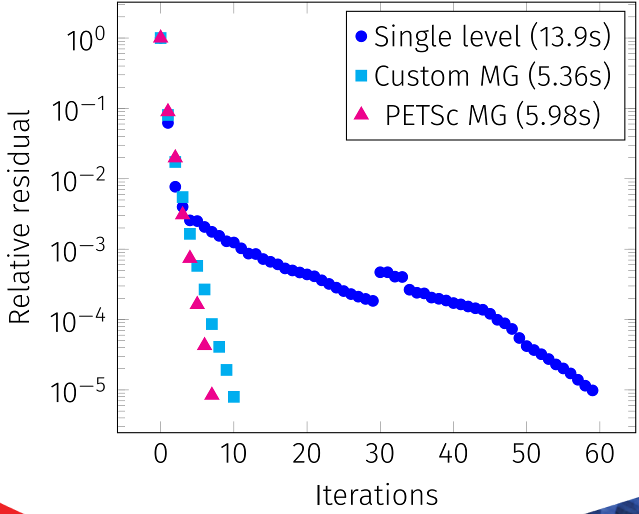
PETSc MG

BoomerAMG with ILU smoother on $\tilde{S} = M_p + \omega_c^2 D M_{v,\mathrm{inv}} D^T$

Algorithmic performance

Convergence with CFL = 8, low-order

$\boldsymbol{v} \in \mathrm{RT0}$, $p \in \mathrm{DG0}$



$18.4 \cdot 10^6$ total unknowns.

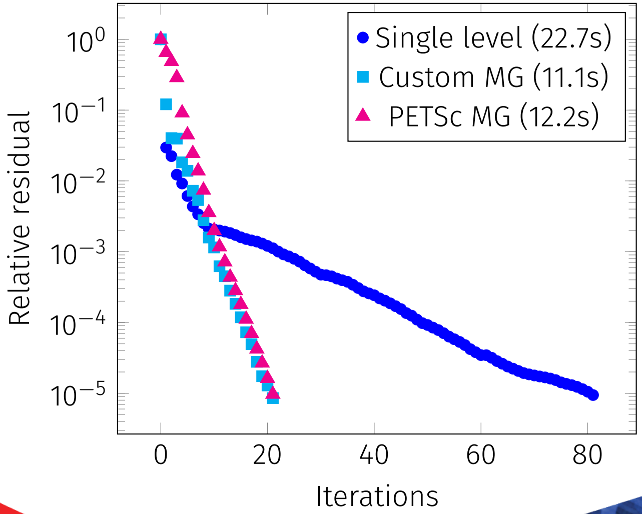
24 cores.

Imperial College London

Algorithmic performance

Convergence with CFL $= 8$, high-order

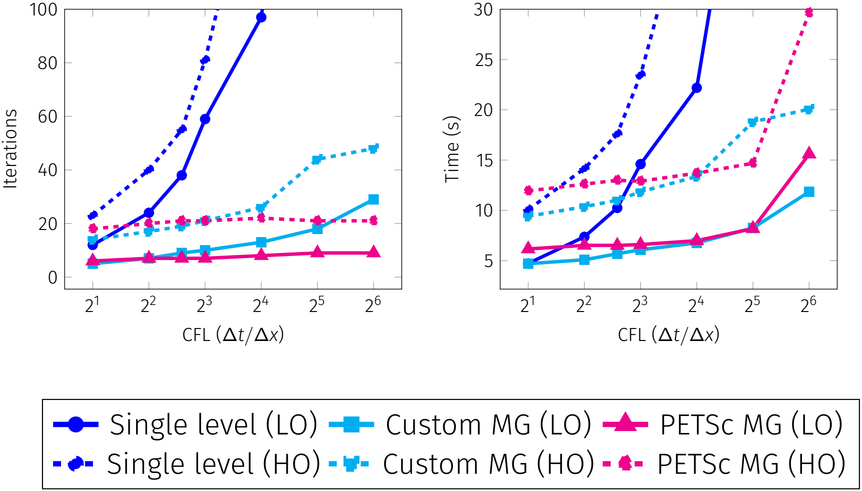
$v \in \mathtt{HDiv}(\mathrm{BDFM1} \otimes \mathrm{P2})$, $p \in \mathrm{DG1}$



$7.9 \cdot 10^{6}$ total unknowns.

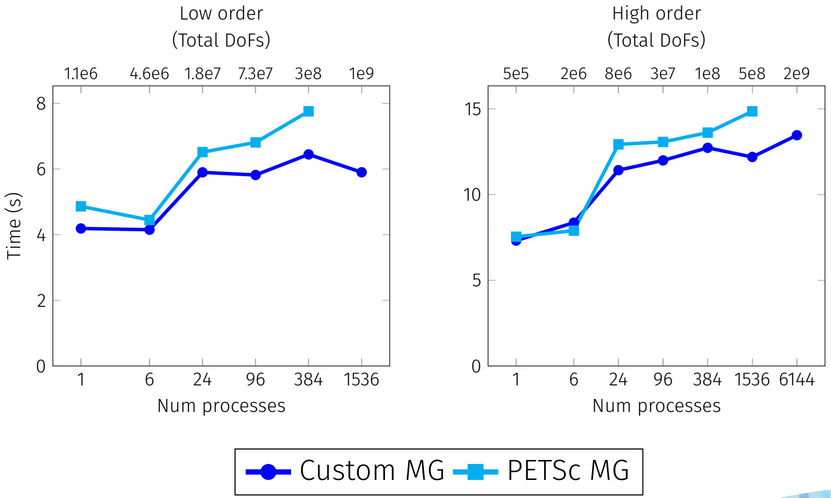
24 cores.

Varying CFL



100
80
60
40
20
0
Iterations
2^1 2^2 2^3 2^4 2^5 2^6
CFL ($\Delta t/\Delta x$)
30
25
20
15
10
5
Time (s)
2^1 2^2 2^3 2^4 2^5 2^6
CFL ($\Delta t/\Delta x$)
Single level (LO)
Custom MG (LO)
PETSc MG (LO)
Single level (HO)
Custom MG (HO)
PETSc MG (HO)

Weak scaling



Imperial College London

Could we go faster?

Sparse matvec is memory bandwidth limited. We count bytes moved assuming a perfect cache.

Achievable memory bandwidth for $a_i = \alpha b_i + c_i$ (STREAM triad) 74GB/s.

Measured bandwidth GB/s

	Low order		High order	
Level	$\tilde{S}$	$\tilde{S}^{-1}$	$\tilde{S}$	$\tilde{S}^{-1}$
0	2.52	1.55	0.70	0.47
1	6.65	4.50	2.30	1.60
2	14.50	4.00	7.23	5.29
3	28.09	29.29	4.69	8.99
4	39.30	40.78	45.92	12.36

Conclusions

- Scalable solver
- Performance within factor 2 of peak
  - Probably 2x available with strong scaling
  - More with judicious low-level optimisation (operational codes)
- Other approaches?
  - H(div) multigrid (Arnold, Falk, and Winther 2000)
  - Distributive smoothing (Uzawa or similar)
  - Auxiliary space (Hiptmair and Xu 2007)
- For more, see `arXiv:1605.00492 [cs.MS]`

References

Arnold, D. N., R. S. Falk, and R. Winther (2000). “Multigrid in $H(\mathrm{div})$ and $H(\mathrm{curl})$”. *Numerische Mathematik* 85. doi:10.1007/s002110000137.

Cotter, C. J. and J. Shipton (2012). “Mixed finite elements for numerical weather prediction”. *Journal of Computational Physics* 231. doi:10.1016/j.jcp.2012.05.020.

Hiptmair, R. and J. Xu (2007). “Nodal auxiliary space preconditioning in $H(\mathrm{curl})$ and $H(\mathrm{div})$ spaces”. *SIAM Journal on Numerical Analysis* 45. doi:10.1137/060660588.

Kirby, R. C. (2010). “From Functional Analysis to Iterative Methods”. *SIAM Review* 52. doi:10.1137/070706914.

Mardal, K.-A. and R. Winther (2011). “Preconditioning discretizations of systems of partial differential equations”. *Numerical Linear Algebra with Applications* 18. doi:10.1002/nla.716.

Imperial College London