Optimal transport and mesh generation on the plane and the sphere

Chris Budd[1], Andrew McRae[2], Colin Cotter[3]

[1]University of Bath

[2]Oxford University

[3]Imperial College London

Delft, December, 2019

Motivation

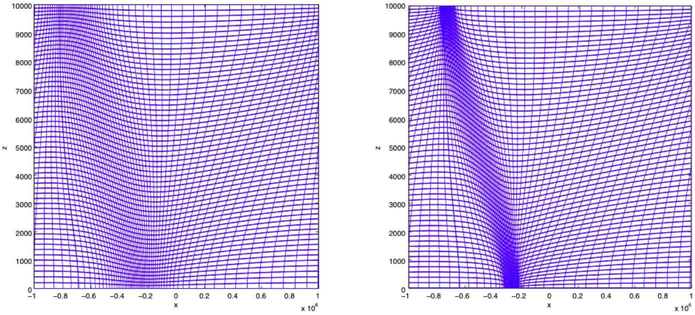
PDE computations often need to use a computational mesh which can

- Capture small scales
- Is aligned with the solution
- Resolve local geometry eg. orography

This is needed

- For accurate numerical computation of anisotropic evolving features eg. storms, fronts possibly on the sphere
- For accurate approximation of anisotropic functions on many scales eg. For data assimilation calculations

Front formation for the Eady Equations of a tropical storm



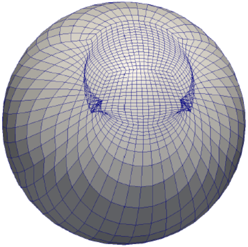
r-adaptivity*

- Want to construct a suitable mesh τ : Various methods eg. mesh refinement, mesh relocation
- *r*-adaptivity: relocate mesh vertices, preserving mesh connectivity/topology.
- When done dynamically, "moving mesh" method.
- Some advantages over *h*-adaptive refinement
  - Can avoid sudden changes in mesh resolution
  - Control over global mesh regularity
  - Mesh topology unchanged $\Longrightarrow$ constant data structures
  - Avoids the obvious load-balance issue when used in parallel

Video: Moving a mesh on the sphere

r-adaptivity

- Some disadvantages:
  - Solution of an extra PDE adds complexity and computational cost
  - Unchanging topology constrains refinement at a global scale
  - Can give rise to skew meshes
  - Poor algorithm can lead to tangling



r-adaptivity

Mesh density controlled by monitor function $m(\vec{x}) > 0$, through **equidistribution**:

$$m(x) \times \text{cell area} = \text{const}$$

(or $\int_{\text{cell}} m \, \mathrm{d}\vec{x} = \text{const}$).

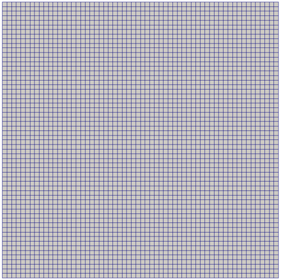
In practice, m will be derived from current simulation state of solution $u(x)$, e.g.

- $m \propto \|\nabla u\|^p, \|\nabla\nabla u\|^p$, etc., so that 'the function $u(x)$ is represented as well as possible' (minimise interpolation error)
- m based on diagnostic derived from physical principles, e.g. vorticity
- Estimates of local interpolation or truncation error

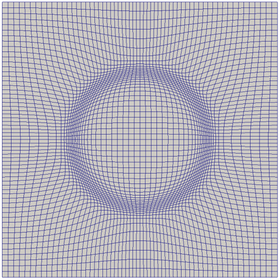
r-adaptivity

Notation:

- Ω_C – "computational" domain eg. plane, sphere (often with uniform mesh)
- Ω_P – "physical" domain
- $\vec{\xi}$ – coordinate in computational domain
- $\vec{x}$ – coordinate in physical domain

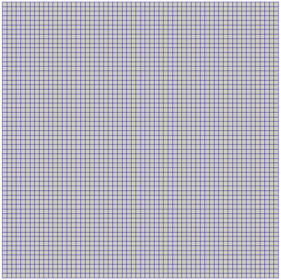


Ω_C, $\vec{\xi}$

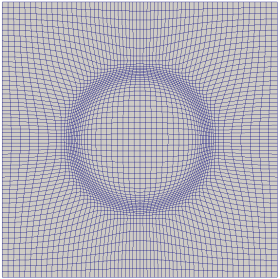


Ω_P, $\vec{x}$

r-adaptivity



Ω_C, $\vec{\xi}$



Ω_P, $\vec{x}$

Adapted mesh is defined by a map $\vec{x}(\vec{\xi})$

The Jacobian of this map is J, $J_{ij} := \frac{\partial x_i}{\partial \xi_j}$

Equidistribution requirement:

$$m(\vec{x}) \det J = \text{const} =: \theta$$

r-adaptivity

Equidistribution requirement:

$$m(\vec{x}) \det J = \theta \tag{1}$$

In 1D, this defines the mesh (almost) uniquely.

In 2D/3D, additional regularisation constraints are needed.

Budd & Williams (2006): subject to (1), pick $\vec{x}(\vec{\xi})$ **minimising**

$$\int_{\Omega_C} |\vec{x}(\vec{\xi}) - \vec{\xi}|^2 \, \mathrm{d}\vec{\xi}.$$

Prevents tangling and reduces skewness.

OT-based mesh generation on the plane

Well-known result from Optimal Transport literature (Brenier, 1991):

There exists a unique map $\vec{x}(\vec{\xi})$ that minimises

$$\int_{\Omega_C} |\vec{x}(\vec{\xi}) - \vec{\xi}|^2 \, \mathrm{d}\vec{\xi}$$

subject to the requirement $m(\vec{x}) \det J = \theta$. Furthermore, this can be written as the gradient of a convex 'potential' function, $\tilde{\phi}(\vec{\xi})$:

$$\vec{x} = \nabla_\xi \tilde{\phi}.$$

Note: convex $\Longrightarrow$ no tangling!

We will actually write

$$\vec{x} = \vec{\xi} + \nabla_\xi \phi \qquad\qquad (\phi \approx \tilde{\phi} - \frac{1}{2}|\xi|^2)$$

Better for periodic domains, and generalises to the Sphere

OT-based mesh generation on the plane

Substituting $\vec{x} = \vec{\xi} + \nabla\phi$ into the definition of J gives

$$J = I + \nabla\nabla\phi.$$

The governing equation is then

$$m(\vec{x})\,\det(I + \nabla\nabla\phi) = \theta.$$

In 2D plane, this is a **Monge–Ampère equation**

$$m(\vec{x})((1 + \phi_{xx})(1 + \phi_{yy}) - \phi_{xy}^2) = \theta.$$

Couple to Neumann or periodic boundary conditions

- Fully nonlinear product of second derivatives of ϕ
- $m(\vec{x})$ is a function of $\nabla\phi$

OT-based mesh generation on the sphere S_2

We use the same approach on the sphere S_2: pick $\vec{x}(\vec{\xi}) : S_2 \to S_2$ **minimising**

$$\int_{\Omega_C} \|\vec{x}(\vec{\xi}) - \vec{\xi}\|^2 \, \mathrm{d}\vec{\xi},$$

where $\| \cdot \|$ is now *geodesic* distance, subject to the equidistribution requirement $m(\vec{x}) \det J = \theta$.

McCann (2001) For a general manifold $\mathcal{M}$ there is a unique such map $\vec{x}(\vec{\xi})$, and there exists a scalar function $\phi(\vec{\xi}) : \mathcal{M} \to R, \quad \nabla\phi \in T_\xi\mathcal{M}$:

$$\vec{x} = \exp(\nabla\phi)\, \vec{\xi}.$$

OT-based mesh generation on the sphere

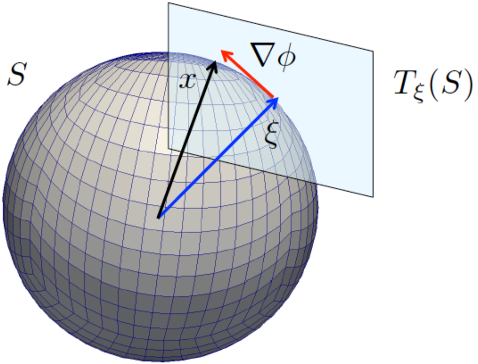
Intuitively, the **exponential map** $\vec{x} = \exp(\nabla\phi)\vec{\xi}$ is:

1. Start at $\vec{\xi}$
2. Travel along geodesic in direction of $\nabla\phi$
3. Stop after distance $|\nabla\phi|$

- On plane, $\exp(\nabla\phi)\vec{\xi} = \vec{\xi} + \nabla\phi$
- On sphere with radius R centred at the origin, Rodrigues' map gives

$$\exp(\nabla\phi)\,\vec{\xi} = \cos\left(\frac{|\nabla\phi|}{R}\right)\vec{\xi} + R\sin\left(\frac{|\nabla\phi|}{R}\right)\frac{\nabla\phi}{|\nabla\phi|}.$$

OT-based mesh generation on the sphere



$$\xi \perp \nabla\phi \in T_\xi(S), \qquad x = e^{\nabla\phi}\xi$$

OT-based mesh generation on the sphere

Now write equidistribution equation $m(\vec{x}) \det J = \theta$ in terms of $\phi(\xi)$

We treat $\vec{x}(\vec{\xi})$ as a map from $\mathbb{R}^3$ to $\mathbb{R}^3$ (partly for software reasons).

J is then rank-deficient. We produce an "equivalent" object of full rank, giving the **Monge-Ampere-like equation**

$$m(\vec{x}) \det((\nabla \exp(\nabla\phi)\vec{\xi}) \cdot P_\xi + \vec{k}_P \otimes \vec{k}_C) = \theta,$$

where $P_\xi := I - \vec{k}_C \otimes \vec{k}_C$ is a projection matrix, $\vec{k}_P$ and $\vec{k}_C$ are unit normal vectors at $\vec{x}$ and $\vec{\xi}$, and the earlier formula is used for the exponential map.

Analogous to adding $(0,0,1) \otimes (0,0,1)$ to a 2x2 matrix to produce an "equivalent" 3x3 matrix.

Numerical solution of MA on the plane

$$m(\vec{x})\,\det(I + \nabla\nabla\phi) = \theta.$$

Mixed finite element approach, based on Lakkis and Pryer (2011, 2013): to get a stable method, introduce discrete variable σ representing $\nabla\nabla\phi$.

Let $\langle -, - \rangle$ denote the obvious inner product $\int_{\Omega_C} \cdot \, \mathrm{d}\vec{\xi}$.

For suitable finite element spaces V_1, V_2, we seek $\phi \in V_1, \sigma \in V_2$ s.t.

$$\begin{aligned} \langle v, m(\nabla\phi)\det(I + \sigma)\rangle &= \langle v, \theta\rangle, & \forall v \in V_1 \\ \langle \tau, \sigma\rangle + \langle \nabla \cdot \tau, \nabla\phi\rangle &= 0, & \forall \tau \in V_2. \end{aligned}$$

On triangles: $V_1 = P_2$ (continuous, piecewise-quadratic), $V_2 = P_2^{(2\times 2)}$
On quads: $V_1 = Q_2$ (continuous, piecewise-biquadratic), $V_2 = Q_2^{(2\times 2)}$

Numerical solution on the plane

Seek $\phi \in V_1, \sigma \in V_2$ such that

$$\langle v, m(\nabla\phi)\det(I+\sigma)\rangle = \langle v, \theta\rangle, \qquad \forall v \in V_1 \tag{2}$$

$$\langle \tau, \sigma\rangle + \langle \nabla\cdot\tau, \nabla\phi\rangle = 0, \qquad \forall \tau \in V_2. \tag{3}$$

We look at two ways of solving the nonlinear system (2)–(3):

- Relaxation method
- Quasi-Newton method

Relaxation method

Given (ϕ^n, σ^n), how to obtain $(\phi^{n+1}, \sigma^{n+1})$?

Iteration:

$$-\nabla^2 \phi^{n+1} = -\nabla^2 \phi^n + \Delta\big(m(\nabla \phi^n) \det(I + \nabla\nabla\phi^n) - \theta^n\big),$$

with Δ some 'step size'.

Full version based on Awanou (2015): **obtain** ϕ^{n+1} by solving

$$\langle \nabla v, \nabla \phi^{n+1} \rangle = \langle \nabla v, \nabla \phi^n \rangle + \Delta \langle v, m(\nabla \phi^n) \det(I + \sigma^n) - \theta^n \rangle, \qquad \forall v \in V_1,$$

then **obtain** σ^{n+1} by solving

$$\langle \tau, \sigma^{n+1} \rangle = -\langle \nabla \cdot \tau, \nabla \phi^{n+1} \rangle, \qquad \forall \tau \in V_2.$$

This **converges to the solution of the nonlinear problem** if $\Delta \ll 1$

Unfortunately, we don't have a good a-priori estimate for 'optimal' Δ.

Relaxation method

Given ϕ^n, σ^n ...

1. Use ϕ^n to evaluate the coordinates $\vec{x}$ of Ω_P via L^2-projection:

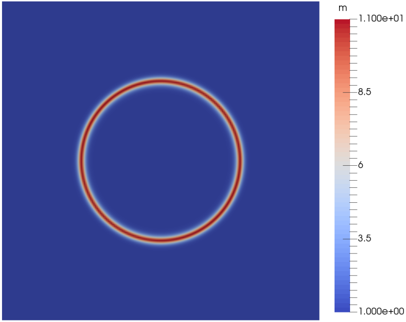
$$\vec{x}(\vec{\xi}) = \vec{\xi} + \Pi_{[P_1/Q_1]^2} \nabla \phi^n(\vec{\xi}).$$

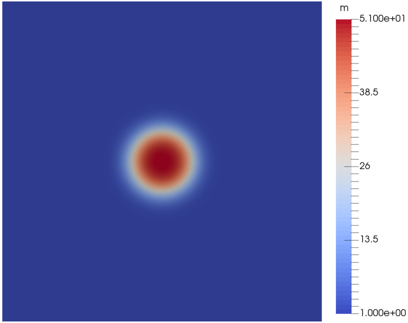
2. Evaluate $m(\vec{x})$ at vertices of Ω_P (assuming analytic expression)
3. Evaluate $\theta^n := \int_{\Omega_C} m \det(I + \sigma^n) \, \mathrm{d}x / \int_{\Omega_C} \mathrm{d}x$
4. Solve Poisson problem to obtain ϕ^{n+1} (CG/GAMG)
5. Solve mass matrix system to obtain σ^{n+1} (CG/ILU)
6. Evaluate termination condition; stop if met.

Plane test cases

$$m(\vec{x}) = 1 + \alpha_1 \operatorname{sech}^2(\alpha_2(|\vec{x} - \vec{x_c}|^2 - a^2))$$

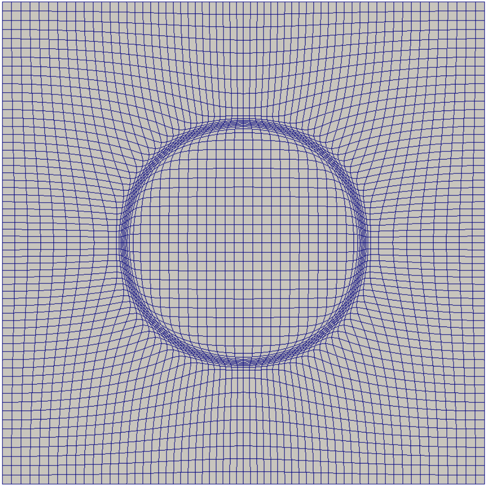
- Ring: $a = 0.25$, $\alpha_1 = 10$, $\alpha_2 = 200$
- Bell: $a = 0$, $\alpha_1 = 50$, $\alpha_2 = 100$





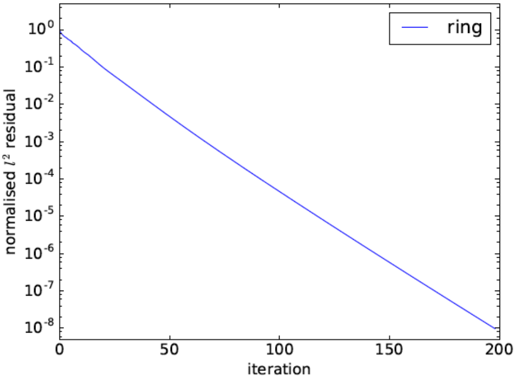
Relaxation method

Final ring mesh:



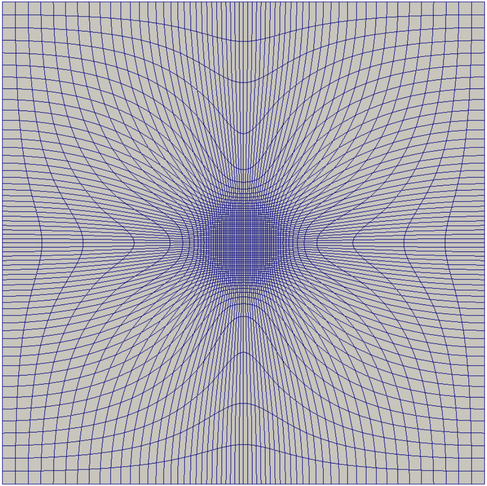
Relaxation method

Convergence for ring test case, $\Delta = 0.1$



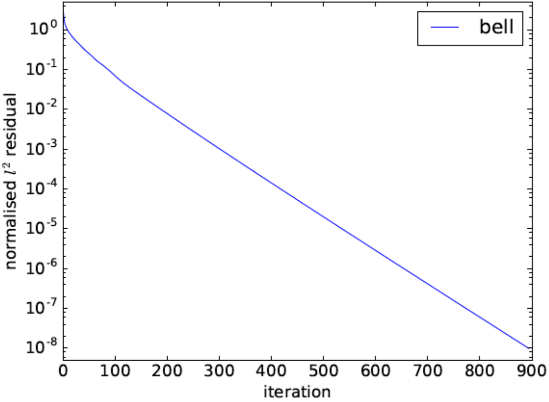
Relaxation method

Final bell mesh:



Relaxation method

Convergence for bell test case, $\Delta = 0.04$



Newton method

Nonlinear residual is

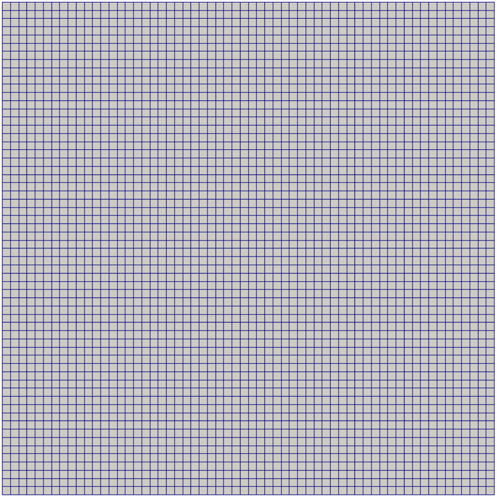
$$R \equiv \langle \tau, \sigma^n \rangle + \langle \nabla \cdot \tau, \nabla \phi^n \rangle - \langle v, m(\vec{x}) \det(I + \sigma^n) - \theta^n \rangle, \quad \forall v \in V_1, \tau \in V_2.$$

Apply a **Newton method** to this.

Full Newton is problematic as Newton steps do not respect convexity

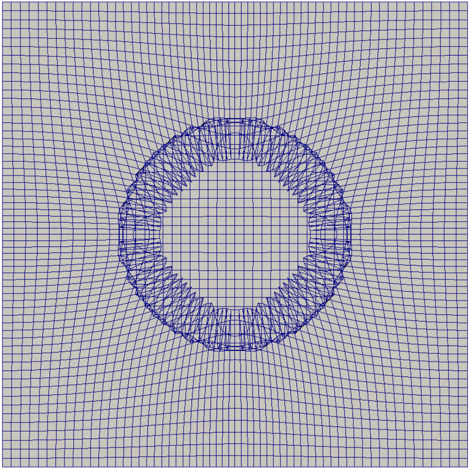
Newton method

Step 0:



Newton method

Step 1:



Quasi-Newton method

Nonlinear residual:]

$$R \equiv \langle \tau, \sigma^n \rangle + \langle \nabla \cdot \tau, \nabla \phi^n \rangle - \langle v, m(\vec{x}) \det(I + \sigma^n) - \theta^n \rangle, \quad \forall v \in V_1, \tau \in V_2.$$

We instead use a **Quasi-Newton method**, with a Jacobian that ignores the dependence of m on ϕ. On the plane, this is

$$\begin{aligned} &\langle \tau, \delta\sigma \rangle + \langle \nabla \cdot \tau, \nabla \delta\phi \rangle \\ &- \langle v, m(\vec{x})(\delta\sigma_{11}(1 + \sigma_{22}^n) + (1 + \sigma_{11}^n)\delta\sigma_{22} - \delta\sigma_{12}\sigma_{21}^n - \sigma_{12}^n\delta\sigma_{21} \rangle. \end{aligned}$$

We omit the $-\langle v, \nabla\delta\phi \cdot \nabla m|_{\vec{x}} \det(I + \sigma^n)\rangle$ term.

Nonlinear solver: l^2-minimisation line search
quite robust but not perfect

Quasi-Newton method*

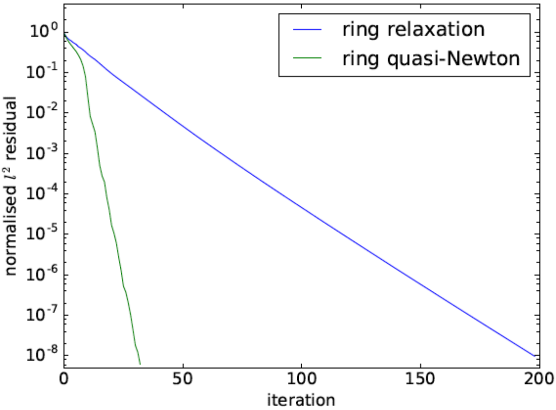
Linear solver: Gory details:

- Preconditioned GMRES on outer system
- 'Riesz map' preconditioner $\langle v, \delta\phi \rangle_{H^1} + \langle \tau, \delta\sigma \rangle_{L^2}$, sufficient for (asymptotically) mesh-independent convergence.
- Outer preconditioner application: block Gauss-Seidel
- Inner solves: PETSc's GAMG for $\delta\phi$ block, ILU for $\delta\sigma$.

Both linear and nonlinear convergence appear to be independent of problem size.

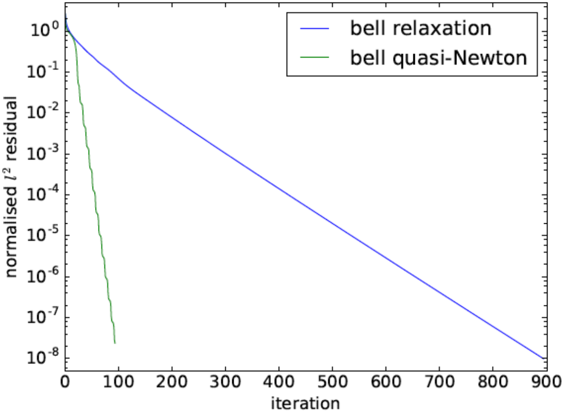
Quasi-Newton method

Comparison of methods on ring test case



Quasi-Newton method

Comparison of methods on bell test case



Numerical solution on the sphere

Now solve the Monge-Ampere-like equation

$$m(\vec{x})\,\det((\nabla \exp(\nabla\phi)\vec{\xi}) \cdot P_\xi + \vec{k}_P \otimes \vec{k}_C) = \theta,$$

Discretise by adapting the mixed finite element methods on the plane.
Set

$$\sigma = \nabla \exp(\nabla\phi)\vec{\xi}.$$

The nonlinear discrete equations are then

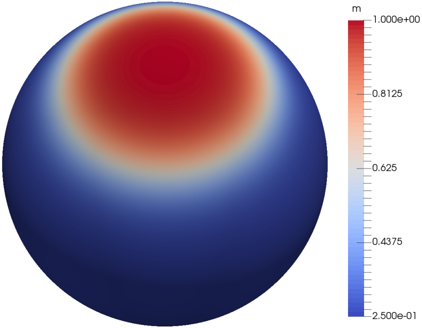
$$\left\langle v, m(\vec{x})\,\det\left(\sigma \cdot P_\xi + \frac{\exp(\nabla\phi)\vec{\xi}}{R} \otimes \frac{\vec{\xi}}{R}\right)\right\rangle = \langle v, \theta\rangle, \quad \forall v \in V_1,$$
$$\langle \tau, \sigma\rangle + \langle \nabla\cdot\tau, \exp(\nabla\phi)\vec{\xi}\,\rangle = 0, \qquad \forall \tau \in V_2.$$

Can solve using relaxation or quasi-Newton methods.

Sphere test cases: 1. Ringler monitor function

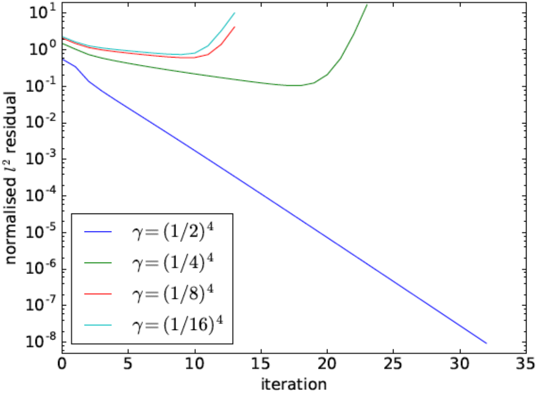
$$m(\vec{x}) = \sqrt{\frac{1-\gamma}{2}\left(\tanh\frac{\beta - \|\vec{x}-\vec{x_c}\|}{\alpha} + 1\right) + \gamma},$$

- $\alpha = \frac{\pi}{20}$, $\beta = \frac{\pi}{6}$, $\gamma = \left(\frac{1}{2}\right)^4, \left(\frac{1}{4}\right)^4, \left(\frac{1}{8}\right)^4, \left(\frac{1}{16}\right)^4$



Sphere: Ringler

'Convergence' of relaxation method on a cubed-sphere mesh



Sphere

We appear to only get convergence for very gentle monitor functions.

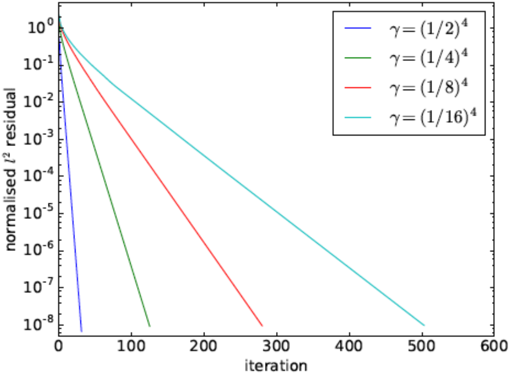
Canadian Breakthrough: use higher-order representation for $\vec{x}$ and $\vec{\xi}$ (not ϕ and σ!)

E.g. one can mesh a sphere using 'flat' triangles. However, in this problem, we don't get convergence for a general monitor function unless each mesh cell is quadratic (or higher).

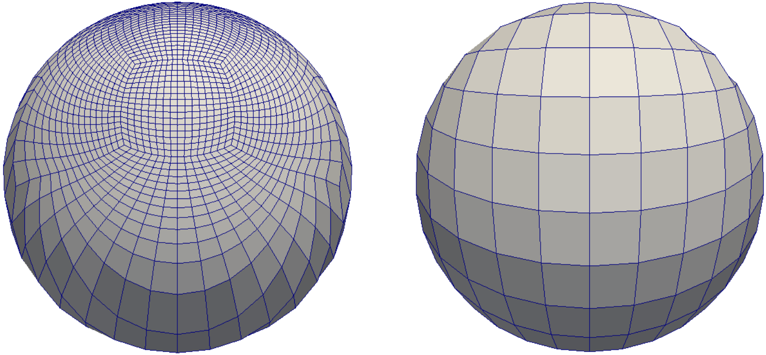
Similarly, on a quadrilateral mesh, need to use a biquadratic representation (or higher) rather than bilinear.

Sphere

Convergence of relaxation method on biquadratic cubed-sphere mesh

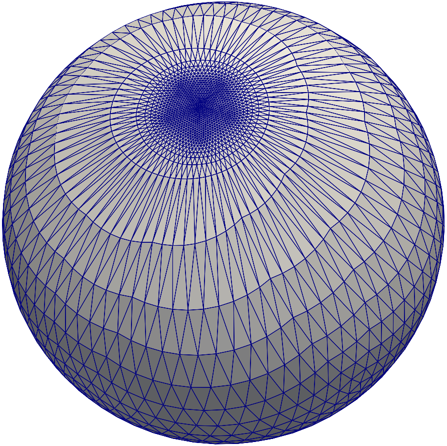


Sphere: Ringler and biquadratic cubed-sphere



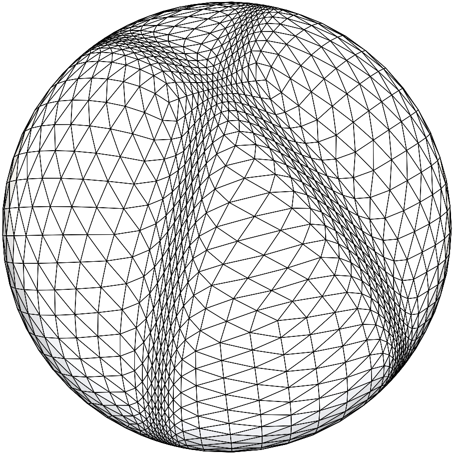
Front/back of mesh, $\gamma = (1/8)^4$

Sphere: 2. Cosine Bell monitor function



Bell shaped monitor function on an Icosahedral mesh

Sphere: 3. Two ring monitor function



Monitor function concentrated in two rings on an icosahedral mesh

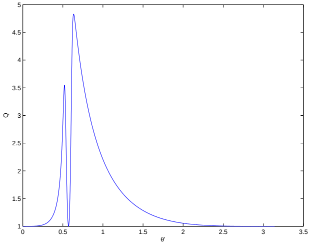
Mesh Regularity

- Mesh regularity follows from the regularity of the solutions of the Monge-Ampere equation and the optimal transport problem
- Studied on the sphere by McCann and Loeper
- Positive curvature of the sphere and lack of a boundary leads to Good Regularity of the MA solutions [Loeper] which is better than the regularity on the plane
- Local regularity of the mesh is related to its scale s and its skewness Q
- If J is the local linearisation of the map with eigenvalues λ_1, λ_2 then

$$s = \lambda_1 \lambda_2, \quad Q = \frac{1}{2}\left[\frac{\lambda_1}{\lambda_2} + \frac{\lambda_2}{\lambda_1}\right].$$

- Uniform mesh $Q = 1$, OT mesh, Q close to one.

Sphere: Skewness



Skewness Q of the Ringler monitor function generated mesh

Implementation

Implemented using Firedrake (`firedrakeproject.org`): software for highly-automated solution of PDEs using FEM. Closely related to FEniCS

```
from firedrake import *

mesh = UnitCubedSphereMesh(refinement_level=4, degree=2)

V1 = FunctionSpace(mesh, "Q", 2)
V2 = TensorFunctionSpace(mesh, "Q", 2)
V = V1*V2

phisigma = Function(V)
phi, sigma = split(phisigma)
xi = Function(mesh.coordinates)
theta = Constant(...)
m = ...

v, tau = TestFunctions(V)
modgphi = sqrt(dot(grad(phi), grad(phi)))
expxi = xi*cos(modgphi) + grad(phi)*sin(modgphi)/modgphi
projxi = Identity(3) - outer(xi, xi)

F = inner(sigma, tau)*dx + dot(div(tau), expxi)*dx - (m*det(outer(expxi, xi) +
     dot(sigma, projxi)) - theta)*v*dx

solve(F == 0, phisigma, solver_parameters={"ksp_type": "gmres",
                                           "pc_type": "fieldsplit",
                                           ...})
```

Implementation

```
static inline void form00_cell_integral_otherwise (double A[9][9] , const double *const restrict
      *restrict coords , const double *const restrict *restrict w_0 , const double *const restrict
      *restrict w_1 , const double *const restrict *restrict w_2 , const double *const restrict *restrict
      w_3 ) {
  static const double t0[5][2] = {{0.953089922969332, 0.046910077030668}, ...
  ...
  for (int ip_0 = 0; ip_0 < 5; ip_0 += 1) {
    double t4 = (((w_1[2][2] * t2[ip_0][0]) + (w_1[5][2] * t2[ip_0][1])) + (w_1[8][2] * t2[ip_0][2]));
    ...
    for (int ip_1 = 0; ip_1 < 5; ip_1 += 1) {
      double t66 = ((t0[ip_1][0] * t65) + (t0[ip_1][1] * t64));
      ...
      for (int k0 = 0; k0 < 3; k0 += 1) {
        for (int k1 = 0; k1 < 3; k1 += 1) {
          t157[k0][k1] = (t2[ip_1][k1] * ((((((((t3[ip_0][k0] * ct79) * ct169) + ((t3[ip_0][k0] * ct79...
        }
      }
      double t158 = ((t70 * t75) + (-1 * (t74 * t71)));
      ...
      for (int k0 = 0; k0 < 3; k0 += 1) {
        for (int k1 = 0; k1 < 3; k1 += 1) {
          ct362[k0][k1] = t157[k0][k1] * ct361;
        }
      }
      for (int j0 = 0; j0 < 3; j0 += 1) {
        for (int j1 = 0; j1 < 3; j1 += 1) {
          double t162 = (t2[ip_0][j0] * t2[ip_1][j1]);
          for (int k0 = 0; k0 < 3; k0 += 1) {
            for (int k1 = 0; k1 < 3; k1 += 1) {
              A[(j0 * 3) + j1][(k0 * 3) + k1] += t162 * ct362[k0][k1];
            }
          }
        }
      }
    }
  }
}
```

Extension: A faster method to generate the mesh

- Improve convergence of Newton solver by using something closer to full Newton.

E.g. add a small fourth-order term to loosen the convexity requirement (Feng & Neilan, 2009).

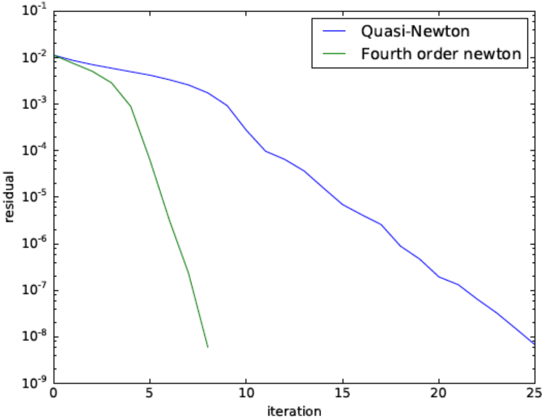
$$-\epsilon \nabla^4 \phi + m(\vec{x}) \det(I + \nabla\nabla\phi) = \theta$$

Sacrifices true equidistribution, but gives much better linear and nonlinear performance.

Preliminary experiments, using FE discretisation of Brenner et al. (2011) for fourth-order term:

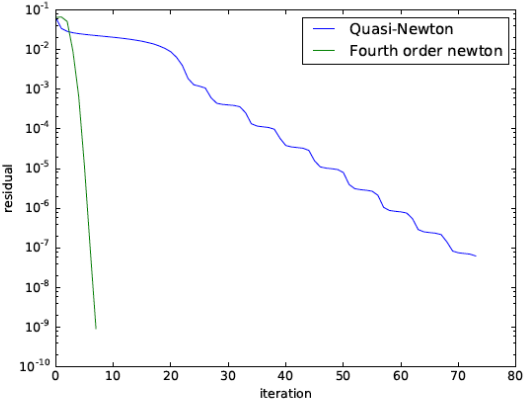
Extension: A faster method to generate the mesh

Ring



Extension: A faster method to generate the mesh

Bell

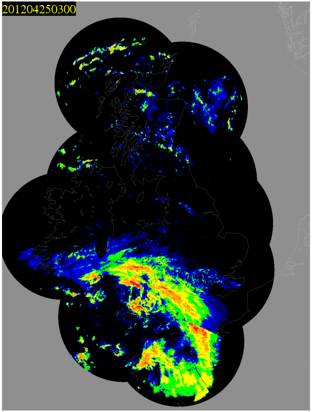


Data assimilation

Eg Operational mesh calculation for meteorological data assimilation

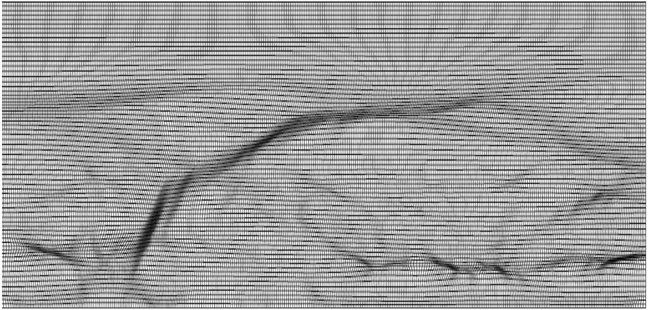
Frontal system:

Rain storm in SW UK



Data assimilation

Take m to be a scaled approximation of the Potential Vorticity of the 3D flow



Coupled to 1d DA procedure [Piccolo, Cullen, Browne]

Solving a PDE on the mesh

- Can couple the moving mesh approach to solve geophysical PDEs

Short term goal: Continuity and Shallow Water Equations.

Preliminary work on advection equation:

$$\frac{\partial q}{\partial t} + (\vec{u} \cdot \nabla) q = 0$$

with scalar q and prescribed $\vec{u}$.

DG approach, SSPRK3 timestepping,

$$m \propto \|\nabla \nabla q\| + const:$$

Solving equations on the moving mesh: Lagrangian approach

So, for the advection equation in the moving frame, mesh velocity $\vec{v}$

$$\frac{\partial q}{\partial t} + ((\vec{u} - \vec{v}) \cdot \nabla) q = 0,$$

we do

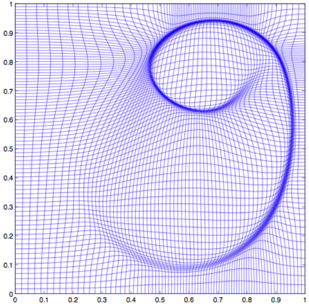
- $\frac{1}{2}\Delta t$ Eulerian continuity $(\vec{u} - \vec{v})$ on old mesh
- Move mesh, and adjust values using $\langle \phi, q \rangle_{\mathrm{new}} = \langle \phi, q \rangle_{\mathrm{old}}$, $\forall \phi \in V$
- $\frac{1}{2}\Delta t$ Eulerian continuity $(\vec{u} - \vec{v})$ on new mesh

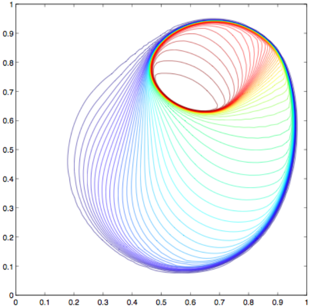
Buckley-Leverett

Eg Buckley-Leverett equation (gas dynamics)

$$u_t = -F_x - G_y + \mu\nabla^2 u, \quad F(u) = u^2/(u^2 + (1-u)^2), \quad G(u) = (1 - 5(1-u)^2)F$$

Solve using simultaneous mesh and solution calculation with m the solution arc-length



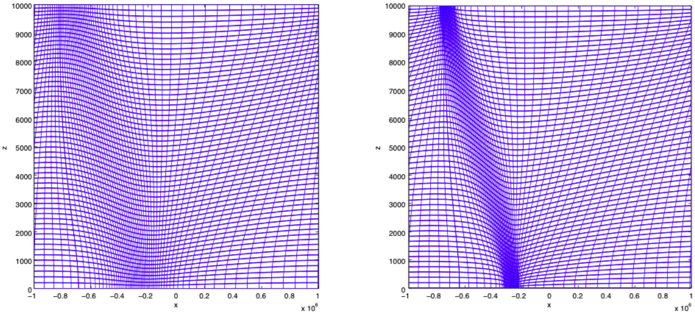


Rezoning: Possibly a more practical method

- Can couple the moving mesh method to much standard software for solving a general time-dependent PDE
- At time level t_k advance the solution the PDE on the current mesh to give solution at time level t_{k+1}. Using a standard software package eg. Discontinuous Galerkin method, Finite element method, Finite volume method (OpenFoam)
- Using the new solution, calculate a new mesh at time level t_{k+1} using the Monge-Ampere based approach
- Interpolate the solution at time level t_{k+1} onto the new mesh.
- Take care to conserve mass (or other desired physical properties) where appropriate. Fine tune the mesh if needed
- **Optional** Repeat the mesh calculation if needed
- Repeat from the top

Eady Equations

Front formation for the Eady Equations of a tropical storm



Potential pitfalls

- By design, we generate small cells, which restricts timestep through CFL for explicit methods ($\implies$ need to control m or use SISL methods or similar)
- Can get $\|\vec{u} - \vec{v}\| \gg \|\vec{u}\|$ in artificial test problems, which restricts Δt further.
- However, Moving mesh can work in our favour – in realistic problems we often get $\vec{v} \approx \vec{u}$ where small cells are present, as the resolution tracks the feature
- Post-processing of m smoothing to control Δx

Summary

- We produce 'optimally-transported' meshes on the plane and sphere
- By solving Monge-Ampere-type PDEs
- This uses a mixed finite element discretisation of the PDEs
- Relaxation approach bulletproof, but takes many iterations
- Quasi-Newton approach robust except for hard m
- Sphere requires higher-order mesh representation for convergence

- Parallel 'just works' (due to Firedrake).
- Coupling to PDEs works so far.

Hilary Weller, Phil Browne, B & Mike Cullen (2016), *Mesh adaptation on the sphere using optimal transport and the numerical solution of a Monge–Ampère type equation*, JCP

Andrew T. T. McRae, Colin J. Cotter, B, (2017) *Optimal-transport-based mesh adaptivity on the plane and sphere using finite elements*, SIAM SISC

Andrew T. T. McRae, Colin J. Cotter, B, (2018) *Scaling and skewness of Optimally-transport-based meshes on the sphere*, JCP .