Multiscale Basis Optimization for Darcy Flow

James M. Rath

work with Todd Arbogast in
the Center for Subsurface Modeling / ICES
at the University of Texas at Austin

Using Newton's method to solve linear systems or: How I learned to stop worrying and love non-linear problems

Joint work with Todd Arbogast @ CSM / ICES / UT-Austin

Problem: solving linear systems

Solving large, sparse linear systems often requires the user of iterative solvers. Storage and speed are the bogeymen.

That's the facts, Jack

Solving large, sparse linear systems often requires the user of iterative solvers. Storage and speed are the bogeymen.

However, generally speaking:

- Solvers get only linear convergence rates toward the solution from the initial guess.
- Large condition numbers mean solvers behave poorly. They require more and more iterations.

What we're after

Solving large, sparse linear systems often requires the user of iterative solvers. Storage and speed are the bogeymen.

However, generally speaking:

- Solvers get only linear convergence rates toward the solution from the initial guess.
- Large condition numbers mean solvers behave poorly. They require more and more iterations.

We want to do better on both these counts.

Dr. Obvious strikes again

Solving linear systems requires nonlinear operations, namely, division.

$$10x = 17 \qquad \Rightarrow \qquad x = \frac{17}{10}$$

Newton's method is da bomb

Solving linear systems requires nonlinear operations, namely, division.

$$10x = 17 \qquad \Rightarrow \qquad x = \frac{17}{10}$$

Nonlinear solvers (Newton's method and its ilk):

- Get fast quadratic convergence to a solution, and
- As applied to discretizations of nonlinear elliptic PDE, are insensitive to mesh size.

Hope beyond hope

Solving linear systems requires nonlinear operations, namely, division.

$$10x = 17 \qquad \Rightarrow \qquad x = \frac{17}{10}$$

Nonlinear solvers (Newton's method and its ilk):

- Get fast quadratic convergence to a solution, and
- As applied to discretizations of nonlinear elliptic PDE, are insensitive to mesh size.

We want to carry over these properties to solving linear systems.

Oopsie!

A naive application of Newton's method to solving a linear system results in a one-step procedure.

Darn!

A naive application of Newton's method to solving a linear system results in a one-step procedure.

Solving:

$$Au = f$$

Objection function:

$$F(u) = f - Au$$

Jacobian:

$$F'(u) = -A$$

Newton step:

$$\begin{aligned} u_{i+1} &= u_i - (-A)^{-1}(f - Au_i) \\ &= u_i + u - u_i \\ &= u \end{aligned}$$

But even worse

To solve your linear system ...

Solving:

$$Au = f$$

Newton step:

$$u_{i+1} = u_i - (-A)^{-1}(f - Au_i)$$

... you must solve your linear system.

But even worse

To solve your linear system ...

Solving:

$$Au = f$$

Newton step:

$$u_{i+1} = u_i - (-A)^{-1}(f - Au_i)$$

... you must solve your linear system.

And that's no fun!

Especially if it's a $10^6 \times 10^6$ sparse, ill-conditioned sytem you want to solve.

If at first you don't succeed ...

To solve your linear system ...

Solving:

$$Au = f$$

Newton step:

$$u_{i+1} = u_i - (-A)^{-1}(f - Au_i)$$

... you must solve your linear system.

We have to try harder to find a nonlinear piece to attack, but it's not obvious where to begin or what will be successful.

Let's examine a 3×3 linear system just to keep things simple.

$$\begin{array}{ccc} A & u & = & f \\ \begin{bmatrix} 10 & -6 & 4 \\ -6 & 17 & 0 \\ 4 & 0 & 9 \end{bmatrix} & \begin{bmatrix} x \\ y \\ z \end{bmatrix} & = & \begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix} \end{array}$$

Let's examine a 3×3 linear system just to keep things simple.

$$\begin{array}{cccc} A & u & = & f \\ \begin{bmatrix} 10 & -6 & 4 \\ -6 & 17 & 0 \\ 4 & 0 & 9 \end{bmatrix} & \begin{bmatrix} \rho\cos\theta\sin\phi \\ \rho\sin\theta\sin\phi \\ \rho\cos\phi \end{bmatrix} & = & \begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix} \end{array}$$

But let's use polar coordinates to represent the unknown.

Let's examine a 3×3 linear system just to keep things simple.

$$A \qquad\qquad U_\sigma \rho \qquad = \quad f$$

$$\begin{bmatrix} 10 & -6 & 4 \\ -6 & 17 & 0 \\ 4 & 0 & 9 \end{bmatrix} \begin{bmatrix} \cos\theta \sin\phi \\ \sin\theta \sin\phi \\ \cos\phi \end{bmatrix} \rho = \begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}$$

But let's use polar coordinates to represent the unknown.
And separate direction (or shape) from magnitude.

Rearranging ...

Let's examine a 3×3 linear system just to keep things simple.

$$\begin{array}{cccc} A & U_\sigma \rho & = & f \end{array}$$

$$\begin{bmatrix} 10 & -6 & 4 \\ -6 & 17 & 0 \\ 4 & 0 & 9 \end{bmatrix} \begin{bmatrix} \cos\theta \sin\phi \\ \sin\theta \sin\phi \\ \cos\phi \end{bmatrix} \rho = \begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}$$

But let's use polar coordinates to represent the unknown.
And separate direction (or shape) from magnitude.

$$\sigma = (\theta, \phi)$$

Objective function:

$$r(\sigma, \rho) = f - AU_{\sigma}\rho$$

Split: some linear, some nonlinear

Objective function:

$$r(\sigma) = f - AU_\sigma \rho(\sigma)$$

Determine ρ as the "best" magnitude for a fixed σ:

$$AU_\sigma \rho = f$$

Split: some linear, some nonlinear

Objective function:

$$r(\sigma) = f - AU_\sigma \rho(\sigma)$$

Determine ρ as the "best" magnitude for a fixed σ:

$$\left(U_\sigma^T A U_\sigma\right) \rho = U_\sigma^T f$$

Where:

- "Best" $=$ best in least-squares sense (in the energy or A-norm).

Split: some linear, some nonlinear

Objective function:

$$r(\sigma) = f - AU_\sigma \rho(\sigma)$$

Determine ρ as the "best" magnitude for a fixed σ:

$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$

Where:

- "Best" $=$ best in least-squares sense (in the energy or A-norm).
- The system $U_\sigma^T A U_\sigma$ is a smaller/coarser linear system to solve.

Algorithm à la Newton

1. Choose a shape σ. (Fix for now.)

Algorithm à la Newton

1. Choose a shape σ.
2. Solve for ρ:
$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$
   This is an "easy" coarsened problem.

Algorithm à la Newton

1. Choose a shape σ.
2. Solve for ρ:

$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$

3. Calculate objective/residual:

$$r(\sigma) = f - AU_\sigma \rho$$

Algorithm à la Newton

1. Choose a shape σ.
2. Solve for ρ:
$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$
3. Calculate objective/residual:
$$r(\sigma) = f - AU_\sigma\rho$$
4. Calculate Jacobian $r'(\sigma)$.

Algorithm à la Newton

1. Choose a shape σ.
2. Solve for ρ:
$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$
3. Calculate objective/residual:
$$r(\sigma) = f - AU_\sigma \rho$$
4. Calculate Jacobian $r'(\sigma)$.
   Oops, oh yeah ...

Algorithm à la Newton

1. Choose a shape σ.
2. Solve for ρ:
$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$
3. Calculate objective/residual:
$$r(\sigma) = f - AU_\sigma\rho$$
4. Calculate Jacobian $r'(\sigma)$.
5. Calculate Newton step:
$$\delta\sigma = -\left(r'\right)^\dagger r$$

Algorithm à la Newton

1. Choose a shape σ.
2. Solve for ρ:
$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$
3. Calculate objective/residual:
$$r(\sigma) = f - AU_\sigma\rho$$
4. Calculate Jacobian $r'(\sigma)$.
5. Calculate Newton step:
$$\delta\sigma = -\left(r'\right)^{\dagger} r$$
6. Update shape σ:
$$\sigma \leftarrow \sigma + \delta\sigma$$

Algorithm à la Newton

1. Choose a shape σ.
2. Solve for ρ:
$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$
3. Calculate objective/residual:
$$r(\sigma) = f - AU_\sigma \rho$$
4. Calculate Jacobian $r'(\sigma)$.
5. Calculate Newton step:
$$\delta\sigma = -\left(r'\right)^\dagger r$$
6. Update shape σ:
$$\sigma \leftarrow \sigma + \delta\sigma$$
7. Repeat as necessary.

Bummer

- Calculating Jacobian $r'(\sigma)$
- Solving the linear system $\left(r'\right)^{\dagger} r$

Bummer

- Calculating Jacobian $r'(\sigma)$
- Solving the linear system $(r')^{\dagger} r$

Calculus ... yuck!

- Calculating Jacobian $r'(\sigma)$
- Solving the linear system $(r')^{\dagger} r$

Jacobians require calculus, and who wants to do calculus?

Linear algebra is my bag, baby

- Calculating Jacobian $r'(\sigma)$
- Solving the linear system $(r')^{\dagger}r$

Jacobians require calculus, and who wants to do calculus?
Blech! I wanna do linear algebra ...

Jacobians are expensive

- Calculating Jacobian $r'(\sigma)$
- Solving the linear system $(r')^{\dagger} r$

Jacobians require calculus, and who wants to do calculus?
Blech! I wanna do linear algebra ...

(Jacobians are expensive to compute, anyway.)

To be lazy, one must do work ...

- Calculating Jacobian $r'(\sigma)$
- Solving the linear system $(r')^\dagger r$

Jacobians require calculus, and who wants to do calculus?
Blech! I wanna do linear algebra ...

We'll use calculus to avoid calculus. (And save the day!)

Chain rule to the rescue!

S'pose instead of computing the Newton step:

$$\delta\sigma = -\left(r'\right)^{\dagger} r$$

Chain rule to the rescue!

S'pose instead of computing the Newton step:

$$\delta\sigma = -\left(r'\right)^{\dagger} r$$

We compute the effect that the Newton step would have on the residual:

$$\delta r = \quad \left(r'\right)\delta\sigma$$

Chain rule to the rescue!

S'pose instead of computing the Newton step:

$$\delta\sigma = -\left(r'\right)^{\dagger} r$$

We compute the effect that the Newton step would have on the residual:

$$\begin{aligned} \delta r &= \quad \left(r'\right) \delta\sigma \\ &= -\left(r'\right)\left(r'\right)^{\dagger} r \end{aligned}$$

A-ha!

S'pose instead of computing the Newton step:

$$\delta\sigma = -\left(r'\right)^{\dagger} r$$

We compute the effect that the Newton step would have on the residual:

$$\begin{aligned} \delta r &= \left(r'\right)\delta\sigma \\ &= -\left(r'\right)\left(r'\right)^{\dagger} r \end{aligned}$$

The operation $\left(r'\right)\left(r'\right)^{\dagger}$ is something familiar: the projection onto the range of r'!

A-ha!

S'pose instead of computing the Newton step:

$$\delta\sigma = -\left(r'\right)^{\dagger} r$$

We compute the effect that the Newton step would have on the residual:

$$\begin{aligned} \delta r &= \quad \left(r'\right)\delta\sigma \\ &= -\left(r'\right)\left(r'\right)^{\dagger} r \end{aligned}$$

The operation $\left(r'\right)\left(r'\right)^{\dagger}$ is something familiar: the projection onto the range of r'!

The range of r' is the tangent space to the manifold of all possible residual vectors. This is an ellipsoid! The normal, it turns out, is easy to compute.

Et voilà!

S'pose instead of computing the Newton step:

$$\delta\sigma = -(r')^{\dagger} r$$

We compute the effect that the Newton step would have on the residual:

$$\begin{aligned} \delta r &= \quad (r')\delta\sigma \\ &= -(r')(r')^{\dagger} r \end{aligned}$$

The operation $(r')(r')^{\dagger}$ is something familiar: the projection onto the range of r'!

The range of r' is the tangent space to the manifold of all possible residual vectors. This is an ellipsoid! The normal, it turns out, is easy to compute.

A projection is a linear algebra sorta thing. And it's a projection onto a low-dimensional space (1-D here). So it's "easy"!

How now brown cow?

S'pose instead of computing the Newton step:

$$\delta\sigma = -\left(r'\right)^{\dagger} r$$

We compute the effect that the Newton step would have on the residual:

$$\begin{aligned}\delta r &= \left(r'\right)\delta\sigma \\ &= -\left(r'\right)\left(r'\right)^{\dagger} r\end{aligned}$$

The operation $\left(r'\right)\left(r'\right)^{\dagger}$ is something familiar: the projection onto the range of r'!

The range of r' is the tangent space to the manifold of all possible residual vectors. This is an ellipsoid! The normal, it turns out, is easy to compute.

A projection is a linear algebra sorta thing. And it's a projection onto a low-dimensional space (1-D here). So it's “easy”!

So how do we use this?

Algorithm v. 2.0

1. Choose a shape σ.

Algorithm v. 2.0

1. Choose a shape σ.
2. Solve coarse problem for ρ:

$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$

Algorithm v. 2.0

1. Choose a shape σ.
2. Solve coarse problem for ρ:

$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$

3. Calculate objective/residual:

$$r(\sigma) = f - AU_\sigma \rho$$

Algorithm v. 2.0

1. Choose a shape σ.
2. Solve coarse problem for ρ:

$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$

3. Calculate objective/residual:

$$r(\sigma) = f - AU_\sigma \rho$$

4. Calculate residual change (Newton step):

$$\delta r = -P_{\tan} r$$

Algorithm v. 2.0

1. Choose a shape σ.
2. Solve coarse problem for ρ:

$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$

3. Calculate objective/residual:

$$r(\sigma) = f - AU_\sigma\rho$$

4. Calculate residual change (Newton step):

$$\delta r = -P_{\mathrm{tan}} r$$

5. Impute new shape from updated residual $r + \delta r$.
   (We leave out these details, but take it on faith that it's easy, too.)

Algorithm v. 2.0

1. Choose a shape σ.
2. Solve coarse problem for ρ:
$$\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$$
3. Calculate objective/residual:
$$r(\sigma) = f - AU_\sigma \rho$$
4. Calculate residual change (Newton step):
$$\delta r = -P_{\tan} r$$
5. Impute new shape from updated residual $r + \delta r$.
6. Repeat as necessary.

You got chocolate in my peanut butter!

We went from a linear problem to a nonlinear one, but ...

What have we *done*?!

We went from a linear problem to a nonlinear one, but ...

We have traded solving a large, ill-conditioned linear problem $Au = f$ for

- solving a much smaller, better conditioned linear problem $\left(U_\sigma^T A U_\sigma\right)\rho = U_\sigma^T f$, and
- solving a small non-linear system (for the shape σ).

An application (the I&A in SIAM)

Steady single-phase flow through a porous medium can be described by:

$$-\nabla \cdot a\nabla p = f$$

This PDE can be discretized in a number of ways. We leave the details of this and our coarsening procedure to another talk.

Challenges

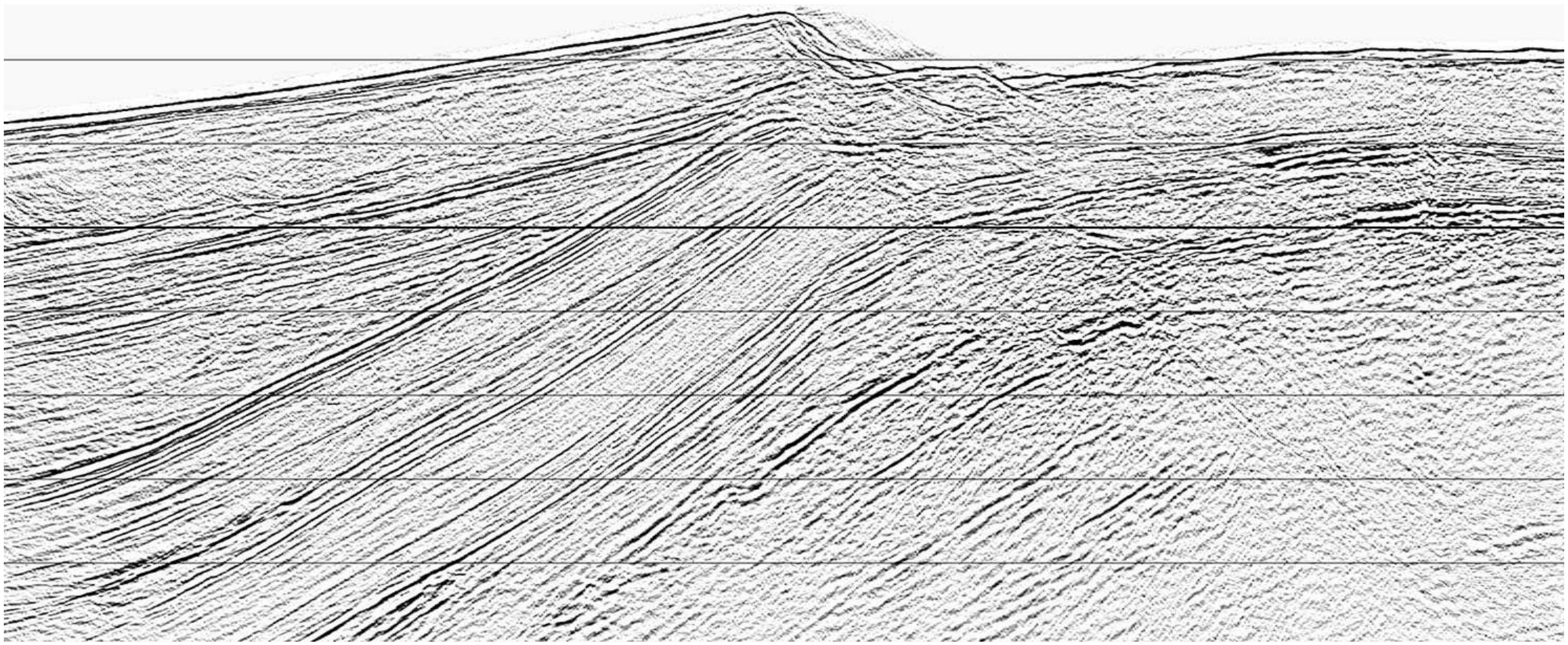
$$-\nabla \cdot a \nabla p = f$$

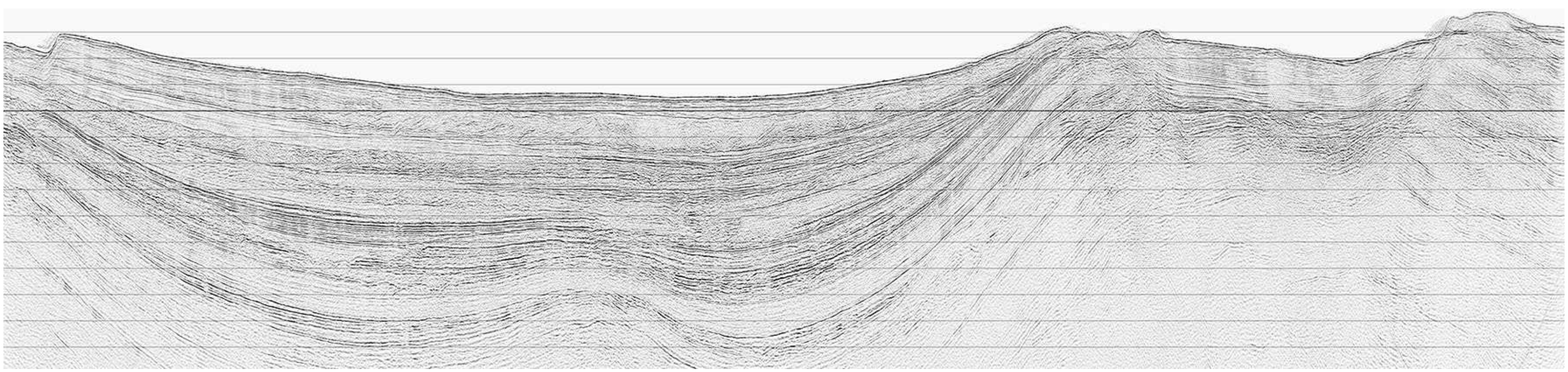
The coefficient a depends on the permeability.

The permeability is often geostatistically generated at high resolution. It can be very heterogeneous.

Together these conditions make for an ill-conditioned and computationally expensive problem.

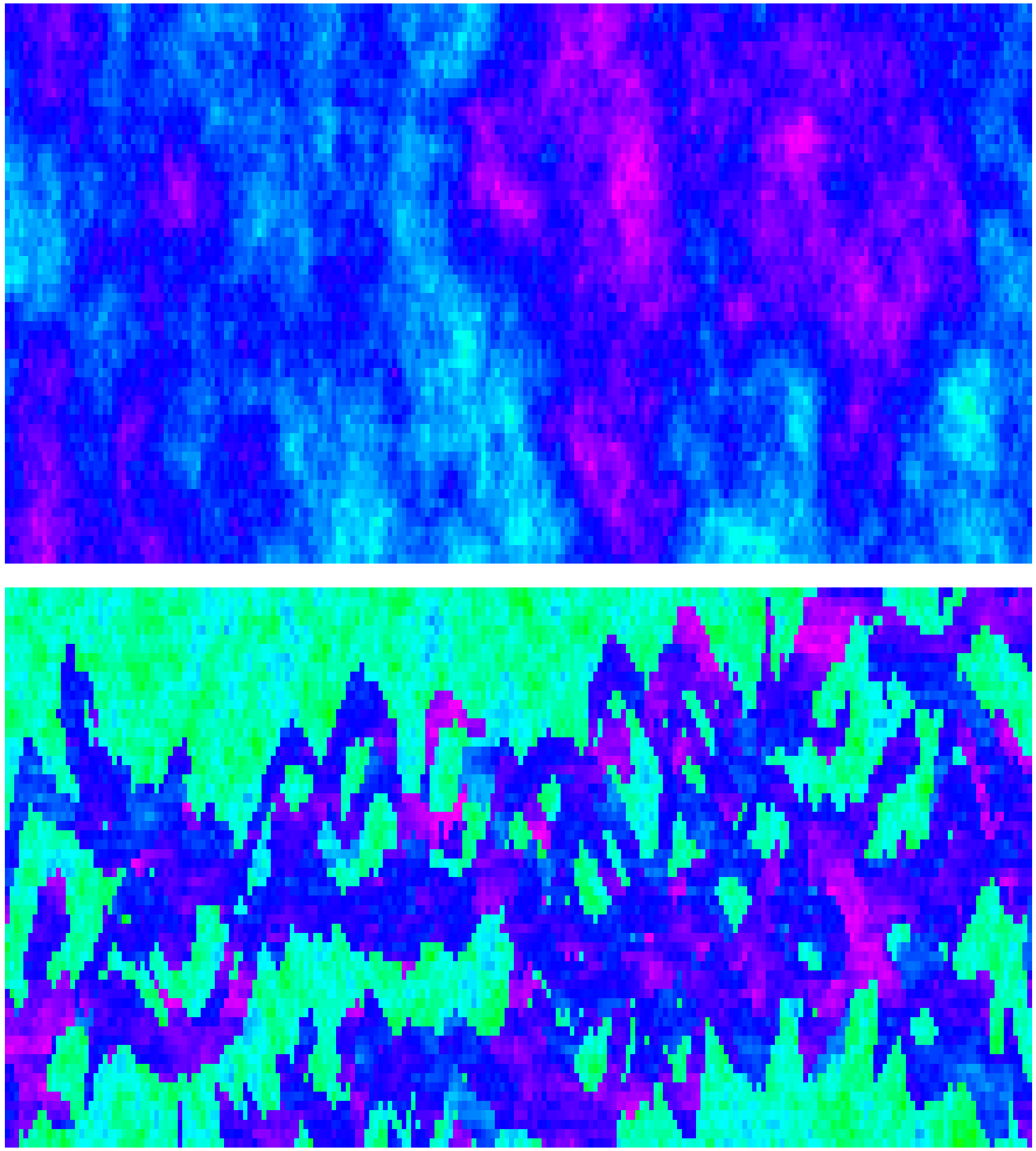
What's “heterogeneous”?





Seismics from a USGS survey in the northern Gulf of Mexico

What's "heterogeneous"?



Simulated fields from the SPE CSP10

Newton's method is da bomb

To reiterate our desired features, Newton's method:

- Gets fast quadratic convergence to a solution, and
- As applied to discretizations of nonlinear elliptic PDE, is insensitive to mesh size.

To reiterate our desired features, Newton's method:

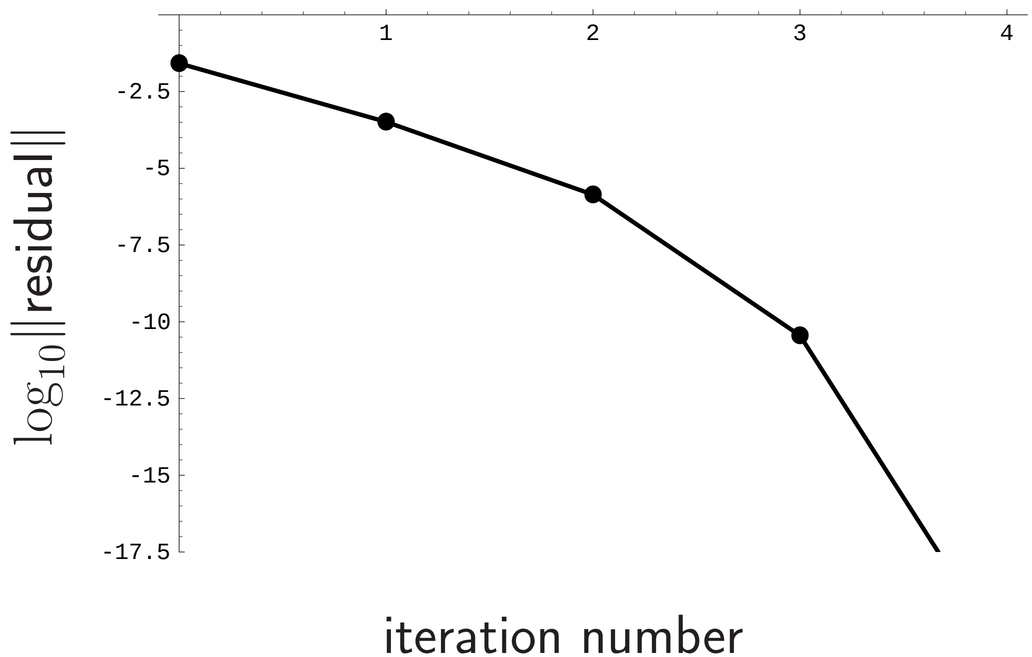
- Gets fast quadratic convergence to a solution, and
- As applied to discretizations of linear elliptic PDE, is insensitive to mesh size.

And a new property:

- Is insensitive to heterogeneity in coefficients (the a in $-\nabla \cdot a\nabla p = f$).

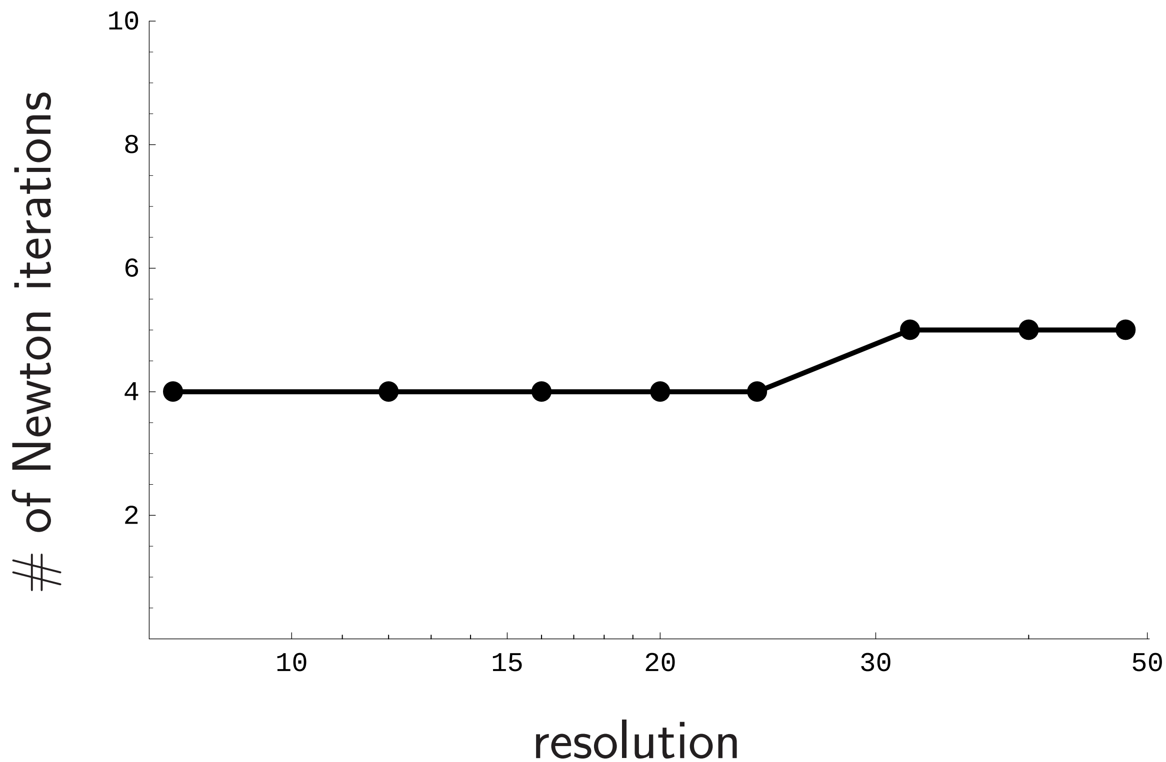
Some easy examples: quadratic convergence

We apply this method to solving the flow problem on the unit square with constant permeability. There are sources at the bottom-left and top-right of the domain (a quarter five-spot) and no gravity. A 2×2 coarse grid is used with a 6×6 subgrid (corresponding to a 12×12 fine grid).



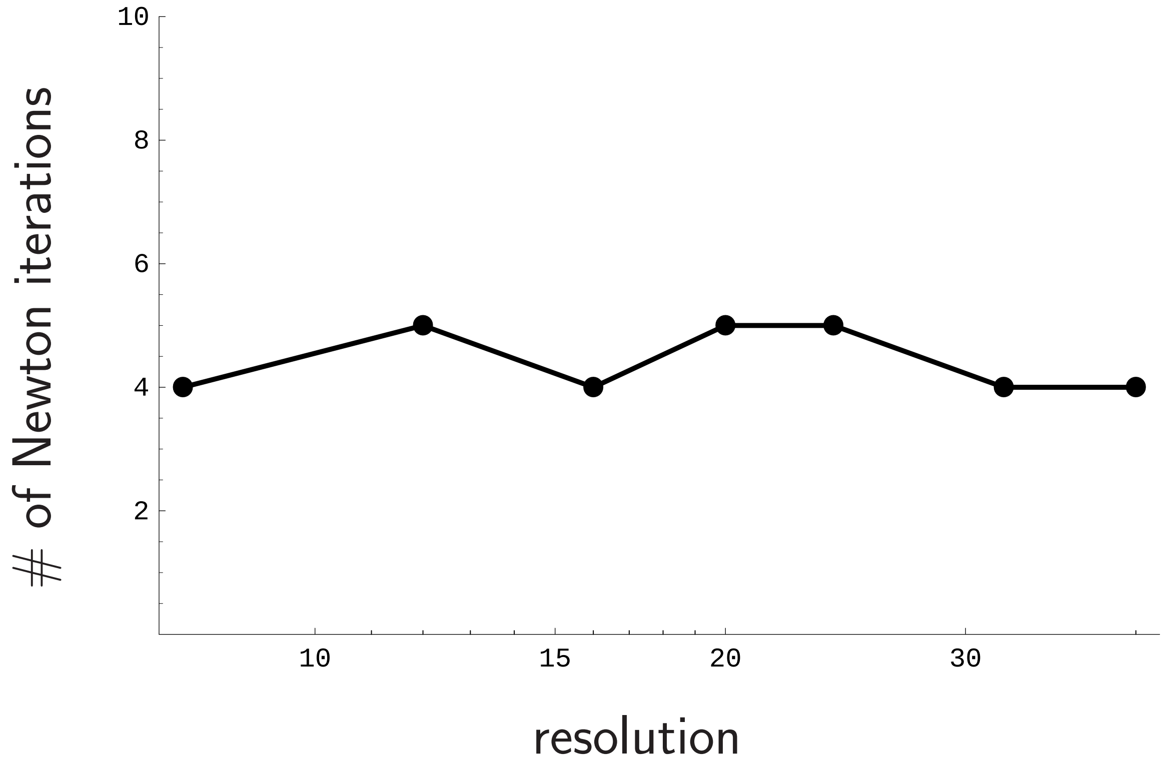
Note the axis scales: we get quadratic convergence — on a linear problem!

Some easy examples: resolution independence I



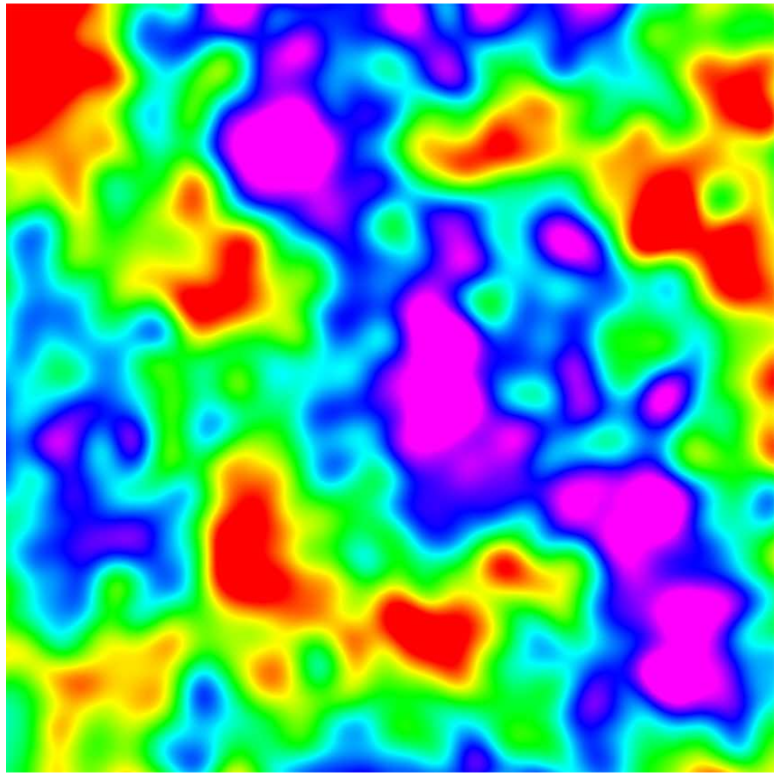
Solve the flow problem with a fixed coarse grid, but let the underlying grid get finer and finer. Only a constant number of Newton iterations regardless of resolution is needed.

Some easy examples: resolution independence II



Solve the flow problem with the coarse grid spacing at a fixed ratio to the fine grid spacing. As the underlying grid gets finer and finer, only a constant number of Newton iterations regardless of resolution is needed.

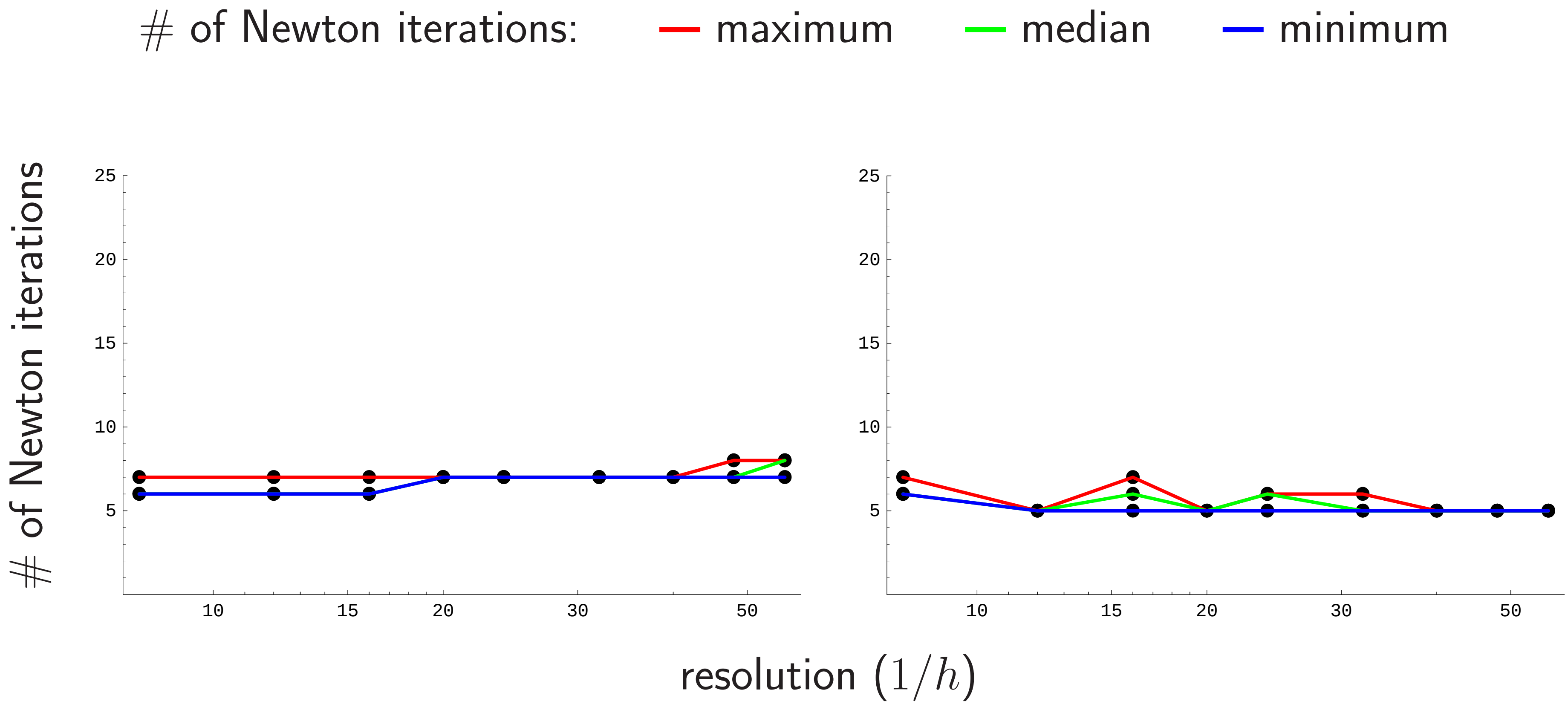
Kick it up a notch: a more heterogeneous permeability



The above graphic plots the variation from a statistically generated permeability field. Red areas indicate low permeability; blue areas indicate high permeability.

The permeabilities span about five orders of magnitude (10^5).

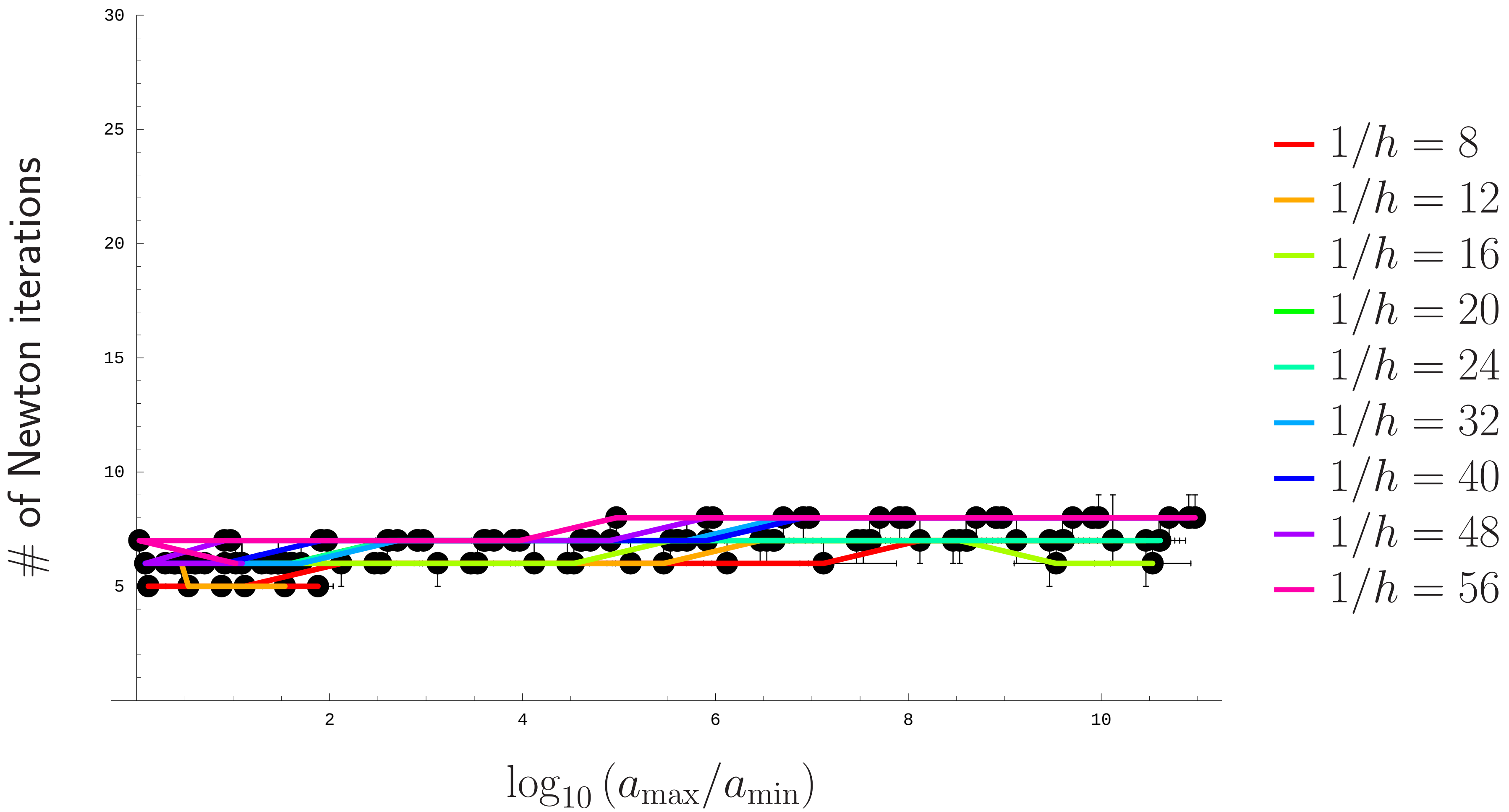
Resolution independence with heterogeneous coefficients



Fixed coarse grid　　　　Fixed coarse/fine ratio

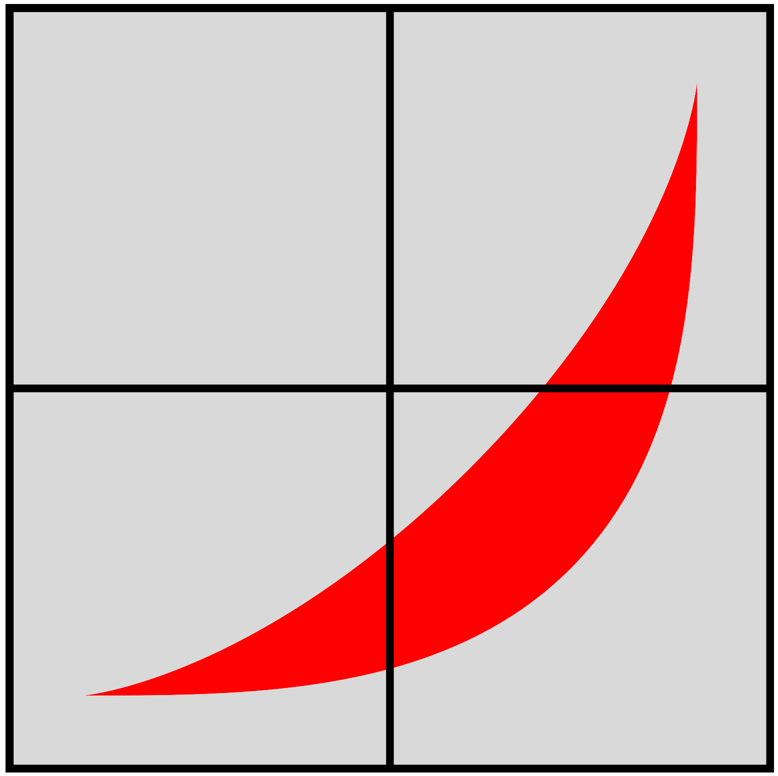
Let the grid get finer and finer (let the resolution increase). At each resolution, grab several statistical subsamples of the permeability field. Roughly a constant number of Newton iterations is needed.

Heterogeneity independence



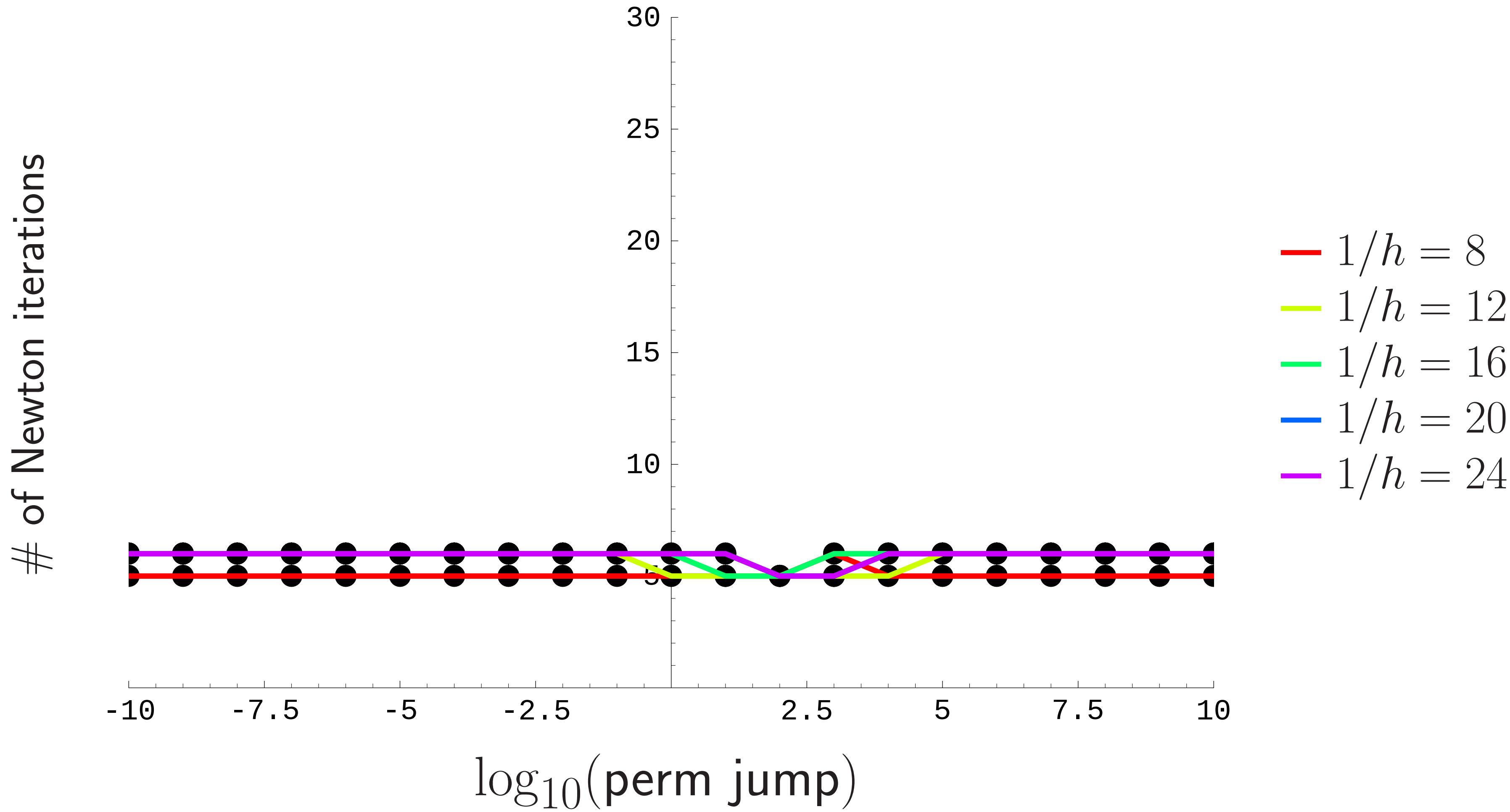
Take a subsample of the heterogeneous permeability field and rescale it so that $a_{\max}/a_{\min}$ gets large.

A channel/barrier permeability field



In the above diagram, gray represents a permeability of 1 and red represents either a high or low permeability. When high, we have a channel; when low, we have a barrier.

Heterogeneity independence for channel/barrier flow



The horizontal axis shows the base-10 log of the permeability of the barrier/channel. Points on the left are for a barrier; points in the center are constant permeability everywhere; points on the right are for a channel.