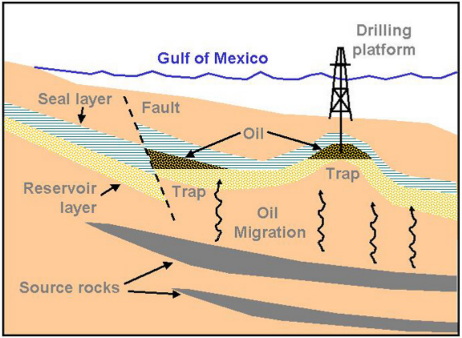
Multi-Multigrid: A Parallelized Multigrid Solver for Python

Tom S Bertalan, Chemical and Biological Engineering;
Faculty Mentor: Roger B Sidje, Mathematics
The University of Alabama

9 April, 2012

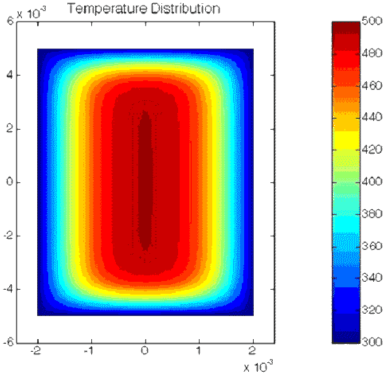
Outline

Petroleum Engineering: pressure driven flow



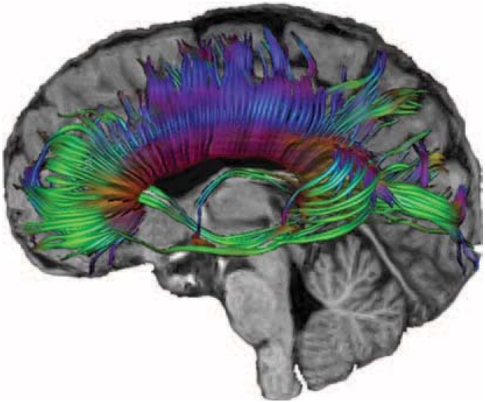
Cross-section of an oil deposit. Photo credit: NOAA, J.Bratton.

Mechanical Engineering: heat transfer



Temperature across a 2D plate in a time-stepping simulation with a Dirichlet boundary condition.

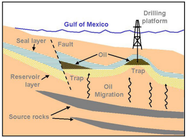
Medical Imaging: anisotropic diffusion



Diffusion Tensor Magnetic Resonance Imaging (DTMRI) uses the diffusion paths of magnetically active particles to map neural tracts in the brain.

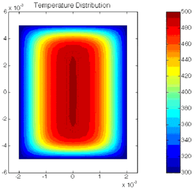
Constituitive Equations

Petroleum Engineering



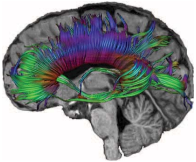
Darcy's Law:
$-\frac{\kappa}{\mu}\nabla P = q$

Mechanical Engineering



Heat Equation:
$\alpha\nabla^2 u = \frac{\partial u}{\partial t}$

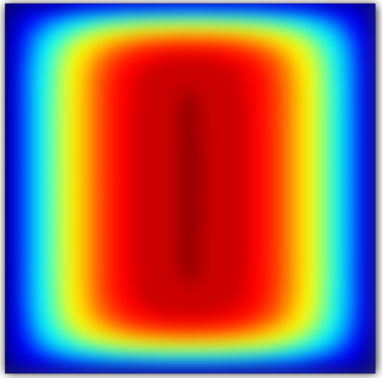
Medical Imaging



Fick's Law:
$\nabla \cdot (D\nabla c) = \frac{\partial c}{\partial t}$

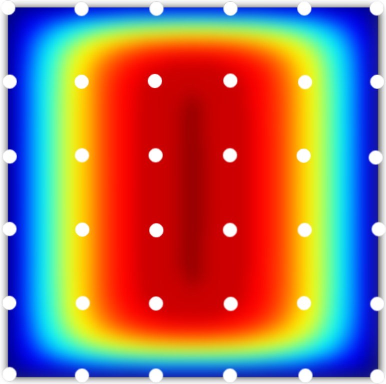
2D Domain: Temperature Profile

A hot conductive square is surrounded by cold, well-mixed fluid.



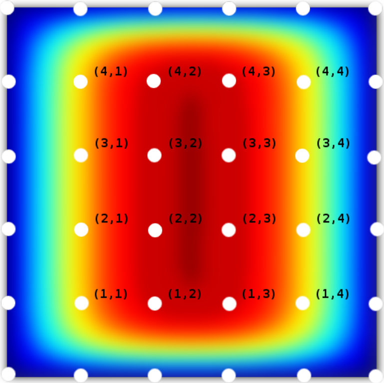
2D Domain: Grid Coordinates

Divide the domain into a discrete grid.



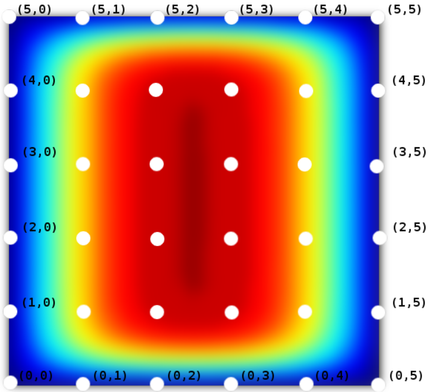
2D Domain: Interior

The set of interior points is called Ω...



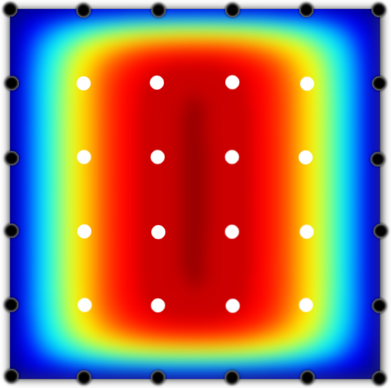
2D Domain: Boundary

... and the set of boundary points is called Γ.



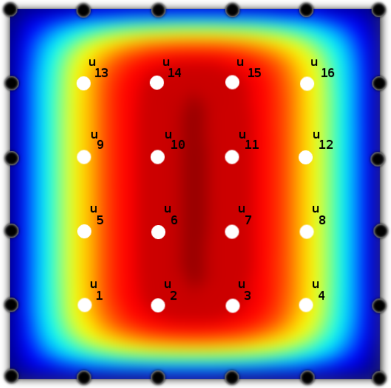
2D Domain: Boundary Condition

The temperature on Γ is known to be that of the fluid.

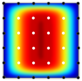


2D Domain: Natural Ordering

The vector u contains all the unknown interior temperatures.



Discrete Equation: Heat Conduction



$$\nabla^2 u = \frac{1}{\alpha}\frac{\partial u}{\partial t} \tag{1}$$

If there are no heat sources or sinks, the divergence of the temperature is zero, that is, $\frac{1}{\alpha}\frac{\partial u}{\partial t} \equiv 0$. In this case, we get the Laplace equation:

$$\nabla^2 u = 0 \tag{2}$$

Laplace Equation

$$\nabla^2 u = 0 \tag{3}$$

∇^2 is the Laplacian operator:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

This is in continuous form, but after discretization of the second partial derivatives, we get

$$Au = b \tag{4}$$

So, somehow we have represented ∇^2 as a matrix A, and included the boundary conditions in b.

Discrete Equation: Partial Derivatives

$$
\begin{array}{cccc}
\cdot & \cdot & u_{i+1,j} & \cdot \\
\cdot & u_{i,j-1} & u_{i,j} & u_{i,j+1} \\
\cdot & \cdot & u_{i-1,j} & \cdot \\
\cdot & \cdot & \cdot & \cdot
\end{array}
$$

Discrete Equation: Partial Derivatives

$\cdot$	$\cdot$	$u_{i+1,j}$	$\cdot$
$\cdot$	$u_{i,j-1}$	$u_{i,j}$	$u_{i,j+1}$
$\cdot$	$\cdot$	$u_{i-1,j}$	$\cdot$
$\cdot$	$\cdot$	$\cdot$	$\cdot$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{\frac{u_{i,j+1}-u_{i,j}}{h} - \frac{u_{i,j}-u_{i,j-1}}{h}}{h} = \left(\frac{1}{h^2}\right)\left(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}\right)$$

Discrete Equation: Partial Derivatives

$\cdot$	$\cdot$	$u_{i+1,j}$	$\cdot$
$\cdot$	$u_{i,j-1}$	$u_{i,j}$	$u_{i,j+1}$
$\cdot$	$\cdot$	$u_{i-1,j}$	$\cdot$
$\cdot$	$\cdot$	$\cdot$	$\cdot$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{\frac{u_{i,j+1}-u_{i,j}}{h} - \frac{u_{i,j}-u_{i,j-1}}{h}}{h} = \left(\frac{1}{h^2}\right)(u_{i,j-1} - 2u_{i,j} + u_{i,j+1})$$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{\frac{u_{i+1,j}-u_{i,j}}{h} - \frac{u_{i,j}-u_{i-1,j}}{h}}{h} = \left(\frac{1}{h^2}\right)(u_{i-1,j} - 2u_{i,j} + u_{i+1,j})$$

Discrete Equation: Partial Derivatives

$$\begin{array}{cccc} \cdot & \cdot & u_{i+1,j} & \cdot \\ \cdot & u_{i,j-1} & u_{i,j} & u_{i,j+1} \\ \cdot & \cdot & u_{i-1,j} & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{array}$$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{\frac{u_{i,j+1}-u_{i,j}}{h} - \frac{u_{i,j}-u_{i,j-1}}{h}}{h} = \left(\frac{1}{h^2}\right)(u_{i,j-1} - 2u_{i,j} + u_{i,j+1})$$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{\frac{u_{i+1,j}-u_{i,j}}{h} - \frac{u_{i,j}-u_{i-1,j}}{h}}{h} = \left(\frac{1}{h^2}\right)(u_{i-1,j} - 2u_{i,j} + u_{i+1,j})$$

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \approx \left(\frac{1}{h^2}\right)(1u_{i-1,j} + 1u_{i,j-1} - 4u_{i,j} + 1u_{i,j+1} + 1u_{i+1,j})$$

Discrete Equation: Matrix Form

The entire matrix equation $Au = b$ then looks like this:

$$
\begin{bmatrix}
-4 & 1 & & 1 & & & & & & \\
1 & -4 & 1 & & 1 & & & & & \\
 & 1 & -4 & 1 & & 1 & & & & \\
\mathbf{1} & & \mathbf{1} & \mathbf{-4} & \mathbf{1} & & \mathbf{1} & & & \\
 & 1 & & 1 & -4 & 1 & & 1 & & \\
 & & \ddots & & \ddots & \ddots & \ddots & & \ddots & \\
 & & & 1 & & 1 & -4 & 1 & & 1 \\
 & & & & 1 & & 1 & -4 & 1 & \\
 & & & & & 1 & & 1 & -4 & 1 \\
 & & & & & & 1 & & 1 & -4
\end{bmatrix}
\cdot
\begin{bmatrix}
u_1 \\ u_2 \\ \\ \cdot \\ \cdot \\ \cdot \\ \\ u_{15} \\ u_{16}
\end{bmatrix}
=
\begin{bmatrix}
b_1 \\ b_2 \\ \\ \cdot \\ \cdot \\ \cdot \\ \\ b_{15} \\ b_{16}
\end{bmatrix}
$$

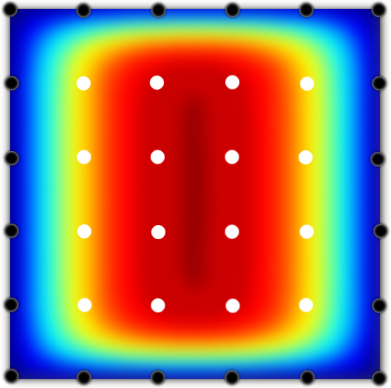
We need to solve for u, the vector of all interior temperatures.

Numerical Methods

- Direct (Gaussian elimination)
  - Advantages: Accurate, finite number of steps.
  - Disadvantages: Slow, not scalable to large problems.
- Iterative (Gauss-Seidel, Jacobi, SOR, etc.)
  - Advantages: Scalable.
  - Disadvantages: Does not always converge.

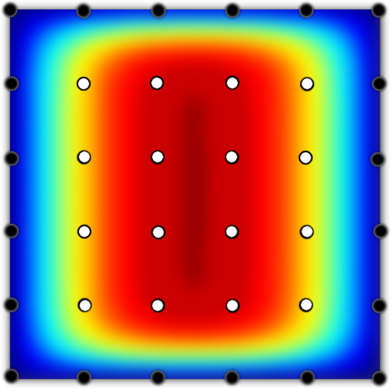
Multigrid: Given

For a Laplace problem, start knowing only the boundary points.



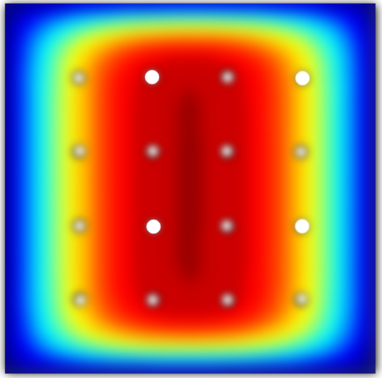
Multigrid: Iterative Approximation

Find u_{approx} via quick iterative solver.
(We now need a correction v s.t. $u_{approx} + v = u$.)



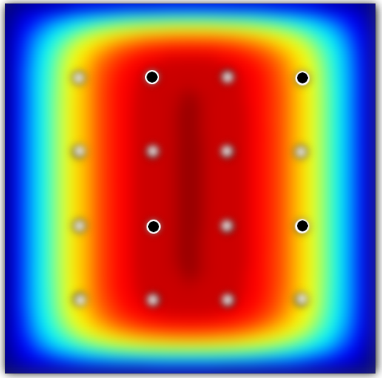
Multigrid: Coarse Points

Define a coarse set, with only $1/4^{th}$ the points of the full domain.



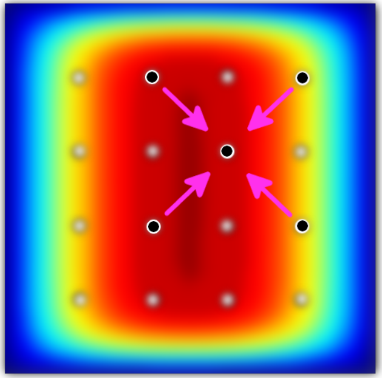
Multigrid: Coarse Correction

Solve for v_{coarse} via accurate direct solver at the coarse resolution.



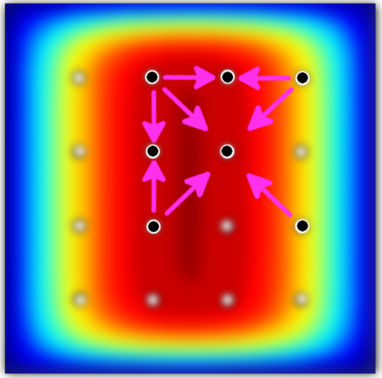
Multigrid: Interpolation to v_{fine}

Interpolate v_{coarse} to v_{fine}.



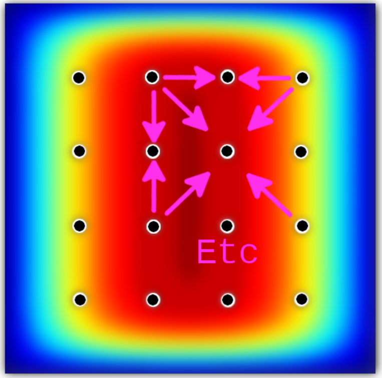
Multigrid: Interpolation to v_{fine}

Interpolate v_{coarse} to v_{fine}.



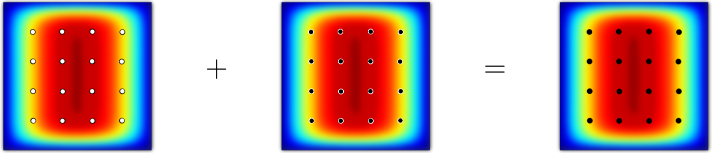
Multigrid: Interpolation to v_{fine}

Interpolate v_{coarse} to v_{fine}.



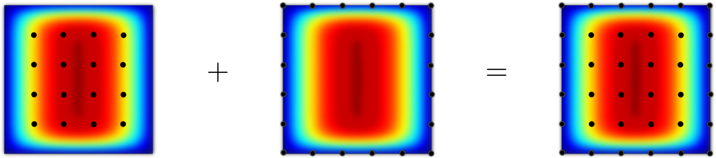
Multigrid: Correction

$$u_{approx} + v_{fine} = u$$

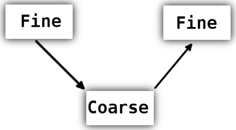


Multigrid: Solution

The full-domain solution is the combination of Ω and Γ points.

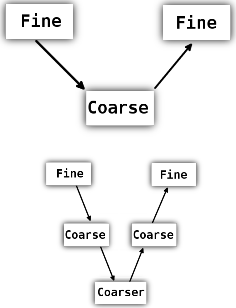


Multigrid: “Multi”



This is a two-grid pattern, with only $N/4$ coarse points for the direct solver.

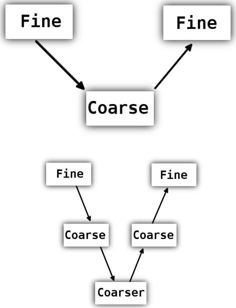
Multigrid: “Multi”



This is a two-grid pattern, with only $N/4$ coarse points for the direct solver.

However, we can also extend to a three-grid pattern, with $N/16$ points in the coarsest set.

Multigrid: "Multi"



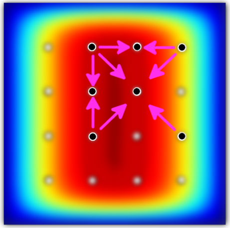
This is a two-grid pattern, with only $N/4$ coarse points for the direct solver.

However, we can also extend to a three-grid pattern, with $N/16$ points in the coarsest set.

By allowing the multigrid cycle function to call itself, recursively, we can extend to arbitrary N-grids.

Future Work: Parallelization

Interpolation, for example, can be parallelized.



Future Work: Parallelization Methods

CPU	(Central Processing Unit)	OpenMP
GPU	(Graphics Processing Unit)	PyCUDA
MPI	Message Passing Interface	OpenMPI
Grid		BOINC; Esimcluster

Acknowledgements

Dr. Roger Sidje	Mathematics
Akand Wahid Islam	Chemical and Biological Engineering
Dr. Eric Carlson	Chemical and Biological Engineering