Multilevel Monte Carlo Solvers for Linear Systems

Stuart Slattery

April 7, 2014

Hardware-Based Motivation

- Modern hardware is moving in two directions (Kogge,2011):
  - Lightweight machines
  - Heterogeneous machines
  - Both characterized by low power and high concurrency
- Some issues:
  - Higher potential for both soft and hard failures (DOE,2012)
  - Memory restrictions are expected with a continued decrease in memory/FLOPS
- Potential resolution from Monte Carlo:
  - Soft failures buried within the tally variance
  - Hard failures mitigated by replication
  - Memory savings over conventional methods

Monte Carlo Methods for Discrete Linear Systems

- First proposed by J. Von Neumann and S.M. Ulam in the 1940's
- Earliest published reference in 1950
- General lack of published work
- Modern work by Evans and others has yielded new applications

Thomas Evans and Scott Mosher, "A Monte Carlo Synthetic Acceleration method for the non-linear, time-dependent diffusion equation", American Nuclear Society - International Conference on Mathematics, Computational Methods and Reactor Physics, 2009.

Monte Carlo Linear Solver Preliminaries

- Split the linear operator

$$\mathbf{Ax} = \mathbf{b} \quad \rightarrow \quad \mathbf{x} = \mathbf{Hx} + \mathbf{b}$$

$$\mathbf{H} = \mathbf{I} - \mathbf{A}$$

- Generate the *Neumann series*

$$\mathbf{A}^{-1} = (\mathbf{I} - \mathbf{H})^{-1} = \sum_{k=0}^{\infty} \mathbf{H}^k$$

- Require $\rho(\mathbf{H}) < 1$ for convergence

$$\mathbf{A}^{-1}\mathbf{b} = \sum_{k=0}^{\infty} \mathbf{H}^k\mathbf{b} = \mathbf{x}$$

Monte Carlo Linear Solver Preliminaries

- Expand the Neumann series

$$x_i = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \dots \sum_{i_k}^{N} h_{i,i_1} h_{i_1,i_2} \dots h_{i_{k-1},i_k} b_{i_k}$$

- Define a sequence of state transitions

$$\nu = i \rightarrow i_1 \rightarrow \cdots \rightarrow i_{k-1} \rightarrow i_k$$

- Use the adjoint Neumann-Ulam decomposition

$$\mathbf{H}^T = \mathbf{P} \circ \mathbf{W}$$

$$p_{ij} = \frac{|h_{ji}|}{\sum_j |h_{ji}|}, \quad w_{ij} = \frac{h_{ji}}{p_{ij}}$$

The Hadamard product $\mathbf{A} = \mathbf{B} \circ \mathbf{C}$ is defined element-wise as $a_{ij} = b_{ij} c_{ij}$.

Evolution of a Solution

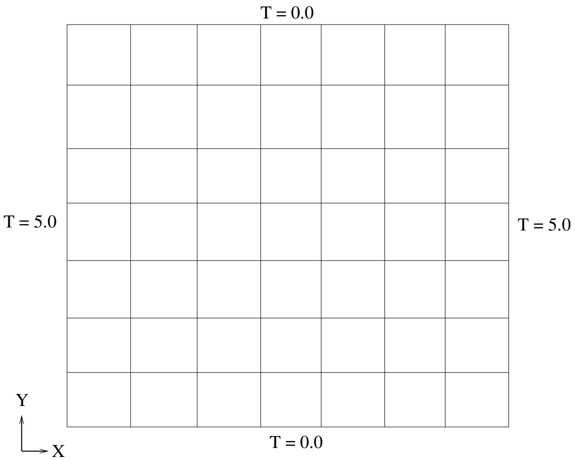

Figure : **Poisson Problem.** *Distributed source of 1.0 in the domain.*

Evolution of a Solution

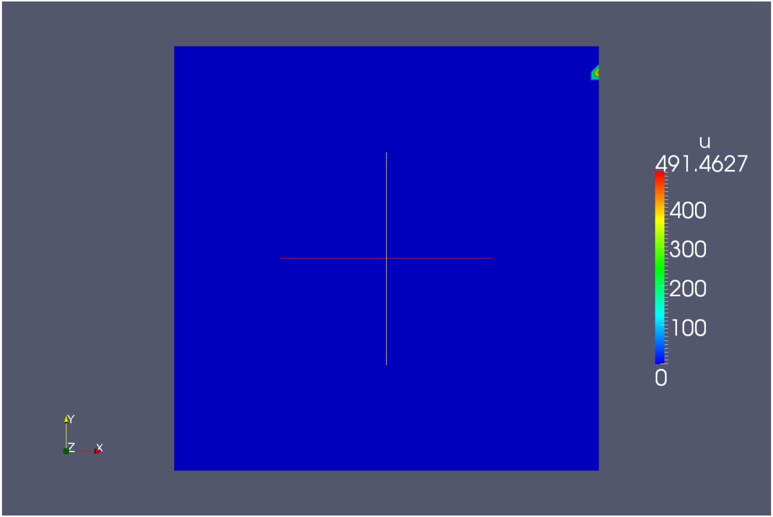

Figure : **Adjoint solution to Poisson Equation.** 1×10^0 *total histories, 0.286 seconds CPU time.*

Evolution of a Solution

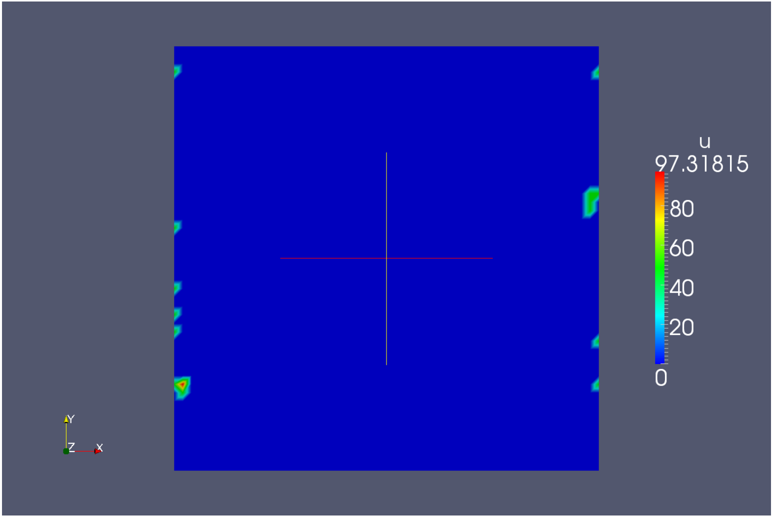

Figure : **Adjoint solution to Poisson Equation.** 1×10^1 *total histories, 0.278 seconds CPU time.*

Evolution of a Solution

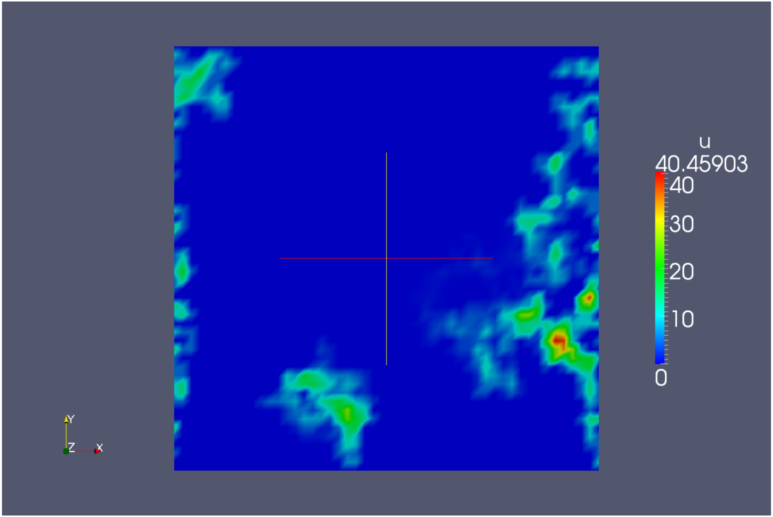

Figure : **Adjoint solution to Poisson Equation.** 1×10^2 *total histories, 0.275 seconds CPU time.*

Evolution of a Solution

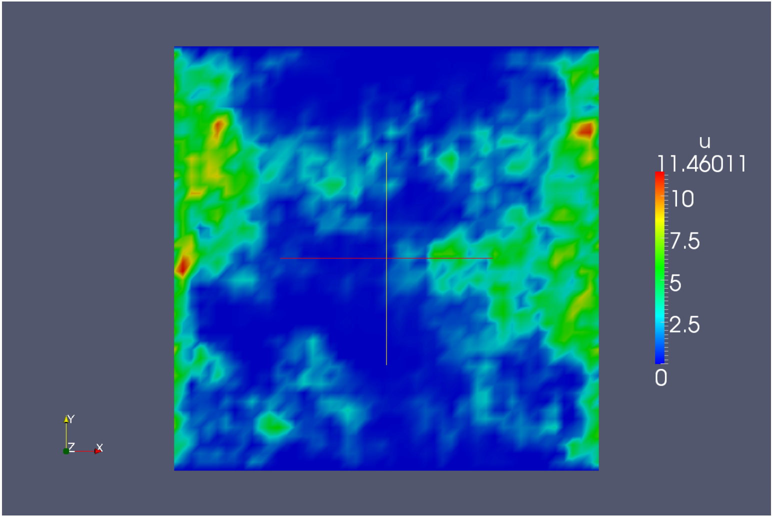

Figure : **Adjoint solution to Poisson Equation.** 1×10^3 *total histories, 0.291 seconds CPU time.*

Evolution of a Solution

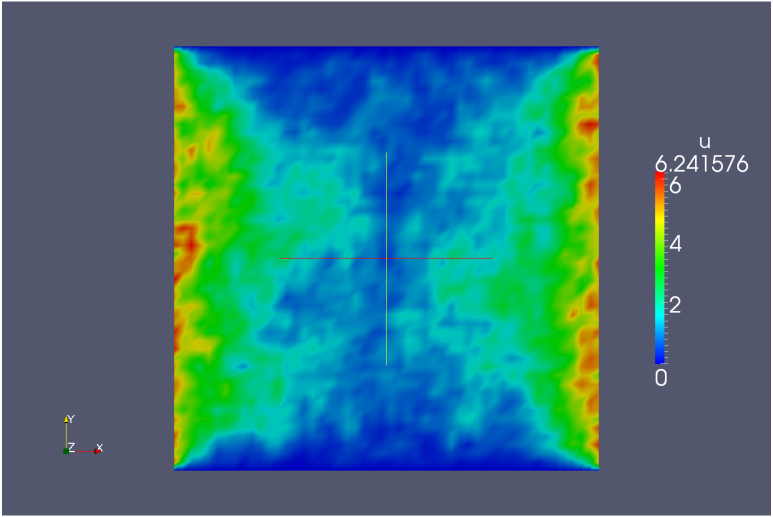

Figure : **Adjoint solution to Poisson Equation.** 1×10^4 *total histories, 0.428 seconds CPU time.*

Evolution of a Solution

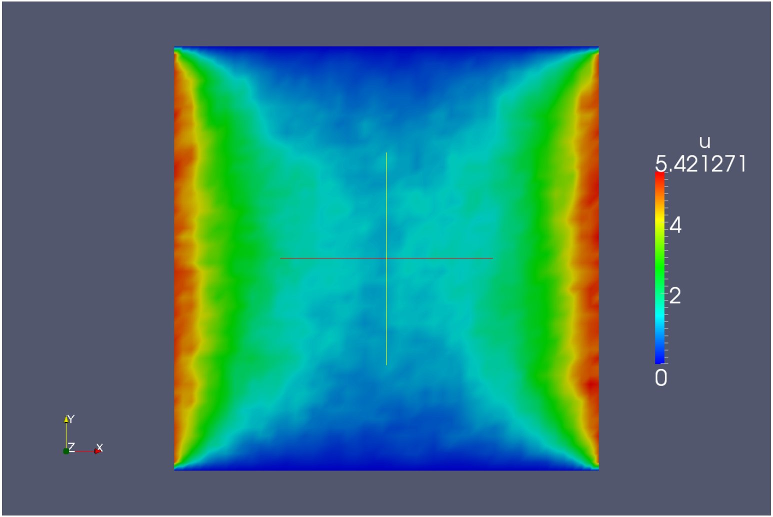

Figure : **Adjoint solution to Poisson Equation.** 1×10^5 *total histories, 1.76 seconds CPU time.*

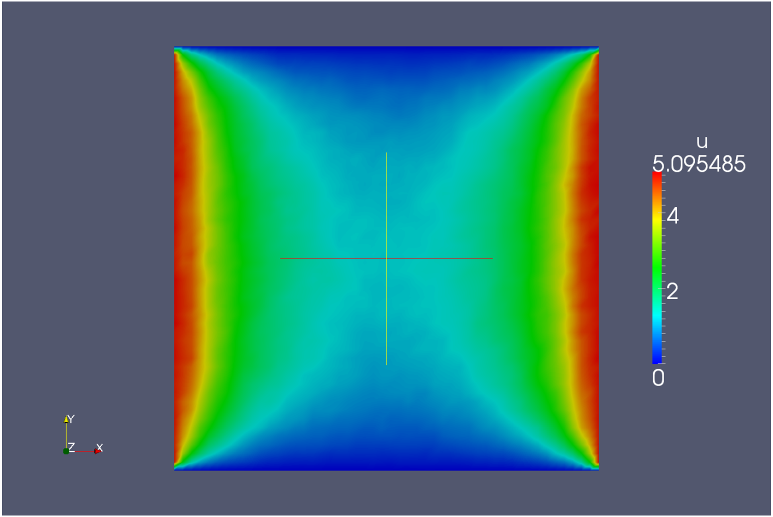

Figure : **Adjoint solution to Poisson Equation.** 1×10^6 *total histories, 15.1 seconds CPU time.*

Evolution of a Solution

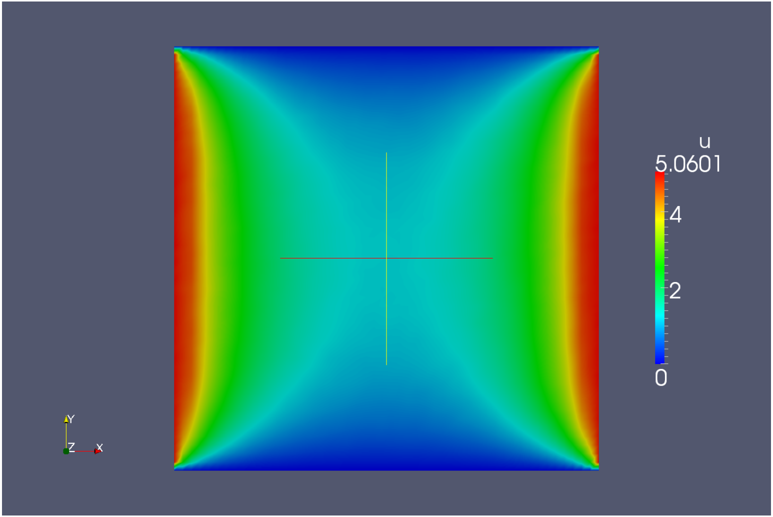

Figure : **Adjoint solution to Poisson Equation.** 1×10^7 *total histories, 149 seconds CPU time.*

Model Problem

Choose a simple homogeneous problem with Dirichlet conditions:

$$\nabla^2 x = 0, \ \ \mathbf{x}_1 = 0, \ \ \ \mathbf{x}_N = 0$$

Second order finite difference:

$$(\nabla \mathbf{u})_i = \frac{\mathbf{u}_{i-1} - 2\mathbf{u}_i + \mathbf{u}_{i+1}}{h^2}$$

Monte Carlo requires $\rho(\mathbf{H})$ so we scale by the diagonal:

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{0}$$

Choose initial guess to be some Fourier mode

$$\mathbf{x}_i^0 = \sin\left(\frac{ik\pi}{N}\right)$$

Error Analysis

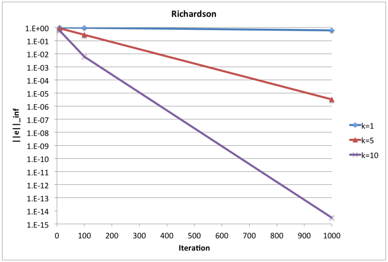

Figure : **Convergence of Richardson's iteration.** *Better for larger wave numbers.*

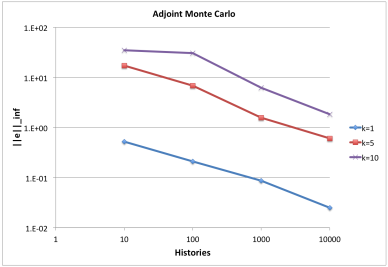

Figure : **Convergence of the adjoint Monte Carlo method.** *Better for smaller wave numbers.*

Error Analysis

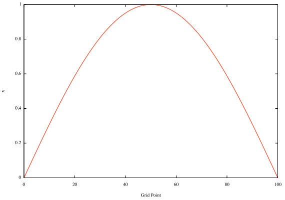

Figure : $k = 1$.

Wave Number	Time per History (s)
1	1
5	0.85
10	0.83

Table : **Normalized average CPU time per history.**

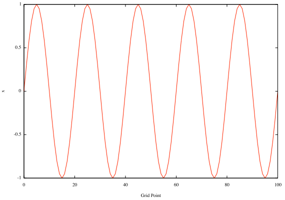

Figure : $k = 10$.

- $\sigma(A)$ dictates the characteristics of the Markov chain
- N dictates the convergence of the Monte Carlo

- Formalized by Heinrich for integral equations in 2001 and by Giles in 2008 for finance calculations
- Recent work includes Bayesian inference techniques for stochastic PDEs in ground water flow

Multilevel Expectation

We start first with the standard Monte Carlo estimator for the solution vector:

$$\hat{\mathbf{x}} = \frac{1}{N} \sum_{m=1}^{N} x^m$$

Consider L levels with level 0 the finest L the coarsest:

$$E(\mathbf{x}_0) = E(\mathbf{x}_L) + E(\mathbf{x}_{L-1} - \mathbf{x}_L) + E(\mathbf{x}_{L-2} - \mathbf{x}_{L-1}) + \cdots + E(\mathbf{x}_0 - \mathbf{x}_1)$$

Reduce to a sum:

$$\hat{\mathbf{y}}_l = \frac{1}{N_l} \sum_{m=1}^{N_l} (x_l^m - x_{l+1}^m)$$

Multilevel Expectation

Build a correction estimator for a given level l:

$$\hat{\mathbf{y}}_l = \frac{1}{N_l} \sum_{m=1}^{N_l} (x_l^m - x_{l+1}^m)$$

Leaving a final multilevel estimator of:

$$\hat{\mathbf{x}} = \sum_{l=0}^{L} \hat{\mathbf{y}}_l$$

Critical observation: x_l^m and x_{l+1}^m must be constructed from the *same* Markov chain

Number of samples at each level should be determined from the estimated variance. For simplicity:

$$N_l = M^{-3(L-l)/2} N$$

Define a *prolongation operator*, $\mathbf{P}_l$, which maps a vector defined on grid $l+1$ to a vector defined on grid l and a *restriction operator*, $\mathbf{R}_l$, which maps a vector defined on grid l to a vector defined on grid $l+1$

$$E(\mathbf{x}_l - \mathbf{x}_{l+1}) = \left(\mathbf{I} - \mathbf{P}_l \mathbf{R}_l\right) \hat{\mathbf{x}}_l$$

Multilevel Monte Carlo Solver

Algorithm 1 Multilevel Monte Carlo Method

for l = 0...L **do**
    $\mathbf{P}_l = P(\mathbf{A}_l)$ {Build the prolongation and restriction operators for the l^{th} level.}
    $\mathbf{R}_l = c\mathbf{P}_l^T$
    $\mathbf{r}_l = \mathbf{b}_l - \mathbf{A}_l\mathbf{x}_l^0$ {Build the l^{th} level residual.}
    $\mathbf{d}_l = \hat{\mathbf{A}}_l^{-1}\mathbf{r}_l$ {Solve the l^{th} level problem with adjoint Monte Carlo}
    if l != L **then**
        $\mathbf{d}_l = (\mathbf{I} - \mathbf{P}_l\mathbf{R}_l)\mathbf{d}_l$ {Apply the multilevel tally}
        $\mathbf{A}_{l+1} = \mathbf{R}_l\mathbf{A}_l\mathbf{P}_l$ {Construct the next level.}
        $\mathbf{x}_{l+1}^0 = \mathbf{R}_l\mathbf{x}_l^0$
        $\mathbf{b}_{l+1} = \mathbf{R}_l\mathbf{b}_l$
    end if
end for
for l = L...1 **do**
    $\mathbf{d}_{l-1} = (\mathbf{I} + \mathbf{P}_l)\mathbf{d}_l$ {Collapse the tallies to the finest grid}
end for
$\mathbf{x} = \mathbf{x}^0 + \mathbf{d}_0$

Numerical Experiments

Geometric Multigrid Example

Geometric Multigrid Example

Algebraic Multigrid Example

Algebraic Multigrid Example

Summary