# Massively Parallel Monte Carlo Methods for Discrete Linear and Nonlinear Systems

Stuart R. Slattery
Engineering Physics Department
University of Wisconsin - Madison

October 28, 2012

# Introduction

- Predictive modeling and simulation enhances engineering capability
- Modern work focused on this task leverages multiple physics simulation (CASL, NEAMS)
- New hardware drives algorithm development (petascale and exascale)
- Monte Carlo methods have the potential to provide great improvements that permit finer simulations and better mapping to future hardware
- A set of massively parallel Monte Carlo methods is proposed to advance multiple physics simulation on contemporary and future leadership class machines

# Physics-Based Motivation

# Physics-Based Motivation

**Predictive nuclear reactor analysis enables...**

- Tighter design tolerance for improved thermal performance and efficiency
- Higher fuel burn-up
- High confidence in accident scenario models

# Physics-Based Motivation

**Predictive nuclear reactor analysis enables...**

- Tighter design tolerance for improved thermal performance and efficiency
- Higher fuel burn-up
- High confidence in accident scenario models

**Multiple physics simulations are complicated...**

- Neutronics, thermal hydraulics, computational fluid dynamics, structural mechanics, and many other physics
- Consistent models yield nonlinearities in the variables through feedback effects
- Tremendous computational resources are required with $O(1 \times 10^9)$ element meshes and $O(100,000)+$ cores used in today's simulations.

# Physics-Based Motivation: DNB

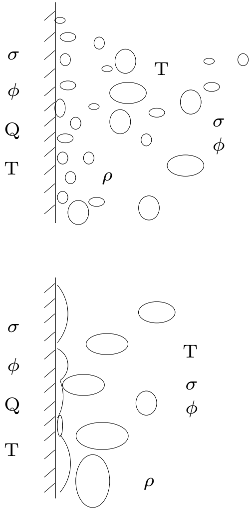


Figure: **Departure from nucleate boiling scenario.**

# Physics-Based Motivation: DNB

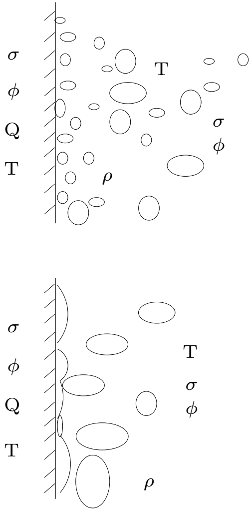


Figure: **Departure from nucleate boiling scenario.**

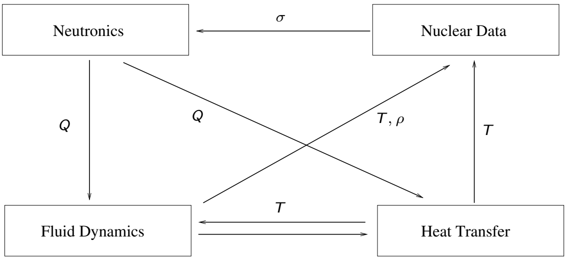


Figure: **Multiphysics dependency analysis of departure from nucleate boiling.**

# Hardware-Based Motivation

- Modern hardware is moving in two directions:
  - Lightweight machines
  - Heterogeneous machines
  - Both characterized by low power and high concurrency
- Some issues:
  - Higher potential for both soft and hard failures
  - 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 are high variance events
  - Memory savings over conventional methods

# Research Outline

- Parallelization of Monte Carlo methods for discrete systems
  - Parallel strategies taken from modern reactor physics methods
  - Research is required to explore varying parallel strategies
  - Scalability is of concern
- Development of a nonlinear solver leveraging Monte Carlo
  - Application to nonlinear problems of interest
  - Memory benefits
  - Performance benefits

# Linear Operator Equations

- We seek solutions of the general linear operator equation

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

$$\mathbf{A} \in \mathbb{R}^{N \times N},\ \mathbf{A} : \mathbb{R}^N \to \mathbb{R}^N,\ \mathbf{x} \in \mathbb{R}^N,\ \mathbf{b} \in \mathbb{R}^N$$

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$$

- $\mathbf{r} = \mathbf{0}$ when an exact solution is found.

# Linear Operator Equations

- We seek solutions of the general linear operator equation

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

$$\mathbf{A} \in \mathbb{R}^{N \times N},\ \mathbf{A} : \mathbb{R}^N \to \mathbb{R}^N,\ \mathbf{x} \in \mathbb{R}^N,\ \mathbf{b} \in \mathbb{R}^N$$

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$$

- $\mathbf{r} = \mathbf{0}$ when an exact solution is found.

**A Requirement**

Assert that $\mathbf{A}$ is *nonsingular*. The solution is then:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

# Stationary Methods

- General stationary methods are formed by splitting the linear operator

$$\mathbf{A} = \mathbf{M} - \mathbf{N}\,.$$

$$\mathbf{x} = \mathbf{M}^{-1}\mathbf{N}\mathbf{x} + \mathbf{M}^{-1}\mathbf{b}\,.$$

- We identify $\mathbf{H} = \mathbf{M}^{-1}\mathbf{N}$ as the *iteration matrix*

$$\mathbf{x}^{k+1} = \mathbf{H}\mathbf{x}^{k} + \mathbf{c}\,.$$

# Stationary Methods Convergence

- The qualities of the iteration matrix dictate convergence
- Define $\mathbf{e}^k = \mathbf{x}^k - \mathbf{x}$ as the error at the $k^{th}$ iterate

$$\mathbf{e}^{k+1} = \mathbf{H}\mathbf{e}^k$$

- We diagonalize $\mathbf{H}$ to extract its Eigenvalues

$$||\mathbf{e}^k||_2 = \rho(\mathbf{H})^k ||\mathbf{e}^0||_2 \ ,$$

- We bound $\mathbf{H}$ by $\rho(\mathbf{H}) < 1$ for convergence

# Projection Methods

- Powerful class of iterative methods
- Provides theory that encapsulates most other iterative methods
- Leveraged in many modern physics codes at the petascale

# Projection Methods

- Powerful class of iterative methods
- Provides theory that encapsulates most other iterative methods
- Leveraged in many modern physics codes at the petascale

## Search Subspace $\mathcal{K}$

Extract the solution from the search subspace:

$$\tilde{\mathbf{x}} = \mathbf{x}_0 + \boldsymbol{\delta}, \;\; \boldsymbol{\delta} \in \mathcal{K}$$

# Projection Methods

- Powerful class of iterative methods
- Provides theory that encapsulates most other iterative methods
- Leveraged in many modern physics codes at the petascale

### Search Subspace $\mathcal{K}$

Extract the solution from the search subspace:

$$\tilde{\mathbf{x}} = \mathbf{x}_0 + \boldsymbol{\delta}, \ \boldsymbol{\delta} \in \mathcal{K}$$

### Constraint Subspace $\mathcal{L}$

Constrain the extraction with the constraint subspace by asserting orthogonality with the residual:

$$\langle \tilde{\mathbf{r}}, \mathbf{w} \rangle = 0, \ \forall \mathbf{w} \in \mathcal{L}$$

# The Orthogonality Constraint

$$\tilde{\mathbf{r}} = \mathbf{r}_0 - \mathbf{A}\delta$$

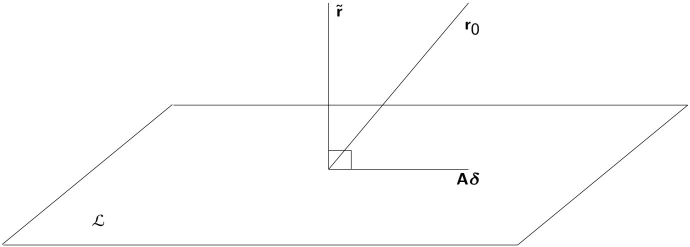


Figure: **Orthogonality constraint of the new residual with respect to $\mathcal{L}$.**

# The Orthogonality Constraint

$$\tilde{\mathbf{r}} = \mathbf{r}_0 - \mathbf{A}\delta$$

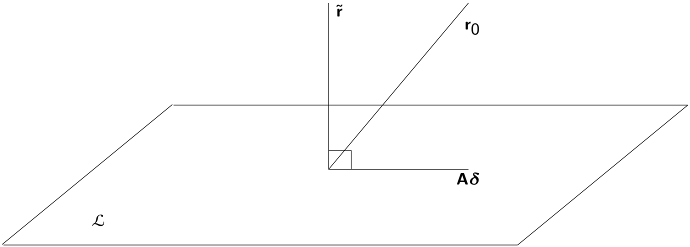


Figure: **Orthogonality constraint of the new residual with respect to $\mathcal{L}$.**

## Minimization Property

The residual of the system will always be *minimized* with respect to the constraints

$$||\tilde{\mathbf{r}}||_2 \leq ||\mathbf{r}_0||_2, \ \forall \mathbf{r}_0 \in \mathbb{R}^N ,$$

# Putting it All Together

- Choose $\mathbf{V}$ as a basis of $\mathcal{K}$ and $\mathbf{W}$ as a basis of $\mathcal{L}$

$$\boldsymbol{\delta} = \mathbf{V}\mathbf{y}, \ \forall \mathbf{y} \in \mathbb{R}^N$$

$$\mathbf{y} = (\mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T\mathbf{r}_0$$

# Putting it All Together

- Choose $\mathbf{V}$ as a basis of $\mathcal{K}$ and $\mathbf{W}$ as a basis of $\mathcal{L}$

$$\boldsymbol{\delta} = \mathbf{V}\mathbf{y}, \ \forall \mathbf{y} \in \mathbb{R}^N$$

$$\mathbf{y} = (\mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T\mathbf{r}_0$$

## Projection Method Iteration

$$\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k$$

$$\mathbf{y}^k = (\mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T\mathbf{r}^k$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{V}\mathbf{y}^k$$

*Update* $\mathbf{V}$ *and* $\mathbf{W}$

$$\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0) = span\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{m-1}\mathbf{r}_0\}$$
$$\mathcal{L} = \mathbf{A}\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$$

- Yields the normal system $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$
- Must generate an orthonormal basis $\mathbf{V}_m \in \mathbb{R}^{N\times m}$ for $\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$
- $\mathbf{W}_m = \mathbf{A}\mathbf{V}_m$
- Typically choose a Gram-Schmidt-like procedure such as Arnoldi or Lanzcos

# GMRES

**Algorithm 1** GMRES Iteration

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$

$\beta := ||\mathbf{r}_0||_2$

$\mathbf{v}_1 := \mathbf{r}_0/\beta$ {Create the orthonormal basis for the Krylov subspace}

**for** $j = 1, 2, \cdots, m$ **do**

  $h_{ij} \leftarrow \langle w_j, v_j \rangle$

  $w_j \leftarrow w_j - h_{ij} v_i$

**end for**

$h_{j+1,j} \leftarrow ||w_j||_2$

$v_{j+1} \leftarrow w_j / h_{j+1,j}$ {Apply the orthogonality constraints}

$\mathbf{y}_m \leftarrow argmin_y ||\beta \mathbf{e}_1 - \mathbf{H}_m\ \mathbf{y}||_2$

$\mathbf{x}_m \leftarrow \mathbf{x}_0 + \mathbf{V_m}\mathbf{y}_m$

# Parallel Projection Methods

- Parallel vector update

$$\mathbf{y}[n] \leftarrow \mathbf{y}[n] + a * \mathbf{x}[n],\ \forall n \in [1, N_g]$$

$$\mathbf{y}[n] \leftarrow \mathbf{y}[n] + a * \mathbf{x}[n],\ \forall n \in [1, N_l]$$

- Parallel dot product

$$d_l = \mathbf{y}_l \cdot \mathbf{x}_l,\ d_g = \sum_p d_l$$

- Parallel vector norm

$$||x||_{\infty,l} = \max_n \mathbf{y}[n],\ \forall n \in [1, N_l]$$

$$||x||_{\infty,g} = \max_p ||x||_{\infty,l}$$

# Parallel Matrix-Vector Multiplication

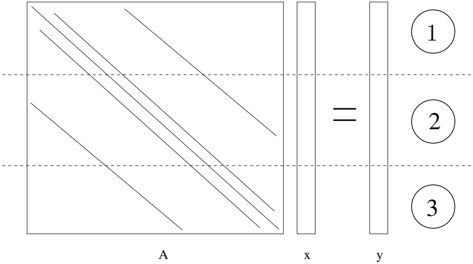


Figure: **Matrix-vector multiply $Ax = y$ operation on 3 processors.**

# Parallel Matrix-Vector Multiplication

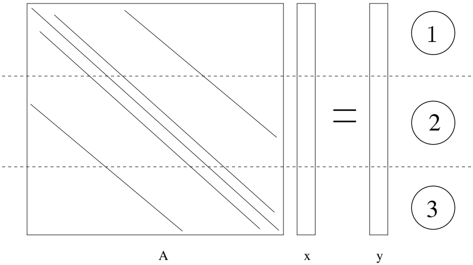


Figure: **Matrix-vector multiply $\mathbf{Ax} = \mathbf{y}$ operation on 3 processors.**

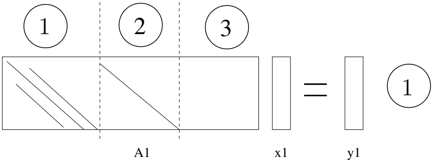


Figure: **Components of multiply operation owned by process 1.**

# Projection Method Notes

- Global reduction operations observed not to impede scalability
  - Dot product
  - Vector norms
- Nearest neighbor computations have poor algorithmic strong scaling
  - Matrix-vector multiply
  - Weak scaling is better

# Monte Carlo Solution 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
- No indication of parallel methods in the literature

# Monte Carlo Linear Solver Preliminaries

- Split the operator

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

$$\mathbf{x} = \mathbf{H}\mathbf{x} + \mathbf{b}$$

- 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 Nuemann 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

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

- Define the *Neumann-Ulam decomposition*[1]

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

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

# Direct Method

- Compute row-normalized transition probabilities and weights

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

- Generate an expectation value for the solution

$$W_m = \sum_{m=0}^{k} w_{i,i_1} w_{i_1,i_2} \cdots w_{i_{m-1},i_m}$$

$$X_\nu(i_0 = i) = \sum_{m=0}^{k} W_m b_{i_m}$$

# Direct Method

- Compute the probability of a particular random walk permutation

$$P_\nu = p_{i,i_1} p_{i_1,i_2} \cdots p_{i_{k-1},i_k}$$

- Generate the estimator

$$E\{X(i_0 = i)\} = \sum_\nu P_\nu X_\nu$$

- Check that we recover the exact solution

$$\begin{aligned} E\{X(i_0 = i)\} &= \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \dots \sum_{i_k}^{N} p_{i,i_1} p_{i_1,i_2} \dots p_{i_{k-1},i_k} w_{i,i_1} w_{i_1,i_2} \dots w_{i_{k-1},i_k} b_{i_k} \\ &= x_i \ , \end{aligned}$$

# Direct Method: Evolution of a Solution

# Adjoint Method

- Solve the adjoint linear system

$$\mathbf{A}^T\mathbf{y} = \mathbf{d}$$

$$\mathbf{y} = \mathbf{H}^T\mathbf{y} + \mathbf{d}$$

- Set the adjoint constraint

$$\langle \mathbf{A}^T\mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{A}\mathbf{y} \rangle$$

$$\langle \mathbf{x}, \mathbf{d} \rangle = \langle \mathbf{y}, \mathbf{b} \rangle$$

# Adjoint Method

- Generate the Neumann series for the adjoint operator

$$\mathbf{y} = (\mathbf{I} - \mathbf{H}^T)^{-1}\mathbf{d}$$

$$\mathbf{y} = \sum_{k=0}^{\infty} (\mathbf{H}^T)^k \mathbf{d}$$

- Expand the series

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

- Pick another constraint to yield the original solution

$$\mathbf{d} = \boldsymbol{\delta}_i, \ \langle \mathbf{y}, \mathbf{b} \rangle = \langle \mathbf{x}, \boldsymbol{\delta}_i \rangle = x_i$$

# Adjoint Method

- Use the adjoint Neumann-Ulam decomposition

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

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

- Build the estimator and expectation value

$$X_\nu = \sum_{m=0}^{k} W_m b_{i_0} \delta_{i,i_m}$$

$$E\{X_j\} = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \dots \sum_{i_k}^{N} b_{i_0} h_{i,i_1} h_{i_1,i_2} \dots h_{i_{k-1},i_k} \delta_{i_k,j}$$
$$= x_j \; ,$$

# Adjoint Method: Evolution of a Solution

# Sequential Monte Carlo

- Neumann-Ulam methods bound by the Central Limit Theorem
- Halton proposed an iterative residual method
- Iteration error decoupled from Monte Carlo error
- Exponential convergence

$$\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k$$

$$\mathbf{A}\boldsymbol{\delta}^k = \mathbf{r}^k$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \boldsymbol{\delta}^k$$

# Monte Carlo Synthetic-Acceleration

- Split the operator to yield Richardson's iteration

$$\mathbf{x} = (\mathbf{I} - \mathbf{A})\mathbf{x} + \mathbf{b}$$

$$\mathbf{x}^{k+1} = (\mathbf{I} - \mathbf{A})\mathbf{x}^{k} + \mathbf{b}$$

- Define the iteration error

$$\delta\mathbf{x}^{k} = \mathbf{x} - \mathbf{x}^{k}$$

$$\delta\mathbf{x}^{k+1} = (\mathbf{I} - \mathbf{A})\delta\mathbf{x}^{k}$$

# Monte Carlo Synthetic-Acceleration

- Subtract $(\mathbf{I} - \mathbf{A})\delta\mathbf{x}^{k+1}$

$$\begin{aligned} \mathbf{A}\delta\mathbf{x}^{k+1} &= (\mathbf{I} - \mathbf{A})(\mathbf{x}^{k+1} - \mathbf{x}^{k}) \\ &= \mathbf{r}^{k+1} \end{aligned}$$

- The following converges in one iteration with exact inversion of $\mathbf{A}$:

$$\mathbf{x}^{k+1} = (\mathbf{I} - \mathbf{A})\mathbf{x}^{k} + \mathbf{b}$$

$$\mathbf{A}\delta\mathbf{x}^{k+1} = \mathbf{r}^{k+1}$$

$$\mathbf{x} = \mathbf{x}^{k+1} + \delta\mathbf{x}^{k+1}$$

# Monte Carlo Synthetic-Acceleration

## MCSA Iteration

$$\mathbf{x}^{k+1/2} = (\mathbf{I} - \mathbf{A})\mathbf{x}^{k} + \mathbf{b}$$

$$\mathbf{r}^{k+1/2} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1/2}$$

$$\hat{\mathbf{A}}\delta\mathbf{x}^{k+1/2} = \mathbf{r}^{k+1/2}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1/2} + \delta\mathbf{x}^{k+1/2}$$

- Adjoint Neumann-Ulam solver computes the correction
- Decouples Monte Carlo error from solution error
- Exponential convergence
- Demonstrated by Evans and colleagues to be competitive with Krylov methods

# Preconditioning Monte Carlo Methods

- No symmetry requirements
- Require $\rho(\mathbf{H}) < 1$
- Choose Jacobi preconditioning at a minimum

$$\mathbf{M} = diag(\mathbf{A})$$
$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$$

- Yields a preconditioned MCSA iteration with no in-state transitions

$$\mathbf{x}^{k+1/2} = (\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})\mathbf{x}^k + \mathbf{b}$$
$$\mathbf{r}^{k+1/2} = \mathbf{b} - \mathbf{M}^{-1}\mathbf{A}\mathbf{x}^{k+1/2}$$
$$\mathbf{M}^{-1}\mathbf{A}\delta\mathbf{x}^{k+1/2} = \mathbf{r}^{k+1/2}$$
$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1/2} + \delta\mathbf{x}^{k+1/2}$$

# Parallelization of Monte Methods

- No literature observed for parallel Neumann-Ulam solvers
- Numerous references for modern parallel Monte Carlo methods in reactor physics
- MCSA parallelism comes from parallel matrix/vector operations

# Domain Decomposition

# Multiple-Set Overlapping-Domain Decomposition

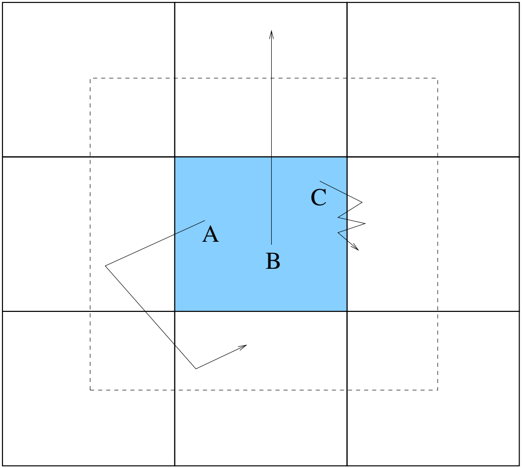


Figure: **Overlapping domain example illustrating how domain overlap can reduce communication costs.**

# Domain-to-Domain Communication

# Load Balancing

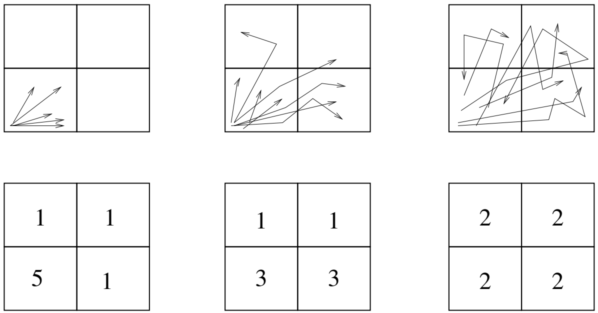


Figure: **Example illustrating how domain decomposition can create load balance issues in Monte Carlo.**

# Reproducability

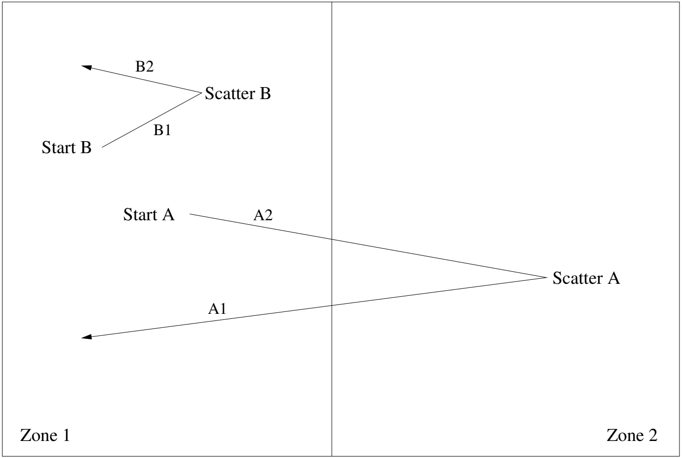


Figure: **Gentile's example illustrating how domain decomposition can create reproducibility issues in Monte Carlo.**

# Parallel MCSA

# Monte Carlo Solution Methods for Nonlinear Problems

# Nonlinear Preliminaries

# Nonlinear Preliminaries

# Newton-Krylov Methods

# Matrix-Free Approximation

# Automatic Differentiation

# The FANM Method

# Jacobian Storage vs. Subspace Storage

# Parallel FANM Method

# Research Proposal

# Experimental Framework

# Progress to Date

# Progress to Date

# Progress to Date

# Monte Carlo Methods Verification

# Proposed Numerical Experiments

# Proposed Challenge Problem

# Conclusion