

# Multi-Method Solvers for Improving the Performance of PDE-Based Simulations

Padma Raghavan

The Pennsylvania State University

In collaboration with:

Sanjukta Bhowmick, Lois Curfman McInnes, and Boyana Norris

Funded by NSF and DOE

## Introduction

- There are several competing (preconditioned, iterative) methods for sparse linear system solution
- It is practically impossible to determine a priori the ‘best’ method— it is often domain or problem dependent and it can vary even across time-steps of the same simulation!
- We consider the development of multi-method sparse linear solvers to improve both the reliability and speed of linear solution
- We demonstrate how such multi-method schemes can improve the performance of large-scale PDE-based simulations

## Background

- Earlier work on multi-method sparse linear solvers include:
  - Ern, Giovangigli, Keyes, and Smooke consider the performance of several linear solution methods for systems from nonlinear elliptic PDEs to motivate development of a ‘polyalgorithmic’ scheme
  - Barrett, Berry, Dongarra, Eijkhout, and Romine, consider a multiprocessor implementation where different types of Krylov methods are applied in parallel to the same system

## Our Approach

- Focus on multi-method solvers for PDE-based applications
- Emphasis on formulating general purpose metrics that can be used to increase reliability and reduce execution times
- Instantiation using advanced object-oriented software systems, such as PETSc (from ANL), with well-defined abstract interfaces and dynamic method selection
- Development with a view to future software component architectures and runtime systems where
  - metrics can be gathered automatically, and
  - used to construct plug-and-play solvers that satisfy application QoS requirements

## Types of Multi-Method Solvers

- We consider two types of multi-method schemes to cater to different aspects of PDE based simulations
  1. **Adaptive Solvers** : primary goal is to match the solver strength to the difficulty/ease of solving a linear system to potentially reduce application time
  2. **Composite Solvers** : primary goal is to provide a highly reliable solver by using a sequence of preconditioned KSP methods until convergence is achieved

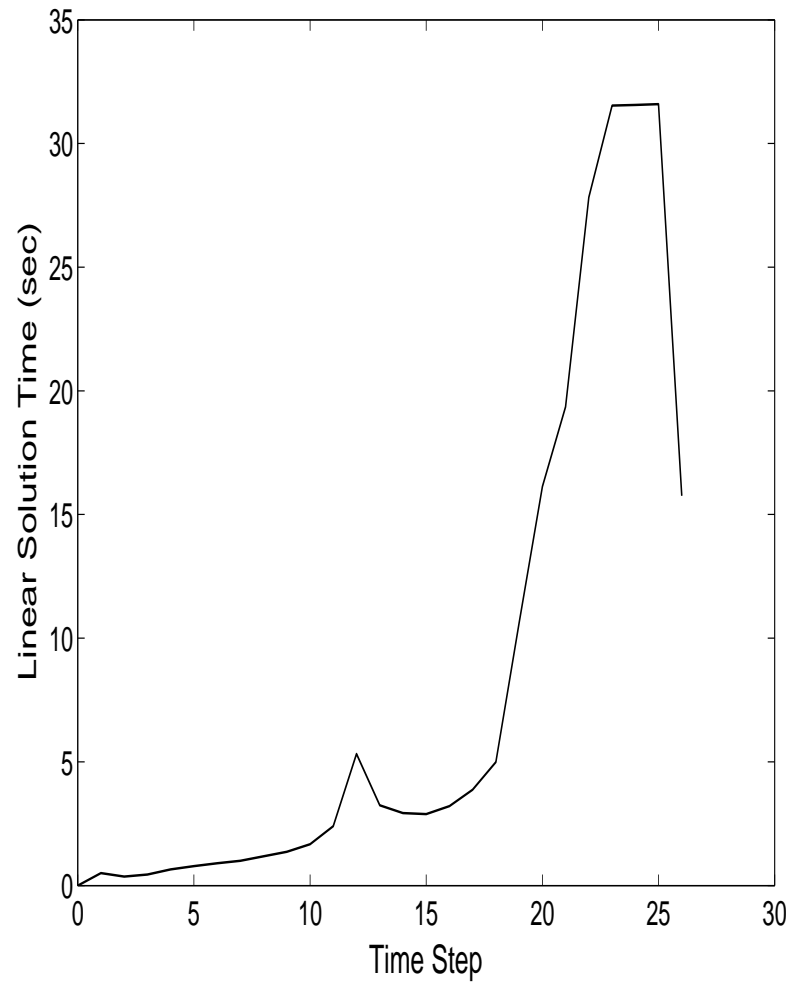
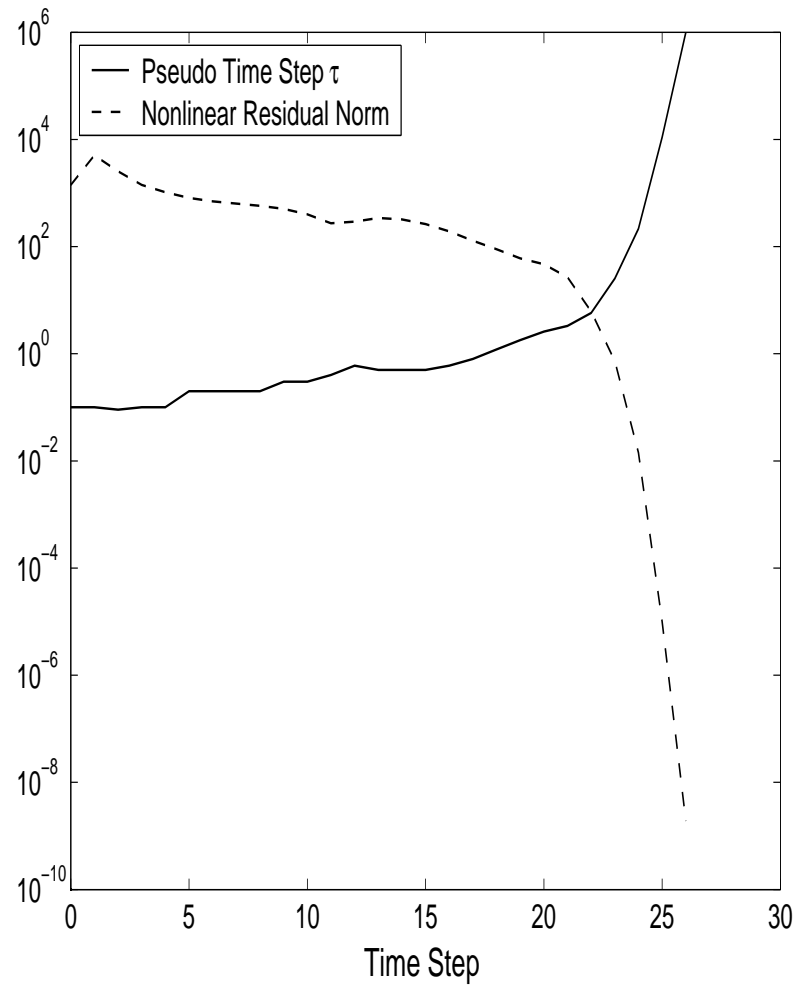
## Driven Cavity Flow Application

- Models driven cavity flow with lid-driven and buoyancy-driven flows in a two-dimensional rectangular cavity
- Uses a velocity-vorticity form of the Navier-Stokes and energy equations and finite differences with a five-point stencil
- Values of Grashof number and lid velocity determine the degree of nonlinearity and affect convergence
- We use inexact Newton methods to solve  $f(u) = 0$ ; for models with high nonlinearity, we use pseudo-transient continuation to solve:  $g_\ell(u) \equiv \frac{1}{\tau^\ell}(u - u^{\ell-1}) + f(u) = 0$ ,  $\ell = 1, 2, \dots$
- Each nonlinear iteration (time-step) requires  $1/2$  sparse linear system solutions

## Adaptive Solvers

- Goal: reducing total time of applications where the numerical properties of linear systems evolve through time-steps
- A typical candidate application is driven cavity flow with pseudo-transient continuation
- The condition number of the Jacobian grows with  $\tau$ , the pseudo-time step
- As the pseudo time step size  $\tau$  increases, the linear systems become more difficult to solve

# Linear System Solution Costs



## Adaptive Method Selection-I

- Consider  $n$  methods  $M_1, M_2, \dots, M_n$  with  $M_i$  associated with  $t_i$ , the normalized execution time per linear iteration
- Assume the methods are ordered in increasing order of  $t_i$  and this ordering implies ordering in increasing order of solver strength
- Assume a simulation has  $T$  time-steps and each time-step requires a sparse linear system solution; the systems change from one time-step to the next
- Consider windows comprised of a fixed number of consecutive time-steps

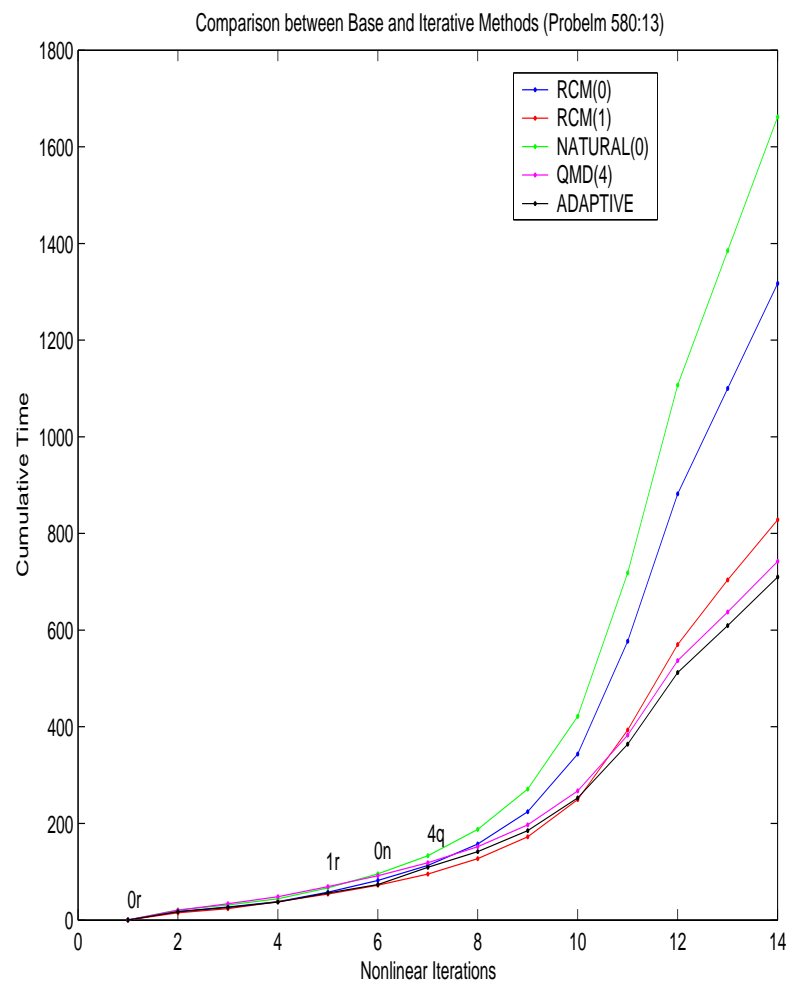
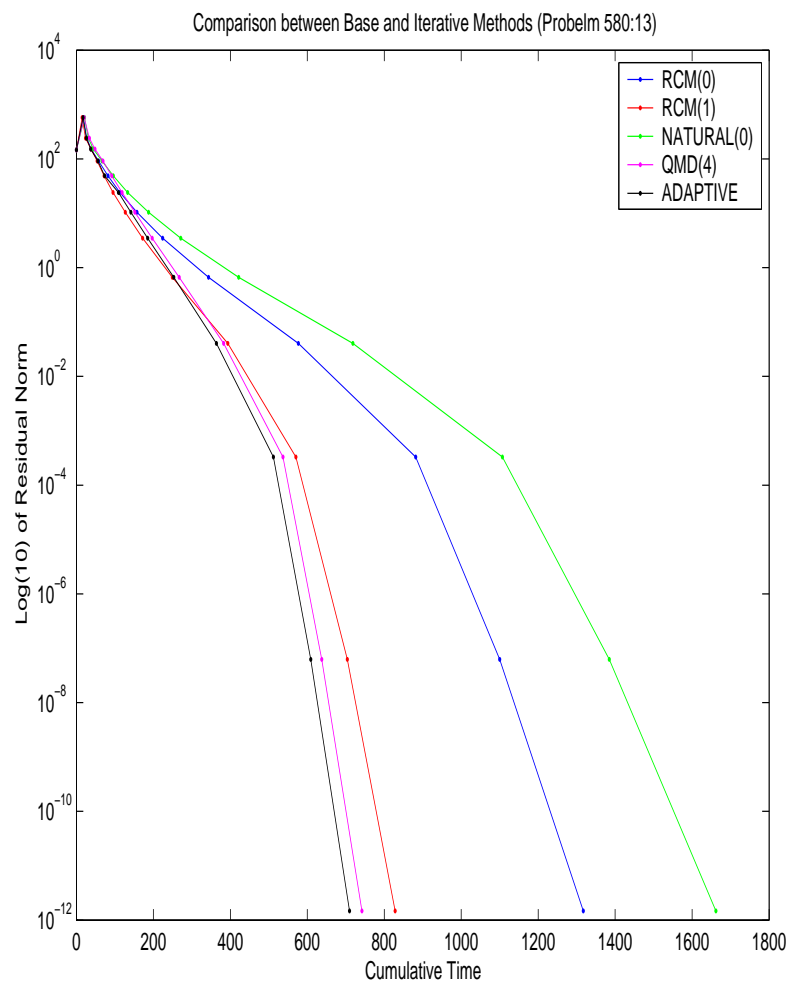
## Adaptive Method Selection-II

- If method  $M_j$  is currently in use in window  $i - 1$  and  $i$ , then automatically determine whether to switch up (down) to method  $M_{j+1}$  ( $M_{j-1}$ ) for next window
- Let  $T(i - 1, j)$  and  $T(i, j)$  be the observed times for all the linear system solutions in windows  $i - 1$  and  $i$
- Method  $j + 1$  is selected for window  $i + 1$  if
  - $T(i, j) > T(i - 1, j)$ , and,
  - $\frac{T(i, j)}{T(i - 1, j)} \geq \alpha \frac{t_{j+1}}{t_j}$ , where  $\alpha$  is a small constant
- A similar scheme can be used for switching down; we have also worked with more complex heuristics taking into account observed convergence rates

## Experiments: Adaptive Solvers

- We used a  $128 \times 128$  grid that gives a system of rank 260,100 with 5.2 million nonzeros
- We used pseudo-transient continuation, and GMRES(30) with the following preconditioners
  1. ILU (0) RCM
  2. ILU (1) RCM
  3. ILU (1) Natural
  4. ILU (4) QMD
- We report performance for window size 2 (overlapped), with  $\alpha = .8$

# Adaptive Solvers: Performance



## Composite Solvers

- Goal: Improving the reliability of linear solution in applications where failures are correlated with poor performance
- Each method represents a trade-off between a performance metric and robustness metric; method failures are independent
- A composite consists of several methods in a sequence; failure of one method results in the execution of the next method
- A composite is significantly more robust than its component methods and its robustness is independent of the order in which methods are selected
- The performance of the composite depends on the order in which methods are selected; Goal: develop composites with optimal performance

## A Combinatorial Framework-I

- There are  $n$  methods  $M_1, M_2, \dots, M_n$ ;  $M_i$  is associated with:
  - normalized execution time:  $t_i$  (worst case)  
 $t_i$  as the time per iteration  $\times$  the maximum number of iterations
  - success rate:  $r_i$  (robustness metric); failure rate  $f_i = 1 - r_i$
- The set  $\mathbf{P}$  contains all permutations (of length  $n$ ) of  $\{1, 2, \dots, n\}$
- $\hat{P}_i$  denotes the  $i$ -th method in  $\hat{P} \in \mathbf{P}$  and it specifies the composite  $\hat{C}$  (i.e.,  $\hat{C}$  executes methods in the order  $\hat{P}$ )

## A Combinatorial Framework-II

- Reliability of  $\hat{C}$ :  $1 - \prod_{i=1}^n f_i$ ; it does not depend on  $\hat{P}$
- Execution time (worst case) of  $\hat{C}$ :

$$\hat{T} = t_{\hat{P}_1} + f_{\hat{P}_1} t_{\hat{P}_2} + \cdots + f_{\hat{P}_1} f_{\hat{P}_2} \cdots f_{\hat{P}_{n-1}} t_{\hat{P}_n}$$

- The performance of a composite depends directly on the permutation and there can be dramatic variations
- Problem: Determine  $\tilde{P} \in \mathbf{P}$  such that the associated composite  $\tilde{C}$  is optimal, i.e., its worst case time  $\tilde{T} = \min\{\hat{T} : \hat{P} \in \mathbf{P}\}$

## Composites of 3 Methods: An Example

$t_1, r_1, f_1$	$t_2, r_2, f_2$	$t_3, r_3, f_3$
1, 0.1, 0.9	1.5, 0.7, 0.3	3.0, 0.8, 0.2

Permutation	Composite Time
1,2,3 'least time'	$3.16 = 1 + .9 \times 1.5 + .9 \times .3 \times 3.0$
3,2,1 'least failure'	$3.36 = 3 + .2 \times 1.5 + .2 \times .3 \times 1.0$
2,1,3	$2.61 = 1.5 + .3 \times 1 + .3 \times .9 \times 3.0$
2,3,1	$2.46 = 1.5 + .3 \times 3 + .3 \times .2 \times 1.0$

Failure rate of any composite is 0.054

## And The Optimal Composite is .....

- Define  $u_i = \frac{t_i}{r_i}$  as the utility ratio of method  $M_i$
- Let  $\tilde{P} \in \mathbf{P}$  and let  $\tilde{C}$  be the associated composite
- Theorem:  $\tilde{C}$  is the optimal composite with  $\tilde{T} = \min\{\hat{T} : \hat{P} \in \mathbf{P}\}$  if and only if  $\tilde{P} = \tilde{P}_1, \tilde{P}_2, \dots, \tilde{P}_n$  is such that

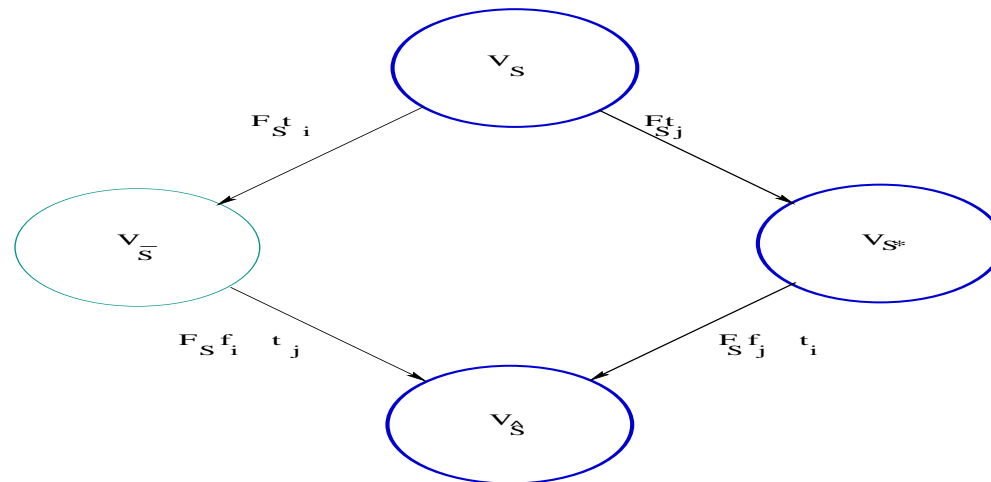
$$u_{\tilde{P}_1} \leq u_{\tilde{P}_2} \leq \dots \leq u_{\tilde{P}_{n-1}} \leq u_{\tilde{P}_n}$$

- The optimal composite is one in which the component methods are arranged in increasing order of their utility ratios
- The proof of the if part is tedious, only-if part is more intuitive and is based on shortest paths in a certain layered graph

## Composites as Paths in a Layered Graph

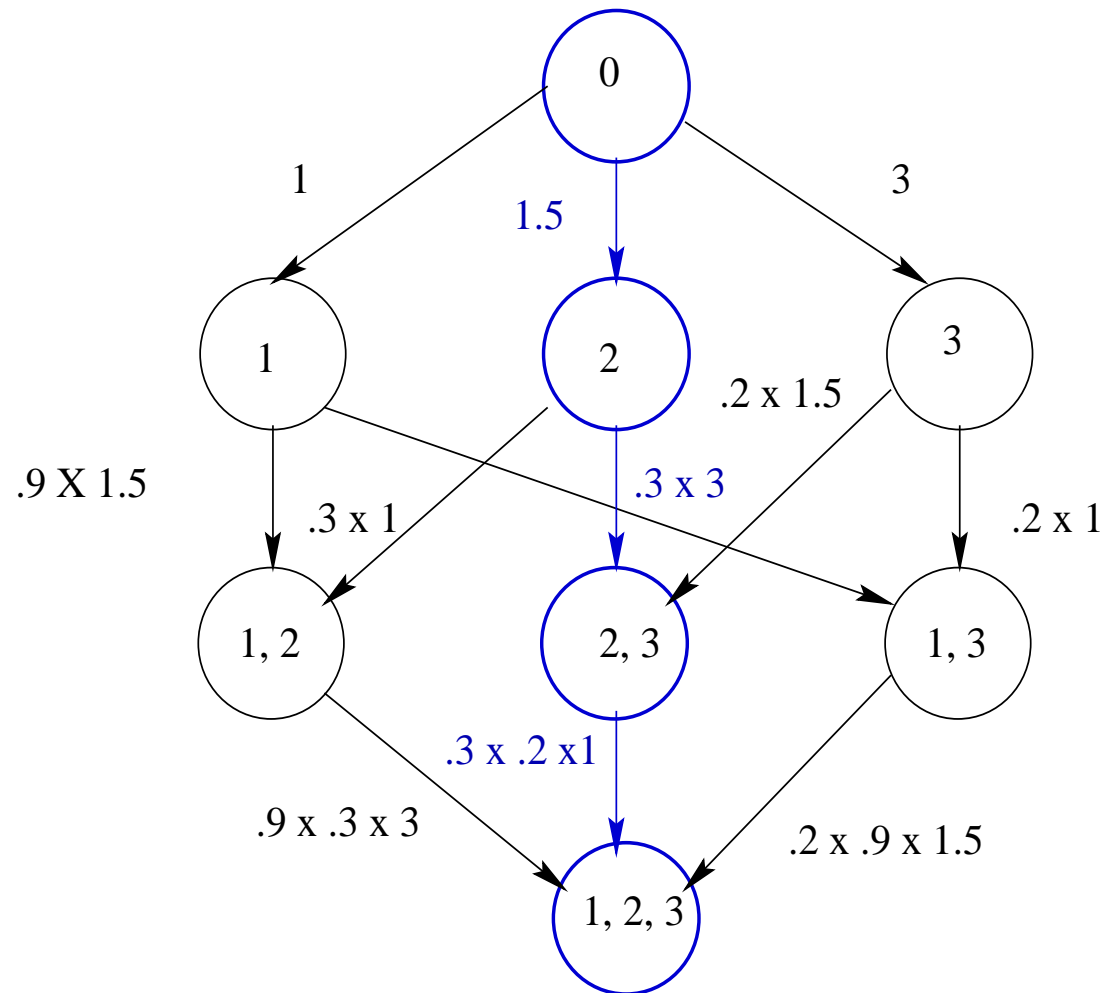
- A graph with vertices at  $n + 1$  levels, a vertex at level  $i$  represents a subset of  $i$  methods ( $V_0$  at level 0 for the empty set)
- An edge connects vertex  $\bar{S}$  to  $\hat{S}$  if they are on adjacent levels,  $|\hat{S} \setminus \bar{S}| = 1$ , and  $\hat{S} \cap \bar{S} = \bar{S}$
- For the subset of methods  $S$ ,  $F_S = \prod_{M_i \in S} f_i$  and  $F_\Phi = 1$ ; edge  $\bar{S} \rightarrow \hat{S}$  has weight  $F_{\bar{S}} t_i$  if  $\hat{S} \setminus \bar{S} = \{i\}$
- A path from  $V_\Phi$  to  $V_{\{1,2,\dots,n\}}$  denotes a composite in the order in which methods were added to vertex sets
- The length of the path is the time for the composite; the optimal composite is the one with the shortest path

# Composites as Paths in a Layered Graph



- $V_S$  and  $V_{\hat{S}}$  are on shortest path and  $\hat{S} - S = \{i, j\}$ .
- There are only 2 paths from  $V_S$  to  $V_{\hat{S}}$ , one with  $V_{\bar{S}}$  ( $\bar{S} - S = \{i\}$ ) and another with  $V_{S^*}$  ( $S^* - S = \{j\}$ )
- By shortest paths,  $T_S + F_S t_j + F_S f_j t_i \leq T_S + F_S t_i + F_S f_i t_j$ ; simplifies to  $t_j(1 - f_i) \leq t_i(1 - f_j)$  and thus  $u_j \leq u_i$

## An Example For 3 Methods



## Composites of 3 Methods: An Example

 $t_1, r_1, f_1, u_1$ 
 $1, 0.1, 0.9, 10$ 
 $t_2, r_2, f_2, u_2$ 
 $1.5, 0.7, 0.3, 2.14$ 
 $t_3, r_3, f_3, u_3$ 
 $3.0, 0.8, 0.2, 3.75$ 

Permutation

Composite Time

1,2,3 'least time'

$$3.16 = 1 + .9 \times 1.5 + .9 \times .3 \times 3.0$$

3,2,1 'least failure'

$$3.36 = 3 + .2 \times 1.5 + .2 \times .3 \times 1.0$$

2,1,3

$$2.61 = 1.5 + .3 \times 1 + .3 \times .9 \times 3.0$$

2,3,1

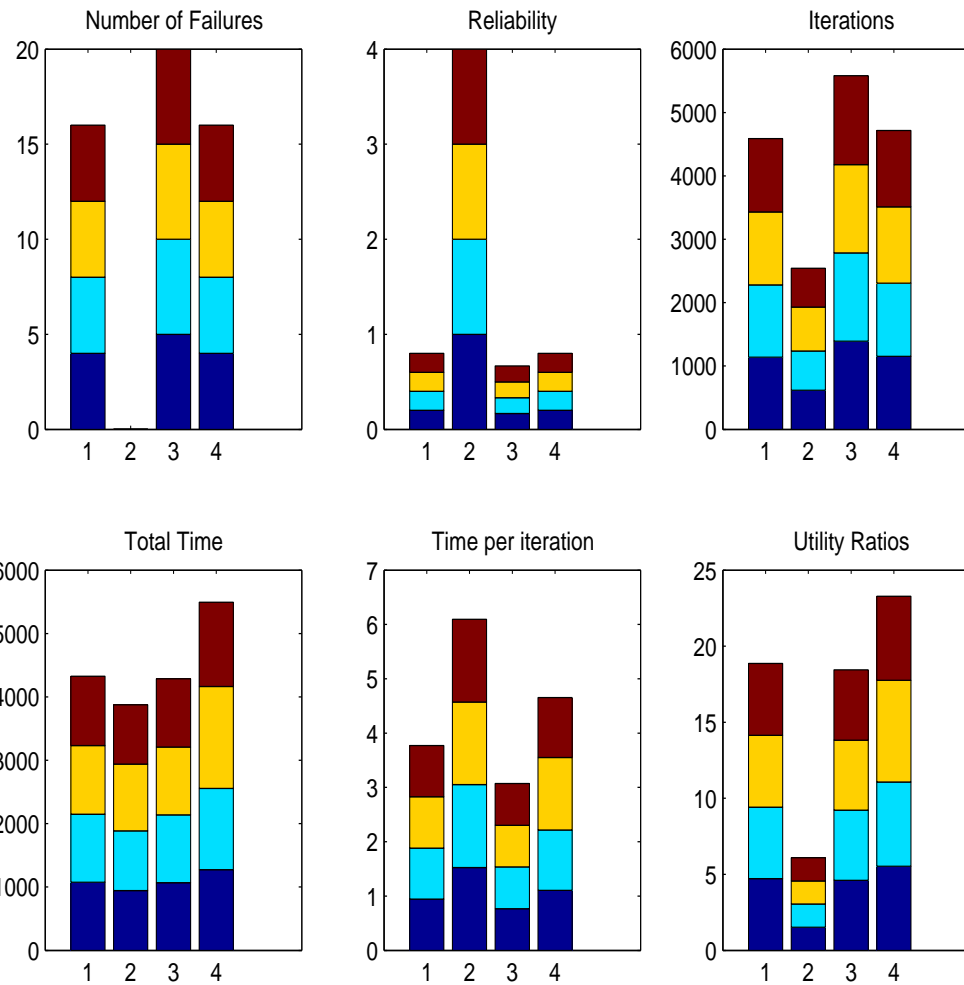
$$2.46 = 1.5 + .3 \times 3 + .3 \times .2 \times 1.0$$

Failure rate of any composite is 0.054

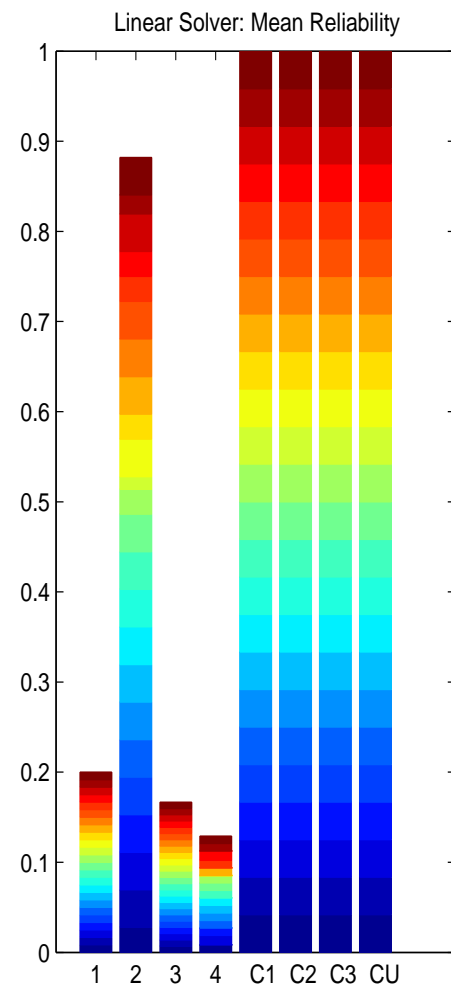
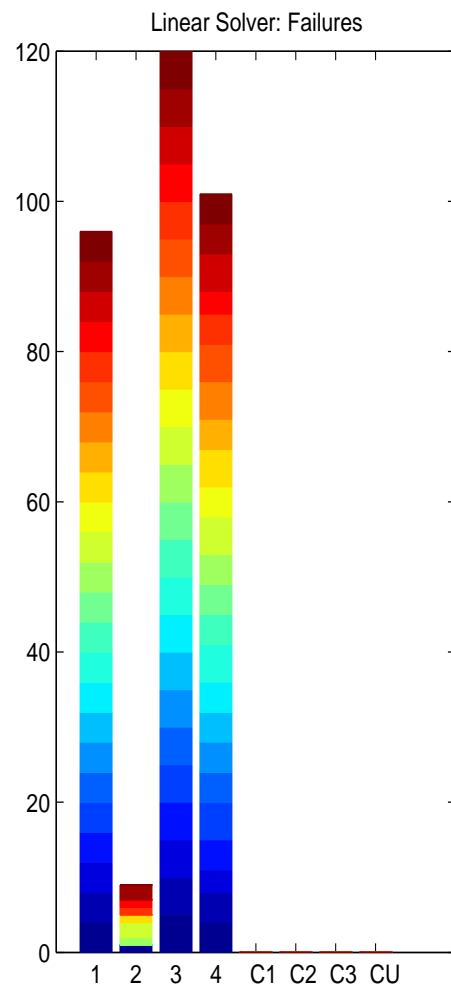
## Experiments: Composite Solvers

- We used a  $128 \times 128$  grid that gives a system of rank 260,100 with 5.2 million nonzeros
- We used the following base solution methods with a maximum iteration limit of 250
  1. GMRES(30), ILU (0) QMD
  2. TFQMR, ILU drop threshold  $10^{-2}$  RCM
  3. GMRES(30), ILU(0) RCM
  4. TFQMR, ILU(0) RCM
- 24 simulations, Grashof numbers: 580, 620, 660, 700, 740, and 780, each with lid velocities: 10, 13, 16, and 20
- CU is the optimal composite: 2, 3, 1, 4; C1: 3, 1, 2, 4; C2: 4, 3, 2, 1; C3: 2, 1, 3, 4

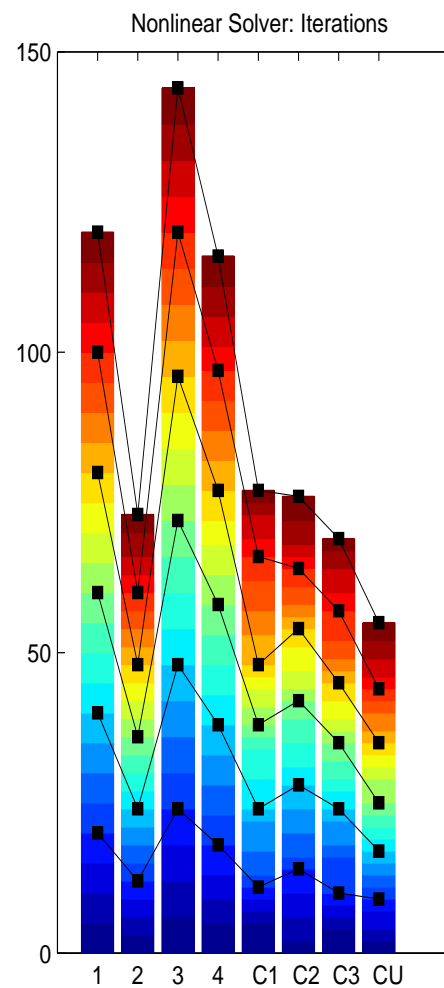
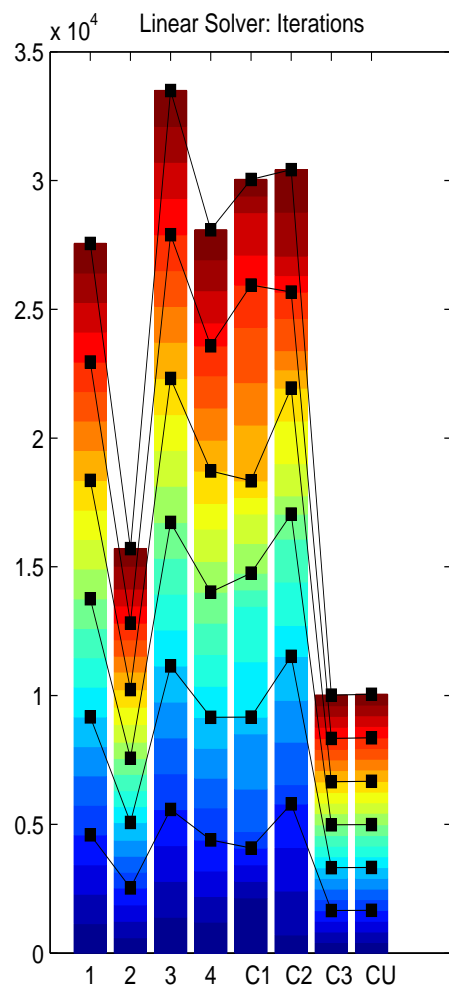
# Observed Metrics At Sample Points



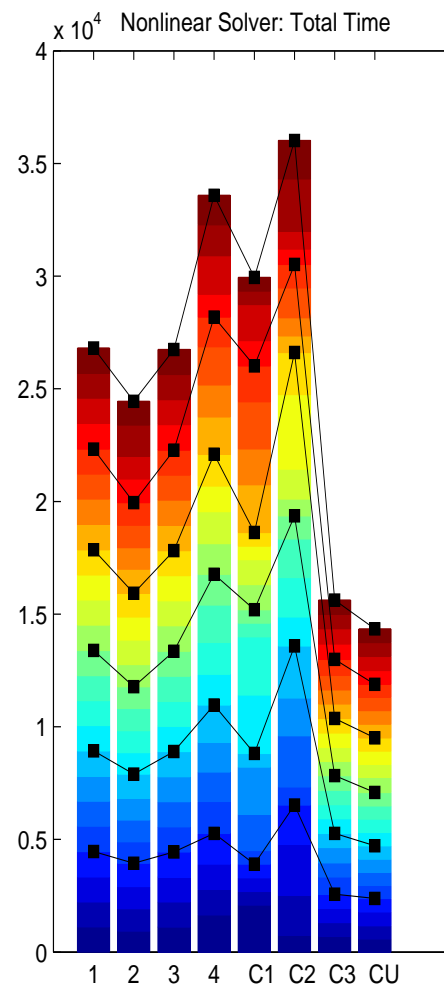
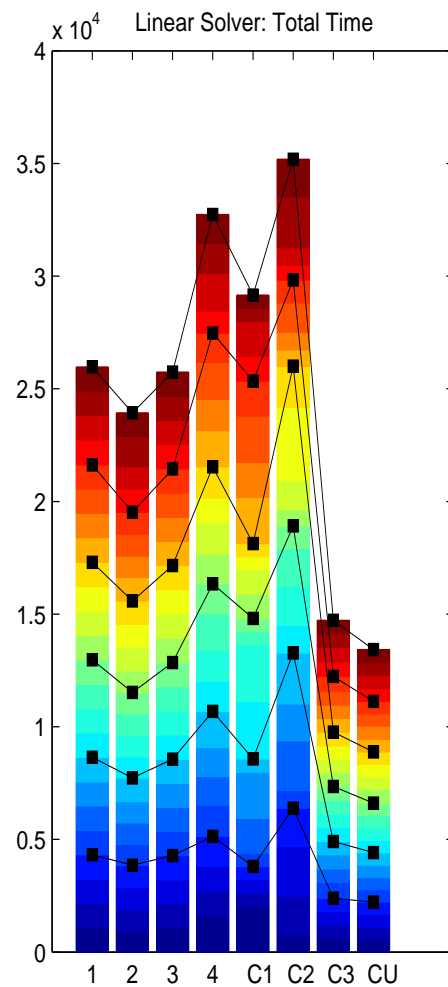
# Failures And Reliability



# Iteration Counts



# Execution Times



## Conclusions

- Adaptive methods show potential in speeding up PDE simulations through automatic method selection
- Composites deliver high reliability and reduce total execution time for the application
- Ongoing work: components that dynamically use observed execution times and failure rates to implement adaptive and composite solvers
- Multi-method solvers are particularly relevant with emerging component architectures and plug-and-play simulation environments

## Related Problems

- Method and Algorithms
  - determining statistically valid performance and reliability metrics by sampling
  - generalization to a parallel computing model
- Software and Runtime Systems
  - specifying QoS metrics for sparse solver components
  - common component interfaces
  - determining models for SPMD computing

## Acknowledgements

- This project is supported through funding from NSF and DOE
- Our implementations use/extend the Portable, Extensible Toolkit for Scientific Computation (PETSc) from Argonne National Labs; special thanks to Barry Smith and Satish Balay for their help