Adaptive numerical components for PDE-based simulations

Boyana Norris1,∗, Lois Curfman McInnes1, Sanjukta Bhowmick2, and Li Li1
1 Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, USA.
2 Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802, USA.

Numerical simulations based on nonlinear partial differential equations (PDEs) using Newton-based methods require the solution of large, sparse linear systems of equations at each nonlinear iteration. Typically in large-scale parallel simulations such linear systems are solved by using preconditioned Krylov methods. In many cases, especially in time-dependent problems, the attributes of the linear systems can change throughout the stimulation, potentially leading to varying times for solving the linear systems during different nonlinear iterations. We present an approach to characterizing the nonlinear and linear system solution and using the resulting application performance information to dynamically select linear solver methods, with the goal of reducing the total time to solution. We discuss the effect of these adaptive heuristics on fluid dynamics and radiation transport codes. We also introduce general component infrastructure to support dynamic algorithm selection and adaptation in applications involving the solution of nonlinear PDEs.

© 2007 WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim

1 Introduction

The implicit solution of nonlinear partial differential equations (PDEs) using Newton-Krylov methods (see, e.g., [1]) is an effective approach for many large-scale scientific simulations. Two applications that motivate our work are an unsteady compressible three-dimensional Euler model using PETSc-FUN3D [2] and a three-dimensional radiation transport model [3]. Both applications employ parallel preconditioned Newton-Krylov methods in the PETSc library [4] to solve nonlinear equations of the form \( f(u) = 0 \), where \( f : \mathbb{R}^n \to \mathbb{R}^n \), at each timestep. The time for solving the linearized Newton systems is a significant fraction of the overall execution time (about 75% for the compressible Euler application and about 35% for the radiation transport code). Various Krylov methods and preconditioners can be combined to produce solvers with vastly differing computational costs and convergence behaviors. It is typically neither possible nor practical to predict a priori which combination will perform best for a given class of problems. Moreover, as explained in [5, 6] for the compressible Euler and radiation transport applications, these linear systems become progressively more difficult to solve as the simulations advance, so that the most effective linear solution method need not be the same at different stages of the simulation.

This situation has motivated us to develop an approach to characterizing the linear and nonlinear systems solution and using the resulting application performance information to dynamically select linear solver methods, with a goal of reducing overall simulation time. Related work on adaptive solvers includes [7–10] as well as additional references discussed in [6].

2 Adaptive Linear Solver Components

Figure 1 introduces the design of prototype software for managing adaptive linear solvers in nonlinear PDEs. Our approach employs the Common Component Architecture (CCA) [11, 12], which has been specifically developed for parallel scientific computing. The infrastructure for adaptive solvers is part of a broader effort to develop tools for computational quality of service (CQoS) [13], or the automatic selection and configuration of components to suit a particular computational purpose.

New components for monitoring the performance of linear and nonlinear solver components are introduced to the initial applications. An adaptive strategy component serves as a proxy for a linear solver component, presenting the same public interface as a non-adaptive linear solver. Thus, at runtime, the application can seamlessly switch among various algorithms.

We have developed several adaptive heuristics with various degrees of generality. An approach that is generally applicable to Newton-Krylov methods is to monitor the nonlinear rate of convergence and then to switch linear solvers when a given threshold is reached. Approaches that also exploit application-specific knowledge can result in more effective adaptive behavior. For example, a change in a physical parameter that is known to strongly impact the attributes of the linear systems can be used to activate substitution of a different linear solver component.

Figure 2 shows preliminary performance of adaptive Newton-Krylov algorithms for a time-dependent radiation transport problem with \( 4.5 \times 10^5 \) unknowns. We employed four processors of the Jazz cluster at Argonne National Laboratory, which has a Myrinet 2000 network and 2.4 GHz Pentium Xeon processors with 1-2 GB of RAM. This adaptive approach employed several base solvers, consisting of a Krylov method (GMRES, FGMRES, or BCGS) and a block Jacobi preconditioner with one block per processor and a specified subdomain solver (SOR or no-fill ILU). By activating different algorithms at key

∗ Corresponding author E-mail: norris@mcs.anl.gov, Phone: 630 252 7908, Fax: 630 252 5986

© 2007 WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim
switching points in the simulation, we achieved an automated adaptive solver that was 42% faster than the slowest method and 1.2% better than the fastest base method; details are explained in [6]. Adaptive linear solvers demonstrated similar benefits in an unsteady compressible Euler application [5]. These preliminary results indicate the promise of adaptive linear solver components, although much work remains in terms of both algorithmic analysis and software infrastructure.

3 Conclusion

We have introduced the design of prototype component software for adaptive algorithms and demonstrated that adaptive linear solvers have the potential to significantly decrease overall execution time in long-running PDE-based simulations. Current work includes developing component-based capabilities for querying and managing runtime and historical databases to further assist with defining application-specific strategies to manage adaptivity. Future plans include incorporating machine learning capabilities [14] to assist with the selection and parameterization of linear solvers in fusion and accelerator modeling.

Acknowledgments We thank Dinesh Kaushik for his assistance in generating Figure 2. We also thank all members of the CCA Forum for stimulating discussions on high-performance scientific software. This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357.

References