UCRL-CR-142046
B511826

Final Report for "FOSPACK"

J. W. Ruge, D. Dean

November 20, 2000

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

Front Range Scientific Computations
1390 Claremont Drive
Boulder, CO 80305

Final Report for "FOSPACK"

Submitted by: John W. Ruge
PI Name: Dave Dean
Title: FOSPACK

FINAL REPORT
For the period ending October 31, 2000.

Prepared for:
University of California
Lawrence Livermore National Laboratory
Attn: Jill Dunaway
P.O. Box 808, L-561
Livermore, CA 94551

Under
B511826

Date prepared
November 20, 2000.

Final Report on
LLNL Subcontract B511826
FOSPACK

PI Name: Dave Dean

1. Project Summary.

The goal of this subcontract was to modify the FOSPACK code, developed by John Ruge, to call the BoomerAMG solver developed at LLNL through the HYPRE interface. FOSPACK is a package developed for the automatic discretization and solution of First-Order System Least-Squares (FOSLS) formulations of 2D partial differential equations (c.f. [3-9]). FOSPACK takes a user-specified mesh (which can be an unstructured combination of triangular and quadrilateral elements) and specification of the first-order system, and produces the discretizations needed for solution. Generally, all specifications are contained in data files, so no re-compilation is necessary when changing domains, mesh sizes, problems, etc. Much of the work in FOSPACK has gone into an interpreter that allows for simple, intuitive specification of the equations. The interpreter reads the equations, processes them, and stores them as instruction lists needed to apply the operators involved to finite element basis functions, allowing assembly of the discrete system. Quite complex equations may be specified, including variable coefficients, user defined functions, and vector notation. The first-order systems may be nonlinear, with linearizations either performed automatically, or specified in a convenient way by the user. The program also includes global/local refinement capability.

FOSLS formulations are very well suited for solution by algebraic multigrid (AMG) (c.f. [10-13]). The original version uses a version of algebraic multigrid written by John Ruge in FORTRAN 77, and modified somewhat for use with FOSPACK. BoomerAMG, a version of AMG developed at CASC, has a number of advantages over the FORTRAN version, including dynamic memory allocation and parallel capability. This project was to benefit both FRSC and CASC, giving FOSPACK the advantages of BoomerAMG, while giving CASC a tool for testing FOSLS as a discretization method for problems of interest there.

The major parts of this work were implementation and testing of the HYPRE package on our computers, writing a C wrapper/driver for the FOSPACK code, and modifying the wrapper to call BoomerAMG through the HYPRE interface.

2. Papers and Book Chapters Supported in Part by the Subcontract.

No books or papers have yet resulted from this contract, although future publications will acknowledge the support provided by LLNL through this contract.

3. Project Highlights.

At the beginning, work proceeded separately on the HYPRE/BoomerAMG implementation and the FOSPACK code. These are described below. The first section is somewhat detailed, and is intended as an aid to implementing FOSPACK at LLNL.

Technical Details of the HYPRE Implementation.

1. The first step was to obtain a tar file of Livermore's HYPRE software library for solving sparse linear systems of equations and load it on Boulder's IBM RS/6000 POWERparallel SYSTEMS Computer (SP) equipped with the IBM Parallel Environment for AIX (PE). The code was placed in the directory ~/hypre. On extracting the data from the tar file, the following directory structure was created:

~/hypre/src
~/hypre/src/test
~/hypre/src/FORT
~/hypre/src/hypre
~/hypre/src/hypre/include
~/hypre/src/hypre/library

All of these were created from the tar file extraction except ~/hypre/src/FORT which was created to hold the AMG FORTRAN source files. Other directories were also created, but these directories are the ones relevant to our project.

2. The next step was to compile and test standard versions of the IJ_linear_solver code that came with the Livermore package. This involved various debugging and portability issues and code modification activities. The HYPRE code actually supports three conceptual interfaces, Structured Grid Interface, IJ Interface and the Finite Element Interface ([1-2]). However, the Structured Grid Interface was not appropriate for our purposes and the Finite Element Inteface was never compiled. Hence, only the IJ Interface was tested. The IJ Interface was, in fact, the most appropriate of the thre for our purpose.

3. The third step developed a C-FORTRAN interface that allowed AMG-FORTRAN code to be mixed with the HYPRE C-code. The IJ_linear_solver.c code was was used as a guide, but was substantially modified in the process. A new prototype header called extern.h was added to the ~/hypre/src/hypre/include directory and a new archive called libfosfort.a was added to the ~/hypre/src/hypre/lib directory. This new archive contains the following objects:

count.o elimbc.o fctnl.o fp_aux.o geometry.o gp.o initu.o ms.o nl.o prob.o readop9.o refine.o seta.o seta0.o td_dummy.o

This archive is maintained using the archive (ar) command. For example, new objects can be added or existing objects can be modified using the command:

ar rv library file.o

If a file.o object already exists in the library archive, it will be replaced with the new one. If file.o is not currently in the library, it will be added.

Simple applications of this interface, using the same data structures as in the FOSPACK code, were written and tested.

4. The final step was to develop a JR_linear_solver mixed C-FORTRAN code that combined certain features of the HYPRE C-code and the AMG-FORTRAN code for solution via BoomerAMG. This is covered in more detail in the next section.

Modifications to the FOSPACK Code

As noted previously, the original FOSPACK code was written in FORTRAN 77. Although HYPRE provides FORTRAN calls, one of the main advantages of BoomerAMG over the original AMG solver, that of dynamic memory allocation, would have been lost without extensive changes. This is especially important because it is often difficult for users to estimate and specify the array sizes beforehand, especially since the automatic choice of coarser grids makes it impossible to predict the storage needed for any particular problem. It was decided to write a new FOSPACK driver in C, leaving the body of the program in FORTRAN, at least for the time being. This required some rather extensive changes to parts of the code. In order to determine memory requirements fairly exactly for the dynamic allocation, and to perform this allocation in the driver, a number of routines had to be written to provide storage requirements for different parts of the code. These often depended on previous counts and used previously allocated memory. This resulted in a "bootstrapping" process, in which the geometry-dependent vectors were counted and allocated first, then storage for the different unknowns and right-hand sides were determined, and finally, the mesh structure and the form of the equations was used to determine requirements for the matrix itself.

The first C/FORTRAN version called the original AMG solver from the driver. That solver had several modifications specifically to deal with some aspects of the matrices, such as handling of free solution components, certain boundary conditions and the presence of "slave" nodes in locally refined meshes. Rather than attempt to modify BoomerAMG in similar ways, these modifications were separated from the AMG solver and called from the driver.

Other changes were needed because currently, BoomerAMG is able to solve only scalar problems (although a version implementing the "unknown" approach for systems is being developed by CASC personnel (c.f., [11,13].) Since any nontrivial FOSLS formulation produces a system of pde's, this is an issue that had to be dealt with. The approach taken was to use a solver with a block-diagonal preconditioner, with the diagonal blocks corresponding to the different unknowns. The problem then simply becomes a number of decoupled scalar problems that BoomerAMG can easily solve. Better convergence is obtainable when coupling between unknowns is included in the AMG solver, but for particular problems of interest, FOSLS theory shows (under somewhat idealized conditions) that such an approach converges at a factor bounded away from 1 independent of the mesh size. Thus, the driver was modified to include this block-diagonal preconditioning. Some extra storage is needed for an additional copy of the diagonal blocks of the matrix, but this is probably more than offset by the advantage of not having to store off-diagonal blocks on the coarser AMG levels. Also, partly to compensate for slower convergence, a conjugate gradient acceleration was incorporated in the C driver around calls to the solver. BoomerAMG does have its own conjugate gradient acceleration, but AMG generally converges quite well on this block problem, and in this setting, more than one cycle (and the conjugate gradient acceleration) per BoomerAMG call would be wasted.

Some capabilities of the original FOSPACK, such as local/global refinements, are still in the process of being implemented in the C/FORTRAN code. In addition, a separate GUI has been written in MATLAB to aid the user in creating the data files needed for FOSPACK, but has not yet been integrated as a front end or wrapper for the code.

The final code is yet another step. The C driver has been integrated with a sample driver provided with HYPRE to run under MPI and call BoomerAMG through the HYPRE IJ interface. As noted, this is the most suitable interface for AMG and the current data structures used. While the driver is not yet written to run in parallel, this is an important step in that direction. Matrix assembly in FOSPACK can be expensive (as in any finite element code), and it is very parallelizable. The plan is to eventually convert the entire FOSPACK code into C, a language probably much better suited to the much of the task (especially interpreting the equations) than FORTRAN.

4. Concluding Remarks.

This project has produced a C/FORTRAN FOSPACK code that uses BoomerAMG as the linear solver, which should prove to be a very useful research tool. FOSPACK is an evolving code, with new capabilities being added and old ones refined (most often in response to user requests.) Work will continue in parallel on the C/FORTRAN and the FORTRAN 77 versions, with most new capabilities implemented first in the FORTRAN version. An effort is being made to keep these as close as possible, so that changes can more easily be transferred from to the C/FORTRAN version, and from there to the HYPRE driver.

5. References.

[1] HYPRE Code Reference, CASC publication, LLNL, 2000.

[2] HYPRE User's Manual (Draft), CASC publication, LLNL, 2000.

[3] M. Berndt. Adaptive Refinement and the Treatment of Discontinuous Coefficients for Multilevel First-Order System Least-Squares (FOSLS), Ph.D. Thesis, University of Colorado at Boulder, 1999.

[4] Cai,Z., Lazarov,R., Manteuffel,T.A., and McCormick,S.F., First-order system least squares for partial differential equations: Part I, *SIAM J. Numer. Anal.* 31 (1994), pp. 1785-1802.

[5] Cai,Z., Manteuffel,T.A., and McCormick,S.F., First-order system least squares for second-order partial differential equations: Part II, *SIAM J. Numer. Anal.* 34 (1997), pp. 425-454.

[6] Z. Cai, T. A. Manteuffel, and S. F. McCormick}, First-order system least squares for the Stokes equations, with application to linear elasticity, *SIAM J. Numer. Anal.*, 34 (1997), pp.1727-1741.

[7] P. Bochev, Z. Cai, T. A. Manteuffel, and S. McCormick, Analysis of velocity-flux least-squares principles for the Navier-Stokes equations: Part I, *SIAM J. Numer. Anal.*, 35 (1998).

[8] B. Lee, First-order system least squares (FOSLS) for Helmholtz equations, Ph. D. Thesis, Department of Applied Mathematics, University of Colorado at Boulder, Boulder, CO, December, 1996.

[9] Z. Cai, T.A. Manteuffel, S.F. McCormick, and S. V. Parter, First-order system least-squares for the pure traction problem in planar linear elasticity, *SIAM J. Numer. Anal.*, 35 (1998), pp. 320-335.

[10] A. Brandt, S. McCormick, J. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. *Sparsity and its Applications* (D.J. Evans, ed.), Cambridge University Press, 1984.

[11] J. Ruge. Final report on AMGO2. Report, GMD, Contract No. 5110/022090, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1985.

[12] J. Ruge, K. Stüben. Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG). *Multigrid Methods for Integral and Differential Equations* (D. J. Paddon and H. Holstein, eds.), Oxford Science Publications, 1985.

[13] J. Ruge, K. Stüben. Algebraic multigrid (AMG). *Frontiers in Appl. Math.* (S. McCormick, ed.), SIAM, Philadelphia, 1987.