INL/EXT-15-31742

Border-Profile Lower-Upper (BPLU) Solver Zero Bandwidth Project

George L. Mesina

September 2003

The INL is a U.S. Department of Energy National Laboratory
operated by Battelle Energy Alliance

INL/EXT-15-31742

Border-Profile Lower-Upper (BPLU) Solver Zero Bandwidth Project

George Mesina

September 2003

Idaho National Laboratory

Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Software Design and Implementation Document

R5/3D-03-06 INL/EXT-14-31742
Revision 1
Effective Date: September 23, 2003

Title: Border-Profile Lower-Upper (BPLU) Solver Zero Bandwidth Project

Reference: General: DOE – RELAP5-3D
Specific: Fortran 90

Developer: _____Date: _____
George Mesina

Qualified Reviewers:

_____Date: _____
Richard Riemke

_____Date: _____
Walter Weaver

Manager: _____Date: _____
Gary Johnsen

Table of Contents

1.0 Summary

According to the description from the funding source, this task has two subtasks, solve the BPLU zero bandwidth problems and make BPLU the default solver for RELAP5-3D.

The linear equation solver, BPLU, is diminished in usefulness by a failure to run certain key problems for one of our customers. The error message produced is that the reordering of unknowns developed by the BPLU analysis subprograms is unusable because the matrix would have a zero-bandwidth, a nonsensical condition. The problem appears to be that there is insufficient memory allocated to the analysis programs, and they overwrite information with other information producing nonsense. The insufficient memory is caused by the current method of allocating space for the information within pre-existing RELAP5-3D storage arrays.

The proposed solution is to create the required storage afresh via FORTRAN 90 allocatable arrays. The arrays will be declared in a FORTRAN 90 module that will also contain a subroutine that both allocates and deallocates storage.

The second subtask is to make BPLU the default solver. This requires the changing of the meaning of option 33. It also necessitates a great deal of testing to make certain that BPLU will work for all known test problems. There are other aspects to this subtask that were only revealed after work began.

2.0 Background

BPLU is an efficient method for solving the kinds of linear systems that arise in nuclear power plant safety analyses and that range in size from small to moderately large. It both reduces solution time and is suitable for speed-up through vectorization and parallelization techniques. BPLU has been applied successfully not only to nuclear safety analyses but also to fusion power systems. It could efficiently solve any system of equations that arises from a network of pipes or wires, such as electricity power grids, airline flight paths, or refrigeration systems.

The BPLU technique is to reorder the variables through a variety of techniques so that, when the linear system is permuted correspondingly, it has a new structure that can be solved much more efficiently than the original system. The new structure is called the border-profile form; it has most of its non-zero entries clustered in a tight band about the main diagonal and all its remaining non-zeroes in a thin block of rows at the bottom and an equally thin block of columns at the right. The block of rows at the bottom and columns at the right are referred to as the border. The non-zero structure resembles an arrow that points down and to the right. The reordering techniques employed to create this structure are Bandwidth Limitation, Constant Removal, Generalized Porsching, Planar, and Reverse Cuthill-McKee.

Not surprisingly, there is no unique border-profile structure for many matrices. Generally speaking, the thinner the band, the thicker the border and vice-versa. The BPLU analysis arrays seek to produce the best reordering by minimizing the theoretical number of operations the LU factorization and back-substitution subprograms will need to solve the reordered linear system. This is done by varying two parameters that control the reordering, producing the reordering for each pair, estimating the operation count for each, and selecting the smallest operation count subject to certain conditions, the most important of which is fitting into the remaining computer memory.

BPLU was written independently of RELAP5 and was first used in fusion applications. When it was initially incorporated into RELAP5/MOD3, a decision was made to create no new storage for the BPLU information arrays, but rather to be efficient with computer memory and reuse scratch space that was no longer needed by the time matrix set-up was initiated. This worked fine when the only reordering that was performed was Generalized Porsching and Bandwidth Limitation. However, as improvements were added and the optimization subprogram was written, more such scratch space was needed. Eventually all of it was used and even reused by the BPLU analysis subprograms, but this has proven insufficient and a new and better approach to laying out memory to BPLU information arrays will be undertaken in the first subtask.

When this problem is corrected, BPLU will be the logical choice as the default solver since it produces answers faster than the current default solver for practically every problem. Therefore, BPLU will be made the default solver; this is subtask two.

3.0 Subtask Two Work Implications

There are several work items that relate to subtask two, making BPLU the default solver. These have been uncovered as work has proceeded, and they were not in the original work scope estimate.

Consider that there are many points in RELAP5-3D where a linear equation solver is invoked. The following linear systems are solved: pressure-drop equation; velocity equation; phasic energy, phasic density, and non-condensible equations; and the PVM coupling equation. With one exception, all these linear systems are solved by a call to subroutine SYSSOL. SYSSOL in turn calls PMINV, BPLU, or PGMRES to solve the system. The exception is the PVM coupling equation. For some reason, it is solved by directly calling PMINV rather than SYSSOL. *Modifying the PVM solution technique to employ BPLU (or SYSSOL) was not part of the original estimate.*

Another potential subtask relates to solving those systems that are bigger than moderately large. Although BPLU is theoretically capable of solving a system of any size, for truly large systems, the amount of memory required is prohibitive and will exceed the amount available on any computer. For such systems, PGMRES was written and included in RELAP5-3D. The conversion of PGMRES, which relies on the BPLU reordering and uses the BPLU factorization of the matrix from time step zero as its pre-conditioner, is another subtask related to subtask two. Once BPLU is modified to make use of a FORTRAN 90 module for its information arrays, PGMRES will cease to work. *Modifying PGMRES was not part of the original work estimate.*

A third aspect is the CRADA protected coding. This is shipped to our defense and real-time customers, but is not available to IRUG. The CRADA protected coding allows the matrix to be built directly inside the BPLU computer array that holds the matrix. Without this coding, the coefficient matrix is built in the storage array for PMINV and is then transcribed into the BPLU storage locations. This is slower, but the coding is much simpler and straightforward.

The construction of the coefficient matrix currently makes use of the same coding for either solver through the array MAPA. Depending on whether MAPA holds the BPLU ordering or the original ordering of unknowns, the matrix is built for BPLU or PMINV respectively. However, this can only happen because the storage for both linear systems is held in the scratch area of RELAP5-3D's FAST array. It is proposed to place all BPLU arrays in a new FORTRAN 90 module. This would make dual purpose coding impossible. *Maintaining two systems was not part of the original work estimate.*

Moreover, *all use of PMINV will be disabled* if BPLU uses a FORTRAN 90 module for the coefficient matrix unless an effort is made to preserve it. It would be imprudent to disable PMINV. If unforeseen problems arise relative to solving linear systems, having PMINV is a necessary safeguard. It is possible for BPLU to use RELAP5-3D storage to hold the coefficient matrix rather than module storage.

4.0 FORTRAN 90 Module Design

The BPLU FORTRAN 90 module is a precursor to all the other FORTRAN 90 conversion work. In the past, RGUI and the PVM executive were written *from scratch* in FORTRAN 90 with modules. This will be the first conversion effort. As such it will set a precedent for all future such conversions.

The naming convention for modules is that its final three letters, before the extension, are MOD. In RGUI we have isomod.F, hsmod.F, and viewmod.F. The BPLU module will be named bpmod.F.

The module must contain a data dictionary that fully defines every scalar and array declared in the data section. Subroutines that have local variables in the CONTAIN section will define their variables within. Also, the data dictionary must precede the declarations. The data dictionary must be alphabetized and not broken into subsections. This makes it easier to find array definitions.

The BPLU module will contain one subroutine, BPMEM. It will allocate and deallocate memory for BPLU arrays. There are four types of arrays: temporary arrays used for analysis and reordering; permanent information arrays used for reordering and efficiency in solving; temporary solution arrays; and permanent solution arrays (the coefficient matrix and right hand side). Temporary arrays can be allocated at the beginning of BPLU analysis or solution and deallocated just before exiting the corresponding phase. The permanent information arrays must be generated before analysis. The permanent solution arrays must be generated before the matrix is constructed. Both kinds of permanent arrays must be maintained until after BPLU is called and produces the solution. If BPLU is called many times for the same system, as it is in RELAP5-3D, the permanent arrays cannot be deallocated until after the the final solution is produced.

BPMEM must test that arrays are not allocated before attempting to allocate them. It must also stop gracefully if there is not enough computer memory for arrays of the requested size.

BPMEM takes two arguments, the type and the action. The type argument is an integer and takes values from one to four corresponding to the four types described in the preceding paragraphs. The action argument is a character variable. It can take on two values corresponding to two memory allocation actions: allocate and deallocate.

Regarding the data section of the module, most of the arrays will be declared with the ALLOCATABLE attribute. Declarations may be grouped into sections according to purpose, as long as the arrays are declared in alphabetical order within each section. There are four sections in BPMOD: (1) temporary information; (2) permanent information; (3) temporary solution; and (4) permanent solution.

5.0 BPLU Coding Modifications for Subtask One

With few exceptions, it is better to pass arrays through the USE statement for BPMOD than to pass them via call argument. Therefore, most BPLU subprograms will no longer list the BPMOD arrays in their call sequences. Their call sequences will be radically reduced or removed entirely.

The exceptions are REORD and the solver subroutines: BORBND, BPLU, BPSQLU, BPSQSL, and BPSUB. REORD is called with array names interchanged pair-wise on numerous occasions. The solver routines will access RELAP5-3D storage for the coefficient matrix, right hand side, and solution. These must be passed through the call sequence.

For all BPLU subprograms, the "*in32" and "in32end" statements must be eliminated as the arrays listed on those lines will be part of the BPLU module and will not be subject to conversion by cnv32. The line that includes the comdeck bplu1.H must also be removed as that comdeck becomes part of BPMOD.

Calls to BPLU routines that send an array to another subprogram and make use of the "[" notation will be eliminated. Again, this was part of the cnv32 conversion and that will be eliminated by use of BPMOD.

Subroutine TSETSL cannot use BPMOD because of naming problems. Therefore, a new subroutine, BPSETUP will be written to call BPPRAM and BPARAM. It will use BPMOD, and it will also access the RELAP5-3D comdecks necessary for arrays ipr and irnr. Those two will be copied into BPMOD arrays ipoint and indexc, respectively, and the copies will be used in their places. BPSETUP will be called from TSETSL.

Similarly, the post-processing done in TSETSL to build permanent BPLU arrays for the solution will be done in BPSETUP.

6.0 Modifications for Subtask 2 and Testing

The meaning of Card 1 option 33 (currently the BPLU option) will be modified so that if it is not present, RELAP5-3D will use BPLU to solve its linear systems. If option 33 is present, it will use the MA18 solver (PMINV). This requires changes in rchng.F and several other subroutines that check option 33. It is especially important to reverse the meaning of array mapa to correspond to the altered meaning of option 33 when it is constructed.

In TSETSL, it is also worthwhile to eliminate the creation of certain scratch space for BPLU. Normally, the information arrays used in the transient solution by BPLU take up more space than the corresponding arrays used by PMINV. Since these arrays will be part of BPMOD, they should not be allocated in RELAP5-3D scratch space.

Three kinds of testing will be performed.

- First, BPLU will be tested against a wide variety of problems. In particular, the input file that causes the zero bandwidth error will be obtained and tested.
- Second, use of PMINV will also be tested in situations where option 33 is present in the input file.
- Third, the results of RELAP5-3D calculations with the new BPLU coding will be compared with the results using the previous BPLU coding.

This third kind of testing requires TSETSL to continue to build the information and set aside space for BPLU the old way until everything check out properly.

After the BPLU to BPLU comparison is carried out, the coding in TSETSL that builds BPLU solution information arrays can be removed and replaced by similar coding in BPSETUP.

7.0 Installation Changes

The scripts that install the code on Unix and Windows machines must be modified to account for the FORTRAN 90 module. Modules must be loaded before any program unit that accesses one of them. In the older Unix scripts, dloadr will be changed. In the Makefile approach, the changes will follow along the lines of the Makefile in the rgui-directory. On the Windows-PC side, the Makefile approach is similar to the Unix Makefile implementation.

The program that pre-processes the FORTRAN source files and makes conversions for 32-bit integers, cnv32.F, will be modified. Currently, it flags any spurious characters in the label field (columns 1-5 of an old-style punch card). In FORTRAN 90 however, an exclamation mark (!) is used to mark the beginning of a comment at any location on the line, including in the label field. In fact, much of the FORTRAN 90 coding in RGUI and the new license subprogram is written this way. A change to ignore an exclamation mark in the label field can be implemented in one line of FORTRAN 90. The line of coding will be protected with a CPP pre-compiler directive, "f90." (This may be unnecessary since FORTRAN 90 has now become the default).

The Unix script, "dutilty" will be modified to compile cnv32.F with this flag active. The Unix script, "doitf," will be modified to compile multiple FORTRAN source files to aid developers.