Analysis of MUMPS and PETSc solvers integrated in PAK software

Nenad Busarac[1], Vladimir Dunić[1], Miroslav Živković[1], Radovan Slavković[1], Vladimir Milovanović[1]
[1]Faculty of mechanical engineering, Kragujevac

Abstract - *In this paper, explanation of using two different solvers: MUMPS and PETSc in PAK software are presented. The MUMPS libraries have been already used as serial and parallel solver while, PETSc libraries are implemented as additional solution in order to improve accuracy and efficiency. Achieved results are also presented through several standard examples and in case of PETSc we can use adequate solver for problem that it is most suited for. The main advantage of used libraries is that they are open source and available free of charge but also other advantages and defects are noticed in work with this solvers.*

1. INTRODUCTION

Efficiency optimization is today one of the most demanded jobs for programmers and software developers. Using of external libraries capable to solve part of our problem is one solution. Implementation of such libraries is often only effort needed for necessary improvements.
For example, in our case, software package PAK uses MUMPS and PETSc libraries for solving systems of equations and the MPI standard for communication between processes in parallel version. The differences obtained using different solvers is shown and the results obtained using both solvers. Also, advantages and disadvantages that were identified during the operation are presented.

2.FINITE ELEMENT METHOD SOFTWARE - PAK

Laboratory of Mechanical Software at the Faculty of Mechanical engineering in Kragujevac has been working for over 30 years on developing its own software package PAK, based on the Finite Element Method. It is a home grown general purpose FEM software. It contains modules for the pre- and post-processing and for solving problems in the fields of: solid mechanics (static and dynamic structural analysis, geometrical and material nonlinear problems, geomechanics, fracture mechanics material fatigue, biomechanics), heat transfer, fluid flow etc. The original methodology, based on published papers in the world journals and books was applied in numerous fields. Also, it has several home solvers for solving systems if equations, but external libraries which contain solver for that purpose were also implemented.

3.MUMPS

MUMPS ("MUltifrontal Massively Parallel Solver") is a package for solving systems of linear equations of the form Ax = b, where A is a square sparse matrix that can be either unsymmetric, symmetric positive definite, or general symmetric. MUMPS uses a multi-frontal technique which is a direct method based on either the LU or the LDLT factorization of the matrix. The main features of the MUMPS package include the solution of the transposed system, input of the matrix in assembled format (distributed or centralized) or elemental format, error analysis, iterative refinement, scaling of the original matrix, detection of zero pivots, and return of a Schur complement matrix. MUMPS offers several built-in ordering algorithms, a tight interface to some external ordering packages such as METIS (strongly recommended) and PORD, and the possibility for the user to input a given ordering. Finally, MUMPS is available in various arithmetics (real or complex, single or double precision).
The software is written in Fortran 90 although a C interface is available. The parallel version of MUMPS requires MPI for message passing and makes use of the BLAS, BLACS, and ScaLAPACK libraries. The sequential version only relies on BLAS.
MUMPS is downloaded from the web site almost once a day on average and has been run on very many machines, compilers and operating systems, although our experience is really only with UNIX based systems. We have tested it extensively on parallel computers from SGI, Cray, and IBM and on clusters of workstations.
MUMPS distributes the work tasks among the processors, but an identified processor (the host) is required to perform most of the analysis phase, to distribute the incoming matrix to the other processors (slaves) in the case where the matrix is centralized, and to collect the solution. The system Ax = b is solved in three main steps:
1.**Analysis**. The host performs an ordering based on the symmetrized pattern $A+A^T$, and carries out symbolic factorization. A mapping of the multifrontal computational graph is then computed, and symbolic information is transferred from the host to the other processors. Using this information, the processors estimate the memory necessary for factorization and solution.
2.**Factorization**. The original matrix is first distributed to processors that will participate in the numerical factorization. The numerical factorization on each frontal matrix is conducted by a *master* processor (determined by the analysis phase) and one or more *slave* processors (determined dynamically). Each processor allocates an array for the so called contribution blocks and for the factors; the factors must be kept for the solution phase.
3.**Solution**. The right-hand side b is broadcast from the host to the other processors. These processors compute the solution x using the (distributed) factors computed during Step 2, and the solution is either assembled on the host or kept distributed on the processors.
Each of these phases can be called separately and several instances of MUMPS can be handled simultaneously. MUMPS allows the host processor to participate in computations during the factorization and solve phases, just like any other processor.

For both the symmetric and the unsymmetric algorithms used in the code, we have chosen a fully asynchronous approach with dynamic scheduling of the computational tasks. Asynchronous communication is used to enable overlapping between communication and computation. Dynamic scheduling was initially chosen to accommodate numerical pivoting in the factorization. The other important reason for this choice was that, with dynamic scheduling, the algorithm can adapt itself at execution time to remap work and data to more appropriate processors. In fact, we combine the main features of static and dynamic approaches; we use the estimation obtained during the analysis to map some of the main computational tasks; the other tasks are dynamically scheduled at execution time. The main data structures (the original matrix and the factors) are similarly partially mapped according to the analysis phase.

4.PETSC

PETSc, the Portable, Extensible Toolkit for Scientific computation, provides sets of tools for the parallel (as well as serial), numerical solution of PDEs that require solving large-scale, sparse nonlinear systems of equations. PETSc includes nonlinear and linear equation solvers that employ a variety of Newton techniques and Krylov subspace methods. PETSc provides several parallel sparse matrix formats, including compressed row, block compressed row, and block diagonal storage. The table below gives an overview of the main numerical components of the PETSc library:

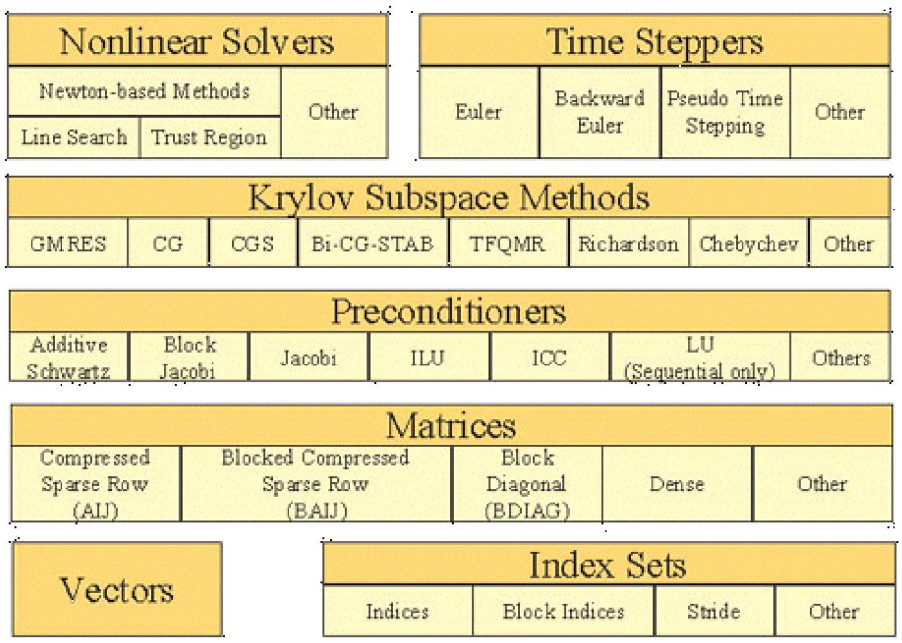

Figure 1

PETSc is designed to facilitate extensibility. Thus, users can incorporate customized solvers and data structures when using the package. PETSc also provides an interface to several external software packages including BlockSolve95, ESSL, Matlab, ParMeTis, PVODE, and SPAI. PETSc is fully usable from Fortran, C and C++, and runs on most UNIX based-systems.

PETSc has several features that make it very convenient for the application programmer. Users can create complete application programs for the parallel solution of nonlinear PDEs without writing much explicit message-passing code themselves. Parallel vectors and sparse matrices can be easily and efficiently assembled through the mechanisms provided by PETSc. Furthermore, PETSc enables a great deal of runtime control for the user without any additional coding cost. The runtime options include control over the choice of solvers, preconditioners and problem parameters as well as the generation of performance logs.

5.IMPLEMENTATION OF PETSC AND MUMPS INTO PAK

We will describe how MUMPS and PETSc solvers are integrated into PAK. Firstly we needed to go through all code and implement MPICH which is needed for MUMPS and PETSc to work in parallel. MPICH must be initialized at the start of program and finalized at the end. Also, all code that was meant to be run sequentially had to be skipped by use of line “if (myid.ne.0) goto X”. When we went to all the code and made sure that we don’t have dead ends in all threads, then MUMPS and PETSc subroutines could be implemented. In figure 2, the procedure where MUMPS and PETSc calls are made is shown.

```
10     CALL MPI_BARRIER(PETSC_COMM_WORLD,ierr)
       CALL MPI_BCAST(iccgg,1,MPI_INTEGER,0,PETSC_COMM_WORLD,ierr)
       CALL MPI_BCAST(imumps,1,MPI_INTEGER,0,PETSC_COMM_WORLD,ierr)
       CALL MPI_BCAST(k,1,MPI_INTEGER,0,PETSC_COMM_WORLD,ierr)
       IF(IABS(ICCGG).EQ.1) THEN
          IF(ICCGG.EQ.1) THEN
             if (rank.ne.0) goto 20
             IF(kkk.EQ.1)THEN
c               PSI=1.D-6
                psi=tolg
                MAXA(JEDN+1)=MAXA(JEDN)+1
C               CALL DRSWRR(A(LSK),MAXA,A(IROWS),JEDN,'SK U')
                CALL ICM(JEDN,NWK,NNZERO,NM,NMREJ,PSI,
     +                   A(LSK),MAXA,A(IROWS),A(LM0),A(LMaxM1),A(LColM),
     +                   A(Lr1),A(Lz1),A(Lp1))
                PRINT*, NWK, NNZERO, NM
C               CALL DRSWRR(A(LSK),MAXA,A(IROWS),JEDN,'SK I')
             ELSE
                CALL JEDNA1(A(Lr1),V,JEDN)
C               EPSILON=1.D-10
                EPSILON=1.D-8
                CALL ICM_CG(V,A(LSK),MAXA,A(IROWS),A(LM0),A(LMaxM1),
     +                   A(LColM),A(Lr1),A(Lz1),A(Lp1),
     +                   JEDN,NWK,NM,EPSILON,TOL,NITER)
20           ENDIF
          ELSE
             if(imumps.eq.1.or.imumps.eq.2) then
c               if(k.eq.2) then
                CALL MPI_BARRIER(PETSC_COMM_WORLD,ierr)
                if(imumps.eq.2) then
                     call pakpetsc(A(IROWS),A(IROWS+nwk),B,V,nwk,nn,k)
                else
                     CALL dmumps1(A(IROWS),A(IROWS+nwk),B,V,nwk,nn,k)
                end if
                IF (rank.ne.0) return
             else
                IF (rank.eq.0) CALL ICCGMA(B,V,MAXA,NWK,NN,
     1                                     K,IZLAZ,TOLG,ALFAG)
             endif
          ENDIF
       ELSE
```

Figure 2

These procedures contain source code written in FORTRAN. Basically they adapt matrix and vectors to format that can be used by solvers. After implementation both solvers ware tested on benchmark test. The accuracy of both solvers satisfied tolerance which is proposed by test itself.

6.ANALYSIS OF MUMPS AND PETSC SOLVERS

Implemented solvers have had to be tested for accuracy. We had benchmark problem (unit cube clamped at one side and loaded by unit pressure on the opposite side) for this purpose that have already was run on dozen of previous solvers, which were developed in ISLAB on Faculty of Mechanical Engineering in Kragujevac, and we had the results to compare with. Since previous solvers were sequential we tested both solvers sequentially.

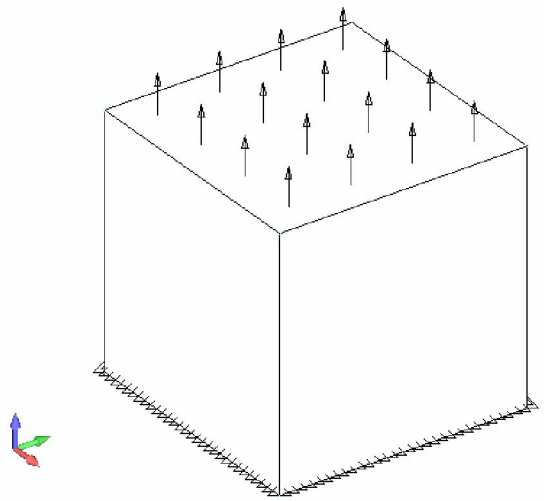

Figure 3

7.CONCLUSION

PAK software for structural analysis using Finite Element Method, was most useful with its old solvers, but the most interesting are its capabilities that are opened with usage of MUMPS and PETSc.

The advantages of using these solvers were numerous: Memory usage when using sparse matrices, speedup of calculation when using solver in parallel on cluster, speedup due to optimized solver code in sequential version, using PETSc internal functions for domain decomposition, etc.

It is very important to mention that MUMPS and PETSc are open source software, which imply that they have great community involved in their development and that they can be used free of charge.

Note: This paper is written in frame of Ministry of science and technologic development project.

8. REFERENCES

[1]Balay, Buschelman, Eijkhout, Kaushik, Knepley, McInnes, Smith and Zhang, *PETSc User Manual*, March 2010

[2] *MUltifrontal Massively Parallel Solver Users guide*, November 2009

[3] Miloš Kojić, Radovan Slavković, Miroslav Živković, Nenad Grujović, *Metod konačnih elemenata, Linearna analiza, Osnove nelinearne analize,* Faculty of Mechanical Engineering, University of Kragujevac

[4] Miloš Kojić, Radovan Slavković, Miroslav Živković, Nenad Grujović, *PAK-S, Program for FE Structural Analysis,* Faculty of Mechanical Engineering, University of Kragujevac