ENGINEERING TRANSACTIONS • *Engng. Trans.* • **65**, 1, 163–170, 2017
Polish Academy of Sciences • Institute of Fundamental Technological Research (IPPT PAN)
National Engineering School of Metz (ENIM) • Poznan University of Technology

Evaluation of Partial Factorization for Reduction of Finite Element Matrices

Paweł JARZĘBSKI, Krzysztof WIŚNIEWSKI

Institute of Fundamental Technological Research
Polish Academy of Sciences
Pawińskiego 5B, 02-106 Warsaw, Poland
e-mail: kwisn@ippt.pan.pl

In this paper, we present the concept of *Partial Factorization* [1] and discuss its possible applications to the Finite Element method. We consider: (1) reduction of the element tangent matrix, which is particularly important for mixed/enhanced elements and (2) reduction of the sub-domain matrices of the Domain Decomposition (DD) equation solvers run either sequentially on a single machine or in parallel on a cluster of computers. We demonstrate that *Partial Factorization* can be beneficial for these applications.

Key words: multi-scale models of multi-layer shells, mixed/enhanced finite elements, parallel computing, domain decomposition, solvers.

1. Introduction

In multi-scale computations of multi-layer shells, the finite element models can have a size of millions of degrees of freedom and must be repeated many times e.g. when a database of material properties is built for the Design Sensitivity Analysis. Therefore, it is necessary to improve efficiency of an finite element (FE) code, especially using concurrent capabilities of multi-core processors and multi-machine clusters.

The minimum is to use one of the available parallel solvers, such as PARDISO, MUMPS, PaStiX, HSL and others. The next step is a parallelization of computations of the FE matrices and vectors, e.g. using OpenMP. In the third, the so-called hybrid approach, additionally the domain decomposition (DD) is performed, e.g. by METIS, and computations for sub-domains are scattered over a cluster of computers, which requires, e.g. MPI.

This, however, requires several changes in the FE code and advanced computational techniques. In [2], we describe a parallelization of the loop over elements

of FEAP while in [3], a performance of the ompFEAP and the parallel solver HSL MA86 was established for two various composite materials.

An objective of this paper is to describe possible applications of *Partial Factorization* (PF) to FE computations. The PF was proposed in [1] for the stochastic optimization problems. We shall establish performance of this technique in two other applications: (1) a reduction of the element tangent matrix, which is particularly important for mixed/enhanced elements using a large number of additional parameters, and (2) a reduction of the sub-domain matrices to the ones expressed in terms of the interface variables for the DD equation solvers, run either sequentially on a single machine or in parallel on a cluster of computers.

2. Partial factorization

2.1. Schur complement for Domain Decomposition

It is well-known that the FE method yields the system of equations $\mathbf{Ku} = \mathbf{f}$ of a banded structure. If we compute these equations using the DD method, i.e. each sub-domain on a different node of a cluster of computers, then for each sub-domain i ($i = 1, ...n$) we have

$$\begin{bmatrix} \mathbf{K}_i & \mathbf{B}_i^{\mathrm{T}} \\ \mathbf{B}_i & \mathbf{C}_i \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_i^{\mathrm{I}} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_i \\ \mathbf{f}_i^{\mathrm{I}} \end{bmatrix}, \tag{2.1}$$

where we distinguish between the interface variables $\mathbf{u}_i^{\mathrm{I}}$ and the domain variables $\mathbf{u}_i$. Besides, we assumed that the matrix is symmetric. The domain variables must be eliminated within each domain, which yields

$$\mathbf{S}_i\, \mathbf{u}_i^{\mathrm{I}} = \mathbf{s}_i, \tag{2.2}$$

where

$$\mathbf{S}_i \doteq \mathbf{C}_i - \mathbf{B}_i \mathbf{K}_i^{-1} \mathbf{B}_i^{\mathrm{T}}, \qquad \mathbf{s}_i \doteq \mathbf{f}_i^{\mathrm{I}} - \mathbf{B}_i \mathbf{K}_i^{-1} \mathbf{f}_i. \tag{2.3}$$

Note that the matrix $\mathbf{S}_i$ is the Schur complement of $\mathbf{K}_i$. Next step is performed for all domains taken together, for which we have to assemble and solve the system of equations for the interface variables,

$$\mathbf{S}\mathbf{u}^{\mathrm{I}} = \mathbf{s}, \tag{2.4}$$

where

$$\mathbf{S} \doteq \sum_{i=1}^{n} \mathbf{S}_i, \qquad \mathbf{s} \doteq \sum_{i=1}^{n} \mathbf{s}_i, \qquad \mathbf{u}^{\mathrm{I}} \doteq \sum_{i=1}^{n} \mathbf{u}_i^{\mathrm{I}}. \tag{2.5}$$

Having obtained the interface variables $\mathbf{u}^{\mathrm{I}}$, we compute $\mathbf{u}_i$ for each domain separately, which ends the solution process.

Note that computations of $\mathbf{S}_i$, $\mathbf{s}_i$ and $\mathbf{u}_i$ can be performed in parallel in sub-domains, which can significantly shorten the solution time. Most time-consuming is computation of the Schur complement $\mathbf{S}_i$ in sub-domains, especially of the product $\mathbf{K}_i^{-1}\mathbf{B}_i^{\mathrm{T}}$. Classically this product is obtained in two steps: (1) factorization of $\mathbf{K}_i$, and (2) one back-substitution for each column of $\mathbf{B}^{\mathrm{T}}$. This, however, is very inefficient and e.g. the speedup on 12 cores is only about 3, see [4]. Much faster is the method presented in the next section.

2.2. Partial factorization and Schur complement

The LU decomposition algorithm for factorization of a matrix has acceptable scalability on multicore machines and this feature is preserved when we use PF to compute the Schur complement. The PF method was proposed in [1] for the stochastic optimization problems; in the current paper we propose its other applications.

The LU decomposition of the matrix of Eq. (2.1) can be written as follows:

$$\begin{bmatrix} \mathbf{K}_i & \mathbf{B}_i^{\mathrm{T}} \\ \mathbf{B}_i & \mathbf{C}_i \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} \\ \mathbf{0} & \mathbf{U}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{11}\mathbf{U}_{11} & \mathbf{L}_{11}\mathbf{U}_{12} \\ \mathbf{L}_{21}\mathbf{U}_{11} & \mathbf{L}_{21}\mathbf{U}_{12} + \mathbf{L}_{22}\mathbf{U}_{22} \end{bmatrix}, \tag{2.6}$$

where $\mathbf{L}_{ij}$ and $\mathbf{U}_{ij}$ are the block factors of the lower and upper triangular parts, respectively. Noting that $\mathbf{L}_{21}\mathbf{U}_{11} = \mathbf{B}_i$ and $\mathbf{L}_{11}\mathbf{U}_{12} = \mathbf{B}_i^{\mathrm{T}}$, we can calculate $\mathbf{L}_{21}$ and $\mathbf{U}_{12}$ first. Then, $\mathbf{L}_{21}\mathbf{U}_{12} = \mathbf{B}_i\mathbf{U}_{11}^{-1}\mathbf{L}_{11}^{-1}\mathbf{B}_i^{\mathrm{T}} = \mathbf{B}_i\mathbf{K}_i^{-1}\mathbf{B}_i^{\mathrm{T}}$ by $\mathbf{L}_{11}\mathbf{U}_{11} = \mathbf{K}_i$. We note that

$$\mathbf{C}_i - \mathbf{L}_{21}\mathbf{U}_{12} = \mathbf{C}_i - \mathbf{B}_i\mathbf{K}_i^{-1}\mathbf{B}_i^{\mathrm{T}} = \mathbf{S}_i, \tag{2.7}$$

i.e. the Schur complement of Eq. (2.3). Note that we do not need to compute $\mathbf{L}_{22}$ and $\mathbf{U}_{22}$ to obtain the Schur complement, which justifies the term "partial" factorization.

3. Reduction of element's matrix

3.1. Set of equations for problems with additional variables

Mixed/enhanced finite elements involve additional local parameters which must be condensed out to reduce the size of a tangent element matrix to the standard one. The number of element parameters can be quite large, sometimes exceeding the number of nodal variables associated with the element, and one of the approaches is to group and condense out all parameters together, which amounts to computation of the Schur complement.

For the considered class of finite elements, the governing functional F depends on the nodal displacements (and rotational parameters) designated as $\mathbf{u}_I$ and the elemental multipliers $\mathbf{q}$. In general, $\mathbf{q}$ are the Lagrange multipliers (including the multipliers of stress modes) and the multipliers of enhancing kinematical modes, used in the enhanced strain methods, such as: Incompatible Displacements or EADG or EAS method.

For kinematically non-linear problems, the stationarity condition of $F(\mathbf{u}_I, \mathbf{q})$ yields a system of equilibrium equations for an element, $\mathbf{r}_u(\mathbf{u}_I, \mathbf{q}) = \mathbf{0}$ and $\mathbf{r}_q(\mathbf{u}_I, \mathbf{q}) = \mathbf{0}$. The linearized (Newton) form of these equations is as follows:

$$\begin{bmatrix} \mathbf{K} & \mathbf{L} \\ \mathbf{L}^{\mathrm{T}} & \mathbf{K}_{qq} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u}_I \\ \Delta\mathbf{q} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_u \\ \mathbf{r}_q \end{bmatrix}, \tag{3.1}$$

where $\mathbf{K} \doteq \partial\mathbf{r}_u/\partial\mathbf{u}_I$, $\mathbf{L} \doteq \partial\mathbf{r}_u/\partial\mathbf{q}$ and $\mathbf{K}_{qq} \doteq \partial\mathbf{r}_q/\partial\mathbf{q}$. Note that $\mathbf{K}$ and $\mathbf{K}_{qq}$ are symmetric and, in general, indefinite.

To eliminate the multipliers $\Delta\mathbf{q}$ at the element level and to reduce the size of a tangent element's matrix to the standard one, defined by the number of nodes on the element and dofs/node, we calculate $\Delta\mathbf{q}$ from the second equation and use it in the first one, which yields

$$\begin{gathered} \mathbf{K}^*\Delta\mathbf{u}_I = -\mathbf{r}^*, \\ \text{where} \qquad \mathbf{K}^* = \mathbf{K} - \mathbf{L}\mathbf{K}_{qq}^{-1}\mathbf{L}^{\mathrm{T}} \qquad \text{and} \qquad \mathbf{r}^* = \mathbf{r}_u - \mathbf{L}\mathbf{K}_{qq}^{-1}\mathbf{r}_q. \end{gathered} \tag{3.2}$$

Comparing to Eq. (2.3), we see that the reduced (or condensed) matrix $\mathbf{K}^*$ is defined as the Schur complement of $\mathbf{K}_{qq}$. Hence, the PF method described in Subsec. 2.2 can be applied to speed up the above condensation. Below, we test and compare its performance for three types of finite elements.

3.2. Numerical results

Tests were performed using only 1 core of a multi-core machine (2 processors Xeon X5650 2.66 GHz with 6 cores each, running under Linux). This is in accord with parallelization of a loop over elements in FEAP, see [2] for details, by which each core processes a different finite element.

The stiffness matrices used in computations were obtained for the central (irregular) elements of the patch tests described in [5]. The computations were repeated 1 million times for each matrix. The speedups are presented in Table 1, where the best results for each matrix are boldfaced.

Solver and method. For reference, we used the scheme of Eq. (3.2) and the LUDCMP routine of Numerical Recipes [6], which performs the *LU* factorization

Table 1. Speedup for particular solvers and methods.

Solver	Method	Shell elements			Solid-shell elements			3D elements		
		EAS10	HW35	HW43	EAS10	HW29	HW47	EADG12	EAS30	HW60
DSYTRF	1RHS	0.26	0.48	0.68	0.30	0.56	0.60	0.29	0.45	0.75
MA64	1RHS	0.57	1.02	1.36	0.72	1.37	1.51	0.84	1.23	1.74
DSYTRF	mRHS	0.97	1.72	2.06	1.04	1.93	1.82	1.12	1.49	2.20
MA64	mRHS	0.62	1.04	1.37	0.77	1.39	1.82	0.90	1.26	1.75
DSYTRF	ownPF	**1.27**	**2.47**	**3.37**	**1.37**	**2.64**	**2.52**	**1.57**	**1.86**	**2.41**
MA64	PF	0.58	1.41	1.91	0.72	1.90	2.10	0.85	1.49	2.31
Matrix density [%]		91.20	100.00	58.70	98.40	48.20	29.20	40.60	22.20	13.70
Reference time [s]		6.77	49.96	129.08	10.46	75.19	108.19	13.34	53.88	192.54

using the Crout method; the times obtained are designated "Reference time" in Table 1. The speedup is computed as a quotient $\text{time}_{\text{method}}/\text{time}_{\text{ref}}$. Besides, we tested two routines based on the Gauss elimination: DSYTRF of LAPACK [7] and MA64 of HSL [8]. To obtain $\mathbf{K}_{qq}^{-1}\mathbf{L}^{\mathrm{T}}$, the back-substitution routine is called in two ways: either for each column of $\mathbf{L}^{\mathrm{T}}$ separately ("1RHS"), or for all columns of $\mathbf{L}^{\mathrm{T}}$ together ("mRHS"). Besides, "ownPF" indicates our modification of the code to perform PF. (Note that the PF is not suitable for the Crout method.)

Finite elements. Three types of elements were tested: 4-node shells, 8-node solid-shells and 8-node 3D elements. Their formulations are designated by: HW – based on the Hu-Washizu functional and enhanced for shell and solid-shell elements [9, 10], EAS – based on the potential energy with the Enhanced Assumed Strain, and EADG – based on the potential energy with Enhanced Assumed Displacement Gradient. The number of additional parameters follows these letters. The results presented in Table 1 indicate that PF is beneficial for all elements and that the solver "DSYTRF" and the method "ownPF" provide the best speedup.

4. Domain decomposition solvers using PF

4.1. Sequential solver for single machine

The parallel algorithm for solving a system of equations of Subsec. 2 can also be performed sequentially on a single machine. This reduces the memory usage, thus allowing to compute bigger examples.

We divide the domain into two sub-domains. The memory usage is reduced because the fill-in of the sub-domain matrix of Eq. (2.1) is less than half of that

for the whole matrix. However, after factorization of the first sub-domain matrix we have to deallocate the memory to compute the second sub-domain matrix. Hence, we need to factorize this matrix once more after solving Eq. (2.4), which increases the time of computation. We used the above method with two solvers: either Pardiso or modMA86, which is the HSL MA86 solver with our implementation of PF.

To test this method we computed a linear example of an elastic cube, loaded by uniform unit forces on the top face and fixed at the bottom face. A mesh of $N \times N \times N$ 3D 8-node standard elements was used with $N = 64$. The results presented in Table 2 indicate that the new method saves around 32–38% memory but is about 37–40% slower than the standard one. We see that each of the solvers tested has its merits.

Table 2. Memory and time usage by a sequential DD solver.

Solver	Memory [GB]			Time [s]		
	Standard	New	Change [%]	Standard	New	Change [%]
Pardiso	14.16	8.71	−38	210.98	288.90	+37
modMA86	14.47	9.91	−32	156.07	218.83	+40

4.2. Parallel solver for cluster of computers

The parallel algorithm for solving a system of equations of Sec. 2 is naturally suited for parallel solution on a cluster of computers. In each sub-domain we use the modMA86 solver, which is the HSL MA86 solver with our implementation of PF. Each sub-domain is computed by different computational node of a cluster. On each node we use multithreaded version of modMA86 solver with use of 12 threads. The equation for interfaces (Eq. (2.4)) is solved using the solver for dense matrices MA64 of HSL. Additional special treatment is necessary for the cases of 4 and 8 nodes; more details on our implementation can be found in [4].

Our solver is compared to the distributed version of WSMP solver of IBM, see [11], by computations on 2, 4, and 8 computational nodes of the type characterized in Subsec. 4.1. The speedup is shown according to run on 1 node with 12 threads. Numerical results for the cube example (the same as in Subsec. 4.1) are presented in Figs. 1a and 1b. We note that our solver compares favorably with WSMP (it uses also 14% less memory). An additional speedup is provided by the OMP parallelization of the MA86 solver which, on the 12-core machine, is about 9.1. Hence, the total speedup provided by our cluster solver is about 24.4.

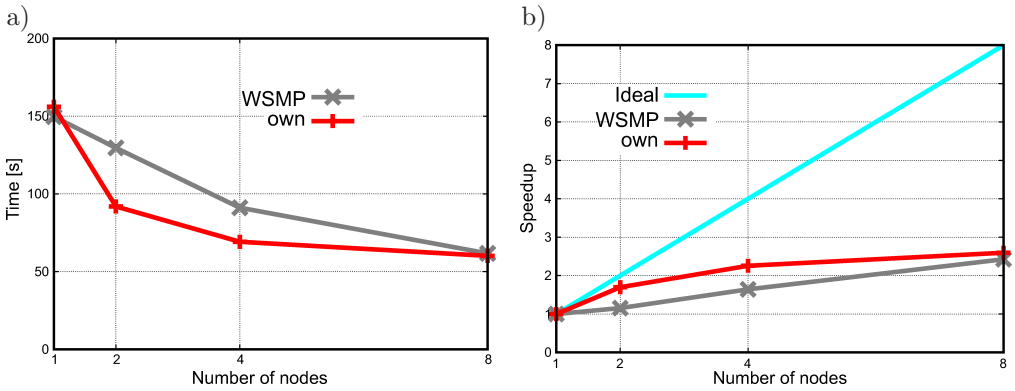

FIG. 1. Comparison of time (a) and speedup (b) of cluster solvers.

5. CONCLUSIONS

We applied the PF technique to two problems involving the matrix reduction and for both the use of this technique has proven beneficial. For the first problem (Sec. 3), it saves time and the best speedup is provided when it is combined with the solver "DSYTRF". For the second problem (Sec. 4), it either saves 32–38% of memory, when run on a single machine, or provides the total speedup 24.4 on the cluster.

REFERENCES

1. PETRA C.G. *et al.*, *An augmented incomplete factorization approach for computing the Schur complement in stochastic optimization*, SIAM J. Sci. Comput., **36**(2): C139–C162, 2014.

2. JARZĘBSKI P., WIŚNIEWSKI K., TAYLOR R.L., *On paralelization of the loop over elements in FEAP*, Computational Mechanics, **56**(1): 77–86, 2015.

3. JARZĘBSKI P., WIŚNIEWSKI K., *Performance of the parallel FEAP in calculations of effective material properties using RVE*, [in:] *Advances in Mechanics*, KLEIBER M. *et al.* [Eds.], Taylor & Francis, London, pp. 241–244, 2016.

4. JARZĘBSKI P., WIŚNIEWSKI K., *Application of partial factorization for domain decomposition solver*, In preparation, 2016.

5. MACNEAL R.H., HARDER R.L., *A proposed standard set of problems to test finite element accuracy*, Finite Element in Analysis and Design, **1**: 3–20, 1985.

6. PRESS W.H. *et al.*, *Numerical Recipes in Fortran 77*, Cambridge Univeristy Press, 1999.

7. ANDERSON E. *et al.*, *LAPACK Users' Guide*, SIAM, Philadelphia, 1999.

8. HSL 2013, *A collection of Fortran codes for large scale scientific computation*, http://www.hsl.rl.ac.uk/.

9. Wiśniewski K., *Finite Rotation Shells. Basic Equations and Finite Elements for Reissner Kinematics*, Springer, 2010.

10. Wiśniewski K., Turska E., *Four-node mixed Hu-Washizu shell element with drilling rotation*, Int. J. Num. Meth. Engng., **90**(4): 506–536, 2012.

11. Gupta A., *WSMP: Watson Sparse Matrix Package*, IBM Research Report, Watson, 2015.

Received October 24, 2016; accepted version January 26, 2017.