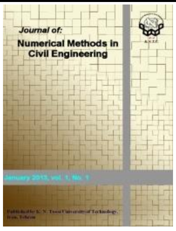

Numerical Methods in Civil Engineering

Reducing computational and memory cost of substructuring technique in finite element models

Mohammad Hadi Kadkhodaee* and Hamid Moslemi**

ARTICLE INFO

Article history:
Received:

Revised:

Accepted:

Keywords:
Substructuring;
Finite element method;
Structural optimization;
Computational cost;
Memory cost.

Abstract:

Substructuring in the finite element method is a technique that reduces computational cost and memory usage for analysis of complex structures. The efficiency of this technique depends on the number of substructures in different problems. Some subdivisions increase computational cost, but require little memory usage and vice versa. In the present study, the cost functions of computations and memory usage are extracted in terms of number of subdivisions and optimized mathematically. The results are presented in the form of tables which recommend the proper substructuring for different number of elements. A combined case is also considered which investigates balanced reduction of computational and memory cost 2D problems. Several numerical examples are analyzed numerically to demonstrate the abilities and efficiency of the proposed computational algorithm for structured and unstructured mesh.

1. Introduction

The accurate analysis and design of large and complex structures remains a challenging`   g task for engineers. Major advances in fast computing technologies have encouraged engineers to consider more complex constitutive models in analysis of structures. The finite element method (FEM) is the most common method used and has played a key role in the development.

Engineers are increasingly interested in accurate analysis and consideration of the nonlinear condition, large deformations, and cases where the approximation is reduced. Where large and highly-complex structures are involved, analysis can take hours and even days. Software producers continually endeavor to reduce analysis time of complex structures. One method of reducing the amount of computation is the technique of substructuring in which a large structure is subdivided into smaller parts that can be analyzed separately. Przemieniecki (1963)[17] first proposed this method for first-level breakdown of complex systems such as a complete airplane for the displacement and force method. The advent of supercomputers has further advanced substructuring technology.

Computational and memory cost confine substructuring technique. The effect of varying the block size on a number of arithmetic operations and storage requirements was investigated by Noor et al. (1978)[14]. They compared multi-level substructuring with the direct method and found that, as the number of substructuring levels increased, the number of arithmetic operations and disk storage requirements decreased. Gurujee and Deshpande (1978)[7] improved substructure analysis method specifically for structures incurring large expense in one direction, such as multi-storied buildings and communication towers. This method reduced the number of arithmetic operations involved and memory space used. Fonseka(1993)[6] reported that the technique could use fixed-sized arrays in the computer program irrespective of the size of the substructure, thus allowing optimal use of computer memory to incorporate substructures into shells of revolution.

Parallel processors opened a challenging area in substructuring to facilitate assignment of substructures to different processors. Kaveh and Roosta (1995)[9] used

* *MSc, Department of Civil Engineering, Shahed University, Tehran, Iran.*
***Corresponding Author: Assistant Professor, Civil Engineering Department, Faculty of Engineering, Shahed University, Tehran, Iran. Email: h.moslemi@shahed.ac.ir*

graph theory to optimize decomposition and proposed a set of balanced subdomains to ensure that the overall computational load be as evenly distributed as possible between processors. Kaveh (2014)[8] minimized the number of interface nodes to reduce the cost of synchronization and/or message-passing between processors. Farhat et al. (1995)[4] proposed subdomains with aspect ratios to improve the convergence rate of domain decomposition based the iterative method and demonstrated that bad element aspect ratios result in poorly-conditioned operators. A simple and efficient algorithm for automatic domain decomposition was proposed by Farhat (1988)[3], who applied it to both regular and irregular two- and three-dimensional finite element mesh. The algorithm was improved by introducing finite element tearing and interconnecting (FETI) requiring less interprocessor communication than does the classical method of substructuring and is suitable for parallel/vector computers with shared memory (Farhat and Roux 1991[5]).

Vanderstraetena and Keunings (1995)[19] proposed optimized partitioning of unstructured mesh in a two-step approach that combines a direct partitioning scheme with a non-deterministic procedure of combinatorial optimization. In the first step, direct partitioning is used to produce initial decomposition of reasonable quality. In the second step, optimization is used to improve on the initial partition. A cost function is introduced that takes into account the interface size and computes the load imbalance between subdomains. Wang et al. (1999)[20] proposed a mixed formulation of the substructure synthesis method in terms of the physics-impedance-modal parameter. This formulation was based on the concept of the parameter-mixed synthesis. A multilevel structural method was implemented by Yang et al. (2011)[21] to reduce the time needed to solve the interface equation system and improve the overall efficiency of parallel substructure finite element analysis. The multilevel approach reduced up to 50% of the time needed for solution of the interface equation system and improved the overall efficiency of parallel substructuring up to 40% in numerical examples.

Substructuring is now used in a variety of applications. Li and Hao (2013)[10] used substructuring to study progressive collapse and for blast loads and have presented a numerical approach with numerical condensation for an efficient simulation of structural response. This approach saves up to 54% of computational time, but the study did not investigate memory sparing in detail. Shen and Yin (2014)[18] proposed a dynamic substructure computational procedure for analysis of impact-induced stress waves in a non-uniform flexible structure and determined the sufficient number of substructures for this purpose. Njomo and Ozay (2014)[13] applied substructuring to sequential analysis modeled on construction. The proposed model produced more accurate results with minimal computer memory and reduced time spent by determining the optimal size of the substructure. Predari et al. (2016)[16] modeled additional constraints with fixed vertices by a direct k-way greedy graph growing partitioning that properly handles fixed vertices. A multilevel tabu search algorithm for balanced partitioning of unstructured grids proposed by Mehrdoost and Bahrainian (2016)[11]. Boo and Oh (2017)[2] introduced automated static condensation method, which was developed for the local analysis of large finite element models. A substructural tree diagram and substructural sets were established in such a way that the omitted substructures were sequentially condensed into the retained substructure to construct the reduced model. A layer-by-layer partitioning of finite element meshes for multicore architecture was presented by Novikov et al. (2017)[15] using a neighborhood criterion to partition the mesh into layers and combining them into blocks and assigning them into different parallel processors. Badia and Verdugo (2018)[1] investigated the use of domain decomposition preconditioners for unfitted finite element methods such as extended finite element method defining the coarse degrees of freedom in the definition of the preconditioner.

Previous studies have been based on a specified number of substructures with the aim of dividing a structure to proper substructures. The number of substructures as the cost parameter, however, also plays a major role in reducing computation and time of analysis. In extreme cases where an entire structure is used as a substructure of which each element is taken as a substructure, this technique does not reduce computational cost. If the proper number of substructures is employed, the computational cost can be reduced as much as possible. The present paper optimized the number of substructures for computational cost and memory by counting FLOP and the need of the memory for data, respectively. The path of partitioning was structured and the size of the substructures was kept as similar as possible. Finding the optimum partitioning path is not investigated in this paper and the optimum number of substructures is studied. Since the nonlinear and dynamic analysis of structures consists of iterative solution of linear governing equations, the proposed algorithm can be employed in nonlinear and dynamic problems.

Section 2 presents classic substructuring theory. Section 3 calculates the computational cost of operations in substructuring with respect to the number of substructures and optimizes them mathematically. In Section 4, the required memory size is computed and optimized with respect to the number of substructures. Section 5 shows that these two optimizations can be combined, depending on the importance of computational cost and memory size for different cases. Section 5 also illustrates the ability and efficiency of the proposed approach using several numerical examples. Section 6 presents the concluding remarks.

2. Substructuring

Przemieniecki (1963)[17] first proposed substructuring for the displacement and force methods. In this technique, the structure is divided into substructures with each substructure containing several elements. The degrees of freedom (DOFs) of a substructure are classified as:

- Internal DOFs: not connected to the DOFs of any other substructure
- Boundary DOFs: connected to at least one other substructure; these usually reside at the boundary nodes placed on the periphery of the substructure

If the equilibrium equation is written in boundary DOFs, the objective will be to eliminate all DOFs associated with internal freedoms. This elimination process is called static condensation or, simply, condensation. The static condensation method assumes that those internal DOFs that can be condensed are arranged in the first *i* DOFs and the remaining boundary DOFs are the last *b* nodal coordinates. This arrangement allows the governing equation for a structure to be written using partitioned matrices as:

$$\begin{bmatrix} [K]_{ii} & [K]_{ib} \\ [K]_{bi} & [K]_{bb} \end{bmatrix} \begin{Bmatrix} \{u_i\} \\ \{u_b\} \end{Bmatrix} = \begin{Bmatrix} \{F_i\} \\ \{F_b\} \end{Bmatrix} \quad (1)$$

where subscripts *i* and *b* represent the internal and boundary DOFs, respectively. A simple multiplication of the partitioned system in Eq. (1) yields the following two matrix equations:

$$[K]_{ii}\{u\}_i + [K]_{ib}\{u\}_b = \{F\}_i \quad (2)$$

$$[K]_{bi}\{u\}_i + [K]_{bb}\{u\}_b = \{F\}_b \quad (3)$$

Solving Eq. (2) for $\{u\}_i$ and substituting it into Eq. (3) arrives at:

$$\{u\}_i = [K]_{ii}^{-1}\left(\{F\}_i - [K]_{ib}\{u\}_b\right) \quad (4)$$

$$\left(\{F\}_b - [K]_{bi}[K]_{ii}^{-1}\{F\}_i\right) = \left([K]_{bb} - [K]_{bi}[K]_{ii}^{-1}[K]_{ib}\right)\{u\}_b \quad (5)$$

Eq. (5) may be written as:

$$\{\overline{F}\} = [\overline{K}]\{u\}_b \quad (6)$$

in which:

$$\{\overline{F}\} = \{\overline{F_b}\} - [K]_{bi}[K]_{ii}^{-1}\{F\}_i \quad (7)$$

and:

$$[\overline{K}] = [K]_{bb} - [K]_{bi}[K]_{ii}^{-1}[K]_{ib} \quad (8)$$

In Eqs. (7) and (8), $[\overline{K}]$, $\{\overline{F}\}$ are the condensed stiffness matrix and force vector, respectively. This technique produces a condensed stiffness matrix and a condensed force vector for each substructure which are associated only with the boundary DOFs. Assume that each substructure is equivalent to an element having stiffness matrix and nodal force, $[\overline{K}]$, $\{\overline{F}\}$ respectively. Classic FEM states that the condensed equations of substructures must be assembled to obtain the condensed governing equation of the whole structure for the total boundary DOFs as:

$$[\overline{K}]_t\{u\}_{b_t} = \{\overline{F}\}_t \quad (9)$$

in which $[\overline{K}]_t$, $\{\overline{F}\}_t$ are the assembled $[\overline{K}]$ and $\{\overline{F}\}$, respectively. Since Eq. (9) is the partitioned form of Eq. (1), the coefficient matrix would not be singular.

This obtains the solution to the Eq. (9) boundary DOFs. By substituting boundary DOFs associated with each substructure into Eq. (4), the internal DOFs in each substructure are computed.

3. Optimization of computational cost

Pr The main computational operations in substructuring include static condensation for each substructure, solving Eq. (9) for total boundary DOFs and substituting it into Eq. (4) to solve for the internal DOFs of each substructure. The parameters considered are the variable of cost functions such as analysis time and the number of floating point operations.

In computing, the floating-point operation per second (FLOPS) is a measure of computer performance and is useful for calculations that have heavy floating-point calculations. In general, as the FLOPS count of the algorithm increases, the analysis time increases. For simple operations such as addition, subtraction, multiplication, and division, FLOP count is considered to be a unit. FLOP count operations such as matrix multiplication and solving a system of linear equations should be calculated based on simple operations.

The FLOP count of the matrix operations were calculated in the lightspeed MATLAB toolbox and are summarized in Table 1 for the size of matrices. The system of linear equations is assumed to be solved using LU decomposition algorithm. Table 1 can be applied for substructuring with the corresponding sizes of the matrices. Table 2 shows computation of the FLOP count for each stage of substructuring where y, x, and z represent the number of boundary degrees of freedom of total of the substructures, internal and boundary degrees of each substructure respectively.

Table.1: FLOP count for matrix operations

Count FLOP	Matrix Operation
$f=(2m-1)nc$	$t=a_{n\times m}\times b_{m\times c}$
$f=m^3+(2n-1.5)m^2+(14n+0.5)m-8$	$b=a_{m\times m}\backslash t_{m\times n}$

Table 2: FLOP count for stages of substructuring

Equation	Operations	FLOP
(7)	$A_{(y\times 1)}=[K]_{ii\,(y\times y)}\backslash\{F\}_{i\,(y\times 1)}$	$f=y^3+0.5y^2+14.5y-8$
	$B_{(z\times 1)}=[K]_{bi\,(z\times y)}\times A_{(y\times 1)}$	$f=(2y-1)z$
(8)	$A_{(y\times z)}=[K]_{ii_{(y\times y)}}\backslash[K]_{ib_{(y\times z)}}$	$f=y^3+(2z-1.5)y^2+(14z+0.5)y-8$
	$B_{(z\times z)}=[K]_{bi\,(z\times y)}\times A_{(y\times z)}$	$f=(2y-1)z^2$
(9)	$\left(\{u\}_{b_t}\right)_{(x\times 1)}=\left(\left[\overline{K}\right]_t\right)_{(x\times x)}\backslash\left(\left\{\overline{F}\right\}_t\right)_{(x\times 1)}$	$f=x^3+0.5x^2+14.5x-8$
(4)	$A_{(y\times 1)}=\{F\}_{i\,(y\times 1)}-[K]_{ib\,(y\times z)}\{u\}_{b\,(z\times 1)}$	$f=(2z-1)y$
	$B_{(y\times 1)}=[K]_{ii\,(y\times y)}^{-1}\times A_{(y\times 1)}$	$f=y^3+0.5y^2+14.5y-8$

Table 2 shows that total computational cost is the sum of the cost functions as follows:

- computational cost for equilibrium equation of total boundary DOFs:

$$f1(x)=x^3+0.5x^2+14.5x-8 \quad (10)$$

- computational cost for equilibrium equation in substructures:

$$f2(y,z)=(3y^3+(2z-0.5)y^2+(14z+19.5)y-8)nsub \quad (11)$$

- computational cost for multiplication of matrix in substructures:

$$f3(y,z)=(2yz^2-z^2+4yz-y-z)nsub \quad (12)$$

where *nsub* indicates the number of substructures.

For structured meshes in 2D problems, the number of elements in each direction are assumed to be *m* and *n*. The number of substructures in each direction are denoted as *na* and *nb*. Parameters *x*, *y*, and *z* can be expressed simply in terms of *na* and *nb*. The complexity of the computational cost function means that optimization does not lead to an explicit solution, but can be solved numerically with a specified number of *m* and *n*. There are different numbers of divisions (*m*, *n*) for different problems and the optimal size of substructure (*na*, *nb*) for some cases are summarized in Tables 3 and 4. It can be seen from these tables that square subdivisions are suitable for minimization of computational cost.

4. Optimization of memory cost

The size of the memory required in FEM confines the analysis of structures with dense mesh. Substructuring reduces the amount of memory needed considerably, but the savings is dependent on the number of substructures. In substructuring, most of the memory is used to save two classes of matrices: the stiffness of each substructure *ksub* ($\left[\overline{K}\right]_t$ in Eq. 9) and the stiffness of the total boundary nodes *kmain* ($[K_{ii}]$ in Eq. 4). The memory used for other matrices could be neglected. It is assumed that all of the analysis operations have been accomplished using the main memory (RAM) of the computer. In large FE models the operating system may use the Hard Disk Drive (HDD) of the computer to simulate RAM and the initial assumption may be violated. As the number of divisions increase, the memory needed for *kmain* increases and the memory for *ksub* decreases. If a low number of subdivisions is chosen, however, a large amount of memory will be needed for *ksub* and less memory for *kmain*. This demonstrated that there is optimal substructuring between these cases. When the matrix is saved directly in the memory, considering the symmetry of the stiffness matrix, the memory cost for *ksub* and *kmain* can be computed as:

memory needed for *ksub*:

$$\frac{y^2+y}{2}\times nsub \quad (13)$$

memory needed for *kmain*:

$$\frac{x^2+x}{2} \quad (14)$$

Table 3: Optimal number of substructures versus number of elements for computational cost (up to 100 elements in each direction)

m / n	10	20	30	40	50	60	70	80	90	100
10	3×3	2×5	2×6	2×7	2×8	2×9	2×10	2×10	2×11	2×11
20	5×2	4×4	3×5	3×6	3×7	3×7	3×8	2×11	2×12	2×12
30	6×2	5×3	4×5	4×5	4×6	3×8	3×8	3×9	3×9	3×10
40	7×2	6×3	5×4	5×5	5×5	4×6	4×7	4×8	4×8	4×8
50	8×2	7×3	6×4	5×5	5×5	5×6	5×6	4×8	4×8	4×9
60	9×2	7×3	8×3	6×4	6×5	5×6	5×7	5×7	5×7	5×8
70	10×2	8×3	8×3	7×4	6×5	7×5	6×6	6×6	5×8	5×8
80	10×2	11×2	9×3	8×4	8×4	7×5	6×6	6×6	6×7	6×7
90	11×2	12×2	9×3	8×4	8×4	7×5	8×5	7×6	6×7	6×7
100	11×2	12×2	10×3	8×4	9×4	8×5	8×5	7×6	7×6	7×7

Table 4: Optimal number of substructures versus number of elements for computational cost (>100 elements in each direction)

m / n	100	200	300	400	500	600	700	800	900	1000
100	7×7	6×11	5×15	5×17	4×23	4×25	4×27	4×29	4×31	4×32
200	11×6	9×9	8×12	7×16	7×17	7×19	6×23	6×25	6×26	6×27
300	15×5	12×8	11×11	10×13	9×16	9×17	8×20	8×22	8×23	8×24
400	17×5	16×7	13×10	12×12	11×14	11×16	10×18	10×20	10×21	9×24
500	23×4	17×7	16×9	14×11	13×13	13×15	12×17	12×18	11×20	11×22
600	25×4	19×7	17×9	16×11	15×13	14×15	14×16	13×18	13×19	12×21
100	27×4	23×6	20×8	18×10	17×12	16×14	15×16	15×17	14×18	14×19
800	29×4	25×6	22×8	20×10	18×12	18×13	17×15	16×16	16×17	15×19
900	31×4	26×6	23×8	21×10	20×11	19×13	18×14	17×16	17×17	17×18
1000	32×4	27×6	24×8	24×9	22×11	21×12	19×14	19×15	18×17	18×18

where y and x are the number of internal DOFs in each substructure and the number of total boundary DOFs, respectively. Similar to computational cost, memory cost also can be optimized using substructures of the proper size. Tables 5 and 6 show the optimal subdivisions for different number of elements in the problem.

5. Combinational memory and FLOPS cost optimization

Sections 3 and 4 showed that the optimal subdivisions for FLOPS and memory cost are not necessarily equal. The importance of each of these two factors leads to the selection of a proper substructure. At times when the amount of memory is insufficient for analysis, memory optimization becomes more important. At such times when analysis is very time-consuming, the focus should be on FLOPS cost optimization.

These two optimizations can be combined by weighting each of them. FLOPS and memory cost are different types of cost and cannot be directly combined; they must be first normalized with respect to the optimal FLOPS and memory and then combined normalized FLOPS cost for substructuring is the ratio of FLOPS cost of substructuring to the optimal FLOPS cost of the problem (Section 3). Normalized memory cost is the ratio of the memory cost of the specified substructuring to the optimal memory cost (Section 4). These two dimensionless costs can be combined with a proper weighting factor.

In this case, the normalized combined cost is:

$$\begin{aligned}&(\text{normalized memory cost}) \times r\\&+ (\text{normalized FLOPS cost}) \times (1-r)\end{aligned} \qquad (15)$$

where r is a weighting factor between 0 and 1 which reflects the importance of memory cost in the problem. For $r = 0.5$, which has equal values for time and memory, the optimal subdivisions are shown in Tables 7 and 8. Since the cost functions are nonlinear functions, this linear combination will result different substructuring in the combinational case.

6. Numerical examples

Several examples were analyzed numerically to illustrate the accuracy and efficiency of proposed optimization strategy with the substructuring technique described in Sections 3,4 and 5. FEM was implemented using the standard isoparametric quadrilateral elements with four Gauss points. In all of the examples the elasticity modulus of

Table 5: Optimal number of substructures versus number of elements for memory cost (up to 100 elements in each direction)

n \ m	10	20	30	40	50	60	70	80	90	100
10	3×3	3×5	2×7	2×8	2×10	2×10	2×12	2×13	2×13	2×14
20	5×3	4×4	4×5	3×7	3×8	3×9	3×10	3×10	3×11	3×12
30	7×2	5×4	5×5	4×6	4×7	4×8	4×8	4×9	3×11	3×12
40	8×2	7×3	6×4	5×5	5×6	5×7	5×7	4×9	4×10	4×10
50	10×2	8×3	7×4	6×5	6×6	5×7	5×8	5×8	5×9	5×9
60	10×2	9×3	8×4	7×5	7×5	6×6	6×7	6×8	6×8	5×9
70	12×2	10×3	8×4	7×5	8×5	7×6	7×7	7×7	6×8	6×9
80	13×2	10×3	9×4	9×4	8×5	8×6	7×7	7×7	7×8	7×8
90	13×2	11×3	11×3	10×4	9×5	8×6	8×6	8×7	7×8	7×8
100	14×2	12×3	12×3	10×4	9×5	9×5	9×6	8×7	8×7	8×8

Table 6: Optimal number of substructures versus number of elements for memory cost (>100 elements in each direction)

n \ m	100	200	300	400	500	600	700	800	900	1000
100	8×8	6×14	6×17	5×22	5×26	5×29	5×32	5×34	4×41	4×44
200	12×7	11×11	10×14	9×18	8×22	8×24	8×27	7×31	7×33	7×35
300	17×6	14×10	13×13	12×16	11×19	11×21	10×25	10×27	10×29	10×31
400	22×5	18×9	16×12	15×15	14×18	13×20	13×22	12×25	12×27	12×29
500	26×5	22×8	19×11	18×14	16×17	16×19	15×21	15×23	14×26	14×27
600	29×5	24×8	21×11	20×13	19×16	18×18	17×20	17×22	16×25	16×26
700	32×5	27×8	25×10	22×13	21×15	20×17	19×20	19×21	18×24	18×25
800	34×5	31×7	27×10	25×12	23×15	22×17	21×19	21×21	20×23	20×24
900	41×4	33×7	29×10	27×12	26×14	25×16	24×18	23×20	22×22	21×24
1000	44×4	35×7	31×10	29×12	27×14	26×16	25×18	24×20	24×21	23×23

the plate is $E = 2.1\times10^6$ kg/cm^2 and the Poisson ratio is $\nu = 0.2$. The thickness of the plate is 1 cm and the plane stress condition is assumed. Substructuring optimization software (SOS) was developed to optimize the substructuring. The first two examples are 2D plates with structured mesh chosen to illustrate the superiority of the proposed technique over the classic substructuring with respect to computational and memory cost. The next example is an asymmetric plate with an unstructured mesh chosen to demonstrate the performance of the proposed technique for unstructured mesh. In all examples, optimization was carried out for computational and memory cost and combinational optimization was carried out as discussed in Section 5. The results of these optimizations will be compared with other subdivisions below.

6.1 Cross-shaped concrete plate

The first example is a cross-shaped concrete plate that is tensioned on four sides. Although one-fourth of the problem could be modeled based on symmetry, half of the problem is modeled to increase the complexity of the FE mesh. The plate is meshed by 40×20 structured 4-noded quadrilateral elements as shown in Fig. 1.

Tables 3 and 5 indicate that the optimal number of substructures for FLOPS and memory cost are 6×3 and 7×3, respectively, as shown in Fig. 2. The bold lines delineate the boundary of the substructures.

Different numbers of substructures were examined using a computer in the same manner and the run times were compared with the optimal states summarized in Table 9. The results indicate that the number of estimated FLOPS is concordant with the run time with slight tolerance. In this mesh, the model can be partitioned using 800 different patterns (1×1,1×2,…,40×20). If these cases are arranged with respect to FLOPS and memory cost (best to worst) and the FLOPS and memory cost for each case is calculated, the decrease in FLOPS and memory cost are as shown in Fig. 3. The drop at the start of the diagrams corresponds to the worst case (1×1) which gains no benefit from substructuring.

For a small number of elements, the optimal FLOPS cost is usually near similar to the optimal memory cost, as demonstrated in Fig. 3. The more dense mesh led to different and distinctive results; thus, the problem was meshed again with 400×200 elements. Combinational optimizations were carried out for different values of *r*. The case of $r = 0$ corresponds to FLOPS optimization and $r = 1$ corresponds to memory optimization. The optimal subdivisions for different

Table 7: Optimal number of substructures versus number of elements for combinational cost (*r*=0.5)(up to 100 elements in each direction)

n \ *m*	10	20	30	40	50	60	70	80	90	100
10	3×3	2×5	2×6	2×8	2×8	2×9	2×10	2×11	2×11	2×12
20	5×2	4×4	3×5	3×6	3×7	3×7	3×8	2×11	2×12	2×13
30	6×2	5×3	4×5	4×5	4×6	3×8	3×9	3×9	3×10	3×10
40	8×2	6×3	5×4	5×5	5×5	4×7	4×7	4×8	4×8	4×9
50	8×2	7×3	6×4	5×5	5×5	5×6	5×7	5×7	4×9	4×9
60	9×2	7×3	6×4	7×4	6×5	6×6	5×7	5×7	5×8	5×8
70	10×2	8×3	9×3	7×4	7×5	7×5	6×6	6×7	6×7	5×8
80	11×2	11×2	9×3	8×4	7×5	7×5	7×6	6×7	6×7	6×8
90	11×2	12×2	10×3	8×4	9×4	8×5	7×6	7×6	7×7	7×7
100	12×2	13×2	10×3	9×4	8×5	8×5	8×6	8×6	7×7	7×7

Table 8: Optimal number of substructures versus number of elements for combinational cost (r=0.5) (>100 elements in each direction)

n \ *m*	100	200	300	400	500	600	700	800	900	1000
100	7×7	6×11	5×16	5×18	4×24	4×27	4×29	4×31	4×33	4×35
200	11×6	9×10	8×13	8×15	7×19	7×21	7×22	6×27	6×28	6×30
300	16×5	13×8	11×11	10×14	10×16	9×19	9×20	9×22	8×25	8×27
400	18×5	15×8	14×10	12×13	12×15	11×17	11×19	10×22	10×23	10×24
500	24×4	19×7	16×10	15×12	14×14	13×16	13×18	12×20	12×21	11×24
600	27×4	21×7	19×9	17×11	16×13	15×15	14×17	14×19	14×20	13×22
700	29×4	22×7	20×9	19×11	18×13	17×14	16×16	16×18	15×20	15×21
80	31×4	27×6	22×9	20×11	20×12	19×14	18×16	17×17	17×18	16×20
900	33×4	28×6	25×8	23×10	21×12	20×14	20×15	18×17	18×18	18×19
1000	35×4	30×6	27×8	24×10	22×12	22×13	21×15	20×16	19×18	19×19

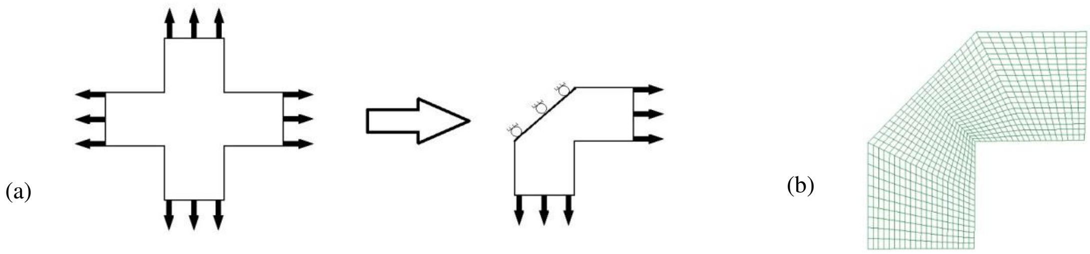

Fig. 1: Cross-shaped plate: (a) geometry; and (b) meshing.

Fig. 2: Optimal substructuring of cross-shaped plate with respect to a) FLOPS cost; b) memory cost

Table 9: Summary of FLOPS and memory cost for subdivisions of cross-shaped plate

State	Subdivisions	FLOP ($\times10^9$)	Memory ($\times10^6$)	Normalized FLOP	Normalized memory	Run time (s)
Arbitrary	3×3	133.0	1.96	2.04	1.37	0.04088
Optimum for FLOPS	6×3	65.4	1.44	1.00	1.01	0.02778
Optimum for memory	7×3	69.4	1.43	1.06	1.00	0.02674
Arbitrary	6×6	157.0	1.78	2.40	1.25	0.05312
Arbitrary (without substructuring)	40×20	3260	8.79	49.75	6.16	0.50988

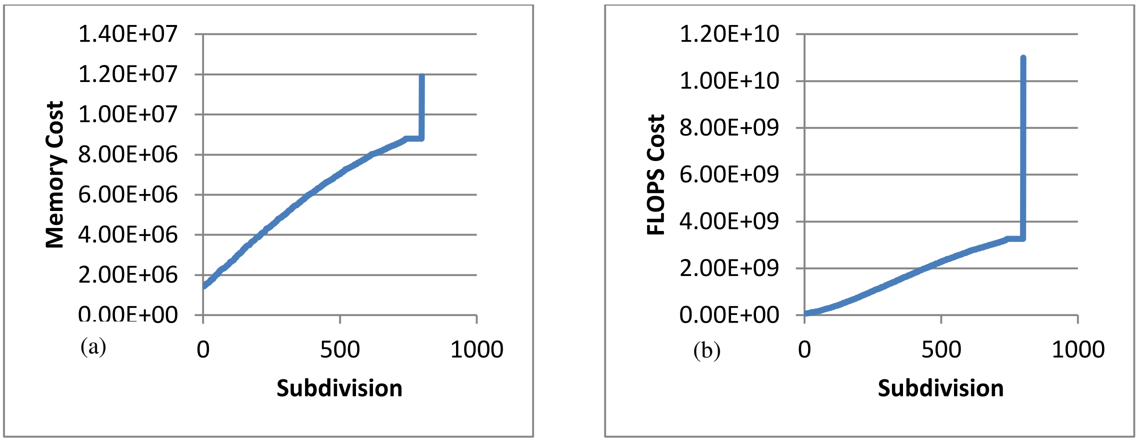

Fig. 3: Variation in memory and FLOPS cost for different subdivisions.

Table 10: Optimum substructuring in cross shaped plate for different values of *r*

r	Subdivisions	FLOP$\times10^9$	Memory$\times10^6$	Normalized FLOPS cost	Normalized memory cost	Normalized average cost
1	18×9	2550.00	1420.00	1.24	1	1
0.8	17×8	2200.00	1440.00	1.07	1.01	1.024
0.5	15×8	2080.00	1480.00	1.01	1.04	1.024
0.2	14×8	2060.00	1510.00	1	1.06	1.013
0	14×8	2060.00	1510.00	1	1.06	1

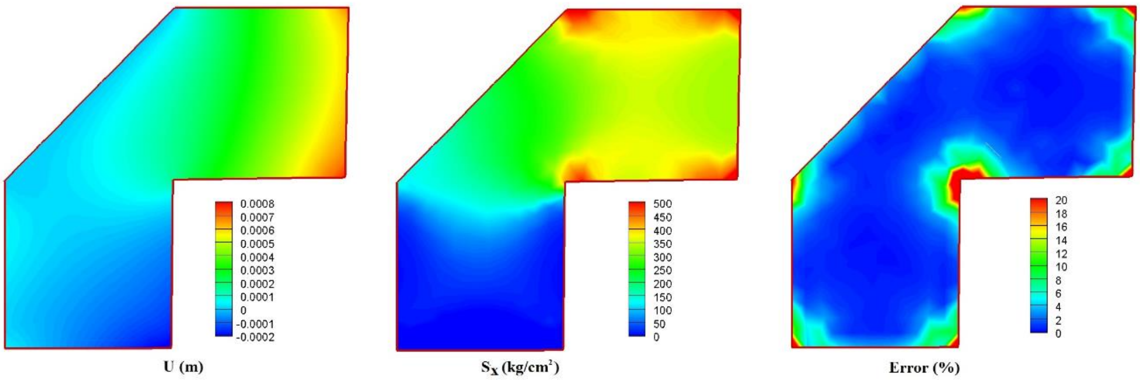

Fig. 4: Contours of horizontal displacement, S_x Stress and estimated error.

optimal subdivisions have resulted for different values of *r* (importance of memory or FLOPS). The contours of the displacement field, stress field and estimated error are shown in Fig. 4.

6.2 Rectangular concrete plate with hole

The next example is a rectangular concrete plate with a central circular hole that is tensioned from opposite sides. Although one-fourth of the problem could be modeled based on symmetry, half of the problem is modeled to make a balanced mesh in two directions, as shown in Fig. 5. The plate was meshed by 30×30 structured 4-noded quadrilateral elements.

Tables 3 and 5 show the optimal number of substructures for FLOPS and memory cost are 4×5 and 5×5, respectively, as shown in Fig. 6. The optimal substructures obtained in the

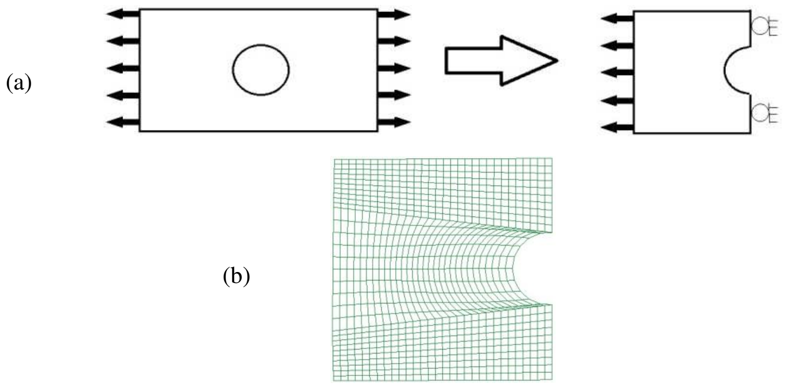

Fig. 5: Plate with a hole: (a) geometry; and (b) meshing.

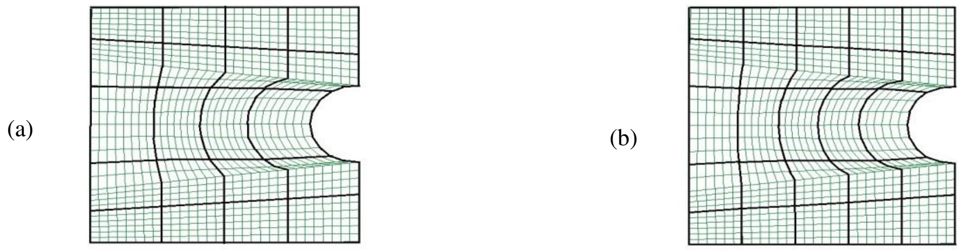

Fig. 6: Optimal substructuring of rectangular plate for: (a) FLOPS cost; (b) memory cost.

Table 11: Summary of flop and memory cost for different subdivisions in rectangular plate.

State	Subdivisions	Flop ($\times 10^9$)	Memory ($\times 10^6$)	Normalized flop	Normalized memory	Run time (s)
Arbitrary	3×2	402	3.11	4.43	1.80	0.08696
Optimum for flop	4×5	91	1.73	1.00	1.01	0.03404
Optimum for memory	5×5	103	1.72	1.13	1.00	0.03572
Arbitrary	6×6	160	1.88	1.76	1.10	0.04028
Arbitrary (without substructuring)	30×30	4760	11.3	52.5	6.59	0.64672

Table 12: Optimum substructuring in rectangular plate for different values of *r*

r	Subdivisions	FLOP$\times 10^9$	Memory$\times 10^6$	Normalized FLOPS cost	Normalized memory cost	Normalized average cost
1	13×13	3320.00	1700.00	1.23	1	1
0.8	12×12	2900.00	1720.00	1.08	1.01	1.024
0.5	11×11	2700.00	1790.00	1	1.05	1.027
0.2	11×11	2700.00	1790.00	1	1.05	1.011
0	11×11	2700.00	1790.00	1	1.05	1

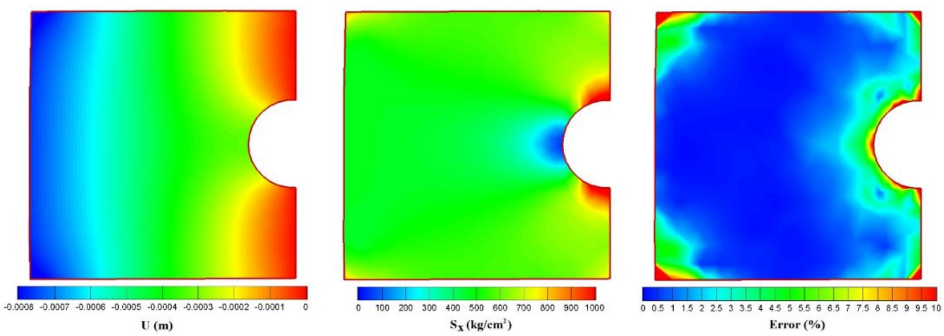

Fig. 7: Contours of horizontal displacement, S_x Stress and estimated error.

first two examples illustrate that square substructures are more efficient than long rectangular subdivisions. Table 11 compares substructuring to reduce FLOPS and memory cost for different subdivisions. It is evident that the greatest percentage of decrease occurred in the 4×5 subdivision for FLOPS cost and 5×5 subdivision for memory cost. For more dense mesh (300×300), the optimal subdivision was 11×11 for FLOPS cost and 13×13 for memory cost. Because these two subdivisions are close together, combinational optimization does not change the optimal subdivision considerably (Table 12). The contours of the displacement field, stress field and estimated error are shown in Fig. 7.

6.3 Asymmetric cracked concrete plate with a hole

The capability of the proposed algorithm for structured mesh has been demonstrated in the previous examples, but this algorithm could also be generalized to unstructured mesh. If unstructured mesh is partitioned into semi-structured regions and the substructuring of each region is optimized, an approximate solution for optimized substructuring of the whole problem could be obtained. An asymmetric cracked concrete plate with a hole was selected to show the robustness of the proposed algorithm for unstructured mesh. The plate was tensioned from two sides and meshed by unstructured quadrilateral elements as shown in Fig. 8.

The plate was partitioned into three regions with semi-structured meshing as shown in Fig. 9. Regions 1 and 2 are 10×20 mesh and region 3 is 20×20 mesh. If these regions are subdivided for FLOPS cost optimization according to Table 3, different subdivisions result for each region (Table 13). The track of the subdivision in each region is shown in Fig. 10.

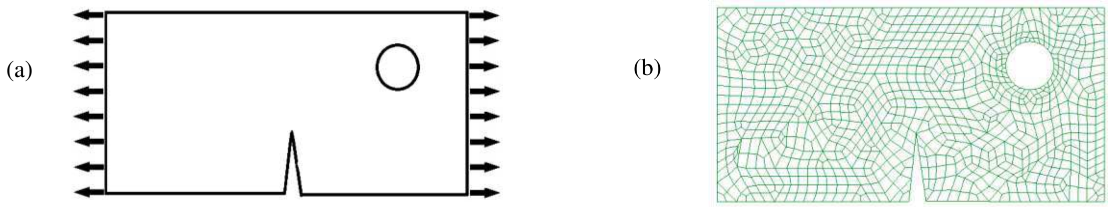

Fig. 8: Asymmetric cracked plate with a hole: (a) geometry; (b) meshing.

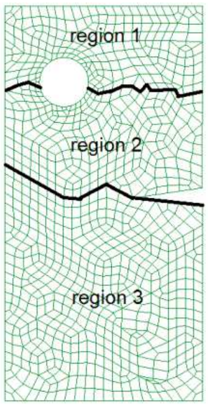

Fig. 9: Semi-structured regions of asymmetric cracked plate with a hole.

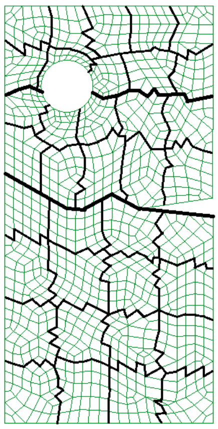

Fig. 10: Track of subdivisions in each region in asymmetric cracked plate.

Table 13: Optimal subdivisions of asymmetric cracked plate by region

Region	Optimal division
1	5×2
2	5×2
3	4×4

Table 14: Summary of FLOPS and memory cost for subdivisions of asymmetric cracked plate

State	Subdivisions	FLOP ($\times10^9$)	Memory ($\times10^6$)	Normalized FLOP	Normalized memory	Run time (s)
Arbitrary	3×3	133.0	1.96	1.98	1.35	0.04239
Optimum for FLOPS (Approximately)	8×4	67.2	1.47	1.00	1.02	0.02738
Optimum for memory (Approximately)	10×4	71.8	1.45	1.07	1.00	0.02962
Arbitrary	6×6	157.0	1.78	2.34	1.23	0.05833
Arbitrary (without substructuring)	40×20	3260	8.79	48.51	6.06	0.58224

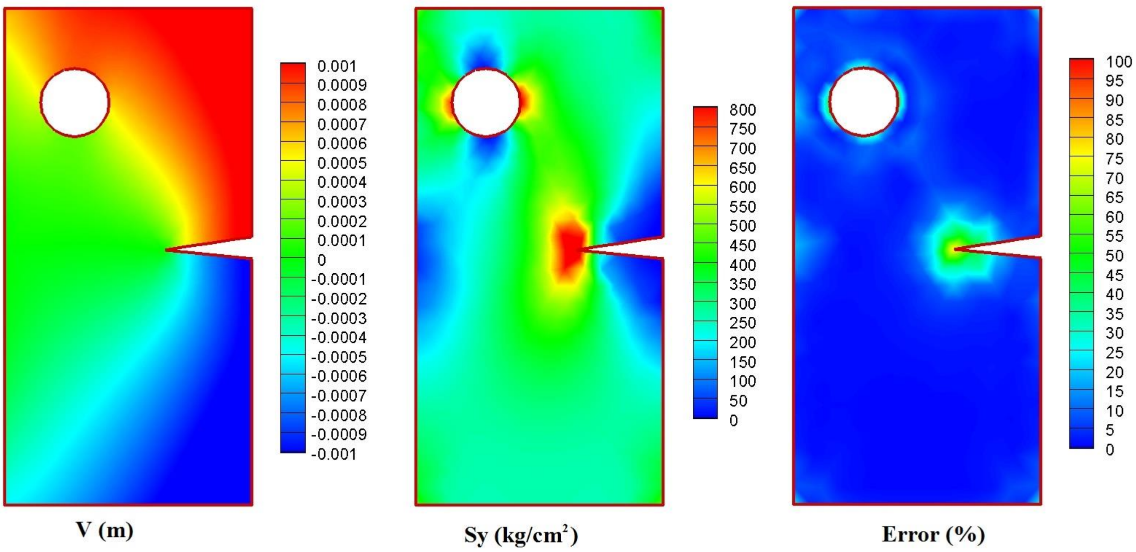

Fig. 11: Contours of vertical displacement, S_y Stress and estimated error.

7. Conclusions

The present study proposed an optimization algorithm for substructuring in the FEM. The cost functions of computation and memory usage are extracted in terms of number of subdivisions and optimized mathematically. The results are presented in the form of tables which recommend the proper substructuring for different number of elements. It was shown that use of the proper number of substructures improved the efficiency of technique for both analysis time and memory required. It also demonstrated that square substructures behave more efficiently than irregular substructures. More advanced optimization for memory and analysis time were investigated according to their respective importance in different problems and on different computers. The FLOPS and memory costs were first normalized with respect to the optimal case and combined with a weighting factor reflecting the value of each factor. The technique was generalized for unstructured mesh by partitioning the mesh to semi-structured regions. The proposed optimization algorithm was validated by analysis of several numerical examples. It was shown that this optimization improved substructuring and reduces analysis time and memory required.

References

[1] Badia, S., Verdugo, F., “Robust and scalable domain decomposition solvers for unfitted finite element methods”, Journal of Computational and Applied Mathematics, vol 344, 2018, p. 740-759.

[2] Boo, S.H., Oh M.H., “Automated static condensation method for local analysis of large finite element models”, Structural Engineering and Mechanics, vol 61, 2017, p. 481-495.

[3] Farhat, C., "A simple and efficient automatic FEM domain decomposer", Comp. and Struc, vol 28, 1988, p. 579-602.
[4] Farhat, C., Maman, N., Brown, G.W., "Mesh partitioning for implicit computations via iterative domain decomposition: impact and optimization of the subdomain aspect ratio", Int. J. Num. Meth. Eng., vol 38, 1995, p. 989-1000.
[5] Farhat, C., Roux, F.X., "A method of finite element tearing and interconnecting and its parallel solution algorithm", Int. J. Num. Meth. Eng., vol 32, 1991, p. 1205-1227.
[6] Fonseka, M.C.M., "A sub-structure condensation technique in finite element analysis for the optimal use of computer memory", Comp. and Struc., vol 49, 1993, p. 537-543.
[7] Gurujee, C.S., Deshpande, V.L., "An improved method of substructure analysis", Comp. and Struc., vol 8, 1978, p. 147-152.
[8] Kaveh, A., "Decomposition for Parallel Computing: Graph Theory Methods", Computational Structural Analysis and Finite Element Methods, Springer Int. Pub, 2014.
[9] Kaveh, A., Roosta, G.R., "Graph-theoretical methods for substructuring, subdomaining and ordering", Int. J. Space Struc., vol 10, 1995, p. 121-132.
[10] Li, J., Hao, H., "Numerical study of structural progressive collapse using substructure technique", Eng. Struc., vol 52, 2013, p. 101-113.
[11] Mehrdoost, Z., Bahrainian, S.S., "A multilevel tabu search algorithm for balanced partitioning of unstructured grids", International Journal for Numerical Methods in Engineering, vol 105, 2016, p. 678-692.
[12] Minka, T. The Lightspeed MATLAB Toolbox, [online] Available: https://github.com/tminka/lightspeed.
[13] Njomo, W., Ozay, G., "Sequential analysis coupled with optimized substructure technique modeled on 3D-frame construction process", Eng. Struc., vol 80, 2014, p. 200-210.
[14] Noor, A.K., Kamel, H.A., Fulton, R.E., "Substructuring techniques: status and projections", Comp. and Struc., vol 8, 1978, p. 621-632.
[15] Novikov, A., Piminova, N., Kopysov, S., Sagdeeva, Y., "Layer-by-layer partitioning of finite element meshes for multicore architectures", Communications in Computer and Information Science, vol 687, 2016, p. 106-117.
[16] Predari, M., Esnard, A., Roman, J., "Comparison of initial partitioning methods for multilevel direct k-way graph partitioning with fixed vertices", Parallel Computing, vol 66, 2017, p. 22-39.
[17] Przemieniecki, J.S., "Matrix Structural Analysis of Substructures", AIAA Journal, vol 1, 1963, p. 138-147.
[18] Shen, Y., Yin, X.C., "Dynamic substructure analysis of stress waves generated by impacts on non-uniform rod structures", Mechanism and Machine Theory, vol 74, 2014, p. 154-172.
[19] Vanderstraeten, D., Keunings, R., "Optimized partitioning of unstructured finite element meshes", Int. J. Num. Meth. Eng., vol 38, 1995, p. 433-450.
[20] Wang, J., Li Q., Zhu, Z., "Mixed substructure synthesis method with physics-impedance-modal parameter", Structural Engineering and Mechanics, vol 8, 1999, p. 505-512.
[21] Yang, Y.S., Hsieh, S.H., Hsieh, T.J., "Improving parallel substructuring efficiency by using a multilevel approach", J. Comp. Civil Eng., vol 26, 2011, p. 457-464.