

ACE 2006

7th International Congress on

Advances in Civil Engineering



October 11-13, 2006

BOOK OF ABSTRACTS

EDITORS
ORGANIZING COMMITTEE

Yıldız Technical University
İSTANBUL - TURKEY

Advisory Committee

Metin AYDOĞAN (ITU), Huriye BİLSEL (EMU),
Erol GÜLER (BU), Tuğrul TANKUT (METU),
Yalçın YÜKSEL (YTU) (Congress Chair)

Local Organizing Committee

Şenay ATABAY, Cem AYDEMİR, Tuba BOSTAN, Ali COŞAR, Havvanur KILIÇ,
Ali KOÇAK (Congress Secretary), Orhun KÖKSAL, İsmail ŞAHİN, Nabi YÜZER

Scientific Committee

Abdulrezak N. MOHAMED (EMU), Abdurrahman GÜNER (Tekirdağ Ü.), Ahmet DURMUŞ (KTÜ), Akın ÖNALP (IKU), Alemdar BAYRAKTAR (KTÜ), Alessandro Di GRAZİANO (Catania Ü), Ali KOÇAK (YTÜ), Ali GÜNYAKTI (EMU), Alper İLKİ (İTÜ), Atakan AKSOY (KTÜ), Atilla AKKOYUNLU (BÜ), Aydın EREL (YTÜ), Ayfer ERKEN (İTÜ), Ayhan İNAL (ODTU), Ayşe EDİNÇLİLER (BÜ), B. Sadık BAKIR (ODTÜ), Bihrat ÖNÖZ (ITÜ), Bülent BARADAN (Dokuz Eylül Ü.), Bülent AKBAŞ (Gebze YTE), Cemal EYYUBOV (Erciyes Ü.), Cengiz KARAKOÇ (BÜ), Cengiz DÜNDAR (Çukurova Ü.), Cumhur AYDIN (Atılım Ü.), Darçın AKIN (Gebze YTE), E. Beyhan YEĞEN (İTÜ), Emin GÜÇBİLMEZ (EMU), Emine AĞAR (İTÜ), Ercan YÜKSEL (İTÜ), Erdinç SOYER (EMU), Erdoğan UZGİDER (İTÜ), Ergun GEDİZLİOĞLU (İTÜ), Esin ÖZKAN Çevik (YTÜ), Faruk YÜKSELER (YTÜ), Fevziye AKÖZ (YTÜ), Feyza ÇİNİCİOĞLU (İÜ), Fikret TÜRKER (Akdeniz Ü.), Filiz PİROĞLU (İTÜ), Gökhan BAYKAL (BÜ), Gökmen ERGÜN (BÜ), Gülay ALTAY (BÜ), Güliz BAYRAMOĞLU (İTÜ), Gülten GÜLAY (İTÜ), Güney ÖZCEBE (ODTÜ), Halil ÖNDER (ODTÜ), Halim CEYLAN (Pamukkale Ü.), Halit ÖZEN (YTÜ), Haluk GERÇEK (İTÜ), Hasan YILDIRIM (İTÜ), Hasan KAPLAN (Pamukkale Ü.), Hasan BİRİCİK (İKÜ), Hasan BODUROĞLU (İTÜ), Havvanur KILIÇ (YTÜ), Hayrullah AĞAÇÇIOĞLU (YTÜ), Hesham RAKHA (Virgina Tech, ABD), Hikmet Hüseyin ÇATAL (Dokuz Eylül Ü.), Hikmet BAYIRTEPE (Gazi Ü.), Hilmi LUŞ (BÜ), Hulusi ÖZKUL (BÜ), Huriye BİLSEL (EMU), Hüsnü CAN (Gazi Ü.), Işıkhan GÜLER (Yüksel Proje A.Ş.), İ. Kutay ÖZAYDIN (YTÜ), İ. Özgür YAMAN (ODTÜ), İbrahim EKİZ (YTÜ), İlhan AVCI (İTÜ), İlker ÖZDEMİR (Eskişehir Osmangazi Ünv), İlknur BOZBEY (İÜ), İrfan COŞKUN (YTÜ), İsmail AYDIN (ODTÜ), İsmail ŞAHİN (YTÜ), İsmail DURANYILDIZ (İTÜ), Joern PACHL (IfVE), K.P. GEORGE (The University of Mississippi, ABD), Kadir GÜLER (İTÜ), Kambiz RAMYAR (Ege U.), Kejin WANG (Iowa State University, ABD), Kemal Selçuk ÖĞÜT (İTÜ), Lei YU (Texas Southern University, ABD), M. Emin BİRPINAR (YTÜ), M. Emin ÖCAL (İTÜ), M. Ruhi AYDIN (Eskişehir Osmangazi Ü.), M. Tuğrul ÖZKAN (İTÜ), M. Uğur MÜNGEN (İTÜ), Mehmet Emin KARAHAN (İTÜ), Mehmet Hakkı OMURTAG (İTÜ), Mehmet Ufuk ERGUN (ODTÜ), Mehmet UYAN (İTÜ), Mehmet BERİLGEN (YTÜ), Melih YANMAZ (ODTÜ), Melike ALTAN (İTÜ), Metin AYDOĞAN (İTÜ), Metin GER (ODTÜ), Metin HÜSEM (KTÜ), Murat KURUOĞLU (İTÜ), Mustafa YILDIRIM (YTÜ), Mustafa ZORBOZAN (YTÜ), Mustafa GÖĞÜŞ (ODTÜ), Mustafa ÖNAL (Gazi Ü.), Mustafa GÜRSOY (YTÜ), Nabi YÜZER (YTÜ), Nadir YAYLA (İTÜ), Nagui ROUPHAİL (North Carolina State University, ABD), Necati AĞIRALİOĞLU (İTÜ), Nesrin YARDIMCI (İTÜ), Nilgün HARMANCIOĞLU (Dokuz Eylül Ü.), Orhan YÜKSEL (Ege Ü.), Orhun KÖKSAL (YTÜ), Perviz AHMEDZADE (Fırat Ü.), Recep İYİSAN (İTÜ), Saadet BERİLGEN (YTÜ), Sedat KAPDAŞLI (İTÜ), Serap KAHRAMAN (Dokuz Eylül Ü.), Serhan TANYEL (Dokuz Eylül Ü.), Sinan HINISLIOĞLU (Atatürk Ü.), Sinan ALTIN (Gazi Ü.), Sönmez YILDİRIM (YTÜ), Sunanda DİSSANAYAKE (Kansas State University, ABD), Süreyya TAYFUR (İSFALT A.Ş.), Syed Tanvir WASTİ (ODTÜ), Talat BİRGÖNÜL (ODTÜ), Tuba BOSTAN (YTÜ), Tuğrul TANKUT (ODTÜ), Turan ÖZTURAN (BÜ), Turgut ÖZTÜRK (İTÜ), Tülay AKSU (İTÜ), Yalçın YÜKSEL (YTÜ), Yaşar UĞUR (Erciyes Ü.), Yeşim ÇELİKOĞLU (YTÜ), Yılmaz AKKAYA (İTÜ), Zalihe NALBANTOĞLU (EMU), Zekai CELEP (İTÜ), Zekai ANGIN (KTÜ), Zekeriya POLAT (YTÜ), Zeki HASGÜR (İTÜ), Zübeyde ÖZTÜRK (İTÜ)

7 ace06-088 INFINITE RESERVOIR ANALYSIS USING AN EXACT TRUNCATION BOUNDARY CONDITION.....479
Safa Bozkurt COŞKUN, Mehmet Tarık ATAY

8 ace06-102 FINITE ELEMENT ANALYSIS OF REBAR CONFIGURATIONS FOR CONCRETE STRUCTURES.....480
Fethi KADIOĞLU, Ramana M. PIDAPARTI

9 ace06-111 YERALTISUYU MODELLEMESİNDE İTERATİF VE DOĞRUDAN ÇÖZÜM ALGORİTMALARININ KARŞILAŞTIRILMASI.....481
Halil KARAHAN , Gürhan GÜRARSLAN, M. Tamer AYVAZ

10 ace06-150 A GENERAL PARALLEL SOLUTION FRAMEWORK FOR THE LINEAR ANALYSIS OF LARGE STRUCTURES ON PC CLUSTERS.....482
Özgür KURÇ, Kenneth M. WILL

11 ace06-208 INVESTIGATION OF SEISMICALLY INDUCED STRESSES IN PILES BY 3D FEM.....483
Ertuğrul ORDU, Bahri ÖZDEMIR, M. Tuğrul ÖZKAN

12 ace06-220 DEVELOPMENT AND VERIFICATION OF HP-FEM FOR ELASTOSTATIC PLANE PROBLEMS.....484
Serkan BEKİROĞLU, Detlef KUHL

13 ace06-239 GÜÇLENDİRİCİ KİRİŞ KONUMUNUN BOŞLUKLU PERDE DAVRANIŞINA ETKİSİ..485
Murat BİKÇE

14 ace06-244 ADAPTIVE APPROACHES IN GENETIC ALGORITHMS TO CATCH THE GLOBAL OPTIMUM.....486
Vedat TOĞAN, Ayşe DALOĞLU

15 ace06-332 PARALLELIZATION OF A SERIAL FRAME ANALYSIS PROGRAM BASED ON MPI.487
Serkan BEKİROĞLU, Karl Robert LEIMBACH

16 ace06-358 MODELING OF THE BEHAVIOR OF CRACKED CONCRETE IN THE NONLINEAR FINITE ELEMENT APPROACH.....488
Yıldır AKKAYA, Zekai CELEP

17 ace06-363 OPTIMIZATION OF TRUSSES WITH STRESS AND STABILITY CONSTRAINTS BASED ON GENETIC ALGORITHM.....489
Nildem TAYŞİ, M. Tolga GÖĞÜŞ, Mustafa ÖZAKÇA

18 ace06-566 A DYNAMIC BAYESIAN NETWORK FOR HURRICANE PLANNING AND PREPAREDNESS.....490
Manoj K. JHA, Onan MARROQUIN

CASE STUDIES in CIVIL ENGINEERING

1 ace06-075 BETONARME YÜKSEK KİRİŞLERDE EN UYGUN DONATI ÇAPLARININ GENETİK ALGORİTMALAR KULLANILARAK BELİRLENMESİ.....491
Mustafa KAYA, Abdussamet ARSLAN

2 ace06-155 DÜZLEM ELASTİSİTE PROBLEMİ OLARAK ELE ALINAN KONSOL BİR KİRİŞİN YAPISAL ŞEKİL OPTİMİZASYONU.....492
Habib UYSAL, Nurcan AŞCI, Ümit UZMAN

3 ace06-202 AKARSU YATAĞI BOZULAN FETREK DEMİRYOLU KÖPRÜSÜNÜN MECRA DÜZENLEMESİ VE KORUMA YAPISI UYGULAMASI.....493
Ebru ERİŞ, Emre ERCAN

7th International Congress on Advances in Civil Engineering, 11-13 October 2006
Yildiz Technical University, Istanbul, Turkey

Parallelization of A Serial Frame Analysis Program Based On MPI

Serkan Bekiroğlu
Karadeniz Technical University, Department of Civil Engineering, Trabzon, Turkiye

Karl Robert Leimbach
Institute for Computational Engineering, Ruhr University Bochum, Bochum, Germany

Abstract

This study presents how to parallelize a current serial program solving 2D and 3D frame structures with examples. A parallelization of the serial program is handled by considering a parallel computer system with distributed memory that uses a parallel programming model, MPI. Before parallelization, the serial program is modified according to a methodology called "Substructure Method". It is pointed out that a parallel process is able to save computation time, regarding of serial process highly time-consuming, and that the arrangement taking substructure method into consideration gets efficiency for sequential computation.

Keywords: Parallelization, 2D and 3D frame structure, Distributed Memory, MPI, Substructure Method.

Introduction

The idea behind parallelization is to execute a set of tasks simultaneously. That means to split higher computational work into lower ones which are instructions and inputs manipulated in tasks or processes. The simultaneous manipulation is achieved as long as Bernstein Conditions are satisfied. Bernstein Conditions briefly mean that the input sets of two tasks are independent of each other's output sets, and their output sets are independent. Only then can tasks be executed in parallel, otherwise, it is not possible. In separation of serial work into tasks, a common tool is Substructure Method. The method is utilized in discretization of whole structure into substructure so that the current program could be adapted in a way that whole structure can be comprised by substructures. In the simple sense, the method enables partial solution of equations system in matrix form so that the value of some unknown can be obtained without solving the whole equation systems, and then the value of the rest of unknown is able to be achieved by depending on the former those. That is to say, the values of common variables of substructures are firstly found and then those of private variables of substructures are secondly found, too. So, the computation process can be simultaneously performed except for the instant when the contributions of substructures put together to get solution for common variables. By means of the method, it can be

managed to divide a structure into individual substructure to which a task corresponds in parallel process. After that, these tasks are executed in parallel. In the mean while, at the beginning of parallel computation, inputs of the program are informed to processes by either allowing to access to that or dispatching that in accordance with parallel programming model. In other words, the inputs can be located in shared memory so that each process can reach them, or they can be broadcasted to distributed memory.

In this study, the analyzer program solving 2D and 3D frame structure by means of Matrix Displacement Method is implemented in parallel process. The implementation of parallelizing the program takes place in the computer system that has memory architecture such as Distributed Memory (DM). The computer system consists of personal computers having the platform LINUX and connecting each other through LAN network. The parallel programming model employed in the system is the model MPI standing for “Message Passing Interface”.

Parallel Architecture

Classification

A parallel computer is a collection of processors and memories prepared in accordance with a parallel architecture. Parallel architecture is assessed by the structure of instruction (I) and data (D); and memory organization. According to the first criterion, it is separated into groups when independent dimensions of instruction and data are considered, which is give in Table 1. Each of these dimensions can have only one of two possible states: Single (S) or Multiple (M) (Leimbach, 2005). SISD machines correspond to serial computer. In SIMD machines, all processors execute the same instruction stream on a different piece of data. As to MIMD machines, each processor can execute a separate stream of instructions on its own local data, too. When memory organization is conceived, parallel computers are crudely split into two portions such as shared memory computer, and distributed memory computer. Shared memory means that memory is common memory to which all processors share access. Distributed memory means that memory is distributed among the processors, rather than placed in a central location.

Table 1. Flynn`s Taxonomy.

D: data I:instruction	SD	MD
SI	SISD	SIMD
MI		MIMD

Common Parallel Architectures

Having different parallel architecture, parallel computers are widely used in practice. The common parallel architecture is *multicomputer* model where each compute resource consists of a von Neumann machine or node: a CPU and memory and has ability to communicate with other compute resource by sending and receiving messages over an interconnection network (URL 1, 2006). The model is indicated in Figure 1(a). A defining attribute of the multicomputer model is that accesses to local (same-compute resource) memory are less expensive than accesses to remote (different-compute resource) memory. That is, read and write are less costly than send and receive. Hence, it is desirable that accesses to local data be more frequent than accesses to remote data, which is called locality (URL 1, 2006). The multicomputer is most similar to what is often called the *distributed-memory MIMD* computer that is illustrated in Figure 1(b).

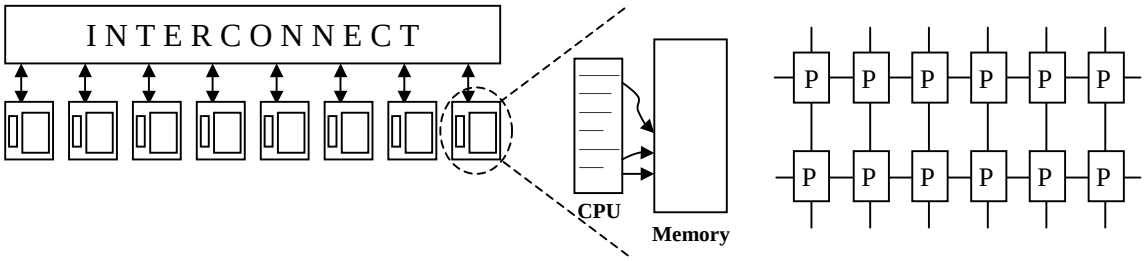


Figure 1. a) Muticomputer model, b) A distributed-memory MIMD computer with a mesh interconnect.

The principal difference between a multicomputer and the distributed-memory MIMD computer is that the cost of sending a message between two nodes may not be independent of node location and other network traffic. Another important class of parallel computer is the *multiprocessor*, or *shared-memory MIMD* computer that is illustrated in Figure 2(a). In multiprocessors, all processors share access to a common memory, typically via a bus or a hierarchy of buses. In practice, the principle of the fact that any processor can access any memory element in the same amount of time usually introduces some form of memory hierarchy. In particular, the frequency with which the shared memory is accessed may be reduced by storing copies of frequently used data items in a *cache* associated with each processor. Access to this cache is much faster than access to the shared memory; hence, locality is usually important, and the differences between multicomputers and multiprocessors are really just questions of degree. Programs developed for multicomputers can also execute efficiently on multiprocessors, because shared memory permits an efficient implementation of message passing. Further, a more specialized class of parallel computer is the *SIMD* (single instruction multiple data) computer. This approach can reduce both hardware and software complexity. Multicomputer algorithms can not in general be executed efficiently on SIMD computers. SIMD and MIMD computer systems are sometimes used as parallel computers trough the local area network (LAN), in which computers in close physical proximity (e.g., the same building) are connected by a fast network, and the wide area network (WAN), in which geographically distributed computers are connected. The logical view of the system is shown in Figure 2(b). They can be viewed for many purposes as multicomputers, albeit with high remote-access costs.

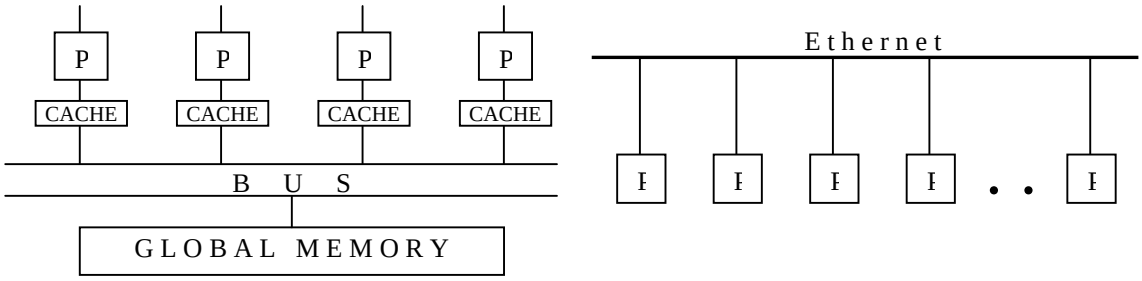


Figure 2. a) A shared-memory multiproccessor, b) A local area network.

Parallel Programming Model

Parallel architecture requires use of parallel programming method with which it is able to express how the computation and data are distributed across the multiple processors and memories (URL 2, 2006). The major parallel programming models are tasks-channels model, data parallelism model, the message passing model, and the shared

memory model (URL 3, 2006). Currently utilized the standard message passing model can be stated as MPI.

Message Passing Model

Message passing model is based on a set of tasks encapsulating local data. This model can be said to implement a SIMD programming model because each task executes the same program but operates on different data, which is designated as domain decomposition. The domain decomposition is implemented by developing an algorithm describing the local computations and local data structures of a single process. Thus, local distribution of input is performed and only locally distributed-input is allocated in a process (Michael, 2000). The locality on input level has to be applied to loop bounds since only the computation depending on local input must be ensured. Hence, the model makes the programmer responsible of expressing the partitioning of computation and data (URL 2, 2006) as well as conceptualizing the communication structure of a parallel program (URL 1, 2006). Message passing model has been implemented on many parallel machines with shared physical memory as well as distributed memory, often with very good communication performance (URL 4, 2006). An advantage of message-passing model is the generality of the model so that it can be used to program almost any algorithm you have, and it applies to almost all kinds of computer systems (URL 5, 2006).

MPI

MPI is the standard programming interface offering a complete set of routines for point-to-point communication, collective communication, one-sided communication so that data can be transmitted from one processor to other processor, (Michael, 2000). These routines are covered in the library called MPICH. That is, the library is needed for implementation of MPI.

Substructure Method

Substructure Method reduces higher computational effort to blocks of the light effort, which refer to partial equation systems in matrix calculation. On the solution of structure system, the method gives an opportunity to divide the structure system into sub-structure systems. Trough the method, firstly, equation system (1) is broken down, where A is a coefficient matrix, X and B are a vector of unknowns and constants, respectively.

$$AX = B, \begin{bmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{bmatrix} \begin{bmatrix} X_I \\ X_B \end{bmatrix} = \begin{bmatrix} B_I \\ B_B \end{bmatrix} \tag{1}$$

Further, indexes I and B stand for internal case and interface boundary case, respectively. Secondly, the system is reduced as

$$\begin{bmatrix} A_{II} & A_{IB} \\ 0 & (A_{BB} - A_{BI}A^{-1}{}_{II}A_{IB}) \end{bmatrix} \begin{bmatrix} X_I \\ X_B \end{bmatrix} = \begin{bmatrix} B_I \\ (B_B - A_{BI}A^{-1}{}_{II}B_I) \end{bmatrix}, \begin{bmatrix} A_{II} & A_{IB} \\ 0 & A^*{}_{BB}) \end{bmatrix} \begin{bmatrix} X_I \\ X_B \end{bmatrix} = \begin{bmatrix} B_I \\ B^*{}_B \end{bmatrix}. \tag{2}$$

For each sub-structure system, the terms corresponding interface boundary in the reduced form $A^*{}_{BB}$ and $B^*{}_B$ are inserted in convenient place in new arrangement which is the equation system just consisting of unknown in the interface boundary. Figure 3 indicates this explanation in the symbolic way.

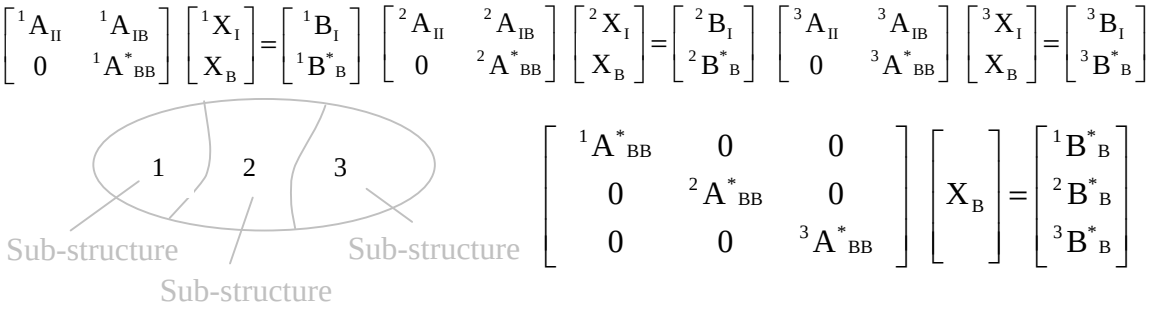


Figure 3. Logical view of partitioning of a structure on the equation system level.

The equation system shown the above figure is solved for the vector X_B of interface boundary unknowns. Then the vector X_I of the internal unknowns is individually calculated in the sub-structure systems in accordance with

$$X_I = A^{-1}{}_{II} B_I - A^{-1}{}_{II} A_{IB} X_B .$$

As a consequence, one equation system is able to be solved in the two steps. In the first step interface boundary unknowns and in the second step internal unknowns are obtained, which makes decomposition of structure system probable in the matrix calculation.

Current and Modified Programs

The analysis program is able to solve 2D and 3D frame structures. The outline corresponding to the current program is given in Figure 4.

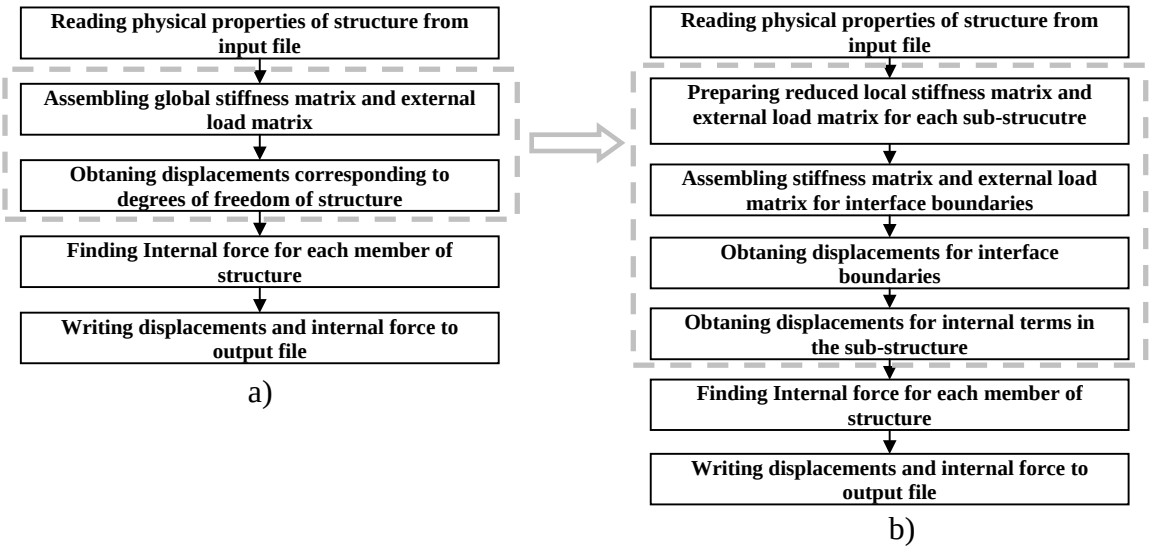


Figure 4. The flowchart of the a) current and b) modified programs.

In the concept of the program, coefficient matrix, vector of unknowns and vector of constants correspond to stiffness matrix, displacements vector and external force vector, respectively. In the way of decomposition of structure, the first step is that assembling of global stiffness matrix and global external force turns to assembling of local stiffness matrix and local external force for each sub-structure. Then each assembling of local terms is reduced to blocks of interface boundary and internal parts in terms of substructure method. Figure 5 illustrates the new algorithm in logical view.

Parallel Implementation

The structure system is broken down into sub-structures as many as the number of processors available for parallel processing. The solution process for each sub-structure is then assigned to a processor. MPI is employed for the information exchange at the interface boundaries of sub-structures. The programming model is conducted in the platform LINUX.

Preparing of Program Based on MPI

A parallel algorithm in the MPI model is based on the master–worker paradigm. For instance, Figure 5 indicates the parallel algorithm of the current program.

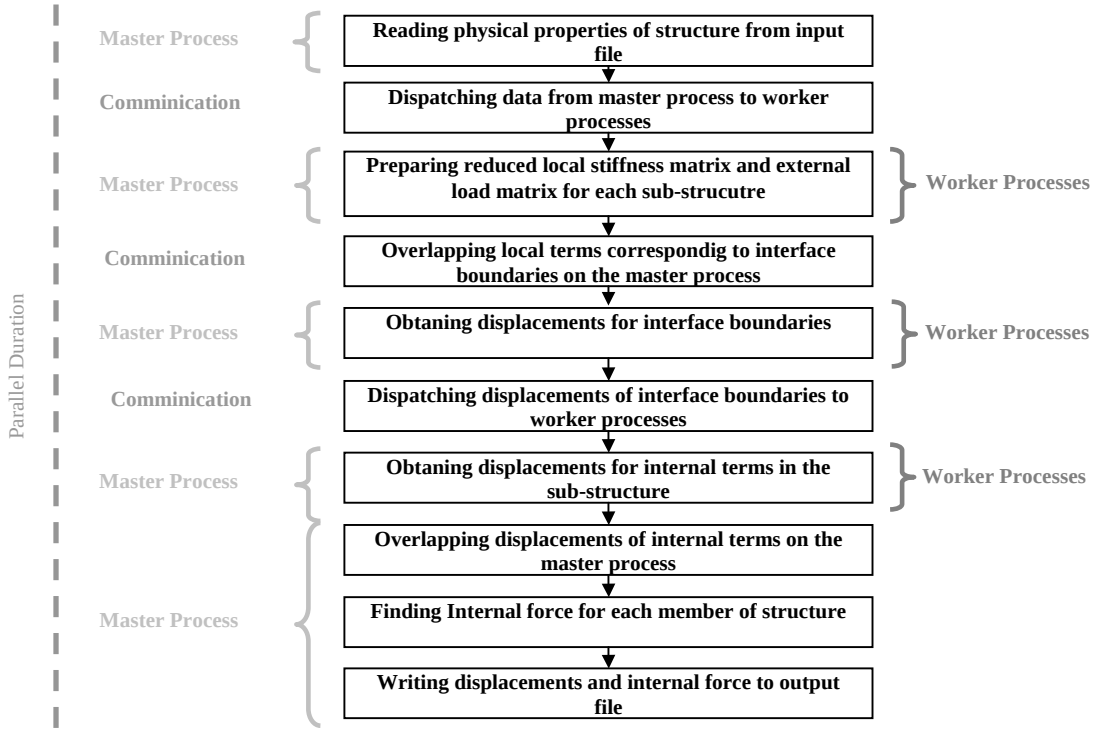


Figure 5. The flowchart of parallel program based on MPI.

The above algorithm is developed over the modified sequence algorithm. Since computational work is made discrete very well in the sequence algorithm, loop bounds are only adapted according to data decomposition. The data decomposition is carried out by distributing the local parts of global arrays to processes. All operations related with domain decomposition and dispatching local data as well as setting physical definition of a structure is manipulated by master process. Then each worker process implements the same instruction as the serial computation and sends results related with the interface boundaries to the master process. After overlapping all terms corresponding to stiffness matrix and external force of interface boundaries on the master process, the related displacements are calculated and broadcasted to worker processes. Worker processes obtain internal displacements by depending on displacements of interface boundaries and also send these results to master. Finally, master finds internal forces of the whole structure and prepares output files for displacements and internal forces of the whole structure. All communications in the parallel duration are performed by collective communication tools such as broadcast and reduce.

Performance Evaluation

Speed-up (S) and efficiency (E) of the parallel processing are evaluated by comparing the execution times of the parallel program corresponding to a single (T1) and p processor (Tp) runs, respectively. The speedup and efficiency are defined as,

$$\text{a) } S = \frac{T_1}{T_p}, \qquad \text{b) } E = \frac{S}{p} \tag{3}$$

It is desired that Speed-up and efficiency reminds close to unity, in the other case, deviation seems and in many times this is unavoidable. As deviation becomes high speed-up and efficiency become smaller.

Examples of Parallel Implementation

The current 2D and 3D frame structure analysis program is implemented over three frame structure examples which are 2D frame structure with 150 members, 3D frame structure with 805 members, and 2435 members. The goal in all examples is to observe effectiveness of substructure method and performance of MPI model. In the consequence of examples, it is noticed that sharing the size of computational work to tasks via sub-structures plays primitive role over the ratio of communication time to computation time. Through view of examples, since comparison of all displacements and internal forces for sequence, modified sequence, and parallel programs can preclude following the study, they are not given here. But, it is ensured that displacement and internal force results of these three programs are sufficiently accurate. Further, the results of three programs perfectly agree with each other.

Frame Structure with 150 members

The three-bay, 15-storey frame structure with 150 members and 79 nodes is demonstrated in Figure 6(a). The layout of the frame structure on 5 sub-structures is also illustrated in Figure 6(b).

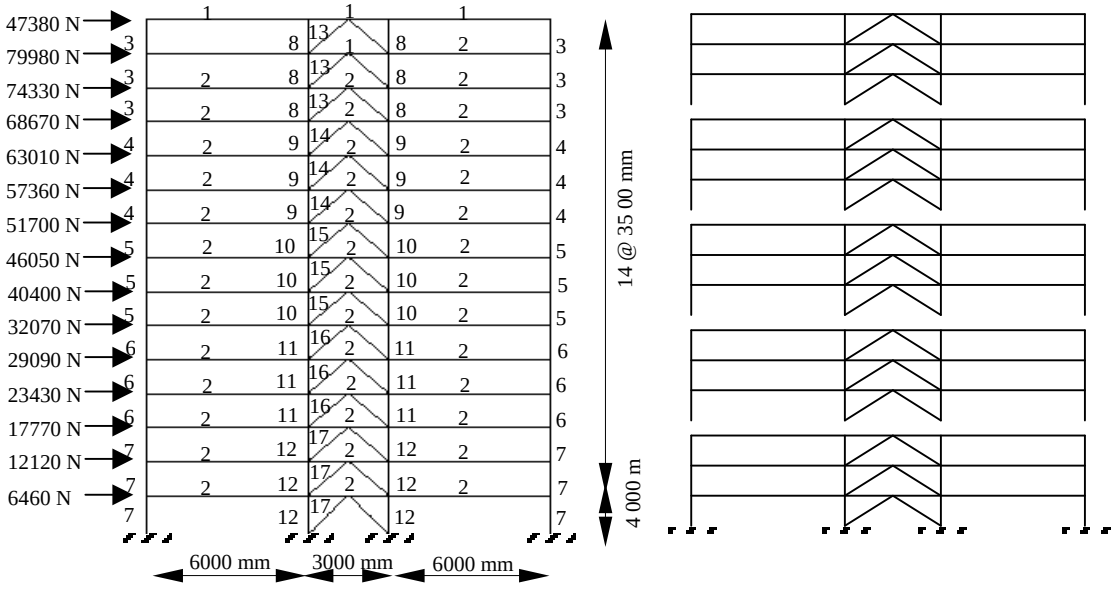


Figure 6. a) Topology of frame structure with 150 members and b) The decomposition model of the frame structure into 5 sub-structures

The dimensions of the frame and loading are shown in Figure 6(a). The frame is subjected to a gravity loading of 24 N/mm on the beams. The lateral loading is earthquake load computed according to Turkish code. The modulus of elasticity is 200000 N/mm^2. There are 17 different groups seen in the above figure whose values of cross-sections and inertia moments are given in Table 2. This sections belong to I profiles taken from American standard AISC.

Table 2. Cross-section and inertia moments of all groups used in the example.

Gp. No	A (x10^{-2} mm^2)	I (x10^{-4} mm^4)	Gp. No	A (x10^{-2} mm^2)	I (x10^{-4} mm^2)	Gp. No	A (x10^{-2} mm^2)	I (x10^{-4} mm^2)	Gp. No	A (x10^{-2} mm^2)	I (x10^{-4} mm^2)
1	90.97	20187.22	6	206.45	51612.70	11	187.74	166076.34	16	220.64	205202.09
2	156.77	76170.35	7	441.93	125285.66	12	301.29	79083.97	17	195.48	124869.43
3	85.81	24391.16	8	75.48	6076.98	13	192.26	100728.00			
4	155.48	36711.61	9	140.64	33132.02	14	129.03	61602.25			
5	201.29	38834.39	10	201.29	38834.39	15	76.13	25473.36			

For the evaluation of performance of the parallel programs, the execution times of serial and parallel programs using five processors, speed-up and efficiency values are compared and summarized in Table 3. As can be seen from Table 3, it is noted that the execution time of the parallel program is much higher than that of serial one, which is reflected by speed-up values much less than unity. The reason is because of the simplicity of the example and the low ratio of computation time to communication time when the collective communication tools i.e. broadcast and reduce are employed for data exchange. That is, this outcome is expected thanks to longer communication time than computation time, which comes at lower speed-up and efficiency. On the other hand, for large frame structures, parallel solution is expected to provide higher speed-up and efficiency values.

Table 3. Performance of sequence programs and parallel program.

The num Of sub-structure	Sequence	Modified Sequence	MPI	Speedup	Efficiency
	Execution time (sec)	Execution time (sec)	Execution time (sec)		
0	0.0229				
5		0.0240	0.2282	0.1052	0.0210

In the meantime, the time score in the above table is quite close for sequence and modified sequence algorithms so that it can not be seen certain difference between them, which can be conceptualized due to lower execution time.

Frame Structure with 805 members

The large-scale 805 member space frame structure with 368 nodes is shown in Figure 7, which is adopted from McGee and Phan (1992). The dimensions of the frame and loading are shown in Figure 7. The frame is subjected to a lateral loading which acts on the node-pair seen the above figure from top to bottom and its value decreases from 13344 N towards 2668 N per 4 stores except for the last 5 store where it is 1334 N. The modulus of elasticity is 200000 N/mm^2. There is no group and all members are I profiles having the cross-section and inertia moment values of which are 5,212,900 mm^2 and 3,171,683,500 mm^4. The frame structure is decomposed into 2, 3, and 6 sub-structures, respectively, in Figure 8 with overlapping boundary nodes at the

intersections and the parallel program is executed by two, three, and six processors, depending on the number of sub-structures in the each decomposition.

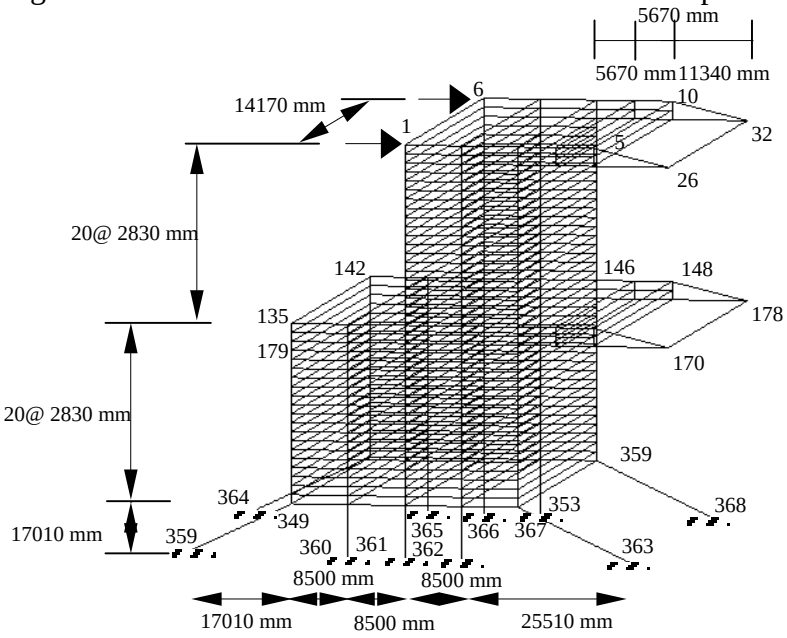


Figure 7. Topology of frame structure with 805 members.

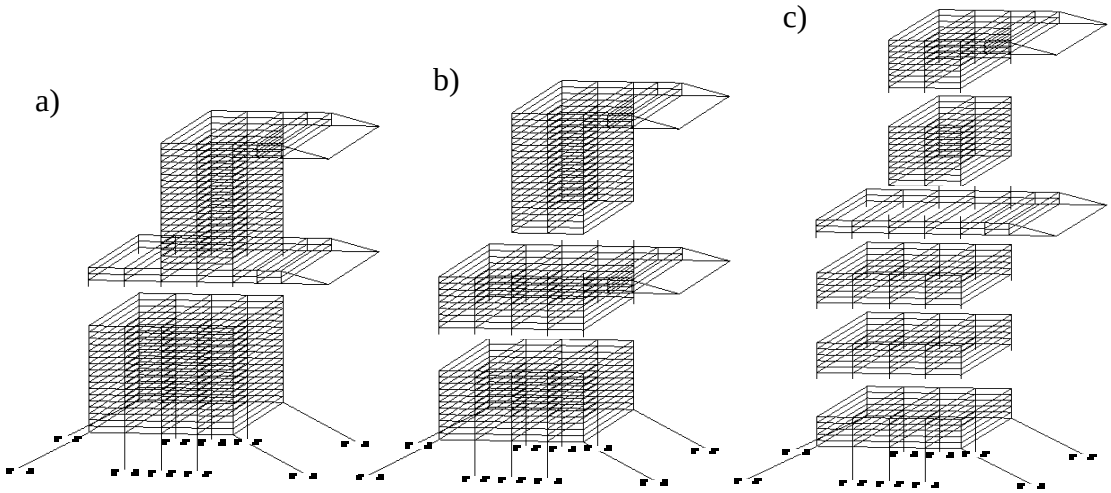


Figure 8. The decomposition models of the frame structure into a) 2, b) 3, and c) sub-structures, respectively.

As can be seen from the speed-up in Table 4, an increase in the number of processors depending on that of sub-structures causes an increase in the speedup with a deviation from the ideal case. It can also be seen from the efficiency. This deviation from the ideal case is getting more as much as the computational load is going down.

Table 4. Performance of sequence programs and parallel program.

The num Of sub-structure	Sequence	Modified Sequence	MPI	Speedup	Efficiency
	Execution time (sec)	Execution time (sec)	Execution time (sec)		
0	8.809				
2		8.992	5.606	1.604	0.802
3		6.415	3.002	2.137	0.712
6		2.785	1.016	2.741	0.457

It points out that the complexity of load corresponding to tasks should cover sufficient quantity to efficiently use parallel process. Another point to be made a difference can be that increase in the number of substructure also cause decrease in the execution time in serial manner via substructure method, which can be seen from Table 4 when made a comparison between sequence and modified sequence program.

Frame Structure with 2435 members

The frame structure having 2435 members and 1104 nodes is obtained by duplicating the former structure three times and concatenating them with 10 members in each connection. The frame is shown in Figure 9. For the structure all sections and materials are the same as the example before. Therefore, the loading state is the same as the former that for each duplicate form, too. Here each duplicate form is considered as a structure shown in Figure 10. As mentioned before, the sharing of the load having favorable complexity results in desirable performance for parallelization, which is seen from the efficiencies of decompositions with 3 sub-structures given in Table 4 and 5. Further, it is straightforward to notice, in Table 5, that increasing complexity in the size of calculation compared to other examples highlights a difference of computation time between sequence and modified sequence in the large-scale problem.

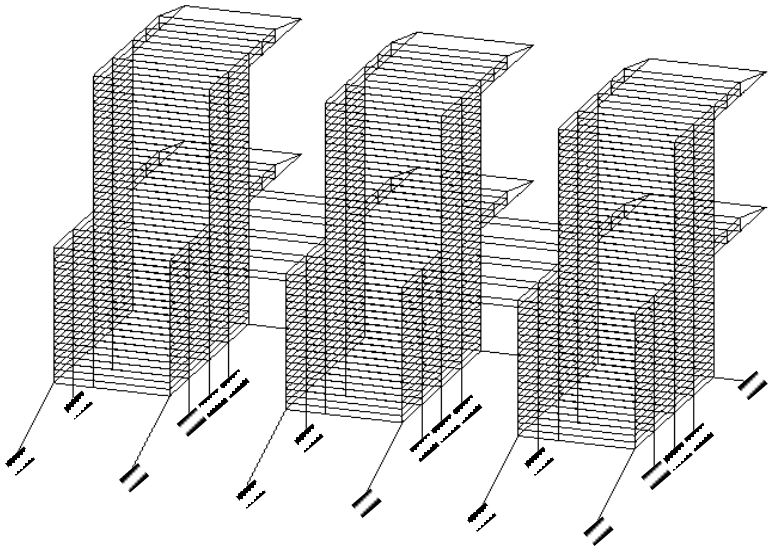


Figure 9. Topology of frame structure with 2435 members.

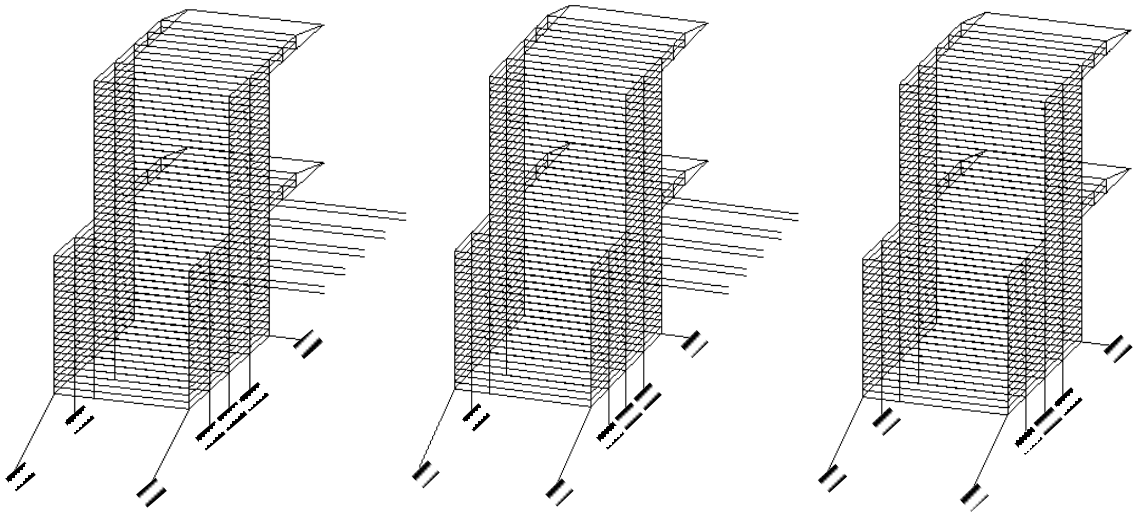


Figure 10. The decomposition model of the frame structure into 3 sub-structures.

Table 5. Performance of sequence programs and parallel program.

The num Of sub-structure	Sequence	Modified Sequence	MPI	Speedup	Efficiency
	Execution time (sec)	Execution time (sec)	Execution time (sec)		
0	532.237				
3		109.237	49.117042	2.224	0.741

Conclusions

This study presented a parallelization strategy for the analysis program solving 2D and 3D frame structure by utilizing the substructure method for computational decomposition and the programming model MPI with collective communication tools. The runs were carried out on computer system incorporating personal computers (PCs), which are equipped with a Pentium III-700MHz processor and with 512MB RAM. The operating system used on these machines is Suse Linux 9.2 and they are on an ethernet network through a 100 Mbps switch. The results obtained by sequence, modified sequence, and parallel programs for the discrete substructure paradigms were compared in terms of execution time. It is concluded that the (i) performance of parallelization depends on the size complexity of sharing computational load, (ii) the collective communication tools of MPI: broadcast and reduce are effectively used in the parallel program without any loss in numerical accuracy, and (iii) substructure method makes a major contribution to development on the performance of the current sequence program for large scale problems and computational decomposition for parallelization.

References

Leimbach, K. R., (2005) Lecture Notes- Parallel Computation. Ruhr University Bochum, Institute for Structural Mechanics, Bochum, Germany.

McGee, O. G. and Phan, K. F. (1992) Adaptable Optimality Criterion Techniques for Large-Scale Space Frames with Multiple Frequency Constraints. Computers & Structures Vol. 42, No. 2, pp. 197-210.

Michael G., (2000) Parallel Programming Models, Tools and Performance Analysis. Modern Methods and Algorithms of Quantum Chemistry, Proceedings, Second Edition, J. Grotendorst (Ed.), John von Neumann Institute for Computing, Jülich, NIC Series, Vol. 3, pp. 27-45.

URL 1- http://www-unix.mcs.anl.gov/dbpp/text/book.html (23 March 2006) MCS Mathematics and Computer Science Division.

URL 2- http://www.ciri.upc.es (23 March 2006) CEPBA-IBM Research Institute.

URL 3- http://en.wikipedia.org/wiki/Parallel_computing (21 March 2006)

URL 4- http://icl.cs.utk.edu/projectsfiles/ftmpi/pubs/mpss-ency.pdf (24 March 2006) Dongarra, J. J., Fagg, G. E., Hempel, R., and Walker, D. W. (1999) Chapter in Wiley Encyclopedia of Electrical and Electronics Engineering.

URL 5- http://radio.weblogs.com/0112083/stories/2002/08/22/usingMpiInParallelProgramm.html (24 March 2006) Haataja, J., Using MPI in Parallel Programming.