UNCONDITIONALLY STABLE CONCURRENT PROCEDURES FOR TRANSIENT FINITE-ELEMENT ANALYSIS

Michael ORTIZ

Division of Engineering, Brown University

Providence, Rhode Island 02912, USA

Bahram NOUR-OMID

Center for Pure and Applied Mathematics, University of California

Berkeley, California 94720, USA

1. Introduction

The need for more powerful computers has prompted the development of a number of multi-processor machines with multi-tasking capabilities. These are often referred to as *parallel* or *concurrent* processors. In this work we are concerned with the development of time-stepping algorithms for transient finite element analysis which lend themselves to an efficient implementation on parallel computers. This requires the modification of present algorithms to suit the new computing environments. In certain instances, algorithms that have been discarded for applications on sequential processors must be re-examined for possible use on the new parallel machines.

Two essential conditions have to be met for an algorithm to be suitable for concurrent computers:

(1) The algorithm must be such that it divides the problem into sub-tasks which require an approximately equal amount of computational effort.

(2) Each sub-task must be as independent as possible.

The first requirement ensures that all the processors start and end their work almost simultaneously, thereby reducing the idle time. The second condition is formulated with a view to minimizing the transfer of information between processors. In [1] Gentleman pointed out that the time for data communication from one processor to another can be substantial in comparison to computation time.

The element-by-element (E×E) solution procedures [2,3,4] were first proposed to reduce storage requirements on sequential computers. In [5] it was suggested that E×E algorithms are potentially

useful for concurrent processing as well. However, a closer examination reveals that although the first aforementioned requirement is met, the data transfer between sub-problems can be substantial. This is mainly due to the fact that E×E methods are based on product algorithms which are *inherently sequential*.

In this paper a new, fully parallelizable class of solution procedures for transient finite element analysis is outlined. Further details about the method can be found in [6]. The algorithms are such that any part of the structure can be processed independently of the rest over a time step. Thus, for any partition of the structure all the members of the partition can be processed independently and simultaneously, i.e., ***concurrently*** over a time step. The proposed algorithms have the structure of an *explicit scheme*. In particular, *no global equation solving effort is involved*. However, the proposed class of algorithms contains an ***unconditionally stable subclass*** for which the choice of time step size is dictated by accuracy considerations alone. This is a typically implicit-like property.

In sum, concurrent procedures may be regarded as a hybrid of implicit and explicit schemes which exhibits some of the best attributes of both types of methods, such as the unconditional stability of implicit algorithms and the concurrency of explicit schemes. This latter feature renders the proposed algorithms particularly well suited for a parallel environment.

2. A class of unconditionally stable concurrent algorithms

Next we discuss a class of time-stepping algorithms a distinct characteristic of which is that they lend themselves to an efficient implementation on parallel processors. The parallel nature of this class of algorithms owes to the fact that, for any partition of the finite element mesh, each subdomain in the partition can be processed independently of the others over a time step. In particular, one can choose a partition in which the subdomains are the finite elements themselves. In this case, all of the finite elements can be processed concurrently and independently of each other, i.e., *in parallel*. It should be emphasized, however, that an element-based partition is just a particular choice among a continuous spectrum of possibilities. In practice, the number of subdomains is limited by hardware considerations such as the number of processors in a parallel computer.

For simplicity, the method is next outlined within the context of linear heat conduction. Further details as well as an extension of the method to the dynamic case can be found in [6]. Upon application of the finite element method as a means of spatial discretization the problem is reduced to a set of

semidiscrete equations

$$\mathbf{M}\dot{\mathbf{d}} + \mathbf{K}\mathbf{d} = \mathbf{f} \tag{1}$$

where $\mathbf{d}$ denotes the nodal temperature array, $\mathbf{M}$ the capacity matrix, $\mathbf{K}$ the conductivity matrix and $\mathbf{f}$ the nodal source vector. In finite element analysis the conductivity and capacity matrices are assembled from element contributions through the assembly operation

$$\mathbf{K} = \sum_e \mathbf{K}^e, \qquad \mathbf{M} = \sum_e \mathbf{M}^e \tag{2}$$

where $\mathbf{K}^e$ and $\mathbf{M}^e$ are the element conductivity and capacity matrices, respectively.

The application of the method requires that the structure be first partitioned into subdomains. In multi-processor computers, the number of such subdomains is typically taken to be equal to the number of processors in the machine. It is interesting to note that, unlike the E×E method, the mesh partitions can here be chosen with no concern for the connectivity of the subdomains. This greatly facilitates the definition of mesh partitions. Let $\mathcal{S}$ denote the domain of analysis and $\{\mathcal{S}^a,\ a = 1,\ldots,N\}$ a given partition of the mesh into N subdomains $\mathcal{S}^a$. We shall use the symbols $\mathbf{M}^a$, $\mathbf{K}^a$ and $\mathbf{d}^a$ to denote the mass and stiffness matrices and the local solution array of substructure $\mathcal{S}^a$. Thus, $\mathbf{d}^a$ contains the nodal values of the solution at nodes within $\mathcal{S}^a$ and it fully determines the state of the subdomain. The local matrices $\mathbf{M}^a$ and $\mathbf{K}^a$ are obtained from a partial assembly (2) extended to the elements contained in subdomain a. Furthermore, let $\Gamma^{int} \equiv \bigcup_{a=1}^{N} \partial\mathcal{S}^a - \partial\mathcal{S}$ denote the 'interior boundary' of the partition. In other words, the interior boundary is the union of the parts of the subdomain boundaries which do not lie on the boundary of the overall domain. The restriction of $\mathbf{d}$ to Γ^{int} will be denoted by $\mathbf{d}^{int}$.

With this terminology, the conceptual algorithm can be stated as follows:

(i) Localize the initial conditions $\mathbf{d}_n$ to subdomains $\mathcal{S}^a$ to obtain an extended array $\tilde{\mathbf{d}}_n \equiv \{\mathbf{d}_n^1,\ldots,\mathbf{d}_n^N\}$.

(ii) Update local arrays $\{\mathbf{d}_n^a\}$ using an implicit algorithm to integrate the decoupled subproblems

$$\mathbf{M}^a\dot{\mathbf{d}}^a + \mathbf{K}^a\mathbf{d}^a = \mathbf{f}^a \tag{3}$$

Let us denote by $\tilde{\mathbf{d}}_{n+1}^*$ the extended predictor so obtained.

(iii) Mass-average $\tilde{\mathbf{d}}_{n+1}^*$ at Γ^{int} to obtain $\mathbf{d}_{n+1}^{int}$.

(iv) Integrate again the decoupled subproblems (3) with initial conditions $\mathbf{d}_n^a$ and prescribed all-around boundary conditions $\mathbf{d}_{n+1}^{int}$ to obtain the updated solution array $\mathbf{d}_{n+1}$.

Thus, the basic algorithm involves a double pass through the subdomains in the mesh partition. The sole purpose of the first pass is to determine the updated solution $\mathbf{d}_{n+1}^{int}$ on the interior boundary Γ^{int}. The second pass updates the remaining degrees of freedom for known values of the solution on the subdomain boundaries. It should be noted that in both passes all subdomains can be processed concurrently. For element-by-element mesh partitions one trivially has $\mathbf{d}_{n+1}^{int} \equiv \mathbf{d}_{n+1}$. Under these conditions, the second pass does not alter the solution and can be dropped from the algorithm. On the other extreme, if the structure is not partitioned at all one trivially recovers the implicit schemes.

REMARK 2.1. The choice of a mass-averaging rule is not arbitrary. It can be shown [**6**] that this is in fact the only choice of averaging rule which renders the algorithm consistent with the global equations of evolution. The mass-averaging rule is implemented as follows. The result of each local update $\mathbf{d}_{n+1}^{*a}$ is first weighted by the local capacity matrix $\mathbf{M}^a$. The resulting local arrays are assembled into a global vector which is then multiplied by $\mathbf{M}^{-1}$. □

REMARK 2.2. The practicality of the method clearly requires the use of a lumped capacity matrix. For most practical purposes, however, this is not a particularly stringent limitation. □

REMARK 2.3. In general, the proposed algorithm can only be expected to be first-order accurate, i.e., $\mathbf{d}_{n+1} = \mathbf{d}(t_n + h) + \mathbf{O}(h^2)$ whenever $\mathbf{d}_n = \mathbf{d}(t_n)$. In [**6**] it is shown how higher-order algorithms can be derived from the first-order scheme discussed here. □

REMARK 2.4. It should be noted that the updates of the subdomains involve local operations only. In particular, the global stiffness matrix need not be assembled at any time during the integration process, much less factorized. □

REMARK 2.5. A particularly promising feature of the proposed class of algorithms is the fact that they are amenable to a fully *parallel implementation*, whereby all the subdomains in the partition are processed concurrently and independently of each other over a time step. It should be emphasized that the mesh partitions can be defined in a completely arbitrary manner, with no concern for the connectivity of the subdomains. This greatly facilitates the definition of mesh partitions. Another interesting aspect of the algorithm is that exchange of information between the subdomains is only required at the end of a time step. This has the effect of reducing the extent of interprocessor communication to a minimum.

All this is in sharp contrast to other 'semi-implicit' schemes proposed in the past which, although parallelizable to some extent, are inherently sequential, require elaborate schemes to define the mesh partitions and involve interprocessor communications during each time step. □

REMARK 2.6. Clearly, the properties of the proposed concurrent procedures depend on the choice of local update algorithm. It can be shown [**6**] based on Iron's bounding principle [7] that if the local algorithms are unconditionally stable then resulting concurrent procedure is also ***unconditionally stable.*** In other words, concurrent procedures preserve the stability of the local algorithms utilized to update the subdomains. □

A first numerical example is shown in Fig. 1. The analysis is **concerned** with linear heat conduction in a bar with prescribed boundary conditions at both ends. The bar was discretized into 100 linear 2-node elements and the resulting mesh partitioned into 4 subdomains each containing 25 elements. The decoupled subproblems were integrated using the backward-Euler algorithm. Fixed time step sizes were utilized throughout the integration process. As may be seen from Fig. 2, the computed results exhibit good accuracy over a wide range of time steps.

3. Accuracy under successive refinements of the partition

The question that naturally arises now is what is the effect on the overall accuracy of the algorithm of successive refinements of the mesh partition. The question is motivated by the observation that the smaller the subdomains in the partition the cheaper is one application of the algorithm. In particular, when element-by-element mesh partitions are utilized the cost of one application of the algorithm is reduced to a minimum. However, numerical experiments inmediately show that increasing the number of subdomains has an adverse influence on the accuracy of the algorithm. This effect is best illustrated by examining the limiting case of element-by-element partitions of the mesh. In this case, the major restriction on the time step size stems from the fact that one application of the algorithm propagates element information to adjacent elements only. This limited flow of information is particularly stringent when analyzing parabolic systems which are far away from equilibrium. In such cases, information needs to be rapidly exhanged between distant sections of the structure. The situation is aggravated by fine meshes for which information has to traverse many elements at the expense of many applications of the algorithm before it is propagated over an appreciable distance. A similar analysis for another class of algorithms has been reported elsewhere [8].

These considerations point to the need of combining element-by-element partitions with a *step-changing technique* to control accuracy. Here the aim is to devise a criterion whereby the time step size is automatically reduced when rapid flow of information is required and increased whenever accuracy permits. A simple strategy is based on Richardson's extrapolation and uses the difference between two solutions $\mathbf{d}_n(h/2)$ and $\mathbf{d}_n(h)$ obtained with step sizes $h/2$ and h, respectively, to estimate the local truncation error as

$$\epsilon_n \approx \| \mathbf{d}_n(h/2) - \mathbf{d}_n(h) \| \tag{4}$$

(see, e.g., [9] where alternative methods are given). Based on this estimate it is possible to determine the extent by which the time step size h has to be reduced or can be increased to satisfy a local trucation error condition

$$\epsilon_n < r \tag{5}$$

for some given tolerance r.

The performance of element-by-element concurrent algorithms can be illustrated by means of the problem stated in Fig. 3. The analysis is concerned with linear heat conduction in a circular region subjected to a sudden temperature rise along the boundary. A mesh of 100 isoparametric 4-node elements was employed. The Crank-Nicolson algorithm (see, e.g., [10]) was utilized for the local updates. The error norm involved in estimate (4) was taken to be $\| \mathbf{d} \| \equiv \left(\frac{1}{n} \sum_{i=1}^{n} d_i^2 \right)^{1/2}$ where n denotes the number of degrees of freedom in the model. Fig. 4a shows a comparison between the exact solution and the results obtained for local truncation error tolerances $r = 10^{-4}$ and 10^{-3}. The more stringent tolerance is seen to result in accurate predictions. As larger local truncation errors are allowed, a loss of accuracy is observed which manifests itself as an overly slow relaxation.

The behavior of the step-changing procedure is exhibited in Fig. 4b. It is seen that during the first stages of the relaxation process when the system is far away from thermal equilibrium accuracy demands the use of small time steps. As the system relaxes, the required step size steadily increases. Whereas for explicit integration this growth has to be stopped at the critical time step h_c to avoid numerical instabilities, concurrent algorithms can be used with time steps of any size as accuracy permits. Fig. 4b shows how the critical time step h_c is eventually exceeded without instabilities in the response or

any significant loss of accuracy. As a result, the 'average time step', i.e., the duration of the analysis divided by the total number of time steps can be substantially larger for concurrent algorithms than for explicit schemes, which renders the former more economical. In view of this numerical evidence, it should be emphasized that an efficient implementation of the method based on element-by-element mesh partitions within the context of parabolic problems requires that it be combined with a time step-changing technique.

The above numerical results clearly indicate that increasing the number of subdomains in the mesh partition has two opposite effects. On one hand, one application of the algorithm becomes increasingly cheaper. On the other hand, to maintain a given level of accuracy the time step has to be decreased, which adds to the cost of the analysis. The question is which effect dominates and whether using concurrent procedures instead of implicit algorithms is cost effective. That this is so can be illustrated by means of a simple example. Consider the nonlinear 3D dynamic analysis of a cube subdivided into N equal subdomains. The case of implicit integration corresponds to $N = 1$. Numerical tests show that to maintain the same level of accuracy obtained from implicit schemes the time step has to chosen so as to satisfy a Courant condition based on the dimensions of the subdomains. Thus, the time increment has to be decreased as $\mathbf{O}(1/N^{1/3})$ as the number of subdomains increases and consequently the number of steps in the analysis has to be increased as $\mathbf{O}(N^{1/3})$. On the other hand, the number of degrees of freedom per subdomain decreases as $\mathbf{O}(1/N)$ and the bandwith as $\mathbf{O}(1/N^{2/3})$. Therefore, the execution time involved in factorizing a local array decreases as $\mathbf{O}(1/N^{7/3})$. Identifying the cost of one application of the algorithm with that of one local factorization then the total execution time for the analysis goes as $\mathbf{O}(N^{1/3}) \times \mathbf{O}(1/N^{7/3}) \approx \mathbf{O}(1/N^2)$. This shows that concurrent algorithms may be expected to cut down significantly on execution times with respect to implicit algorithms. Similar estimates hold for 2D hyperbolic and 2D and 3D parabolic problems.

4. Summary and conclusions

A new family of algorithms has been outlined which would appear to be particularly well-suited for implementation in a parallel environment. This owes to the fact that for any partition of the mesh each subdomain in the partition can be processed over a time step simultaneously and independently of the rest. The method eliminates the need for assembling and factorizing large global arrays while retaining the unconditional stability properties of the algorithms used at the local level. To critically appraise the proposed methodology, two limiting cases may be considered:

Element-by-element mesh partitions. An appealing feature of element-by-element partitions is that they render the implementation of the method a trivial exercise. Thus, for any finite element code with an explicit algorithm the method can be implemented by merely replacing the usual element stiffness and conductivity matrices by suitably modified ones. Furthermore, this choice of partition has the effect of minimizing storage requirements and arithmetic operations per time step. It is interesting to note that the first order method requires the same number of arithmetic operations per time step as the single pass E×E method. However, for dynamic problems numerical experiments demonstrate the superior accuracy of concurrent algorithms over the E×E method discussed in [3]. For parabolic problems, concurrent algorithms based on element-by-element partitions share the same accuracy limitations as explicit schemes and E×E methods. These limitations arise as a consequence of the limited flow of information per time step allowed by the algorithms. However, as shown above the combination of concurrent algorithms with a step changing technique results in an accurate and reliable procedure which can be significantly more economical that explicit schemes.

Coarse mesh partitions. The use of coarse mesh partitions is a natural choice when implementing the method in concurrent computers. In a parallel environment, the number of subdomains in the partition is dictated by the number of processors in the machine. Remarkably, the numerical evidence presented above shows that the use of coarse mesh partitions is also optimal from the standpoint of both accuracy and cost efficiency. Thus, it would appear that by far the most promising characteristic of the proposed algorithms is their suitability for an efficient and straightforward parallel implementation. By contrast, in this context E×E procedures are cumbersome particularly when applied to structures with complicated topologies. **Even for regular meshes the E×E method may not be amenable for a fully parallel** implementation. For instance, for a rectangular domain with a regular mesh some degree of parallelism can be obtained from the E×E method as a consequence of the fact that the mesh can be partitioned into four disjoint groups. Then, the elements in each group can be processed concurrently but the groups have to be processed sequentially. Thus, even in this simple case full parallelism cannot be achieved with the E×E method. For arbitrary 2D and 3D topologies a graph coloring algorithm has to be implemented to partition the mesh into disjoint subdomains. This task is by no means trivial. Furthermore, simple bar models can be formulated for which no degree of parallelism at all can be obtained from the E×E method.

In addition, even in the cases where disjoint groups can be easily found, when the processing of a group is completed a set of data pertaining to the intermediate solution has to be transferred between processors. The time and cost involved in these operations can be substantial [1]. By contrast, the method presented here requires no special partitioning schemes and performs fewer interprocessor communications.

In conclusion, whereas the proposed methodology can be useful in sequential machines as well, it would appear to be most promising as it bears on parallel computation. It should also be emphasized that extensions of the method to nonlinear problems are possible.

References

[1] W. M. Gentleman, Some complexity results for matrix computation on parallel processors, J. Ass. Comput. Mach., Vol. 25 (1978) 112-115.

[2] B.Nour-Ohmid and B. N. Parlett, Element preconditioning, Report PAM-103, Center for Pure and Applied Mathematics, University of California, Berkeley, California, Oct. 1982.

[3] M. Ortiz, P. Pinsky and R. L. Taylor, Unconditionally stable element-by-element algorithms for dynamic problems, Comp. Meth. Appl. Mech. Eng., Vol. 36, No. 2 (1983) 223-239.

[4] T. J. R. Hughes, I. Levit and J. Winget, An element-by-element solution algorithm for problems of structural and solid mechanics, Comp. Meth. Appl. Mech. Eng., Vol. 36, No. 2 (1983) 241-254.

[5] J. M. Winget, Element-by-element solution procedures for nonlinear transient heat conduction analysis, Ph. D. Dissertation, Report CLASSIC-83-05, Center for Large Scale Scientific Computation, Stanford University, Stanford, California, Nov. 1983.

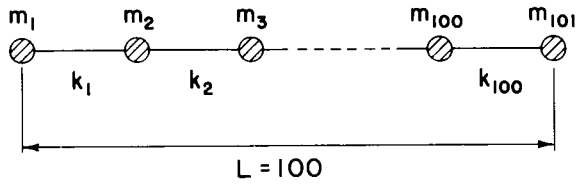
[6] M. Ortiz and B. Nour-Ohmid, Unconditionally stable concurrent procedures for transient finite element analysis, Comp. Meth. Appl. Mech. Engr., (to appear).

[7] B. M. Irons, Applications of a theorem on eigenvalues to finite element problems, (CR/132/70) University of Wales, Department of Civil Engineering, Swansea, 1970.

[8] R. Mullen and T. Belytschko, An analysis of an unconditionally stable explicit method, Computers and Structures, Vol. 16, No. 6 (1983) 691-696.

[9] C. A. Felippa and K. C. Park, Direct time integration methods in nonlinear structural dynamics, Comp. Meth. Appl. Mech. Eng., Vol. 17/18 (1979) 277-313.

[10] T. J. R. Hughes and T. Belytschko, A precis of developments in computational methods for transient analysis, J. Appl. Mech., Vol. 50 (1983) 1033-1041.



MATERIAL PROPERTIES :

$m_1 = m_{101} = 0.5$

$m_2 = m_3 = \text{----} = m_{100} = 1.0$

$k_1 = k_2 = \text{----} = k_{100} = 1.0$

BOUNDARY CONDITIONS :

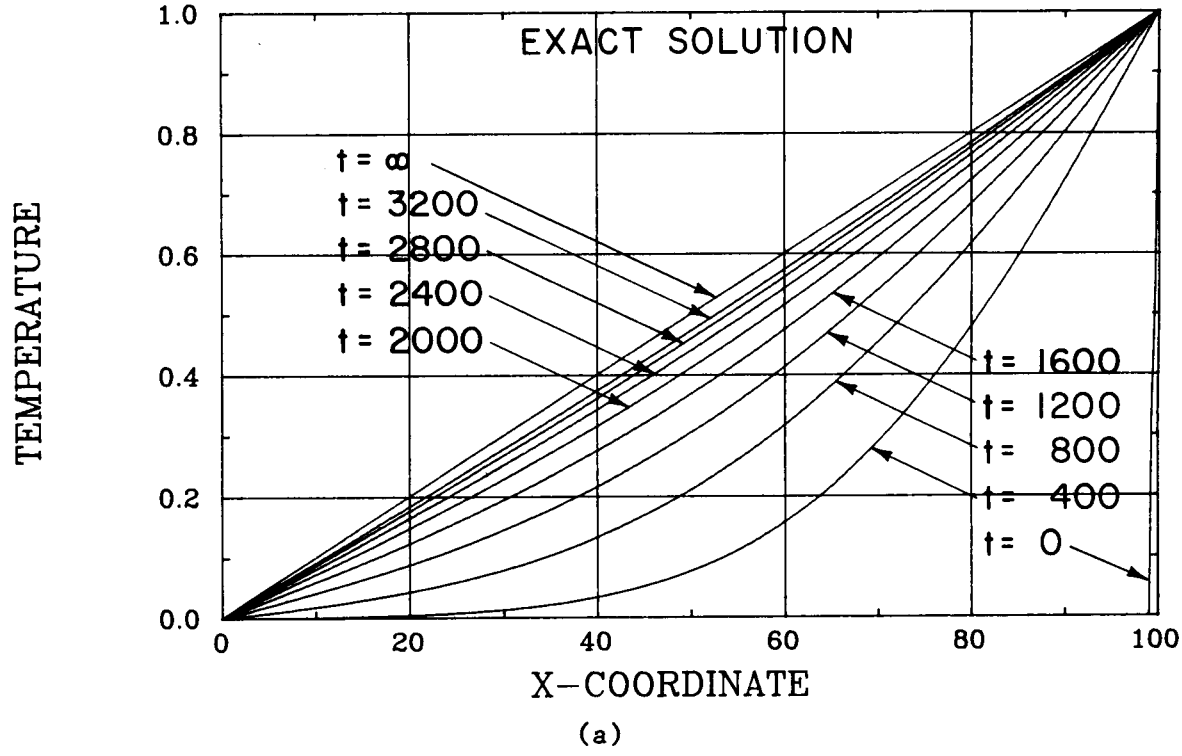
$d_1(t) = 0.0$

$d_{101}(t) = 1.0$

INITIAL CONDITIONS :

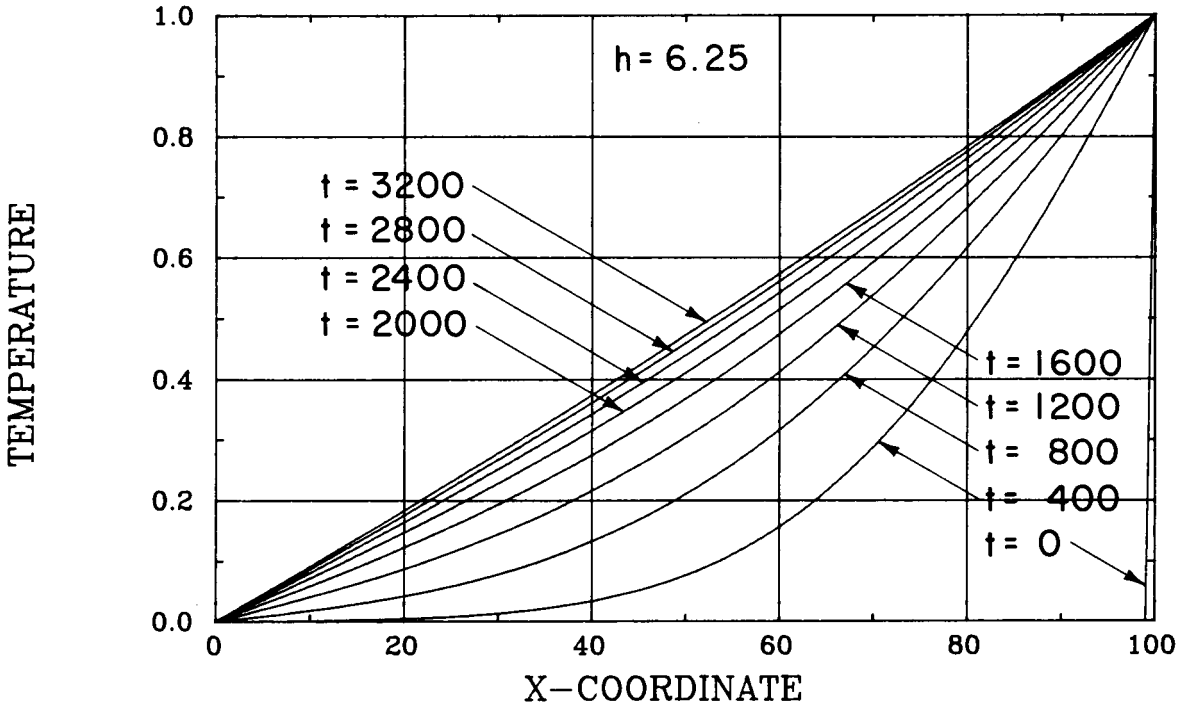
$d_2 = d_3 = \text{----} = d_{100} = 0.0$

Fig. 1. Definition of test problem: heat conduction in a bar with prescribed temperatures at both ends.

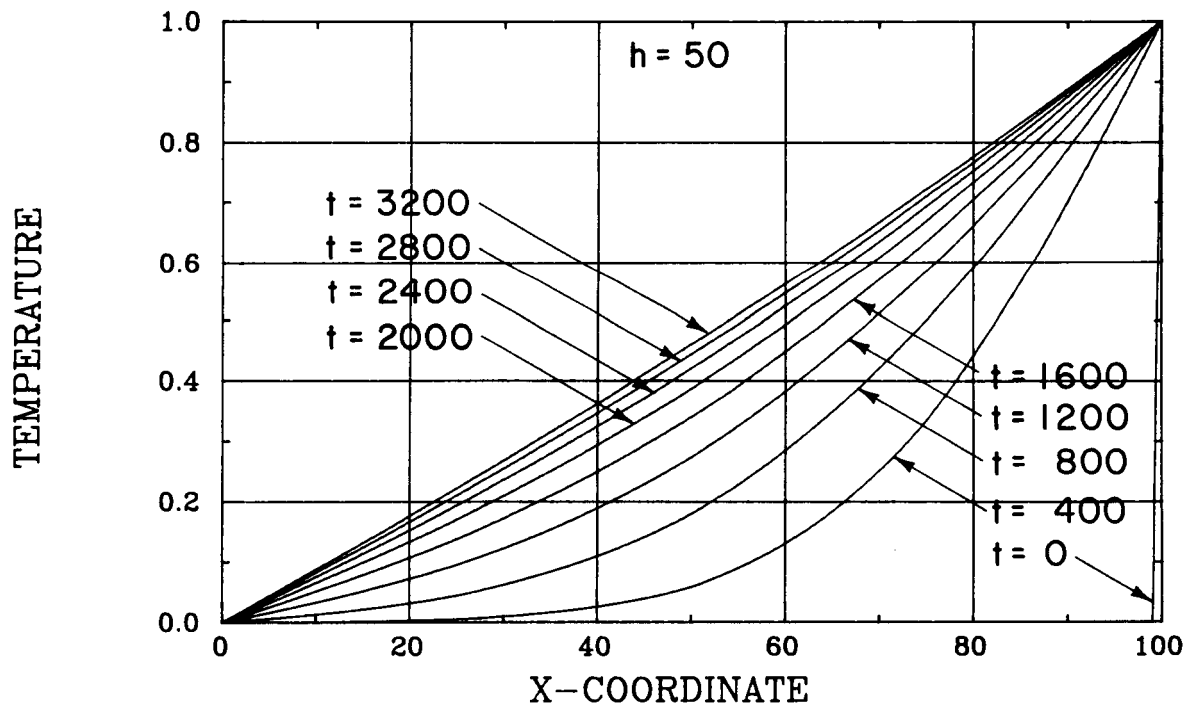


(a)

Fig. 2. Computed results for problem in Fig. 1 using concurrent algorithm based on a partition of the mesh into four subdomains.

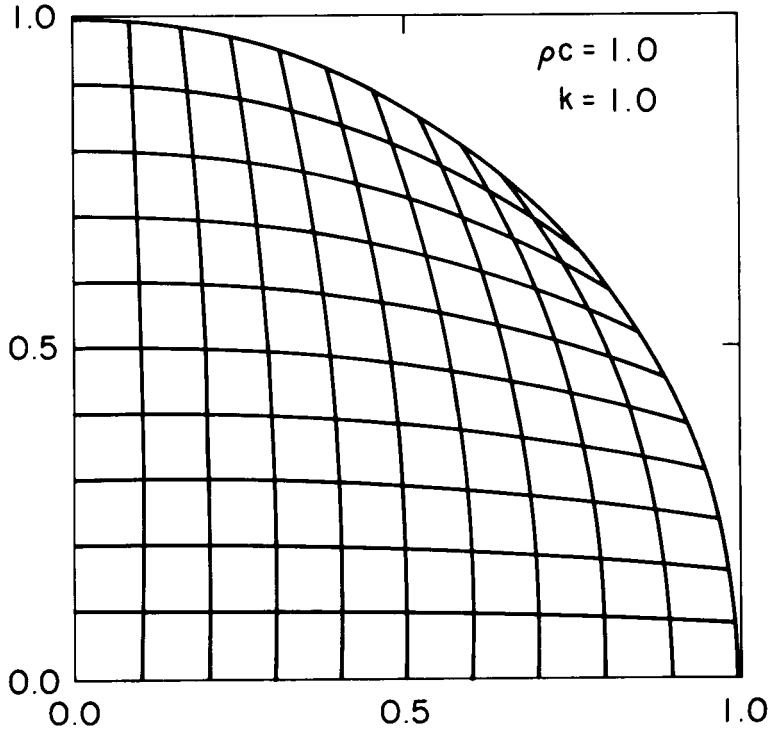


(b)

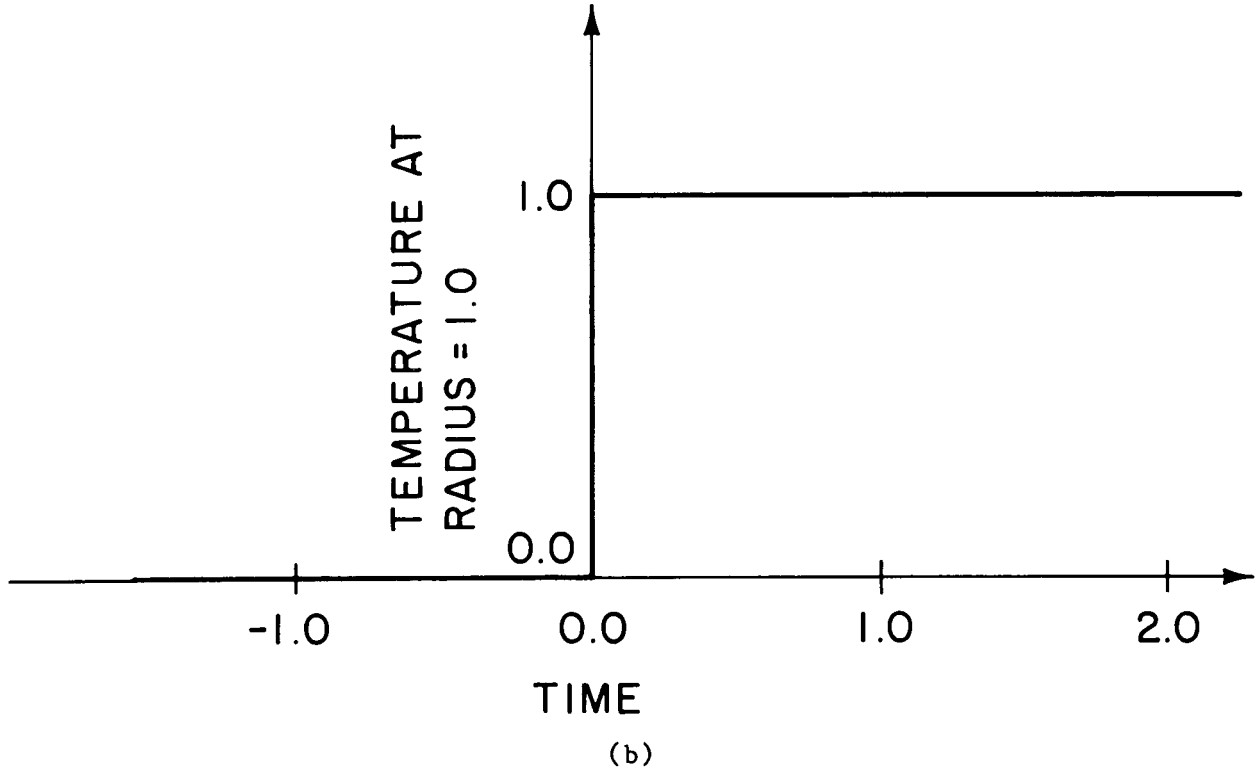


(c)

Fig. 2. Concluded.

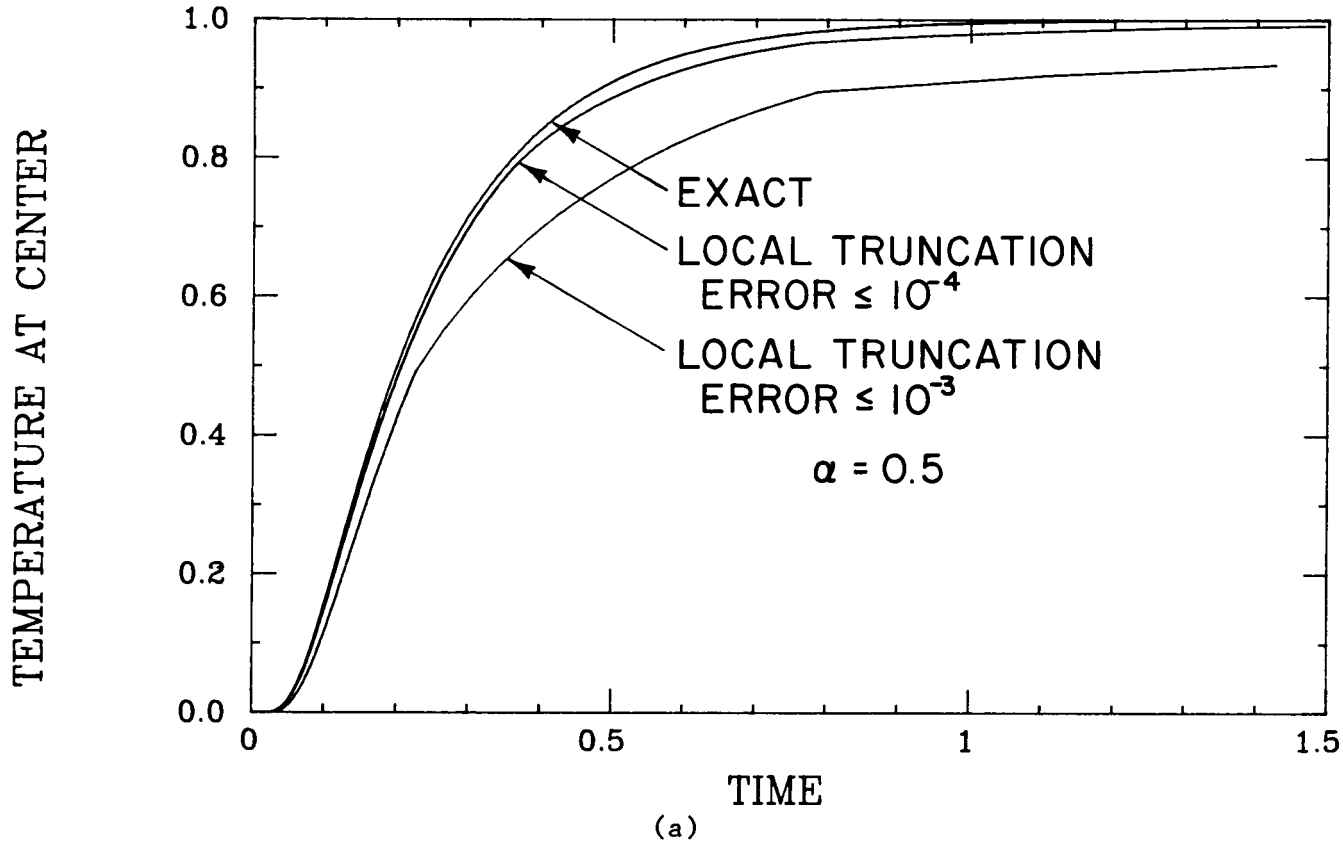


(a)



(b)

Fig. 9. Definition of test problem: heat conduction in a circular domain subjected to a sudden rise in temperature along its exterior boundary.



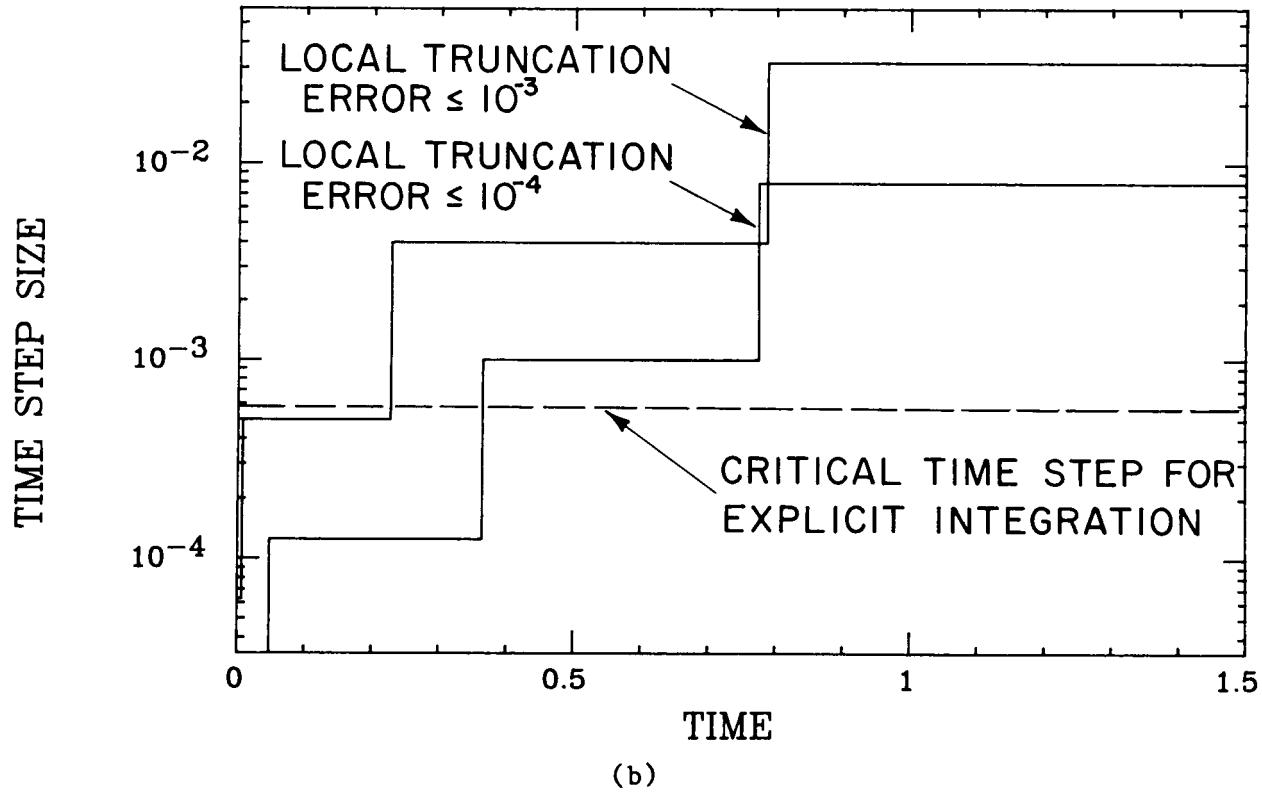


Fig. 4. Results computed from element-by-element concurrent algorithm with step-changing scheme for problem in Fig. 3. a) Time evolution of temperature at center node. b) Step size variation.