On the Computational Efficiency and Implementation of Block-Iterative Algorithms for Nonlinear Coupled Problems

M. Cervera
R. Codina
M. Galindo

Publication CIMNE Nº 43, February 1993

On the Computational Efficiency and Implementation of Block-Iterative Algorithms for Nonlinear Coupled Problems

M. Cervera
R. Codina
M. Galindo

Publication CIMNE Nº 43, February 1993

International Center for Numerical Methods in Engineering
Gran Capitán s/n, 08034 Barcelona, España

ON THE COMPUTATIONAL EFFICIENCY AND IMPLEMENTATION OF BLOCK–ITERATIVE ALGORITHMS FOR NONLINEAR COUPLED PROBLEMS

M. CERVERA, R. CODINA AND M. GALINDO

E. T. S. Ingenieros de Caminos, Canales y Puertos
Technical University of Catalonia
Gran Capitán s/n, Edificio C1, 08034 Barcelona, Spain

ABSTRACT

In this paper we outline a general methodology for the solution of the system of algebraic equations arising from the discretization of the field equations governing coupled problems. In our examples, we shall consider that this discrete problem is obtained from the finite element discretization in space and the finite difference discretization in time. Our motivation is to preserve software modularity, to be able to use existing single field codes to solve more complex problems, and to exploit computer resources optimally, emulating parallel processing. To this end we deal with two well-known coupled problems of computational mechanics—the fluid-structure interaction problem and thermally driven flows of incompressible fluids. The possibility of coupling the block-iterative loop with the nonlinearity of the problems is demonstrated through numerical experiments, which suggest that even a mild nonlinearity drives the convergence rate of the complete iterative scheme, at least for the two problems considered here. The paper discusses the implementation of this alternative to the direct coupled solution, stating advantages and disadvantages. The need for on–line synchronized communication between the different codes used is also explained, together with the description of the master code who will control the overall algorithm.

KEYWORDS: Nonlinearity Coupling Block-iterative techniques Fluid-structure interaction Thermally driven flows

INTRODUCTION

Coupled problems arise frequently in engineering applications. As defined in reference 1, *"coupled systems and formulations are those applicable to multiple domains and dependent variables which describe different physical phenomena and in which (a) neither domain can be solved while separated from the other; and (b) neither set of dependent variables can be eliminated at the differential equation level"*. It is also usual[1] to classify coupled systems in two categories: (I) those problems in which coupling occurs on domain interfaces via the boundary conditions imposed there, and (II) those problems in which the coupling comes from different physical phenomena which occur on (totally or partially) overlapping domains. The methodology to be described in the following will be applied to two well–known coupled problems, one falling in each of these two groups: fluid–structure interaction and thermally driven flows of incompressible fluids.

Numerical methods applied to these coupled problems lead to the solution of a set of nonlinear algebraic equations which necessarily involve the (nodal) variables corresponding to the various domains (for Class I) or to the various physical phenomena (for Class II). Thus, the alternatives to solve a coupled problem are two–fold:

(a) To treat all the domains simultaneously. This leads to a single set of algebraic equations involving all the relevant variables. In general, these variables will not be homogeneous, as they represent discretization of different domains and/or different physical phenomena.

(b) To treat the domains one at the time, considering the coupling terms as forcing terms on the right–hand side of the equations. This leads to several sets of algebraic equations (one per domain), each of them to be solved solely for the variables related to one domain, but with the right hand side depending on variables related to the other domains.

Strategy (a) necessarily requires the development of a special–purpose code, probably involving collaboration from different expertise areas. Standard engineering software developed for uncoupled problems may be of little help when writing such a program, due to its particular structure. The outcome of this *may well be* a complicated code, difficult to maintain, modify or upgrade, and even difficult to use. This program could only be parallelized at basic instruction level. Moreover, even though for a 'standard' coupled problem this alternative could make sense, the effort that it involves is hardly affordable for all the coupled problems one must be ready to solve in engineering practice. For example, let us mention two other particular problems to which we have been faced recently. One of them arises when the deformation of the roll in the rolling process of a flat metal plate has to be taken into account in order to reproduce more precisely the final shape of the metal piece. Another coupled problem has been encountered in the thermal analysis of a mould filling simulation of a casting process. The boundary condition for the temperature inside the cavity of the mould depends on the temperatures of the mould itself via the classical Robin condition, and these temperatures are in turn determined by those of the fluid filling the cavity.

Strategy (b), on the other hand, allows each domain/problem to be tackled on its one. The codes used may be either new or existing programs, slightly modified to account for the coupling terms. Each of these codes may be developed by a different expert or team of experts on the particular field, using optimal (and different) strategies for each of them. The outcome of this *should be* a set of (relatively) simple programs, easy to maintain, modify or upgrade, one independently of the others. Note that this approach is parallel *by construction*, and at module level. On a multi–user non–parallel machine, this approach turns the coupled problem being run into several interconnected processes. So, the kernel of the operating–system is working, automatically and inadvertedly, as a pseudo–parallel processor emulator.

Our first concern in this paper will be to study the application of Strategy (b) to the two nonlinear coupled problems mentioned earlier and, in particular:

- To cast the fully discrete finite element equations of both problems in the general framework of block-iterative techniques.

- To consider the linearized forms of these equations and to justify a unique iterative loop to deal both with the coupling and the nonlinear terms.
- To study the numerical performance of the final scheme from the standpoint of cost–effectiveness and rate of convergence.
- To justify the use of block–iterative methods vs the direct solution of the original coupled systems.

Of course, Strategy (b) has some drawbacks, and for some specific coupled problems these may prove this alternative impractical. In any case, if the goal outlined above, namely, modularity of the software used for coupled problems, is to be achieved, a corner stone will be required: a *master code* that will perform, at least, three tasks, namely, the transfer of information from the different problems, the checking of convergence and the synchronization of the overall process.

Having these general objectives in mind, the paper is organized as follows. In the next two sections we describe in more detail options (a) and (b), viewing the latter as a block–iterative procedure which can be coupled with nonlinear iterative loops. Next, this strategy is applied to the fluid–structure interaction problem and to the numerical solution of thermally driven flows, showing some numerical results that illustrate our discussion. Once the use of block–iterative methods is justified, the last section is concerned with the description of the basic characteristics of a computer code whose goal is to interact single field analyzers in order to solve coupled problems. The modifications to be introduced to the original codes are also indicated. It is shown that only a few coding lines have to be added, so that the solution of most coupled problems turns out to be an extremely simple task once the above mentioned master code is available.

ALGORITHMIC SOLUTION FOR STRATEGY (a)

The discretization of the continuous problems to be considered will lead to a nonlinear algebraic system of the form

$$\begin{bmatrix} \mathbf{A}_{11}(\mathbf{x}) & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22}(\mathbf{x}) \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix} \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the vectors of nodal unknowns at a certain time step of the two fields under consideration, $\mathbf{f}_1$ and $\mathbf{f}_2$ are the vectors of 'force' terms and $\mathbf{A}_{ij}$, $i,j = 1,2$, are matrices, the dependence of which on the unknown $\mathbf{x}$ has been explicitly indicated. The discussion of problem (1) is enough for our purposes, although it would be straightforward to extend what follows to other situations, such as several-fields problems or other nonlinear dependencies. Observe that in problem (1) a *linear* coupling of the first equation with the second is assumed, as well as a *linear* behaviour of $\mathbf{y}$ for a given $\mathbf{x}$.

The algorithm for the direct solution of problem (1), Strategy (a), can be chosen among the variety of linearization schemes available for the solution of nonlinear problems. Here we can mention the well–known Newton–Rahpson method (or any of its variations, known as *modified* Newton–Rahpson method), the Picard method, or the somehow more sophisticated, Quasi– and Secant–Newton methods (see, for instance, reference 2 for a discussion

on the relative merits of theses schemes). We will make use of the modified Newton–Rahpson and Picard methods when dealing with the particular problems of fluid-structure interaction and thermally driven flows. The steps in the solution process would follow exactly those necessary to solve an uncoupled nonlinear problem of similar characteristics. One disadvantage of this strategy is that the structure of the global matrices $\mathbf{A}_{ij}$ is such that entries come from the two different fields, and so, either integrals have to be evaluated in two different domains (Class I problems), or they represent physically different magnitudes (Class II problems). Another disadvantage is the larger size of the global matrix as compared with the ones arising from the different domains/fields. On the other hand, the advantage is that the final algorithm is easily and clearly defined, and its analysis, regarding for instance convergence, is feasible.

ALGORITHMIC SOLUTION FOR STRATEGY (b)

Let us consider now the use of block-iterative algorithms to solve problem (1). This will reduce the size of the resulting subproblems at the expense of iterating. Assuming that the first equation in (1) is solved first, there are two possible block-iterative schemes, namely,

$$\begin{aligned} \mathbf{A}_{11}(\mathbf{x}^{(i)})\mathbf{x}^{(i)} &= \mathbf{f}_1 - \mathbf{A}_{12}\mathbf{y}^{(i-1)} \\ \mathbf{A}_{22}(\mathbf{x}^{(i-1)})\mathbf{y}^{(i)} &= \mathbf{f}_2 - \mathbf{A}_{21}\mathbf{x}^{(k)}, \ k = i-1 \text{ or } i \end{aligned} \tag{2}$$

Here, superscripts in parenthesis refer to iteration counters. For $k = i - 1$ this is the *block-Jacobi* (or *block-total-step*) method, whereas for $k = i$ it is the *block-Gauss-Seidel* (or *block-single-step*) method. Observe that the matrix $\mathbf{A}_{22}$ is evaluated using $\mathbf{x}^{(i-1)}$. The use of $\mathbf{x}^{(i)}$ would imply a certain treatment of the nonlinear term $\mathbf{A}_{22}(\mathbf{x})\mathbf{y}$, as will be shown below. From elementary numerical analysis it is known that, under certain conditions, both the Jacobi and the Gauss-Seidel methods converge linearly when applied to linear systems, the convergence rate of the latter being twice higher than that of the former. In their *block* counterparts these properties are inherited, the convergence rate depending now on the spectral radius of the matrices involved.

On the other hand, problem (1) (and also problem (2)) is nonlinear, so that an iterative procedure must be used to deal with this nonlinearity. Both the Picard (or fixed point) and the Newton-Raphson methods are defined by approximating

$$\mathbf{A}_{11}(\mathbf{x}^{(i)})\mathbf{x}^{(i)} \approx \mathbf{A}_{11}(\mathbf{x}^{(i-1)})\mathbf{x}^{(i)} + \beta_1 \mathbf{A}^*_{11}(\mathbf{x}^{(i-1)})(\mathbf{x}^{(i)} - \mathbf{x}^{(i-1)}) \tag{3}$$

$$\mathbf{A}_{22}(\mathbf{x}^{(i)})\mathbf{y}^{(i)} \approx \mathbf{A}_{22}(\mathbf{x}^{(i-1)})\mathbf{y}^{(i)} + \beta_2 \mathbf{A}^*_{22}(\mathbf{x}^{(i-1)}, \mathbf{y}^{(i-1)})(\mathbf{x}^{(i)} - \mathbf{x}^{(i-1)}) \tag{4}$$

where $\beta_j = 0$ or 1 determines if the jth equation in (1) ($j = 1, 2$) is linearized using the Picard or the Newton-Raphson schemes, respectively. Matrices $\mathbf{A}^*_{11}$ and $\mathbf{A}^*_{22}$ arise from the derivation of $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ with respect to $\mathbf{x}$.

Using expressions (3) and (4), the linearized version of problem (1) is

$$\begin{bmatrix} \mathbf{A}_{11}(\mathbf{x}^{(i-1)}) + \beta_1 \mathbf{A}^*_{11}(\mathbf{x}^{(i-1)}) & \mathbf{A}_{12} \\ \mathbf{A}_{21} + \beta_2 \mathbf{A}^*_{22}(\mathbf{x}^{(i-1)}, \mathbf{y}^{(i-1)}) & \mathbf{A}_{22}(\mathbf{x}^{(i-1)}) \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(i)} \\ \mathbf{y}^{(i)} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 + \beta_1 \mathbf{A}^*_{11}(\mathbf{x}^{(i-1)})\mathbf{x}^{(i-1)} \\ \mathbf{f}_2 + \beta_2 \mathbf{A}^*_{22}(\mathbf{x}^{(i-1)}, \mathbf{y}^{(i-1)})\mathbf{x}^{(i-1)} \end{bmatrix} \tag{5}$$

Problem (2) is nonlinear and problem (5) is linear, but coupled. For each problem, either the nonlinearity or the coupling could be dealt with in a nested iterative loop. However, there is the strong temptation to use a single iterative loop to deal both with the nonlinearity and the coupling. Starting from problem (5), this would lead to:

$$\left[\mathbf{A}_{11}(\mathbf{x}^{(i-1)}) + \beta_1 \mathbf{A}^*_{11}(\mathbf{x}^{(i-1)})\right] \mathbf{x}^{(i)} = \mathbf{f}_1 + \beta_1 \mathbf{A}^*_{11}(\mathbf{x}^{(i-1)})\mathbf{x}^{(i-1)} - \mathbf{A}_{12}\mathbf{y}^{(i-1)} \tag{6}$$

$$\mathbf{A}_{22}(\mathbf{x}^{(i-1)})\mathbf{y}^{(i)} = \mathbf{f}_2 + \beta_2 \mathbf{A}^*_{22}(\mathbf{x}^{(i-1)}, \mathbf{y}^{(i-1)})\mathbf{x}^{(i-1)} - \beta_2 \mathbf{A}^*_{22}(\mathbf{x}^{(i-1)}, \mathbf{y}^{(i-1)})\mathbf{x}^{(k)} - \mathbf{A}_{21}\mathbf{x}^{(k)} \tag{7}$$

where $k = i - 1$ or $k = i$. Either for $\beta_2 = 0$ (Picard linearization of $\mathbf{A}_{22}(\mathbf{x})\mathbf{y}$) or for $k = i - 1$ (block-Jacobi coupling) Eq. (7) reduces to

$$\mathbf{A}_{22}(\mathbf{x}^{(i-1)})\mathbf{y}^{(i)} = \mathbf{f}_2 - \mathbf{A}_{21}\mathbf{x}^{(k)} \tag{8}$$

From Eq. (4) it may be concluded that for $\beta_2 = 1$ (Newton-Raphson's linearization of $\mathbf{A}_{22}(\mathbf{x})\mathbf{y}$) *and* $k = i$ (block-Gauss-Seidel coupling) Eq. (7) is simply

$$\mathbf{A}_{22}(\mathbf{x}^{(i)})\mathbf{y}^{(i)} = \mathbf{f}_2 - \mathbf{A}_{21}\mathbf{x}^{(i)} \tag{9}$$

This is perhaps the simplest choice to implement: once $\mathbf{x}^{(i)}$ is known by solving Eq. (6), it is used to evaluate the matrix $\mathbf{A}_{22}$ and then Eq. (9) may be solved. We stress that this *natural* approach implies a high order linearization of the second equation in problem (1) *and* the use of the block-Gauss-Seidel coupling.

The algorithm for the block–iterative solution of problem (2), Strategy (b), is outlined in Box 1 *for field number 1.* Obviously, field number 2 could follow an identical procedure, although the block–iterative approach allows for different schemes to be used for the various fields present in the problem. Note that the use of the Picard or Newton–Raphson schemes has been assumed. It must be remarked that the depicted algorithm follows very closely the standard process for an uncoupled nonlinear problem, apart from the addition of the *export/import operations* and the evaluation of the interaction term.

The advantage is now that the structure of the matrices in the left hand–side (LHS), $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$, is such that entries come only from the field currently considered, and so, integrals have to be evaluated only in that domain, and they represent physically homogeneous magnitudes. Moreover, for many practical applications these matrices are symmetric. The difficulty of evaluating the terms appearing in the right hand–side (RHS) of Eq. (2) is problem dependent. For Class I problems, they only involve integrals over the "interaction boundary", and this is "seen" from both domains. For Class II problems,

INITIALIZE
$i := 0, \quad \mathbf{x}^{(0)} := \mathbf{x}^0$

REPEAT (* For each iteration *)

INCREMENT
$i := i + 1$

COMPUTE RESIDUAL
$\mathbf{r}_1 := \mathbf{f}_1 - \beta_1 \mathbf{A}^*_{11}(\mathbf{x}^{(i-1)})\mathbf{x}^{(i-1)}$

IMPORT
$\mathbf{y}^{(i-1)}$

COMPUTE INTERACTION TERM
$\mathbf{f}_{c1} = \mathbf{A}_{12}\mathbf{y}^{(i-1)}$

MODIFY RESIDUAL
$\mathbf{r}_1 := \mathbf{r}_1 - \mathbf{f}_{c1}$

ASSEMBLE
$\mathbf{A} := \mathbf{A}_{11}(\mathbf{x}^{(i-1)}) + \beta_1 \mathbf{A}^*_{11}(\mathbf{x}^{(i-1)})$

SOLVE
$\mathbf{x}^{(i)} = \mathbf{A}^{-1}\mathbf{r}_1$

EXPORT
$\mathbf{x}^{(i)}$

CHECK
CONVERGENCE STATUS for $\mathbf{x}$

EXPORT
LOCAL CONVERGENCE STATUS for $\mathbf{x}$

IMPORT
LOCAL CONVERGENCE STATUS for $\mathbf{y}$

UNTIL
GLOBAL STATUS = CONVERGED

Box 1. Computational algorithm for Strategy (b).

the domains where the two equations in Eq. (2) hold overlap totally or partially, so that difficulties can only arise if different meshes or interpolations are used for both fields. Finally, note that the *two* systems of equations to be solved are smaller in size and with reduced band–width, as well as better conditioned, as compared to that yielding from Strategy (a).

On the other hand, the disadvantage of the block–iterative solution of Eq. (2), is that iterations will be needed *even if the problem is linear* (note the need to check global or overall convergence in Box 1). This is not especially inconvenient if the problem is nonlinear, as equilibrium iterations would be required anyway, or if the coupling effect is not too strong. In the following, we shall apply these ideas to two particular problems, showing that dealing with the nonlinearity and the coupling within a single iterative loop is effective. Roughly speaking, the number of iterations due to the nonlinearity and the coupling *do not* add up. Even with weak nonlinearities, they are who govern the global process, so that the coupling is achieved automatically, and unexpensively.

It needs to be said that coupled problems are usually time–dependent, and their governing equations include time derivatives. Therefore, equation system (2) (or, indeed, system (1)) arise from the discretization in space *and time* of the corresponding partial differential equations. In this case, an appropriate step–by–step procedure has to be introduced to obtain the solution of the problem in time, and the casting of our model problem, (1) or (2), corresponds to the solution of a given time step of this procedure.

If the time dimension is involved, the analysis of any proposed solution strategy must consider the time integration stability of the approach. If direct solution of the coupled problem is considered (Strategy (a)), the stability analysis is analogous to that of a standard uncoupled transient problem[1,3]. In this case, conditions for unconditional stability (for implicit–implicit schemes) or conditional stability (for implicit–explicit or explicit–explicit schemes) are generally well known for linear situations. Regarding the stability analysis of the block–iterative solution (Strategy (b)), it is certainly complicated and very problem (and scheme) dependent. It may happen that, even if an unconditionally stable algorithm has been used for every one of the fields, the overall block–iterative algorithm may still be conditionally stable[1,4]. Stabilization methods have been proposed to ensure unconditional stability of the block–iterative solution for specific problems under certain circumstances[1,5,6], but they require matrix operations that destroy the modularity (and so, the main motivation) of this approach. In practice, a conditionally stable approach is feasible and competitive if: (i) the limitation of stable time step size is compensated by a significant reduction of computational effort per time step, or (ii) the size of the time step is mostly limited by accuracy considerations rather than stability. Reason (i) is the motivation of most of the explicit schemes used in computational mechanics. The model problems described in the next sections will show that feasibility of block–iterative methods can be also justified by reason (ii).

An important point to remark regarding the stability of the block–iterative technique is that time integration stability will also depend on the tolerance demanded to achieve overall convergence. As a limit case, if no check on the overall convergence is made, the approach becomes block–explicit (also known as *"staggered" methods*), and it will be obviously conditionally stable, or, in some unfortunate cases, unconditionally unstable[4,7–8]. As an opposite limit case, if the solution of problem (2) is iterated until *full* overall convergence is achieved, then the stability characteristics of the approach are identical to those of the direct solution (problem (1)). However, there are two points to consider. Firstly, the convergence characteristics of the block–iterative procedure will depend on the time step size, since the spectral radius of the matrices involved depends on it. It may happen that the time step limitation for *convergence of the iterative solution* be more restrictive than that for *stability of a block–explicit approach*. In any case, both time step limitations must not be confused. Secondly, it must be realized that achieving full overall convergence is impossible in real computations; therefore, a new source of instability will arise from the convergence tolerance used for a specific analysis. Sensibility to this factor is again very problem dependent.

Unfortunately, it does not exist a general theory to establish convergence and time stability conditions for block–iterative schemes applied to linear coupled problems, and only partial results relative to particular problems are available. The situation is even

more speculative when nonlinear effects are present, and numerical experiments have to be performed. The next two sections present results referred to two well–known coupled problems. It will be shown that the scheme has great potential for practical engineering use. No practical restrictions on the time step size were found in the numerical examples run for fluid–structure interaction even if slack convergence tolerances were used. On the other hand, some dependence of the stability on the convergence tolerance used was evident in some thermally–coupled flows for specific situations.

A CLASS I PROBLEM: FLUID–STRUCTURE INTERACTION

Doubtless, this is one of the best known coupled problems in engineering. In this case, the coupling occurs at the interface between two different domains, occupied one by the solid and the other by the fluid. Neither the structure nor the fluid can be solved independently of each other, since the motion of the structure depends on the pressures of the fluid at the interface, and the fluid pressures depend in turn on the normal acceleration of the wet wall of the solid.

To fix ideas, a particular problem will be considered, namely, the transient analysis of a dam subject to a dynamic excitation. The mathematical model considered here is the Helmholz equation for the fluid and the conservation of momentum for the solid, both equations being coupled through the (moving) interface boundary terms. The constitutive relationship adopted for the structure is a nonlinear isotropic damage model suitable for concrete, whereas a linear elastic model is assumed for the foundation[7,9].

Without going into details (see, e.g., reference 1), the semidiscrete problem arising from the standard Galerkin spatial discretization reads:

$$\begin{aligned} \mathbf{M}_s\ddot{\mathbf{a}} + \mathbf{C}_s\dot{\mathbf{a}} + \mathbf{S}(\mathbf{a}) &= \mathbf{f}_s - \mathbf{Q}^T\mathbf{p} \\ \mathbf{M}_f\ddot{\mathbf{p}} + \mathbf{C}_f\dot{\mathbf{p}} + \mathbf{K}_f\mathbf{p} &= \mathbf{f}_f + \rho_f\mathbf{Q}\ddot{\mathbf{a}} \end{aligned} \tag{10}$$

where subscripts s and f refer to the solid and the fluid, respectively. The notation involved in (10) is as follows. The mass and damping matrices are $\mathbf{M}$ and $\mathbf{C}$, $\mathbf{K}_f$ is the matrix arising from the discretization of the Laplace operator, $\mathbf{f}$ are the vectors of force terms, $\mathbf{a} = \mathbf{a}(t)$ is the vector of nodal displacements unknowns in the solid, $\mathbf{p} = \mathbf{p}(t)$ the vector of nodal pressures in the fluid, whose density is ρ_f, $\mathbf{S}(\mathbf{a})$ is the vector of internal forces in the solid ($\mathbf{S}(\mathbf{a}) = \mathbf{K}_s\mathbf{a}$ if a linear constitutive model is adopted), $\mathbf{Q}$ is the coupling matrix and the dot denotes differentiation with respect to time t. The damping matrices account for both the natural damping (assumed to be of Rayleigh type) and radiation boundaries (see, e.g., reference 7 for details). The rectangular matrix $\mathbf{Q}$ comes from the integral over the fluid-solid interface of the product of pressure and displacement shape functions and the unit normal to this interface.

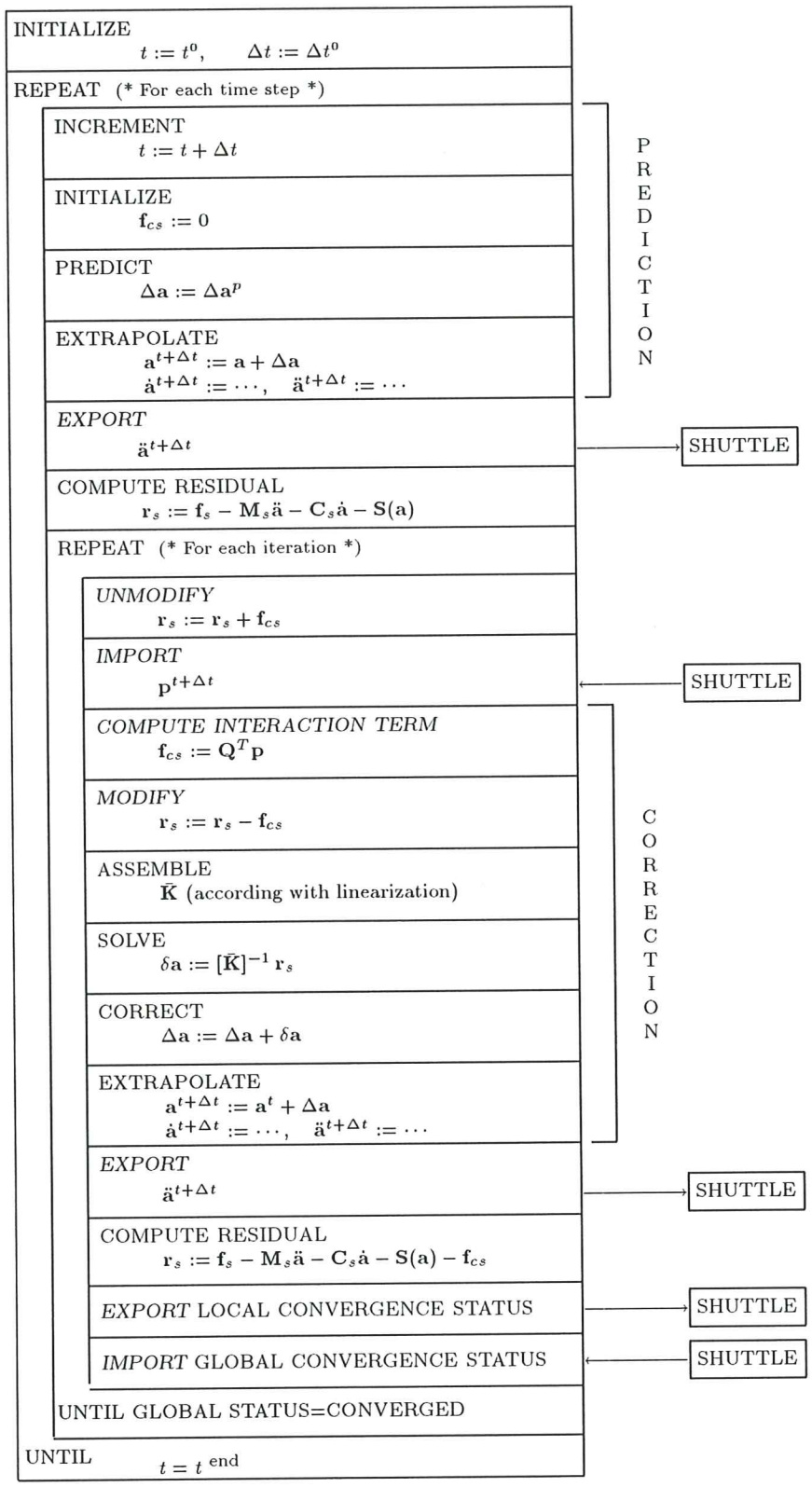
What is interesting for us is the structure of the equations (10). Observe that the coupling is *linear* and the only nonlinearity of the problem comes from the constitutive model for the solid (the dam). Therefore, after using a suitable time discretization (the Newmark scheme, for example) one will be led to a nonlinear algebraic system of the form (1), but now matrix $\mathbf{A}_{22}$ being independent of $\mathbf{x}$, provided this is identified with the

displacements and $\mathbf{y}$ with the fluid nodal pressures at a certain time step. Moreover, in this case the system can be easily symmetrized by scaling properly the equation for the fluid pressure.

We are now in a position to apply the iterative techniques described in the previous section. As the equation to be solved is second order in time an appropriate step–by–step procedure must be used. Popular options would be the Newmark method (if an implicit scheme is preferred) or central differences in time (to obtain an explicit scheme), although many alternatives exist[1,3]. Box 2 presents the computational algorithm for Strategy (b) applied to the solid phase of the fluid–structure interaction problem. Note that here an implicit predictor–multicorrector scheme[8,10,11] has been assumed for the solution of the nonlinear transient problem, without loss of generality. Obviously, the computational algorithm for the fluid phase could follow an identical or similar procedure. The squares labelled "SHUTTLE" refer to the inter–communication program used to interchange data between the solid and the fluid phases in the problem. Note also the *modify/unmodify* steps introduced for the evaluation of the residual force vector, which save unnecessary re–evaluations of the internal forces $\mathbf{S}(\mathbf{a})$.

The results to be presented here correspond to the transient analysis of the dam in Fig. 1, where also the finite element meshes for the dam, the foundation and the reservoir are shown. Note that the "solid" and "fluid" phases are discretized in separate domains (meshes). The dam selected resembles very closely Koyna Dam in India (107 m high) that has been studied by many researchers interested in seismic analysis. The water level in the reservoir is 100 m. The material properties for the dam are: $E = 31.64$ GPa (elastic modulus), $\nu = 0.2$ (Poisson ratio) and $\rho_s = 2690$ Kg/m^3, with a tensile strenght $f_t = 0.85$ MPa. The soil is considered elastic with $E = 18$ GPa and $\nu = 0.2$. The fluid properties are $\rho_f = 1019$ Kg/m^3 and $c = 1439$ m/s (speed of acoustic waves). The left and right boundaries of the foundation are forced to have equal displacements ('repeatibility condition'), while the bottom boundary is modelled as a radiating one with a prescribed incoming seismic wave[7,9]. The left boundary of the reservoir is also modelled as a radiating boundary, and water pressures are prescribed to zero in the free surface. No damping was considered apart from that provided by the radiating boundaries, and the nonlinear material behaviour of the dam. The bottom boundary of the soil is excited by a prescribed incoming velocity wave in the horizontal direction. This is a senoidal wave of varying amplitude, also senoidal ($\dot{U}(x,t) = 0.05\sin(2\pi t/6.144)\sin(2\pi t/0.384)$ m/s for $t \leq 3.072$ and $\dot{U}(x,t) = 0$ for $t > 3.072$). Due to the damage model we have employed, the displacements in the dam do not depend linearly on the amplitude of the input wave.

Given the dimensions of the smallest elements in the mesh and the material properties, we can use Iron's theorem to estimate the critical time step size for an explicit scheme as $\Delta t_{cr} \simeq 10^{-4}$ s in the solid phase, and $\Delta t_{cr} \simeq 10^{-3}$ s in the fluid phase (bigger elements and lower speed for the acoustic waves). On the other hand, the first fundamental period of the dam–soil system is $T_1 = 0.4355$ s. Therefore, a reasonable time step size for an implicit scheme would be $\Delta t \simeq 10^{-2}$ s, around $T_1/40$, if accurate results are to be obtained. We have performed implicit–implicit (Newmark method) analyses using block–iterative algorithms with time step sizes ranging from $\Delta t = 10^{-2}$ s to $\Delta t = 10^{-1}$ s. No problems have arisen in convergence nor in stability, although accuracy is greatly reduced for $\Delta t >$



Box 2. Computational algorithm for Strategy (b) applied to the solid phase of the fluid–structure interaction problem.

2×10^{-2} s, and the nonlinear nature of the response is completely overlooked for $\Delta t > 4 \times 10^{-2}$ s. This shows that stability is dramatically improved by the iterative schemes, with regard to purely block–explicit (staggered) schemes. It is possible to use time step sizes of the order that would be reasonable to use with unconditionally stable schemes, mostly limited by accuracy.

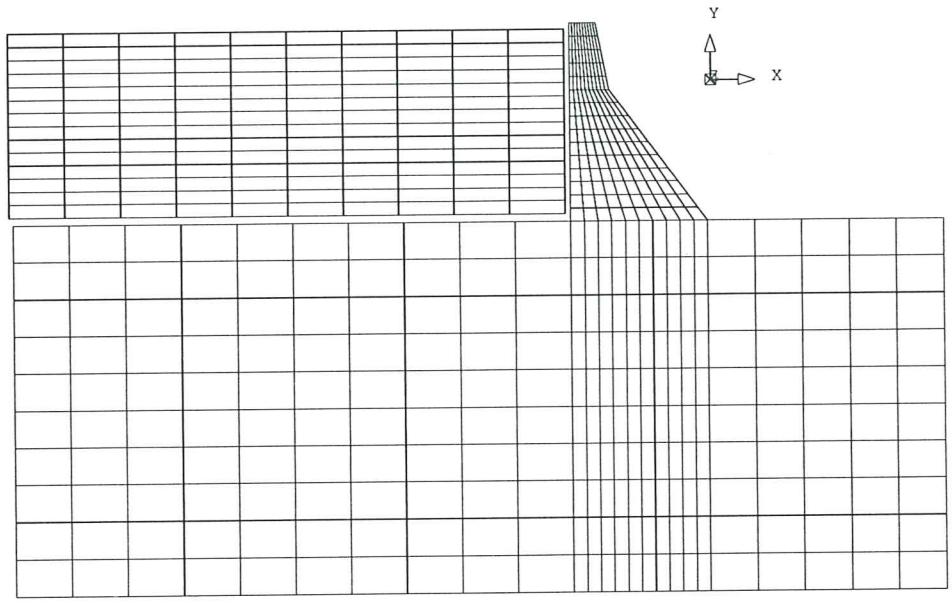


Figure 1. Computational model showing the dam with the foundation and the reservoir (the "solid" and "fluid" phases are in separate meshes)

The results depicted below correspond to the analysis with $\Delta t = 10^{-2}$ s. The convergence tolerance in displacements has been set to 0.01 %, although it is worthwhile to mention that almost the same accuracy is obtained with a tolerance of 0.1 %, which reduces the number of iterations almost by 50 %, making this approach much more cost–effective. The horizontal displacement computed at the left top of the dam is plotted in Fig. 2(a), and the evolution of the global damage index (mean square value) in Fig. 2(b).

Let us discuss now the convergence of the numerical method using the block-Jacobi and the block-Gauss-Seidel schemes. The nonlinearity of the problem will be included in the same iterative loop. However, from the evolution of the global damage index it is observed that a nonlinear behaviour will be encountered only in some time intervals when this parameter increases. Therefore, this example will serve to compare the performance of the iterative processes both for linear and nonlinear situations.

The number of iterations required to converge vs time is plotted in Fig. 3(a). In this case, the Picard scheme has been used to linearize the equations. The peaks correspond to times when the damage index increases, that is, to nonlinear situations. It is observed that both the Jacobi and the Gauss-Seidel methods need the same number of iterations, and hence it may be concluded that *convergence is driven by the nonlinearity*. When the

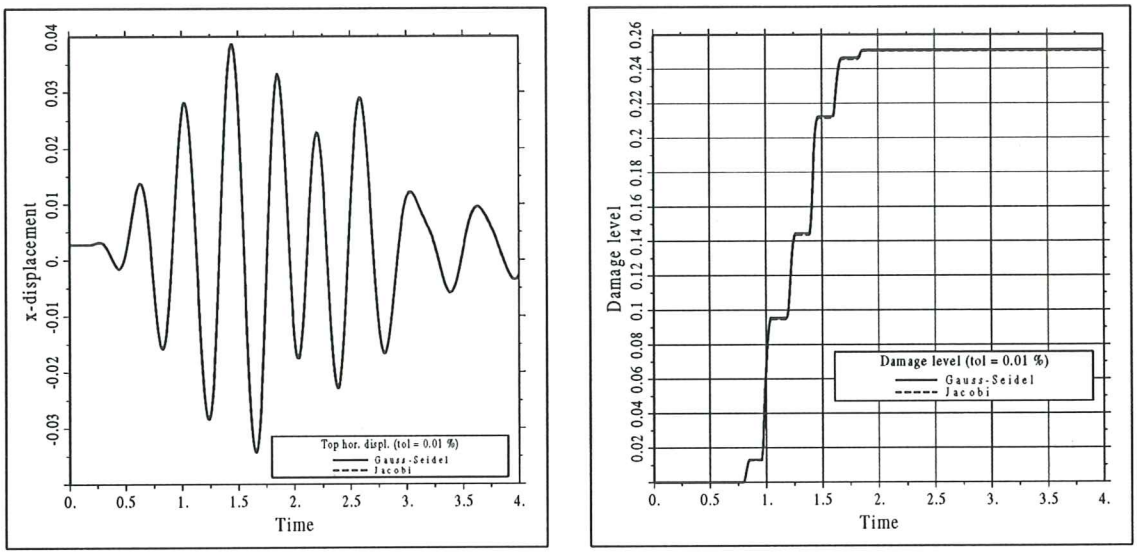


Figure 2. (a) Horizontal displacement of the left top of the dam, (b) Time evolution of the global damage index

dam behaves linearly, the Jacobi scheme needs roughly double number of iterations than the Gauss-Seidel scheme, as expected.

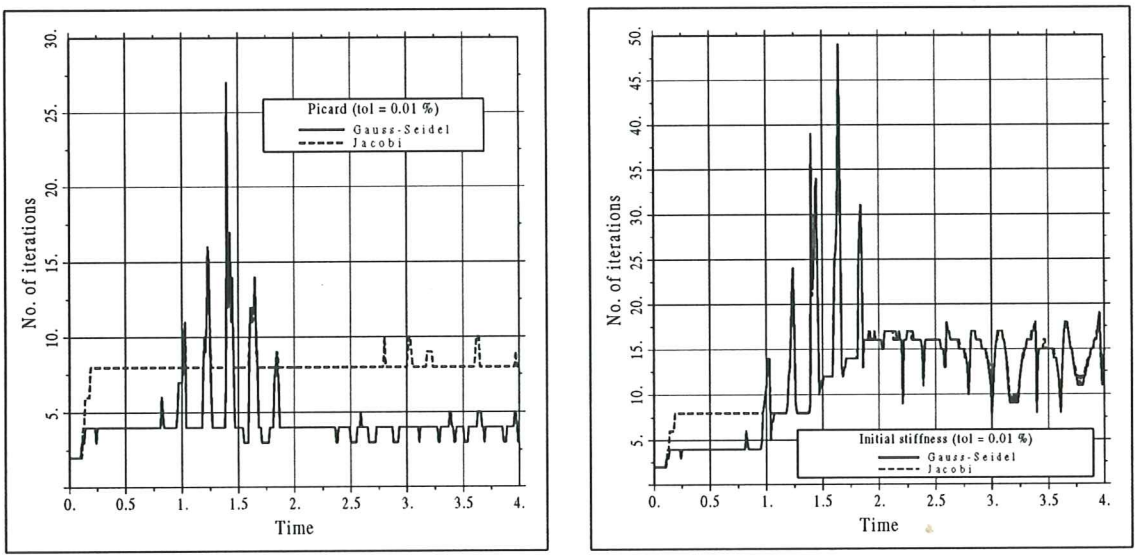


Figure 3. Number of iterations vs time using (a) the Picard scheme, (b) the initial stiffness method

The Picard method necessitates the formation of the stiffness matrix for the solid at each iteration of each time step. One can also try to use always the initial stiffness as an iteration matrix. We have also considered this possibility, *also including the new iterations within*

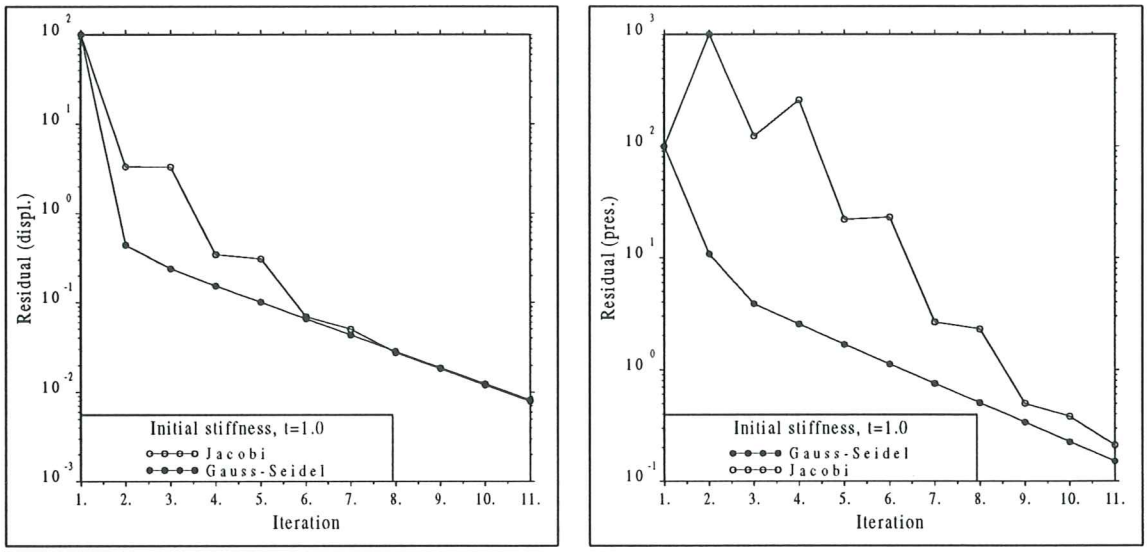


Figure 4. Convergence history for $t = 1$. (a) Displacements; (b) Pressures

the coupling/nonlinearity loop. The number of iterations vs time in this case is shown in Fig. 3(b). It is clearly observed that roughly up to $t = 1$ the coupling drives the process. Then the nonlinearity dominates and, if we compare with Fig. 3(a), from t approximately 1.5 onwards the reason to iterate is the use of the initial stiffness. The fact that both the Jacobi and the Gauss-Seidel methods yield the same numerical response indicates that *the coupling will be achieved automatically*. The convergence history for $t = 1$ is shown in Fig. 4, from where it is observed that the nonlinearity governs the iterative process after the sixth iteration in displacements and the ninth in pressures.

A CLASS II PROBLEM: THERMALLY DRIVEN FLOWS

Several coupled problems are encountered in computational fluid dynamics. Here, we shall consider the coupling of the incompressible Navier-Stokes equations with the energy balance equation using the classical Boussinesq model. In this case, body forces in the Navier-Stokes equations are proportional to the temperature, which depends on the velocity through the convective term of the heat equation.

After discretizing the continuous equations in space, one is led to a system of first order ordinary differential equations of the form

$$\begin{aligned} \mathbf{M}_u\dot{\mathbf{u}} + \mathbf{K}_u(\mathbf{u})\mathbf{u} - \mathbf{G}\mathbf{p} + \mathbf{C}\mathbf{T} &= \mathbf{0} \\ \mathbf{G}^T\mathbf{u} &= \mathbf{0} \\ \mathbf{M}_T\dot{\mathbf{T}} + \mathbf{K}_T(\mathbf{u})\mathbf{T} &= \mathbf{0} \end{aligned} \tag{11}$$

where subscripts u and T refer to the velocity and the temperature, respectively. In Eqs. (11), $\mathbf{u}$ is the vector of velocity nodal unknowns, $\mathbf{T}$ the vector of temperature nodal

unknowns, $\mathbf{p}$ the vector of pressure nodal unknowns, $\mathbf{M}$ are the mass matrices, matrices $\mathbf{K}$ account for both the convective and viscous (or diffusive) terms, $-\mathbf{G}$ is the discrete gradient matrix and $\mathbf{C}$ is the coupling matrix. Observe that no source terms have been included in Eqs. (11). Motion will be induced by the boundary conditions to be prescribed to the velocity and the temperature.

Only a brief outline of the numerical model shall be given here. For further details the reader is referred to 12. To arrive at Eqs. (11), we employ a mixed interpolation for the velocity and the pressure satisfying the so called inf-sup or Babuška-Brezzi stability condition. In particular, for the numerical example of this section we have used the Q_2/P_1 element, constructed using a continuous biquadratic interpolation for the velocity and a discontinuous piecewise linear pressure (see, e.g., reference 13).

It is well known that the standard Galerkin formulation yields oscillatory results when applied to problems where convection is dominant. In order to overcome this problem, we have used the streamline-upwind/Petrov-Galerkin (SUPG) method[14], based on a modification of the Galerkin test function for the velocity, say $\mathbf{v}$, to $\mathbf{v} + \tau(\mathbf{u} \cdot \nabla)\mathbf{v}$, where the perturbation only affects the element interiors. The parameter τ is the so called *intrinsic time*, which depends on the element size and the local Reynolds number, and ∇ is the gradient operator. A similar procedure is applied for the heat equation, now computing τ based on the cell Péclet number. Observe that the presence of the velocity in the perturbation of the Galerkin test function will introduce another nonlinearity in the problem.

To discretize Eqs. (11) in time one can use the generalized trapezoidal rule, also called θ-method. Once this is done, the nonlinear algebraic system to be solved at each time step will have the form

$$\begin{bmatrix} \mathbf{A}(\mathbf{u}) & -\mathbf{G} & \mathbf{C} \\ \mathbf{G}^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}(\mathbf{u}) \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \\ \mathbf{T} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_u \\ \mathbf{0} \\ \mathbf{f}_T \end{bmatrix} \tag{12}$$

This system of equations has the form (1) if we identify $\mathbf{x}$ with $(\mathbf{u}, \mathbf{p})$ and $\mathbf{y}$ with $\mathbf{T}$. Observe that in this case $\mathbf{A}_{12} = \mathbf{0}$ and $\mathbf{A}_{22}$ depends on $\mathbf{x}$.

The presence of a zero matrix in the diagonal is an important drawback of Eqs. (12). It makes the matrix in this system not definite and, if a direct solver is used, pivoting is needed. The penalty method allows to circumvent this problem. Moreover, if the pressure is interpolated using discontinuous polynomials, one can eliminate the element pressure unknowns in terms of the velocity nodal unknowns, thus reducing the size of the system. The only inconvenience of the penalty method is the ill-conditioning found when the penalty parameter is very small. This can be alleviated by using an iterative penalty method as described in reference 12. Eqs. (12) are replaced by

$$\begin{bmatrix} \mathbf{A}(\mathbf{u}^{(i)}) & -\mathbf{G} & \mathbf{C} \\ \mathbf{G}^T & \epsilon \mathbf{M}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}(\mathbf{u}^{(i)}) \end{bmatrix} \begin{bmatrix} \mathbf{u}^{(i)} \\ \mathbf{p}^{(i)} \\ \mathbf{T}^{(i)} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_u \\ \epsilon \mathbf{M}_p \mathbf{p}^{(i-1)} \\ \mathbf{f}_T \end{bmatrix} \tag{13}$$

where ϵ is a small number (penalty parameter) and $\mathbf{M}_p$ is any symmetric and positive-definite matrix. We take it as the Gramm matrix arising from the pressure interpolation.

It is proved in reference 12 that the iterative penalization converges towards the incompressible solution for the uncoupled Navier-Stokes equations provided that ϵ is sufficiently small.

The use of the iterative penalty method to satisfy the incompressibility constraint introduces another iterative procedure in the problem, that shall be included also in the coupling/nonlinearity loop. It is possible to consider several combinations as in the previous sections. For brevity, we shall only present the final algorithm obtained using the block-Gauss-Seidel method. Taking into account that it is possible to eliminate the pressure at the element level if discontinuous pressures are used, the final algorithm is as follows:

$$\left[\mathbf{A}(\mathbf{u}^{(i-1)}) + \beta_1 \mathbf{A}^*(\mathbf{u}^{(i-1)}) + \frac{1}{\epsilon}\mathbf{G}\mathbf{M}_p^{-1}\mathbf{G}^T\right]\mathbf{u}^{(i)} = \mathbf{f}_u + \\ + \mathbf{G}\mathbf{p}^{(i-1)} - \mathbf{C}\mathbf{T}^{(i-1)} + \beta_1 \mathbf{A}^*(\mathbf{u}^{(i-1)})\mathbf{u}^{(i-1)} \tag{14}$$

$$\mathbf{p}^{(i)} = \mathbf{p}^{(i-1)} - \frac{1}{\epsilon}\mathbf{M}_p^{-1}\mathbf{G}^T\mathbf{u}^{(i)} \tag{15}$$

$$\mathbf{B}(\mathbf{u}^{(i)})\mathbf{T}^{(i)} = \mathbf{f}_T \tag{16}$$

where $\beta_1 = 0$ or 1 determines if the nonlinear convective term of the Navier-Stokes is linearized up to first or second order. As explained earlier, the use of $\mathbf{u}^{(i)}$ in the heat equation implies a high order linearization of the convective term in this equation.

Let us make several remarks concerning this iterative scheme. The linearization of the SUPG term, the iterative penalization and the block-Jacobi or block-Gauss-Seidel methods can only yield a linear convergence rate, with a more or less steep slope in a plot iterations vs logarithm of the residual. Sooner or later, convergence will be driven by the slowest of these rates as the iterative procedure goes on, even though $\beta_1 = 1$ be selected to linearize the Navier-Stokes equations. We have found from numerical experiments that the Newton-Raphson method is only useful when the coupling with the heat equation is weak. Otherwise, it only contributes to increase the computational cost, without reducing the number of iterations needed to reach a prescribed convergence tolerance.

If instead of using $\mathbf{T}^{(i-1)}$ in Eq. (14) and $\mathbf{u}^{(i)}$ in Eq. (16) they are replaced by the temperature and velocity nodal values of the previous time step, one is led to the so called 'staggered algorithms', in which the coupling between the Navier-Stokes and the energy equations is accomplished by means of the time stepping. The algorithm in time in this case is block explicit, regardless of the value of the parameter θ of the generalized trapezoidal rule used to advance in time. Therefore, a critical time step exists above which the algorithm becomes unstable.

Referring again to the stability in time, if a fully converged solution is obtained for the algorithm (14)–(16) then stability should be ensured provided that $\theta \geq 1/2$. Obviously, the block iterative method will not give exactly the same solution as the full nonlinear system. An error will remain that may affect the stability of the algorithm in time. Numerical experiments indicate that this in fact happens. We have found that $\theta = 1/2$ (Crank-Nicolson) is very sensitive to the convergence tolerance adopted for each time step. The higher it is, the sooner instabilities begin to appear, leading to the numerical blow-up after a few time steps. In this sense, the backward Euler scheme ($\theta = 1$) has been found to be much more robust. We have never found instability problems using this method.

Let us apply this numerical method to the finite element simulation of the problem sketched in Fig. 5. It consists of a 2D laminar flow suddently heated from below (in Fig. 5, ϑ denotes the temperature). The dimensionless parameters of the problem have been taken as $Re = 10$ (Reynolds number), $Fr = 1/150$ (Froude number) and $Pe = 40/9$ (Péclet number). The average inlet velocity, the height of the channel and the temperature difference between the top and bottom walls have been chosen as reference values for velocity, length and temperature, respectively. These values result in a thermoconvective instability of the basic Poiseuille flow. The stable solution turns out to be periodic in time. The domain $[0, 10] \times [0, 1]$ has been discretized using a uniform mesh of $30 \times 15 = 450$ Q_2/P_1 elements.

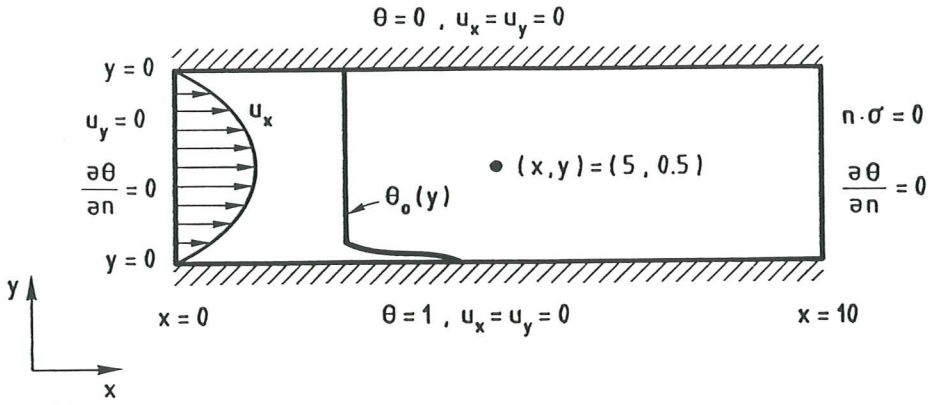


Figure 5. Geometry, initial and boundary conditions

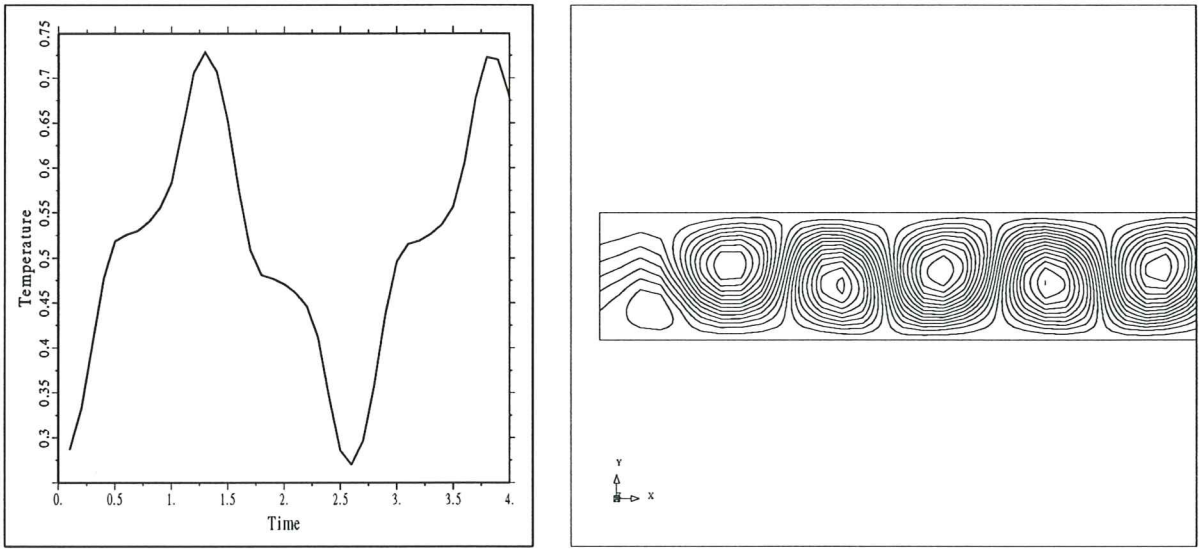


Figure 6. (*a*) Temperature vs time at the central point; (*b*) Streamlines for $t = 1.3$

The evolution of the temperature at the central point is plotted in Fig. 6(a). Fig. 6(b) shows the streamline pattern for one half of the domain at $t = 1.3$. As it has already been mentioned, we have found that the Crank–Nicolson method is very sensitive to the convergence tolerance. The time step size has been taken as $\Delta t = 10^{-3}$. For a tolerance of 1 %, instability problems have been found at time step number 37, whereas for a tolerance of 0.1 % they do not appear until time step 121. Using the backward Euler method ($\theta = 1$) the time stepping algorithm has been found to be stable in all the cases. The results presented here have been obtained using this method.

In Figs. 7(a) and 7(b) we have plotted the convergence history in velocities and the evolution of the norm of the incompressibility constraint for the first and second time steps (in Fig. 7(b), $\mathbf{B} \equiv \mathbf{G}^T$). The convergence history for the temperature shows a similar behaviour (not shown). Thus, with a single iterative loop the nonlinearity, the coupling and the approximation to the incompressibility constraint are achieved at once.

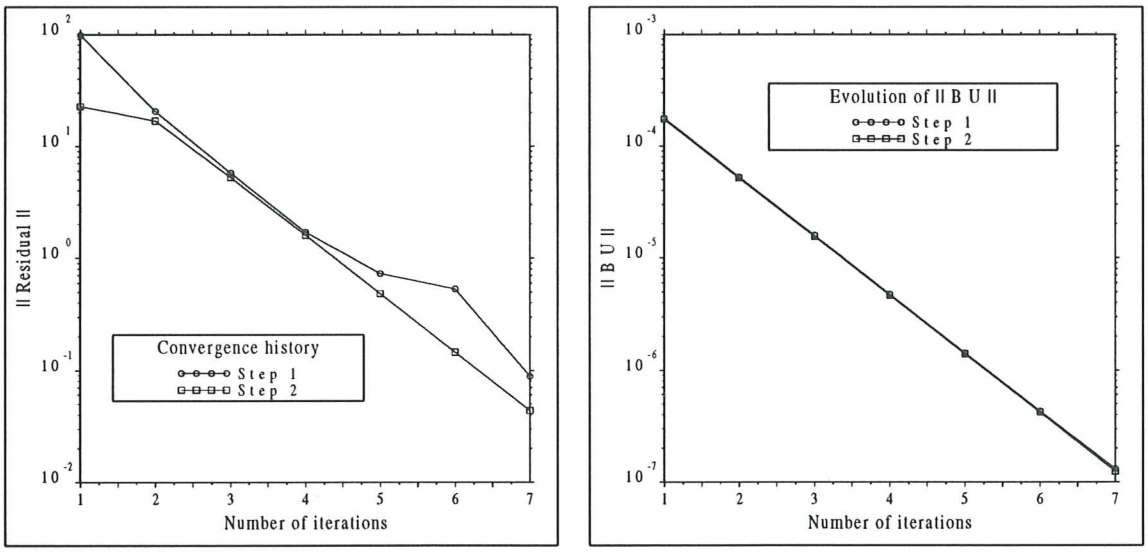


Figure 7. (a) Convergence history; (b) Evolution of the incompressibility

MODULAR IMPLEMENTATION OF BLOCK-ITERATIVE TECHNIQUES

From the point of view of software development for coupled problems, maybe the most important advantage of the block–iterative strategy is the possibility of using different codes for the analysis of each domain. On the other hand, from the point of view of software execution, the most important advantage is the possibility of parallel processing of the different codes, running even in different connected machines, which helps to optimize the use of the existing hardware in the solution of coupled problems.

A first alternative for the implementation of the block–iterative techniques is to develop a single program which calls the different domain solvers (now transformed in modules of a more general program) sequentially. This may look as the most practical approach, but it has several drawbacks. Such a program would be quite long and problems could arise,

for instance, if the same names have been used for variables of subroutines of the different uncoupled problems. However, the most important disadvantage is that this alternative precludes all possibility of parallel processing of the different domains.

A second alternative is to execute the processes in parallel and handle the interchanging of data using files that can be accessed by the different codes. This alternative poses difficulties in process synchronization, as it is necessary for a given process to wait for other processes to finish writing the necessary interface data. This may overload the computer system with a number of processes permanently accessing the disk to see if the desired data is already there.

A third alternative, presented and recommended here, is to allow the interconnexion of several *slave codes* via a *master code*, which is responsible for the communication and synchonization of the different processes, controlling the overall convergence of the coupled system for each time step. The implementation of this is based on the facilities for inter–process communication that the *UNIX* operative system has avalaible to the user. Under this system, any process can create a *socket* that immediately is made accessible to other processes. When a process creates a socket, the operating system stops this process until another one connects itself to the same socket. At this moment, both processes are re–started and the system establishes a communication pipe between them so that any information written by a process into the pipe can be accessed by the other. Details on the use of this *UNIX* facility are not given here, but they can be found in any standard user manual.

For the process interconexion methodology to be established it is necessary:

- To develop a single *master code* (here called *SHUTTLE*) to which all the *slave codes* are connected. The functions to be performed by this code are explained in the next section.
- To develop a library of routines to allow the *slave codes* to perform the necessary import/export operations of nodal variables. No additional effort is needed here apart from including the desired CALL statements to this routines in the single field solvers.
- To develop the necessary routines (which are problem dependent) so that the *slave codes* can evaluate the corresponding RHS interaction terms from the imported nodal values.

Master code SHUTTLE

The interconexion program described in this section is designed to assist in the solution of *nonlinear transient coupled problems* using the block–ietrative techniques presented previously. A step–by–step procedure is used to advance in time, cast in a predictor–multicorrector form, together with a convenient linearization of the nonlinear problem for each time step.

In our work, *SHUTTLE* is the *master code* that controls the overall block–iterative algorithm described earlier. To achieve this, the program must perform the following tasks, regarding the execution of the *slave codes* who are responsible for the solution of each of the single field equations in system (2):

- To communicate the different *slave codes/processes*, transferring information from one program/domain to another, if (and only if) required.
- To control the overall coupled algorithm, checking convergence, forcing the *slave codes/ processes* to iterate if (and only if) required; and
- To synchronize the execution of the different *slave codes/processes*, ensuring that each process runs as efficiently as possible, and that all of then iterate and/or advance in time simultaneously.

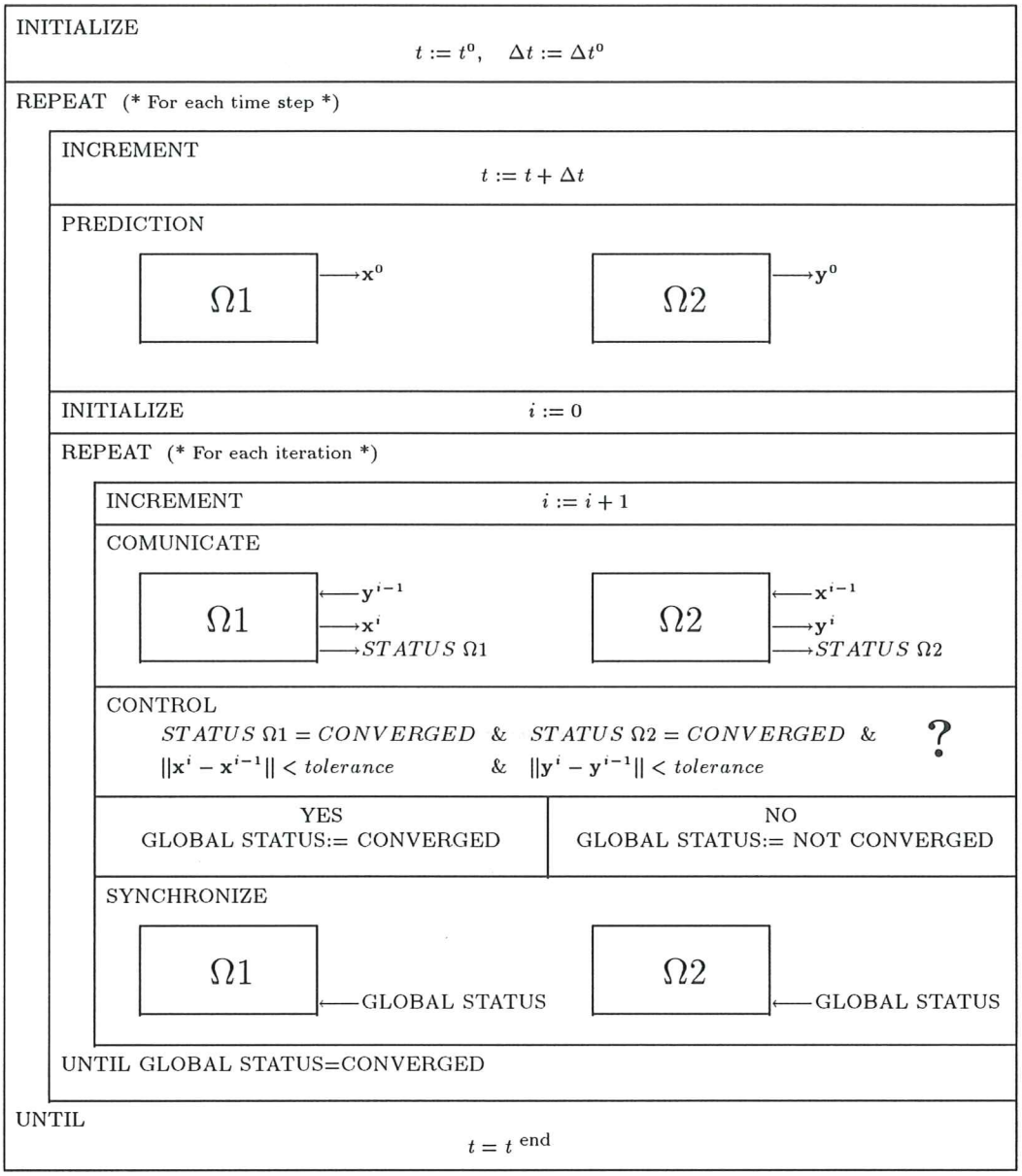
The Task Diagram for the master code is shown in Box 3. Here we have labelled as $\Omega 1$ and $\Omega 2$ each of the *slave codes* used to solve the equations corresponding to the fields/domains involved in the problem. Referring, for instance, to system (2), $\Omega 1$ would solve the first equation for variable $\mathbf{x}$, and $\Omega 2$ would solve the second equation for variable $\mathbf{y}$. Note that Box 3 follows, as closely as possible, the Task Diagram of each slave code (see Box 2). Note also that the task labelled "PREDICTION" consists of importing the individual predictions from the slave codes. The task labelled "CORRECTION" consists of: (a) exporting to all the processes the variables from the previous iteration, and, (b) importing from all of them the iterative corrections and the local convergence status. The task labelled "CONTROL" consists of checking that all the tolerances specified for each of the processes are satisfied, both at local (individual) level as globally. Finally, the task labelled "SYNCHRONIZE" consists on exporting to each process the overall convergence status for the current iteration.

The code is programmed following a hierachy of objectives. Each objective is fulfilled by achievement of sub–objectives. The main objective is obviously to integrate the governing equation from the initial time of the analysis to the final time; this is done in a step–by–step fashion. For each time step, the objective is to achieve global and local convergence; this is attained by forcing all the codes to perform the necessary iterative corrections. During each iteration, the objective is to furnish the slave codes with the required data to evaluate the coupling terms; this is done by transferring the necessary information through the *sockets*. Finally, for each communication the objective is to preserve the pre–established turn and the natural dependencies of the coupled problem. So, the key–word for the whole procedure is *synchronized communication*. This framework of objectives can be sketched as follows,

- To arrive at the final time of the analysis, advancing in a step–by–step fashion.
  - For each step: to achieve local and global convergence, iterating if necessary.
    - For each iteration: to furnish the required data to each one of the slave codes through the sockets.
      - For each communication: to ensure the pre–established turn and the natural dependencies of the coupled problem.

Slave codes. Import/export operations

As an example, the computing algorithm for the solid phase of a fluid–structure interaction problem has been shown in Box 2. It is indicated there which variables must be *exported* and *imported*, and the exact place to do so. In order to show the modifications that must be performed in a standard program, Figure 8(a) presents a standard generic algorithm



Box 3. Task Diagram for Program *SHUTTLE*.

for an uncoupled problem, while Figure 8(*b*) presents the algorithm for the same process as part of a coupled problem solved by the block–iterative technique (using *SHUTTLE*). Comparing both of them, the necessary modifications can be seen as reduced to the addition of the *import/export* operations that are indicated in figure (*b*). Also, it must be remarked that the box labelled "VERIFICATION" consists of *exporting* to *SHUTTLE* the *local* convergence status of the process and *importing* from *SHUTTLE* the *global* convergence status of the coupled problem as a whole (see also Box 2).

Efficiency of the proposed scheme

The system constructed along the lines described in the previous sections achieves a fringe, but not minor, benefit: to exploit the computer resources optimally, forcing the execution of

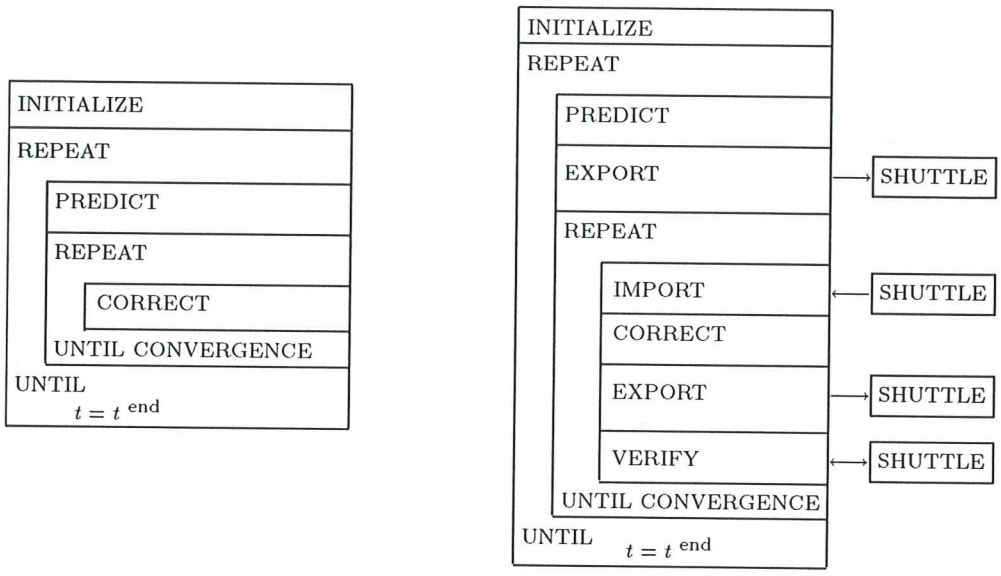


Figure 8. Task Diagrams for:
(a) A process in an uncoupled problem; (b) A process in a coupled problem.

the slave codes to proceed *in parallel*, or as much in parallel as possible. Let us direct some attention to this matter. As discussed previously, for a two field problem two alternatives are possible for the iterative multi–correction procedure: (a) block–Gauss–Seidel, in which one process, say Ω_1, iterates in first turn, while the other, Ω_2, follows; and (b) block–Jacobi, in which both processes, Ω_1 and Ω_2, iterate simultaneously. Figure 9 depicts graphically both schemes.

Alternative (a), block–Gauss–Seidel, does not seem to allow for parallel processing. However, and referring to Box 2, each process exports its variables as soon as possible, that is, before getting entangled with the cumbersome residual forces evaluation. Therefore, just after Ω_1 has exported its variables to *SHUTTLE*, they can be imported by Ω_2, who can start correcting long before Ω_1 has finished doing so. For nonlinear problems, where residual forces evaluation can take quite a percentage of the total CPU time, this is as close as one can get to parallel processing (see Figure 10). Recall as well that it has been shown in the previous sections than dealing with the coupling and the nonlinearity within the same iterative loop is highly effective.

We can quantify this by referring to the fluid-structure interaction example described previously. For the particular case using the initial stiffness as iteration matrix (see Fig. 3(b)) the percentage of CPU time invested in the solution of the fluid phase is only 11.20 % of that needed for the solid, whereas the communication process needs the 0.41 % of that time. Considering that the evaluation of residual forces takes 90.96 % of the total time spent by the solid, it is clear that the fluid can be solved *fully in parallel*. Thus, improvement of the overall efficiency can only be achieved by improving the solid solver (for instance, by using the Picard iterative method, see Fig. 3(a)). For a 3D fluid–structure

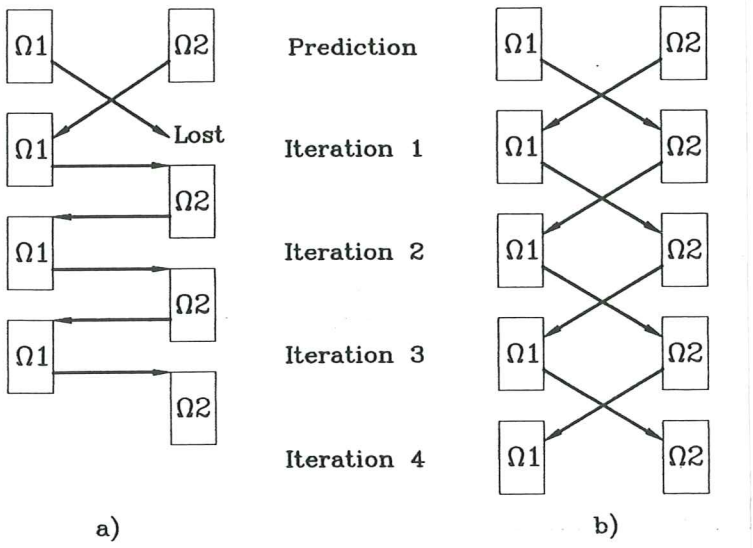


Figure 9. Alternatives scheme for iterative multi–correction:
(a) block–Gauss–Seidel method: Ω_1 goes first and Ω_2 follows.
(b) block–Jacobi method: Ω_1 and Ω_2 run simultaneously.

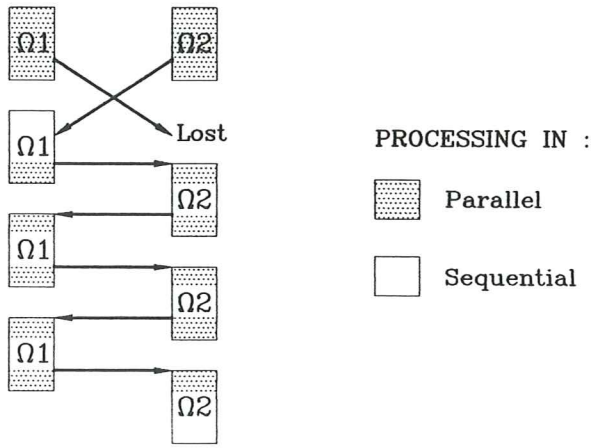


Figure 10. Parallel and sequential processing for block–Gauss–Seidel scheme.

interaction problem, the relative cost of the fluid phase would be even smaller.

On the other hand, alternative (b), block–Jacobi, seems the obvious choice for parallel processing, but this may be misleading. Consider, for instance, the case when *both problems are linear*, and iterating is only required due to the block–iterative scheme adopted. The situation is depicted in Fig. 11. Note that the S–shaped sequence marked as A is exactly the same for both alternatives. However, alternative (b) spends almosts double CPU time. This expresses graphically the well–known fact that, for linear problems, the Gauss–Seidel method exhibits double asymptotic convergence rate than the Jacobi method

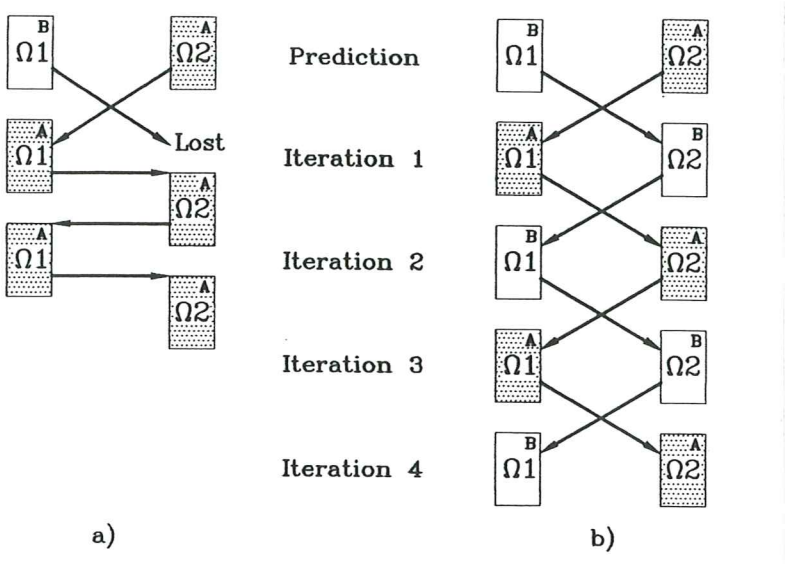


Figure 11. Block–Gauss–Seidel (a) and block–Jacobi (b) schemes applied to a linear–linear coupled problem.

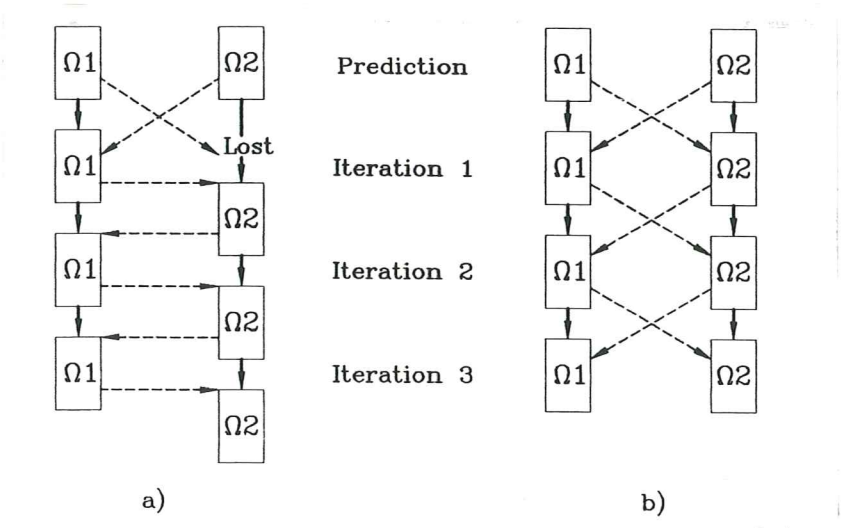


Figure 12. Block–Gauss–Seidel (a) and block–Jacobi (b) schemes applied to a strongly nonlinear coupled problem.

(cf. Fig. 3(a)). Nevertheless, alternative (b) can be satisfactory when the problems to be solved are strongly nonlinear on their own, and the interaction terms play a minor role in the iterative process. In this case, both processes would be running in parallel and fully exploiting the computational resources, and the global check on convergence will be automatically satisfied as soon as both processes converge locally. This situation is depicted in Figure 12.

CONCLUSIONS

An algorithmic approach to *nonlinear* coupled problems based on the partition of the discrete equations of the system in its single field components has been discussed. The alternative can be seen as a block–iterative solution of the original system. If this technique is used together with a procedure for synchronized communication of the processes involved, two benefits are obtained: modularity of the necessary software and optimal use of the computer resources. Theoretical analysis of the resulting algorithm regarding convergence (and stability for transient problems) is very problem dependent. Moreover, the values adopted for the tolerance of the overall convergence within a time step do affect stability in time if a step–by–step scheme is used. However, the procedure has been successfully applied to two very different in nature coupled problems such as fluid structure interaction and thermally coupled incompressible flows. It has been shown that even if unconditional stability cannot be ensured in general, stability is dramatically improved by the iterative schemes, with regard to purely block–explicit (staggered) schemes. Thus, the approach is considered to be not only feasible, but highly competitive with respect to other alternatives. Additionally, and considering that our main interest is dealing with nonlinear problems, numerical experiments suggest that in most of the cases the rate of convergence will be driven by the linearization of the nonlinear terms, so that the coupling of the equations is achieved with very little (or not at all) additional cost.

REFERENCES

1 Zienkiewicz, O.Z. and Taylor, R.L. *The finite element method.* McGraw Hill (1991).

2 Crisfield, M.A. *Non–linear Finite Element Analysis of Solids and Structures. Volume 1: Essentials.* John Wiley (1991).

3 Hughes, T.J.R. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis.* Prentice–Hall International (1987).

4 Felippa, C.A. and Park, K.C. Staggered transient analysis procedures for coupled mechanical systems: Formulation. *Com. Meth. Appl. Mech. and Engng.*, **24**, 61–111, (1980).

5 Zienkiewicz, O.C., Paul, D.K. and Chan, A.H.C. Unconditionally stable staggered solution procedures for soil–pore fluid interaction analysis. *Int. J. Num. Meth. Eng.*, **19**, 1669–1673, (1986).

6 Chan, A.H.C. *A unified Finite Element Solution to Static and Dynamic problems of Geomechanics.* Doctoral Thesis, University of Wales (1988).

7 Galindo, M. *Una metodología para el análisis numérico del comportamiento resistente no lineal de presas de hormigón con cargas estáticas y dinámicas* . Doctoral Thesis, Technical University of Catalonia (1993).

8 Wisniewski, K., Turska, E., Simoni, L. and Schrefler, B.A. Error analysis of staggered predictor–corrector scheme for consolidation porous media, in *The Finite Element Method in the 1990's.* Springer–Verlag/CIMNE (1991).

9 Galindo, M., Cervera, M. and Oliver, J. Efficient solution schemes for fluid–structure–soil interaction problems, in *Proceedings of 10th World Congress on Earthquake Engineering* (1991).

10 Paul, D.K. *Efficient Dynamic Solutions for Single and Coupled Multiple Field Problems.* Doctoral Thesis, University of Wales (1982).

11 Hughes, T.J.R. and Liu, W.K. Implicit–explicit finite elements in transient analysis. *J. Appl. Mech. Engng.*, **45**, 371–374 and 375–378 (1978).

12 Codina, R. *A finite element model for incompressible flow problems.* Doctoral Thesis, Technical University of Catalonia (1992).

13 Cuvelier, C., Segal, A. and van Steenhoven, A. *Finite element methods and Navier–Stokes equations.* Reidel (1986).

14 Books, A.N. and Hughes, T.J.R. Streamline upwind/Petrov–Galerkin formulations for convected dominated flows with particular emphasis on the incompressible Navier–Stokes equations. *Com. Meth. Appl. Mech. and Engng.*, **32**, 199–259 (1982).