Using hybrid discretizations in parallelized structural dynamic analyses

Luís A. M. Mendes[a,b,], Luís M. S. S. Castro[a,c]

[a]*Instituto de Engenharia de Estruturas, Território e Construção, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal*
[b]*Laboratório Nacional de Engenharia Civil, Av. do Brasil 101, 1700-066 Lisboa, Portugal*
[c]*Instituto Superior Técnico, Universidade Técnica de Lisboa, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal*

Abstract

This paper presents a new approach to perform incremental dynamic analyses on reinforced concrete buildings using the so-called *hybrid discretizations*, taking advantage of parallelized computations and domain decomposition techniques to enhance the capability and performance for the analysis of large-scale problems.

The concept of hybrid discretization consists in the combination of different modelling approaches for the three-dimensional structural elements. Where most of the non-linear phenomena are expected to occur, refined meshes and more complex constitutive relations are adopted. Elsewhere, simplified structural models are considered.

Special attention is devoted to the definition of adequate techniques to treat the transition zones between different structural models. The efficiency and accuracy of alternative kinematic constraint techniques are studied and assessed.

The paper ends with two validation examples that test the accuracy and the computational performance of the proposed methodology.

Keywords: Hybrid Discretization, Parallelization, Primal Substructuring,

Email address: `luismendes77@gmail.com` (Luís A. M. Mendes)

1. Introduction

The main objective of this paper is to propose a combined strategy to enhance the computational performance of dynamic analyses on *reinforced concrete* (*RC*) buildings.

This strategy integrates the use of parallelization, adapted solution procedures and the combination of different discretization techniques, which will be addressed in this paper as *hybrid discretizations* (*HD*).

To cope with this objective, the paper is organized as follows: at first, the formulation and some implementation issues related to the parallelization and to the domain decomposition method adopted are addressed. Afterwards, the concept of *hybrid discretizations* is discussed in detail, underlying their main advantages and drawbacks. The following section is devoted to the use of *kinematic constraints* (*KC*) to enforce the transition between different mesh types, as required for the *HD*. Two validation examples close the paper. The first is dedicated to test the accuracy of the proposed methodology in *incremental dynamic analyses* (*IDA*), and the second, focuses on studying the use of different techniques to enforce the *KC* and assessing their impact in the overall performance of the simulation.

2. Parallelization

The basic concept of substructuring applied to the finite element method is to solve the original problem in a two-level format, using: i) a *reduced* or *coarse* problem, commonly defined at the subdomain (*s*) boundaries, which acts as an interconnecting and load balancing mechanism, and ii) an *internal* problem defined at the subdomain-level with all the condensed and boundary unknowns associated with each subdomain.

This procedure requires the implementation of the *Domain Decomposition* (*DD*) method to perform the analysis. In this work only the *Primal Substructuring* (*PS*) method [1, 2, 3, 4, 5] will be considered due to its simplicity and applicability to the problems addressed. The formulation can be established by considering the stiffness-based definition of a static linear mechanical system subjected to external loads ($\mathbf{Q}_e$), in which the structural domain is divided into n_s non-overlapping subdomains Ω^s with interconnecting boundaries $\Gamma^{s,s}$ (see Figure 1).

The unknowns of this problem are the nodal displacements that can be reordered by grouping first the interior *degrees-of-freedom* (*DOF*), gathered by subdomain (s), and afterwards the *DOF* at the boundary. Adopting this procedure, the following governing system would be obtained:

$$\begin{bmatrix} \mathbf{K}_{ii}^1 & \mathbf{0} & \mathbf{0} & \mathbf{K}_{iB}^1 \\ \mathbf{0} & \ddots & \mathbf{0} & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{K}_{ii}^{n_s} & \mathbf{K}_{iB}^{n_s} \\ \mathbf{K}_{Bi}^1 & \cdots & \mathbf{K}_{Bi}^{n_s} & \mathbf{K}_{BB} \end{bmatrix} \begin{bmatrix} \mathbf{q}_i^1 \\ \vdots \\ \mathbf{q}_i^{n_s} \\ \mathbf{q}_B \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_{e,i}^1 \\ \vdots \\ \mathbf{Q}_{e,i}^{n_s} \\ \mathbf{Q}_{e,B} \end{bmatrix}, \tag{1}$$

where the stiffness sub-matrices associated with the boundary *DOF* are defined by:

$$\mathbf{K}_{Bi}^s = \mathcal{A}_{bB}^{s,t}\,\mathbf{K}_{bi}^s, \tag{2}$$

$$\mathbf{K}_{iB}^s = \mathbf{K}_{ib}^s\,\mathcal{A}_{bB}^s, \tag{3}$$

$$\mathbf{K}_{BB} = \sum_{s=1}^{n_s} \mathcal{A}_{bB}^{s,t}\;\mathbf{K}_{bb}^s\;\mathcal{A}_{bB}^s, \tag{4}$$

where superscript s represents the subdomain and the subscripts (i, b and B) represent the (internal, subdomain boundary and global boundary) *DOF*.

The matrices $\mathcal{A}$ are primal Boolean operators used to define mappings between global and local operators. The symbol was chosen to be consistent with the terminology used for the analyses without substructuring, because these matrices can be seen as assembler operators.

The system of equations presented in equation (1) clearly shows the independence of the blocks $\mathbf{K}^s_{ii}$, $\mathbf{K}^s_{iB}$ and $\mathbf{K}^s_{Bi}$, as these sub-matrices have contributions only from elements belonging to one subdomain. Additionally, the sub-matrix $\mathbf{K}_{BB}$ can be computed from the sum of the independent subdomain contributions $\mathbf{K}^s_{bb}$ using equation (4).

Already at this stage, the computation and the assemblage of the governing system could be more efficient by taking advantage of parallelization. Nevertheless, the parallelization will be taken one-step further, using a multi-step approach to solve the linear system. To achieve this goal, a static condensation method will be adopted to eliminate the internal *DOF* of each subdomain from the coarse problem.

Let's start by simplifying the notation, using the symbols $\mathcal{S}$ and $\mathcal{Q}$ for the condensed stiffness matrix and for the condensed force vector, where $\mathcal{S}$ is also known as the *Schur Complement*, which can be computed by assembling the contribution of each subdomain, using:

$$\mathcal{S} = \sum_{s=1}^{n_s} \mathcal{A}^{s,t}_{bB}\ \mathcal{S}^s\ \mathcal{A}^s_{bB}, \tag{5}$$

$$\mathcal{Q} = \sum_{s=1}^{n_s} \mathcal{A}^{s,t}_{bB}\ \mathcal{Q}^s, \tag{6}$$

where:

$$\mathcal{S}^s = \mathbf{K}^s_{bb} - \mathbf{K}^s_{bi}\left(\mathbf{K}^s_{ii}\right)^{-1}\mathbf{K}^s_{ib}, \tag{7}$$

$$\mathcal{Q}^s = \mathbf{Q}^s_{e,b} - \mathbf{K}^s_{bi}\left(\mathbf{K}^s_{ii}\right)^{-1}\mathbf{Q}^s_{e,i}. \tag{8}$$

The reduced governing system (9) can be solved for the global boundary displacements and the internal *DOF* displacements can be recovered using (10):

$$\mathcal{S}\,\mathbf{q}_B = \mathcal{Q}, \tag{9}$$

$$\mathbf{q}_i^s = \left(\mathbf{K}_{ii}^s\right)^{-1}\left(\mathbf{Q}_i^s - \mathbf{K}_{ib}^s\,\mathbf{q}_b^s\right), \tag{10}$$

where:

$$\mathbf{q}_b^s = \mathcal{A}_{bB}^s\,\mathbf{q}_B. \tag{11}$$

It should be emphasized that no approximation is introduced by solving the system of equations (9) instead of (1), and consequently, the same results should be recovered, apart from numerically-related accuracy losses. Furthermore, it should be noted that the size of the reduced system (9) is generally much smaller than the one without using substructuring (1). On the other hand, one aspect that should be taken into consideration is that the condensed operators (7) are potentially full matrices and this may represent a significant overhead in the computations.

To enhance the computational performance, the structural analysis algorithm should be able to take advantage of multi-processing units. Commonly, part of the computations are centralized on one *root* or *master processing unit* (*MPU*) and the rest are performed in parallel, at the *subdomain processing units* (*SPU*).

This method starts by computing the subdomain contributions to the global Schur Complements. This computation is executed in parallel at the *SPU*, using expression (7).

To adapt the formulation to non-linear analyses, the terms associated with the *external force vector* in equation (8) should be replaced by the *unbalanced force vector*, defined by:

$$\mathbf{g} = \mathbf{Q}_e - \mathbf{Q}_i, \tag{12}$$

where $\mathbf{Q}_i$ represents the *internal force vector* defined ahead in equation (38), resulting in the following subdomain contribution to the coarse system right-hand side (6):

$$Q^s = -\mathbf{g}_b^s + \mathbf{K}_{bi}^s \left(\mathbf{K}_{ii}^s\right)^{-1} \mathbf{g}_i^s. \tag{13}$$

Afterwards the reduced problem is assembled at the *MPU*, using equations (5) and (6), and the coarse governing system (9) is solved for the global boundary *DOF* increments using a direct solver.

The last steps consist of sending the relevant boundary *DOF* displacement increments to the *SPU* using equation (11) and computing the internal unknowns at the *SPU*, through:

$$\delta\mathbf{q}_i^s = \mathbf{K}_{ii}^{s,-1} \left(-\mathbf{g}_i^s - \mathbf{K}_{ib}^s \, \delta\mathbf{q}_b^s\right), \tag{14}$$

and at the end, it is necessary to update the displacement field, using:

$$\mathbf{q}^s = \mathbf{q}^s + \begin{bmatrix} \delta\mathbf{q}_i^s \\ \delta\mathbf{q}_b^s \end{bmatrix}. \tag{15}$$

3. Hybrid Discretization

One of the most significant drawbacks of using three-dimensional meshes to perform dynamic analyses is related to the high number of unknowns associated with this type of discretization. To mitigate this problem and improve the efficiency of the simulations, the use of the so-called *hybrid discretization* (*HD*) is studied.

The term hybrid was chosen to indicate that the resulting mesh combines two different natures of discretization. Accordingly, a refined mesh is used when the most relevant stress concentrations are expected to occur, and consequently, to develop a significant non-linear response. In contrast, the zones anticipated to remain elastic or with minor non-linear effects are modelled with a simplified discretization. The typical choice for the simplified mesh of common frame *RC* buildings are Euler-Bernoulli or Timoshenko beams. Solid elements (hexahedra) are used for the refined part of the mesh and these were combined with truss or beam elements for simulating the steel reinforcements.

In general, the *HD* proposed in this work is expected to introduce the following advantages:

1. Reduce the number of *DOF* and therefore make the analysis more efficient and feasible, without a significant impact on the accuracy of the numeric simulation;
2. Promote the use of different modelling strategies within a single simulation;
3. Create a convenient and efficient partition between the subdomains that can be used in the substructured analysis (see Figure 2-a).

This methodology requires making an assumption regarding where most of the non-linear phenomena will occur. This is not considered a significant problem due to two reasons. First, the zones with stress concentrations can be reasonably well predicted for *RC* structures, by knowing the characteristics and the intensity of the loads acting on the structure. Secondly, the mesh can always be readjusted and the analysis rerun if the structural behaviour is different from what was predicted.

Figure 2-a shows what would be a feasible *HD* for the main structural elements of a common *RC* structure subjected to predominant horizontal loading. In this case, the refined mesh would be concentrated at the joints and at the extremities of the beams and columns. The interior spans of these members are supposed to develop, at most, only minor non-linear effects, which is a reasonable hypothesis when an earthquake loading is considered.

High permanent loads or earthquakes with intense vertical accelerations could change these assumptions. Normally, the first case would only result in stabilized cracking and stress redistribution that would have a minor impact on the global response of the structure. Moreover, considering that the permanent loading effects are present before the earthquake loading, their effect can be simulated by creating a segment of the beams with reduced stiffness. Regarding the second case, it is always possible to use non-linear beam elements or extend the refined mesh to the interior of the beam span.

In order to generate the mesh, it is necessary to estimate the size of the refined mesh segments. The basic parameter to be defined is the length of the refined mesh segments near the edges of the elements (L_h in Figure 2-a). This parameter is strongly related to the well-known *plastic hinge length* that is discussed in several works, *e.g.* [6, 7, 8, 9]. A first estimation for this parameter can be made assuming that it depends only on the cross-section dimensions:

$$L_h = \lambda_h \max(w_{x2}, w_{x3}), \text{ with } \lambda_h = [2.0, 3.0], \tag{16}$$

where w_{x2} and w_{x3} represent the transversal widths in the local direction x_2 and x_3, respectively.

Linear kinematic constraints were used to connect the beam and solid elements, although other possibilities were considered, like *transition elements* that are formulated to handle different types of *DOF* resulting from non-compatible elements (see for example [10, 11, 12, 13]).

The transition elements present some disadvantages when compared to the kinematic constraints, mainly because they introduce additional complexities to the model, related to the numerical behaviour of these elements. The discretization would also present additional difficulties and the usage of transition elements increases the number of elements in the mesh. On the other hand, kinematic constraints present a predictable structural behaviour and represent a minor increase in the complexity of the model. Furthermore, this approach does not increase the number of nodes or elements. Nevertheless, there will be an increase in unknowns if the constraints are enforced using Lagrange Multipliers. At the end, the choice tended clearly towards using kinematic constraints, which will be discussed in more detail in the following section.

The *HD* increases the possibilities regarding the use of different modelling strategies within a single simulation. To illustrate this, Figure 3 presents a schematic representation of the mesh used to model a four storey building subjected to an earthquake loading. In this case, the lower storeys were simulated using refined meshes and more complex constitutive models, because this is where most of the non-linear phenomena are expected to occur. On the other hand, simplified meshes and models were adopted for the upper storeys, where the structure is expected to have a nearly elastic response.

Fibre models implemented in beam-column elements could be used for the simplified mesh when some non-linear response may still develop at the upper storeys. Otherwise, using linear constitutive relations would be the most adequate solution. Global response models with concentrated plasticity could also be considered. However, this formulation presents some disadvantages for three-dimensional models, in particular for columns subjected to biaxial bending, due to the increased complexity associated with the definition of the global response rules.

4. Kinematic Constraints

Kinematic constraints (*KC*) can be used to prescribe a specific behaviour on the nodal displacements. Within the scope of this work, the *KC* are linear because the analyses are also geometrically linear. In addition, the mathematical formulation for the *KC* presented is written under the assumption of small displacements. A typical use for the kinematic constraints is to impose a specific structural behaviour such as connecting different parts of the structure or enforcing certain types of rigid-body behaviour.

A set of kinematic constraints can be expressed in the matrix format:

$$C_K \, \mathbf{q} = \mathbf{d}_K, \tag{17}$$

where C_K is the *kinematic constraint matrix*, also known as *connectivity matrix*, which contains only constant values and $\mathbf{d}_K$ is a constant vector. The matrix C_K has a number of rows equal to the number of kinematic constraint equations (n_k) and a number of columns equal to the total number of *DOF*.

4.1. Beam2Solid constraint

Within the scope of the *HD* proposed in this work, *KC* are used to impose the connection between beam and solid elements, which present 6 and 3 *DOF* per node for the general three-dimensional case, respectively (see Figure 2-b). These specific constraints are called *Beam2Solid* in this paper, which consist of enforcing part of the equation of the well-known diaphragm and plate constraints. Only the equations associated with the translational *DOF* of the solid elements are enforced (see Figure 4), hence reducing the number of constrained equations to half.

Considering the case of the constraint axis located along the global direction x_1, the equations adopted by the *Beam2Solid* constraint are:

$$q_1^S = q_1^M + x_3\,\theta_2^M - x_2\,\theta_3^M, \tag{18}$$

$$q_2^S = q_2^M - x_3\,\theta_1^M, \tag{19}$$

$$q_3^S = q_3^M + x_2\,\theta_1^M, \tag{20}$$

where $q_i^S \left(q_i^M\right)$ represents the displacement of the slave (master) node along the global coordinate x_i and $\theta_i^S \left(\theta_i^M\right)$ represents the rotation about the axis x_i of the slave (master) node. For other constraint axes, the equations can be defined using a direct cycle permutation of the direction indices.

The *Beam2Solid* constraint will be implemented by the *Master-Slave Elimination* (*MSE*) method [14] or using *Lagrange Multipliers* (*LM*) [15, 14], which will be discussed in the following paragraphs.

4.2. Master-slave Elimination

The *Master-Slave Elimination* (*MSE*) method, also known as *Transformation Equations* method, consists in the elimination of a group of *DOF* (slave *DOF*) using the equations that relate them to a set of special *DOF* (master *DOF*). This can be seen as a static condensation. However, in this case a set of custom kinematic relations are used instead of the relations already included in the governing system.

If the *DOF* are ordered leaving the slave *DOF* at the end, it is possible to rewrite equation (17) as:

$$\begin{bmatrix} C_{K,m} & C_{K,s} \end{bmatrix} \begin{bmatrix} \mathbf{q}_m \\ \mathbf{q}_s \end{bmatrix} = \mathbf{d}_K, \tag{21}$$

where the subscript m represents the master (retained) *DOF* and the subscript s represents the slave (condensed) *DOF*.

After some simple algebraic manipulations on equation (21), the slave *DOF* displacements can be defined in terms of the master *DOF* displacements, using:

$$\mathbf{q}_s = C_{K,sm}\,\mathbf{q}_m + \mathbf{d}_{K,s}, \tag{22}$$

with:

$$C_{K,sm} = -C_{K,s}^{-1}\,C_{K,m}, \tag{23}$$

$$\mathbf{d}_{K,s} = C_{K,s}^{-1}\,\mathbf{d}_K. \tag{24}$$

It is possible to rewrite equation (22) as:

$$\begin{bmatrix} \mathbf{q}_m \\ \mathbf{q}_s \end{bmatrix} = \mathbf{T}_K\,\mathbf{q}_m + \begin{bmatrix} \mathbf{0} \\ \mathbf{d}_{K,s} \end{bmatrix}, \tag{25}$$

where $\mathbf{T}_K$ can be seen as a transformation matrix defined by:

$$\mathbf{T}_K = \begin{bmatrix} \mathbf{I} \\ C_{K,sm} \end{bmatrix}, \tag{26}$$

where **I** represents the identity matrix with size equal to the number of master *DOF*.

The governing system of a mechanical linear system subjected to static forces, where the master *DOF* are ordered first and the slave *DOF* at last, can be written as:

$$\begin{bmatrix} \mathbf{K}_{mm} & \mathbf{K}_{ms} \\ \mathbf{K}_{sm} & \mathbf{K}_{ss} \end{bmatrix} \begin{bmatrix} \mathbf{q}_m \\ \mathbf{q}_s \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_{e,m} \\ \mathbf{Q}_{e,s} \end{bmatrix}, \tag{27}$$

after introducing equation (25) and pre-multiplying by $\mathbf{T}_K^t$, it is possible to write:

$$\mathbf{T}_K^t \begin{bmatrix} \mathbf{K}_{mm} & \mathbf{K}_{ms} \\ \mathbf{K}_{sm} & \mathbf{K}_{ss} \end{bmatrix} \mathbf{T}_K\,\mathbf{q}_m + \mathbf{T}_K^t \begin{bmatrix} \mathbf{K}_{mm} & \mathbf{K}_{ms} \\ \mathbf{K}_{sm} & \mathbf{K}_{ss} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{d}_{K,s} \end{bmatrix} = \mathbf{T}_K^t \begin{bmatrix} \mathbf{Q}_{e,m} \\ \mathbf{Q}_{e,s} \end{bmatrix}, \tag{28}$$

then the governing system of equations can be rewritten as:

$$\mathbf{K}^C \mathbf{q}_m = \mathbf{Q}^C - \mathbf{d}^C, \tag{29}$$

where:

$$\mathbf{K}^C = \mathbf{T}_K^t \, \mathbf{K} \, \mathbf{T}_K = \mathbf{K}_{mm} + C_{K,sm}^t \, \mathbf{K}_{sm} + \mathbf{K}_{ms} \, C_{K,sm} + C_{K,sm}^t \, \mathbf{K}_{ss} \, C_{K,sm}, \tag{30}$$

$$\mathbf{Q}^C = \mathbf{T}_K^t \, \mathbf{Q_e} = \mathbf{Q}_{e,m} + C_{K,sm}^t \, \mathbf{Q}_{e,s}, \tag{31}$$

$$\mathbf{d}^C = \mathbf{T}_K^t \; \mathbf{K} \begin{bmatrix} \mathbf{0} \\ \mathbf{d}_{K,s} \end{bmatrix} = \left(\mathbf{K}_{ms} + C_{K,sm}^t \, \mathbf{K}_{ss}\right) \mathbf{d}_{K,s}. \tag{32}$$

After solving (29) for the master displacements, the slave *DOF* displacements can be computed from equation (22). This method presents the advantage of eliminating the constrained *DOF* from the governing system, leading to a smaller number of equations to be solved. Nevertheless, this benefit can be largely outweighed by the necessary manipulations and by the need of solving the additional system of equations to recover $\mathbf{q}_s$. The major disadvantage of this method consists in the required rearrangement of the governing system.

4.3. Lagrange Multipliers

This method introduces new variables called *Lagrange Multipliers* (*LM*) to the system of equations. The *LM* can be grouped into the following vector:

$$\boldsymbol{\lambda}^t = \begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_{nc} \end{bmatrix}, \tag{33}$$

where n_c is the number of constraint equations.

This vector can be used to impose the constraint equations (17) using:

$$\boldsymbol{\lambda}^t \left(C_K \mathbf{q} - \mathbf{d}_K \right) = 0. \tag{34}$$

This equation can be seen as the energy necessary to enforce the constraints and can be added to the total potential energy functional of the mechanical system:

$$\Pi(\mathbf{q}, \lambda) = \int \underline{\sigma} : \underline{\varepsilon}(\mathbf{q})\, d\Omega - \mathbf{Q}_e{}^t\, \mathbf{q} + \lambda^t \left(C_K\, \mathbf{q} - \mathbf{d}_K\right), \tag{35}$$

where $\underline{\sigma}$ and $\underline{\varepsilon}$ represent the stress and strain tensors.

Introducing the relation ($\boldsymbol{\varepsilon} = \mathbf{B}\,\mathbf{q}$) and imposing the stationary condition, it is possible to obtain the following groups of equations:

$$\frac{\partial \Pi}{\partial \mathbf{q}} = \mathbf{Q}_i - \mathbf{Q}_e + C_K^t\, \lambda = \mathbf{0}, \tag{36}$$

$$\frac{\partial \Pi}{\partial \lambda} = C_K \mathbf{q} - \mathbf{d}_K = \mathbf{0}, \tag{37}$$

where the *internal force vector* is given by:

$$\mathbf{Q}_i = \int \mathbf{B}^t\, \sigma\, d\Omega. \tag{38}$$

Grouping these equations into an augmented governing system results in:

$$\begin{bmatrix} \mathbf{K} & C_K^t \\ C_K & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ \mathbf{d}_K \end{bmatrix}, \tag{39}$$

where it can be seen that C_K^t acts like an equilibrium operator, and therefore, the nodal forces ($\mathbf{Q}_c$) that impose the constraints, can be computed from:

$$\mathbf{Q}_c = C_K^t \lambda. \tag{40}$$

As highlighted by Liu *et al.* [15], there are two main disadvantages associated with the use of this method: the number of equations increases and the expanded stiffness matrix becomes indefinite due to the zeros in the diagonal terms. The latter reduces the efficiency in solving the system of equations and some solvers rely on positive definiteness. On the other hand, the main advantage of this method is related to the fact that it is not necessary to rearrange or to perform any additional operation to the system of equations. This last feature can compensate in many cases for the fact of having a larger number of equations to solve [16].

4.4. Implementation

Within the framework of this work, *KC* were used for three different purposes:

1. for prescribing nodal displacements;
2. for implementing the *HD* technique;
3. for enforcing a specific structural behaviour.

The *LM* method was used for prescribing nodal displacements *e.g.* base displacements to simulate earthquakes, and for enforcing constraints that are defined within the scope of more than one subdomain, *e.g.* rigid floor diaphragms. This technique was implemented in the reduced problem and resulted in an augmented governing system with a new set of unknowns, the Lagrange Multipliers. Within the scope of this paper, this technique will be referred to as *Global Kinematic Constraints* (*GKC*).

Alternatively, the *MSE* method was used when no interaction with the other subdomains is necessary. This corresponds to the general case of enforcing the *Beam2Solid* constraint. This methodology reduces the number of unknowns in the governing system by associating the *DOF* to be condensed with the ones retained in the model using mathematical relations. To avoid being confused with the *DOF* split used in the *PS* method, from this point on the condensed (retained) *DOF* will be referred to as slave (master) and this enforcing technique will be called *Local Kinematic Constraints* (*LKC*). Figure 5 presents an example of a possible distribution of *GKC* and *LKC*.

4.4.1. *Local Kinematic Constraints*

The *LKC* are imposed at the subdomain level and the following *DOF* ordering is adopted:

$$\mathbf{q}^s = \begin{bmatrix} \mathbf{q}_m^s \\ \mathbf{q}_s^s \\ \mathbf{q}_b^s \end{bmatrix}, \tag{41}$$

where the subscripts *m* and *s* refer to master (retained) and slave (condensed) internal subdomain *DOF* and *b* refers to the subdomain boundary *DOF*. As a consequence, equation (25) should be rewritten as:

$$\begin{bmatrix} \mathbf{q}_m^s \\ \mathbf{q}_s^s \\ \mathbf{q}_b^s \end{bmatrix} = \mathbf{T}_L^s \begin{bmatrix} \mathbf{q}_m^s \\ \mathbf{q}_b^s \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{d}_{L,s}^s \\ \mathbf{0} \end{bmatrix}, \tag{42}$$

where the new kinematic transformation matrix (26) is defined by:

$$\mathbf{T}_L^s = \begin{bmatrix} \mathbf{I}_{mm}^s & \mathbf{0} \\ C_{L,sm}^s & C_{L,sb}^s \\ \mathbf{0} & \mathbf{I}_{bb}^s \end{bmatrix}, \tag{43}$$

in which $C_{L,sm}^s$ and $C_{L,sb}^s$ are the extension to the subdomain scope and to the proposed *DOF* reordering of the matrix $C_{K,sm}$ defined in expression (23). The submatrices $\mathbf{I}_{mm}^s$ and $\mathbf{I}_{bb}^s$ are identity matrices of size equal to the number of master and boundary subdomain *DOF*, respectively. The remaining entries are zero matrices defined with consistent sizes.

The usage that will be given to the local constraints implies that $\mathbf{d}_{L,s}^s = \mathbf{0}$, so that the last term in equation (42) becomes irrelevant and will be omitted from now on. It should be emphasized that the subdomain matrices $\mathbf{T}_L^s$ are highly sparse and very prone to be stored in matrix sparse format.

Under these assumptions, the subdomain structural operators can be defined as:

$$\mathbf{K}^{C,s} = \mathbf{T}_L^{s,t}\,\mathbf{K}^s\,\mathbf{T}_L^s, \tag{44}$$

$$\mathbf{C}^{C,s} = \mathbf{T}_L^{s,t}\,\mathbf{C}^s\,\mathbf{T}_L^s, \tag{45}$$

$$\mathbf{M}^{C,s} = \mathbf{T}_L^{s,t}\,\mathbf{M}^s\,\mathbf{T}_L^s, \tag{46}$$

and the dynamic effective stiffness associated with the α-Method time integration scheme [17, 18] is given by:

$$\hat{\mathbf{K}}^{C,s} = (1+\alpha)\left(\mathbf{K}^{C,s} + \frac{\gamma}{\beta\,\Delta t}\mathbf{C}^{C,s}\right) + \frac{1}{\beta\,\Delta t^2}\mathbf{M}^{C,s}. \tag{47}$$

After computing the subdomain boundary *DOF* displacements in substructured analyses, the rest of the unknowns can be computed using equation (42).

4.4.2. Global Kinematic Constraints

Using *GKC* the incremental coarse problem used for substructured structural analysis (9) becomes augmented and can be written as follows:

$$\begin{bmatrix} \hat{\mathcal{S}} & C_G^t \\ C_G & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\mathbf{q}_B \\ \lambda \end{bmatrix} = \begin{bmatrix} -\hat{\mathcal{Q}} \\ \delta\mathbf{d}_G \end{bmatrix}. \tag{48}$$

The matrix C_G is the global constraint matrix of size $n_g \times n_B$, where n_g represents the number of global kinematic constraints and n_B represents the number of boundary *DOF* in the reduced problem. In addition, λ is a vector of size n_g that holds the Lagrange Multipliers.

In this case, the right-hand side vector $\mathbf{d}_G$ cannot be eliminated as before, because this vector will be used to enforce absolute values on the unknowns (*e.g.* prescribed displacements).

The operators $\hat{S}$ and $\hat{Q}$ are respectively the dynamic versions of the Schur Complement and of the unbalanced forces vector defined for the coarse problem.

Using local kinematic constraints, the subdomain contributions to these operators are defined as follows:

$$\hat{S}^{C,s} = \hat{\mathbf{K}}_{bb}^{C,s} - \hat{\mathbf{K}}_{bi}^{C,s}\left(\hat{\mathbf{K}}_{ii}^{C,s}\right)^{-1}\hat{\mathbf{K}}_{ib}^{C,s}, \tag{49}$$

$$\hat{Q}^{C,s} = -\hat{\mathbf{g}}_{b}^{C,s} + \hat{\mathbf{K}}_{bi}^{C,s}\left(\hat{\mathbf{K}}_{ii}^{C,s}\right)^{-1}\hat{\mathbf{g}}_{i}^{C,s}, \tag{50}$$

where the superscript C indicates that these operators are defined after condensing the internal slave *DOF*, as in equation (44).

Expressions similar to (5) and (6) can be used for assembling the contribution of all subdomains, because these are defined for the global boundary *DOF* and do not change by eliminating the slave condensed *DOF*.

5. Example 1: Elastic cantilever column

The purpose of this example is to test the algorithm on *IDA* using substructuring and the HD technique proposed in this paper. To cope with this objective, the elastic cantilever column, presented in Figure 6, is chosen. The material response is considered elastic for simplicity and four meshes are adopted. The meshes include *8-noded hexahedrons* (*H8*) and *Euler-Bernoulli Beams* (*EBB*). Table 1 summarizes the main characteristics of the meshes.

GKC were adopted to enforce the *Beam2Solid* constraints and the *PS* method was used as the *DD* technique.

The column was subjected to a prescribed displacement at the base (see Figure 6). The time history presented in Figure 7-a was considered as a displacement record (in centimetres) for the prescribed displacement at the bottom of the column. In this case, the accelerations associated with the imposed displacement may be computed by double differentiation of the displacement record, resulting in an acceleration record with $PGA = 1\ m/s^2$. The elastic response spectrum is displayed in Figure 7-b, revealing that this record was generated to match the elastic response spectrum of Eurocode 8 [19] for ground type A and 5% of damping.

The mass was concentrated on the top surface of the column and a lumped mass matrix was considered. The Newmark's [20] *average acceleration method* ($\gamma = 0.50; \nu = 0.25$) was used as the time integration algorithm and proportional damping was considered by imposing 5% of damping at 1 Hz and 10 Hz, leading to $\alpha = 5.7119 \times 10^{-1}$ and $\beta = 1.4468 \times 10^{-3}$ [14].

The results are compared with the solution obtained with the finite element software ADINA [21] adopting the Mesh #1 discretization, which is considered as the reference for the tests.

The *IDA* resulted in the displacements at node A (X direction) presented in Figure 8 for the analyses #1 and #2, and in Figure 9 for the analyses #3 and #4, together with the results obtained by the reference analysis. These results show that the analyses #1 and #2 recovered the reference solution, as a result of using the same mesh, element type and finite element formulation. On the other hand, for the analyses #3 and #4 the response shows small differences in the amplitude and a minor time offset. This results from having different structural responses that change the amplification and the main vibration frequency.

In conclusion, the differences in the response of the meshes with and without *hybrid discretization* are small, expected and related to the different modeling techniques adopted in the simulations. This demonstrates that the proposed technique is a valid and feasible way to reduce the problem size and still retrieve good results.

6. Example 2: Elastic frame

The purpose of this example is to test the accuracy and efficiency of the different methods used to enforce the kinematic constraints required for the *HD* technique. Both *GKC* and *LKC* are tested in this example.

The problem chosen is the single-bay frame 5.0 m wide and 3.0 m high represented in Figure 10. All materials are considered to be linear elastic and two types of meshes are used to simulate the structure. Mesh #2 was created by implementing the partition associated with expression (16) setting $\lambda_h = 2.5$. The overall layout of the structure is symmetric although this property is not used to simplify the analyses. The considered loading consists of a distributed vertical load in Mesh #1 and a concentrated load in Mesh #2.

Once again, the results are compared with those obtained using the software ADINA [21] adopting the same discretization adopted in Mesh #1, which is considered as the reference in this example.

Three numbers of subdomains are considered in the analyses for both meshes (see Table 2). When only one subdomain is considered, the analysis was executed with the sequential and the parallel version of the code, in order to assess the parallelization overhead (see Table 3). Moreover, when two subdomains are adopted the subdomain boundary was positioned in the symmetry axis, hence resulting in uniform element and *DOF* divisions, apart from Mesh #1 where the elements passing through the symmetry axis were included in subdomain #1. Furthermore, when four subdomains are adopted the division was made by separating each column and respective foundation, and the interior part of the beam into two nearly identical halves (partitions P1, P2, P3 and P4 represented in Figure 10). The same geometric division was used for Mesh #2 (partitions P5, P6, P7 and P8). This resulted in a non-uniform *DOF* distribution, with both column-beam subdomains having around 2/3 of the total number of interior *DOF* and the remaining 1/3 divided in equal shares by both column bases.

The characteristics of the analyses and of the kinematic constraints implemented are also presented in Table 3. This data shows that when more than one subdomain is considered, the *PS* method is adopted and that the analyses #5 to #8 were made using 756 *GKC* and the analyses #9 to #12 with 756 *LKC*.

Table 3 also presents the size of the governing system to be solved. For the analyses with only one subdomain, all *DOF* are considered as interior and are present in the governing system. When *GKC* are used, the 756 constraint equations are added to the system, because the *LM* method is used. Moreover, when *LKC* are implemented, the *DOF* associated with the 756 constraint equations are eliminated from the governing system.

For the case of more than one subdomain, the governing system to be solved is the reduced problem defined by the inter-subdomain *DOF*. When *GKC* are implemented, the reduced problem is also augmented with the equations associated with the constraints. On the other hand, when *LKC* are used the size of the reduced problem does not change because the constraints are applied for the internal *DOF* of each subdomain, which is an important difference between these two methods.

6.1. Accuracy of the results

Table 4 presents the results for the vertical displacement at point A and for the work performed by the discrete external forces, which in this case is equal to the symmetric of the elastic strain energy. The results show that for Mesh #1 the solution matches the reference data, which is expected because the mesh, the element types and the global finite element formulations are the same.

Regarding the analyses with Mesh #2, two important observations must be made concerning the data presented. At first, the same results were obtained for all domain partitions and methods used to enforce the kinematic constraints. Secondly, the results are very similar to the analyses without using *HD* ($e_r \leq 1.20\%$), both in terms of the displacements and of the work performed by external forces. This result shows that the accuracy degradation in the computation of the global structural behaviour using the *HD* technique is small when compared to the case where the structure is modelled using only solid elements.

From these results, it is possible to confirm that the proposed methodology does not introduces any significant loss of accuracy in the results. Additionally, both methodologies for imposing the kinematic constraints are equivalent in terms of the overall results.

6.2. Performance analysis

This analysis is made for assessing the efficiency of the KC-enforcing methods, when combined with the parallelization procedure.

All computations were executed on a 8 GB RAM multi-core computer (model *Intel Core i7 720QM*) with 4 *processing units* (*PU*), which runs at a variable clock rate between 1.6 and 2.8 GHz, due to an automatic dynamic hardware-implemented overclock procedure. These specifications lead to a theoretical peak speed between 25.6 and 44.8 GFlop/s, considering 4 instructions per cycle. Furthermore, the computations were made using a windows implementation of the Matlab software [22] and using specific toolboxes for the parallelization [23, 24].

The following measures were taken to mitigate some uncertainties associated with the computational performance. All non-critical processes and services running in the operating system were terminated and all processes related to the computation (Matlab's client session, Matlab workers, and the *spmd* and *mpiexec* processes) were executed with high priority levels. In addition, the computing times of all analyses were affected by a factor γ, in order to take into account the variation of the CPU's clock rate. For the processor installed in the computer, the adopted clock rate distribution was $\{2.80; 2.20; 1.60; 1.60\}$ GHz for $\{1; 2; 3; 4\}$ Matlab workers or active cores, leading to the correction factors $\{1.000; 0.786; 0.571; 0.571\}$. This procedure was validated by estimating the number of floating point operations per second of two different computations using parallel computations. The following expression is used:

$$F^* = \gamma \, F_s \, n_w, \tag{51}$$

where n_w is the number of Matlab workers and F_s is the number of floating point operations per second for the sequential code run in the Matlab client session. The computations used were matrix-by-matrix multiplication and linear system solving using Matlab's "\" operator and in both cases the results showed that the relative errors are always below 8% and most of the time below 5%, therefore demonstrating that the computational uncertainties are restrained at a satisfactory level (see Mendes [25] for more details).

The computing times of five incremental steps from the *IDA* were measured and the average values are used to assess the performance of the structural analysis. These average computing times ranged between 97.060 s for the sequential version of the code and using mesh #1, and 11.581 s when 4 subdomains, mesh #2 and *LKC* are considered.

A deeper analysis regarding the computational efficiency can be made by introducing the following parameters. The *computational performance gain* (CPG) for the homogeneous system used in the computations is defined as the ratio between the sequential code computing time and the parallelized version of the code using n_p processing units. In addition, the *efficiency* of a homogeneous system (E) is defined as the computational performance gain per processor: $E = CPG/n_p$. In all cases, the number of processing units considered in each case matches the number of subdomains ($n_p = n_s$). The term *speed-up* is avoided because the computations in the sequential and in the parallelized versions of the code are different.

Let us start by assessing the effect of the parallelization for each mesh and KC-enforcing method. These results are presented numerically in Table 5 and graphically in Figure 11. The analyses with one subdomain confirmed a small computing time increase for the parallelized version of the code due to the implicit overhead. Moreover, for the analysis with two and four subdomains, the computing time decreased significantly for Mesh #1 and Mesh #2 combined with *LKC* and this decrease was less significant for Mesh #2 and *GKC*. This data clearly shows that for two subdomains using Mesh #1 and Mesh #2 with LGK, the CPG values show linear scalability with the parallelization level and even super-linear scalability, *i.e.* efficiency above 100%, due to the different solving technique associated with the *PS* method.

The CPG values are significantly smaller when Mesh #2 is used with *GKC*. This response can be associated with two main aspects. At first, when the *GKC* method is implemented in the reduced problem, which is solved at the *MPU*, it does not benefit from the parallelization, contrary to the *LKC*, which are enforced at the subdomain level (*SPU*), and hence, computed in parallel. Secondly, for this case in particular, enforcing the *GKC* results in nearly doubling the size of the reduced problem, which also contributes to worse results in terms of the CPG and efficiency values.

For the analyses with four subdomains, the CPG and efficiency values are reduced bellow the linear scalability level. This behaviour is expected due to the load unbalancing associated with the mesh partition with four subdomains, as presented in Table 2 and discussed before. This aspect results in a significant loss of efficiency because under the data parallelism approach, the light-weighted *PU* must wait idle until the most loaded units finish their computations.

Another valuable source of information can be extracted from the CPG values grouped for the different meshes and KC-enforcing methods. These results are numerically presented in Table 6 and graphically in Figure 12. The curves associated with one subdomain show higher CPG values for Mesh #2 than for Mesh #1, as a result of the decrease in the number of *DOF* associated with the *HD* technique. Moreover, comparing the result using *GKC* and *LKC* it is possible to observe that similar results were obtained. This is expected because for one subdomain it is not possible to gain from parallelization for the *LKC* and the supplementary equations added to the governing system using *GKC*, represent only a minor increase in the size, with almost insignificant effect on the overall performance. Furthermore, the curves associated with two and four subdomains show CPG values of about 2.0 for Mesh #2 using *GKC* when compared to Mesh #1. These gains in efficiency are mainly associated with the decrease in the number of *DOF* using the *HD* technique. The CPG values increase even further when using *LKC* due to the additional parallelization associated with this technique.

The next step in the analysis consists in combining the effect of using the parallelization, the *PS* method and the *hybrid discretization* technique, which led to the results presented in Table 7 and in Figure 13. These results were normalized by the analysis made with the sequential version of the code, considering only one subdomain and using Mesh #1. This data shows a combined CPG value between 4 and 6, when two or four subdomains are used together with the *GKC*. In addition, the efficiency gains are even more significant when *LKC* are adopted, reaching values between 9 and 12, which represent very significant and encouraging performance gains.

At this point, it should be stressed that even better performance gains could have been achieved if the load balancing had been more uniform for Mesh #2 with four subdomains. Moreover, better results may also be accomplished using higher level of parallelization and by adjusting the number of subdomain interior *DOF* to optimal values, *e.g.* to around 10000-20000 *DOF*, rather than the 2000-5000 considered in this example using four subdomains.

For the interpretation of these results, it is convenient to extend the performance analysis to the different code segments of typical finite element algorithms. The incremental procedure was divided into the following four sections: i) *Stiffness Matrix*; ii) *Corrector*; iii) *Unbalanced Forces*; and iv) *Other*. The *Stiffness Matrix* segment includes the computation of the element stiffness matrices and the assemblage of the global or subdomain stiffness matrices. The *Corrector* segment includes the procedure used to obtain the iterative correction of the discrete displacements in order to reduce the unbalanced forces. Moreover, the *Unbalanced Forces* part includes the prior computations for the element's strains, state determination, stresses, and finally, the global or subdomain internal forces. The *Other* section includes the time step initialization, the computation of the applied external forces, retrieving and saving variables, and the auxiliary computations associated with the time step ending.

Figure 14 presents the percentage of computing time spent on each code segment for the different problem sizes considered. The data presented led to the following conclusions. The *Other* segment reduces its percentage for larger problem sizes (Mesh #1) and tends to increase with parallelization. This is expected as the computations included in this section are much less sensitive to the problem size and for the parallel version of the code results in a large overhead (*e.g.* retrieving the unknowns from the Matlab workers for ending the time step). Additionally, in this segment there are several tasks that are executed in the Matlab client session, therefore, without the possibility to scale with the number of subdomains (*e.g.* storing the unknowns on hard disk and ending the time step).

It is also possible to observe that the *Unbalanced Forces* segment maintains roughly its relative weight. This is related to the fact that most of the computations associated with this segment are performed at the element level and with good conditions for scalability.

In addition, the *Corrector* segment tends to increase its relative weight for the analyses with substructuring and the *Stiffness Matrix* segment presents the inverse behaviour. This is a fundamental aspect for the algorithm performance and deserves an in-depth analysis. The first issue to be noted is that for the analyses with parallelization the corrector phase is completely different and more complex. For the analyses performed with only one subdomain, the corrector segment corresponds basically to the solution of the linearized system of equations using a direct method. On the other hand, the use of the *PS* method requires the previous computation of the structural operators, such as the stiffness matrices and the Schur complements, which are first assembled at the subdomain level (7,8) and only in the corrector segment are joined together for computing the reduced problem (5,6). Therefore, it is possible to conclude that for multi-domain computations there is computational load transference from the *Stiffness Matrix* segment to the corrector phase and that the computing times are not just a matter of parallelization efficiency, but result from using different solving schemes. Regarding this issue, the most relevant fact in Figure 14 is that this effect is considerably less significant for the analyses associated with Mesh #2 considering two and four subdomains and using *LKC*.

The reasons for this behaviour can be associated with two main aspects: At first, the size of the reduced problem for the analyses with *LKC* is extremely small (*e.g.* 6 or 18 *DOF*, see Table 3); Secondly, the enforcement of the KC is made in the Corrector segment, but for the case of *LKC*, it is made in the scope of the subdomain, which has the possibility of gaining efficiency from the parallelization.

Figure 15 presents the stress field σ_{xx} plotted on the deformed mesh for the analysis with Mesh #2, four subdomains and using *LKC*. In this figure is possible to confirm that the expected deformed shape and stress distribution is retrieved.

7. Conclusions

In this paper, the computational performance was improved by taking advantage of concurrent computations (parallelization), domain decomposition techniques and the so-called *hybrid discretization*.

The use of concurrent computations was implemented in the finite element code developed and makes it possible to take advantage of the data parallelism associated with the finite element method and partitioned subdomains, *e.g.* for the element's state determination and for the computation of the structural operators. The adopted domain decomposition method (Primal Substructuring), was also used to enhance the gains from using concurrent computations by adapting the structural analysis algorithm.

Hybrid discretizations combine different types of meshes and can significantly reduce the size of the problem to be analysed. To illustrate this, Example 2 reveals a reduction of the total number of unknowns of about 48% and it was possible to expose CPG values of 3.5 using sequential computations and 4.1 using concurrent computations.

The *HD* proposed also presents the following two additional benefits: it creates a natural and efficient partition between the subdomains that can be used in the substructured analysis and it increases the modelling flexibility by promoting the use of different element types in different parts of the structure.

It was necessary to test different methods to enforce kinematic constraints associated with the *HD*. This research concluded that the best technique to enforce the *Beam2Solid* kinematic constraint is to use a subdomain-based approach by using the Master-Slave Elimination method, known as *Local Kinematic Constraints* (*LKC*) in the scope of this work. Using this technique, the model can benefit from the parallelization and scale with the number of processors. In addition, when the objective is to enforce inter-subdomain behaviour, then the best solution is to use *Global Kinematic Constraints* (*GKC*), enforced by Lagrange Multipliers and using additional *LKC* to reduce the number of unknowns added to the coarse problem by creating local master and slave nodes. Lagrange Multipliers were also used to prescribe displacements on the structure, and as before, trying to reduce the number of unknowns added to the reduced problem using additional *LKC*.

Furthermore, when combining the improvements associated with concurrent computations with the ones resulting from substructuring techniques and with the ones resulting from the *HD*, it was possible to reveal CPG values of up to 11.7 using only four processing units, even though for that case the mesh partition led to some load unbalance. These results exceeded the initial expectations and were considered very promising and extremely encouraging.

The main future development for this work is to implement and test the proposed technique in a more efficient computational environment (*e.g.* cluster), with larger problem-sizes and higher parallelization levels. In addition, further testing is required to assess the accuracy and performance of the proposed methodology for solving non-linear problems.

8. Acknowledgements

The authors would like to acknowledge and emphasize the importance of the financial support given to this research by *Fundação para a Ciência e a Tecnologia* through the doctoral grant with reference SFRH/BD/21491/2005/J03122797V2. In addition, all the support gave by Laboratório Nacional de Engenharia Civil (LNEC), in particular by Dr. Ema Coelho, is greatly recognized and appreciated.

References

[1] T. P. A. Mathew, Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations, Vol. 61 of Lectures Notes in Computational Science and Engineering, Springer Berlin Heidelberg, Berlin, 2008.

[2] T. P. A. Mathew, Schur complement and iterative substructuring algorithms, in: Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations, Vol. 61 of Lecture Notes in Computational Science and Engineering, Springer Heidelberg, Berlin, 2008, pp. 107–230.

[3] D. T. Nguyen, Finite element methods - Parallel-sparse statics and eigen-solutions, Springer Science, Inc., 2006.

[4] J. S. Przemieniecki, Matrix structural analysis of substructures, Am. Inst. Aero. 1 (1963) 138–147.

[5] A. Toselli, O. Widlund, Domain Decomposition Methods - Algorithms and Theory, Vol. 34 of Springer Series in Computational Mathematics, Springer, New York, 2005.

[6] W. Corley, Rotational capacity of reinforced concrete beams, Journal of Structural Division 97-ST5 (1966).

[7] A. Mattock, Discussion of "rotational capacity of reinforced concrete beams" by W. Corley, Journal of Structural Division 93-ST2 (1967).

[8] T. Paulay, M. J. N. Priestley, Seismic design of reinforced concrete and masonry buildings, Wiley, New York, 1992.

[9] M. Priestley, R. Park, Strength and ductility of concrete bridge colums under seismic loading, ACI Structural Journal 84-S8 (1987).

[10] E. Garusi, A. Tralli, A hybrid stress - assumed transition element for solid-to-beam and plate-to-beam connections, Computers & Structures 80 (2002) 105–115.

[11] T. Gmur, R. Kauten, 3d solid to beam transition elements for structural dynamics analysis, International Journal for Numerical Methods in Engineering 36 (1993) 1429–1444.

[12] Lim, Song, Seog, Formulation method for solid-to-beam transition finite elements, KSME International Journal 15 (11) (2001) 1499–1506.

[13] Ziyaeifar, Noguchi, A refined model for beam elements and beam column joints, Computers and Structures 76 (2000) 551–564.

[14] R. D. Cook, D. S. Malkus, M. E. Plesha, Concepts and Applications of Finite Element Analysis, 3rd Edition, John Wiley & Sons, Inc., New York, 1989.

[15] G. R. Liu, S. S. Quek, The finite element method - A practical course, Butterworth-Heinemann, 2003.

[16] F. Martín, A. Benavent-Climent, R. Gallego, A scheme for imposing constraints and improving conditioning of structural stiffness matrices, Int. J. Numer. Meth. Biomed. Engng 26 (2008) 1117–1124.

[17] H. M. Hilber, T. J. R. Hughes, R. L. Taylor, Improved numerical dissipation for time integration algorithms in structural dynamics, Earthquake Engineering and Structural Dynamics 5 (3) (1977) 283–292.

[18] M. A. Crisfield, Non-Linear Finite Element Analysis of Solids and Structures - Volume 2: Advanced Topics, John Wiley & Sons Ltd, New York, 1997.

[19] CEN, Eurocode 8: Design of structures for earthquake resistance. Part 1: General rules, seismic actions and rules for buildings, Draft No 6, Stage 49 (2003).

[20] N. M. Newmark, E. Rosenblueth, Fundamentals of Earthquake Engineering, 1st Edition, Prentice-Hall, Inc, New Jersey, 1971.

[21] ADINA R&D Inc, ADINA - Automatic Dynamic Incremental Nonlinear Analysis, version 8.5 (2008).

[22] MathWorks, MATLAB - The language of technical computing (2010).

[23] MathWorks, MATLAB - Parallel computing toolbox - User's guide (2010).

[24] MathWorks, MATLAB - Distributed computing server - System administrator's guide (2010).

[25] L. A. M. Mendes, Refined three-dimensional seismic analysis of reinforced concrete structures, Ph.D. thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa (2011).

Figure1

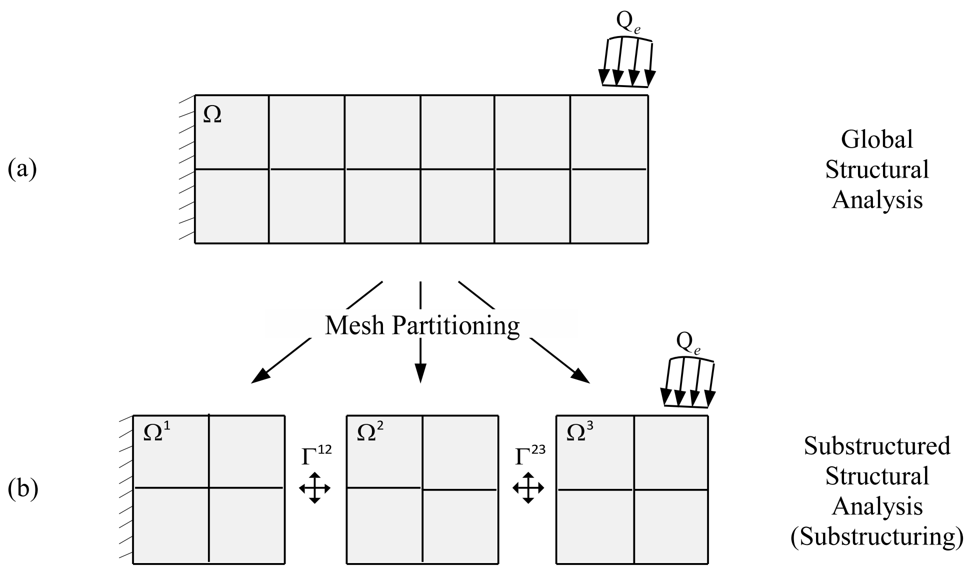
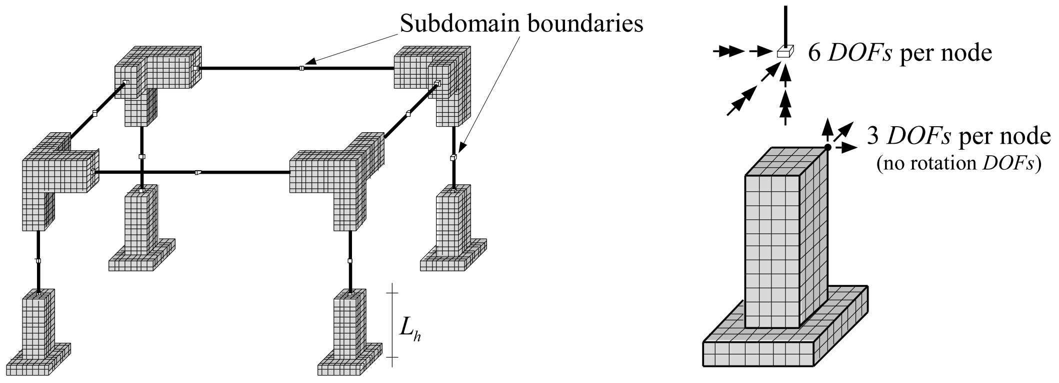


Figure 1: Structural analysis techniques.

Figure2



(a) (b)

Figure 2: Schematic representation of the *hybrid discretization* approach.

Figure3

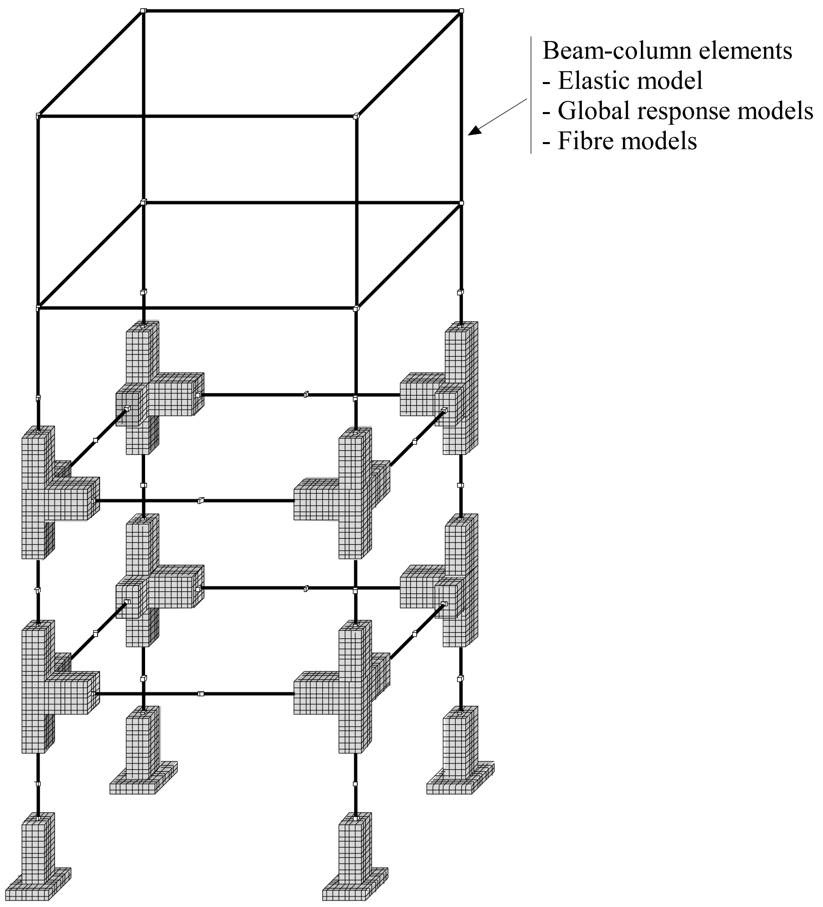


Figure 3: Different possibilities arising from using *hybrid discretizations*.

Figure4

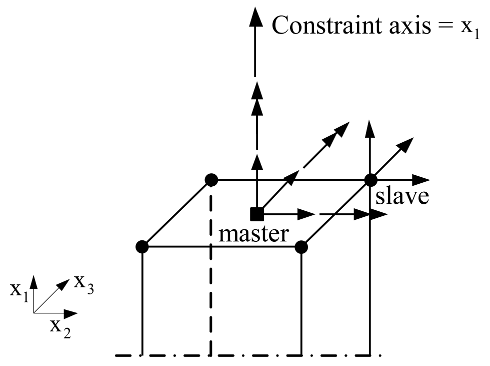
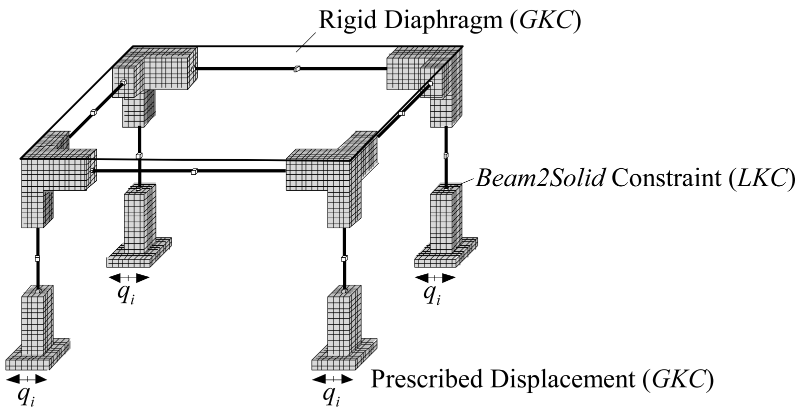


Figure 4: Implementation of the *Beam2Solid* kinematic constraint.

Figure5



Rigid Diaphragm (*GKC*)
Beam2Solid Constraint (*LKC*)
q_i
q_i
q_i
q_i
Prescribed Displacement (*GKC*)

Figure 5: Different methods to enforce kinematic constraints.

Figure6

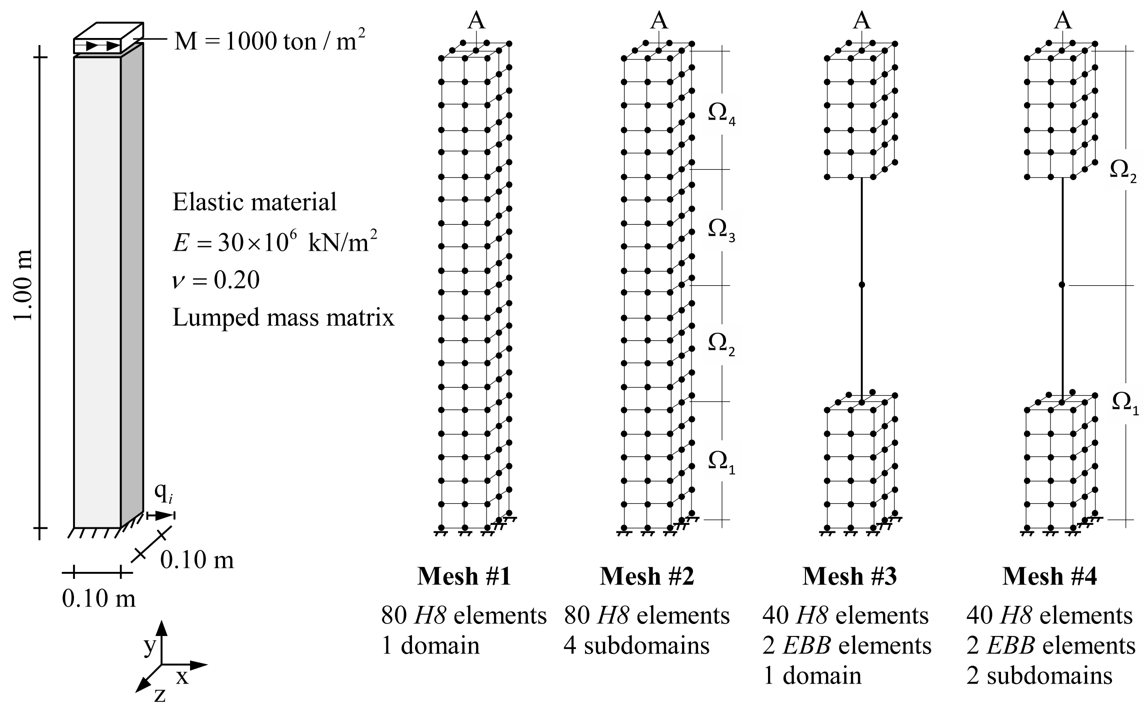
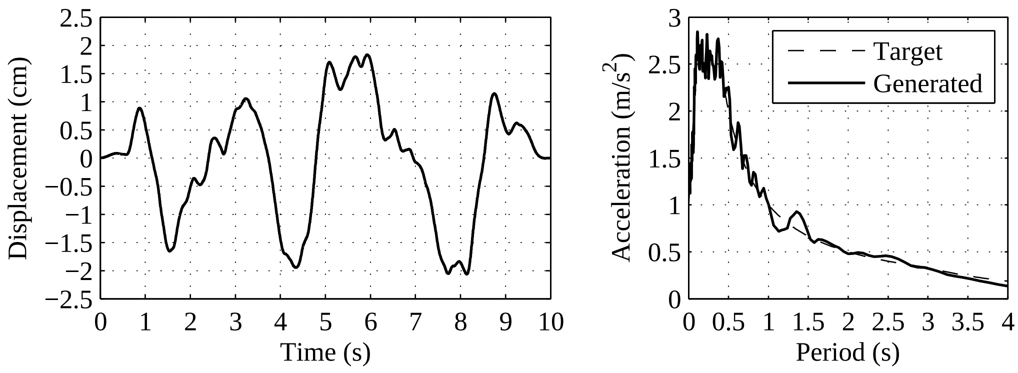


Figure 6: Example 1 - Definition of the problem.

Figure7



(a) time-history (b) response spectra

Figure 7: Example 1 - Earthquake record.

Figure8

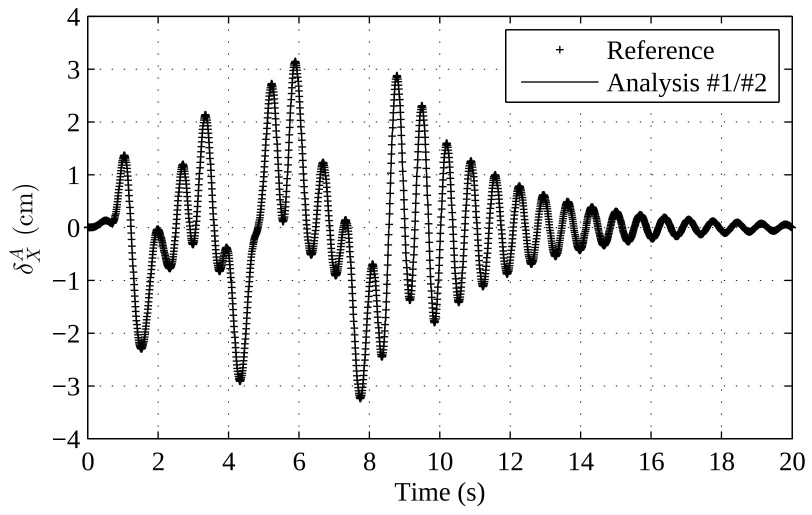


Figure 8: Example 1 - Displacement at node A for analyses #1 and #2.

Figure9

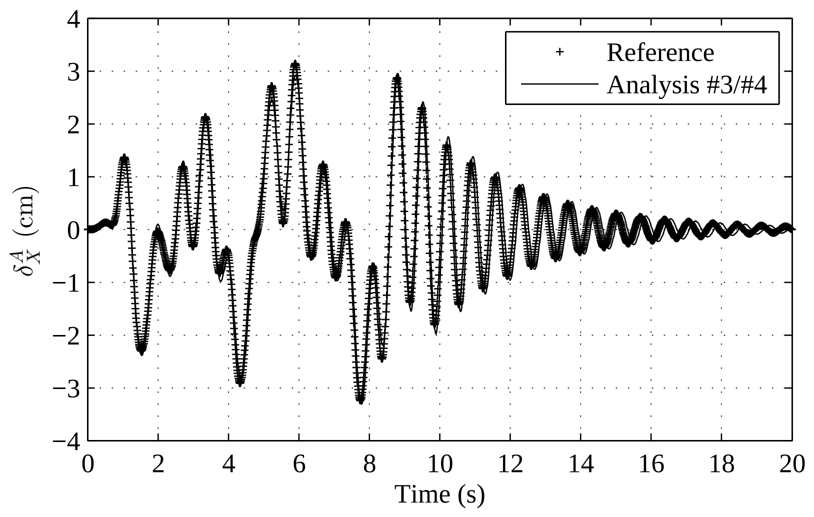


Figure 9: Example 1 - Displacement at node A for analyses #3 and #4.

Figure10

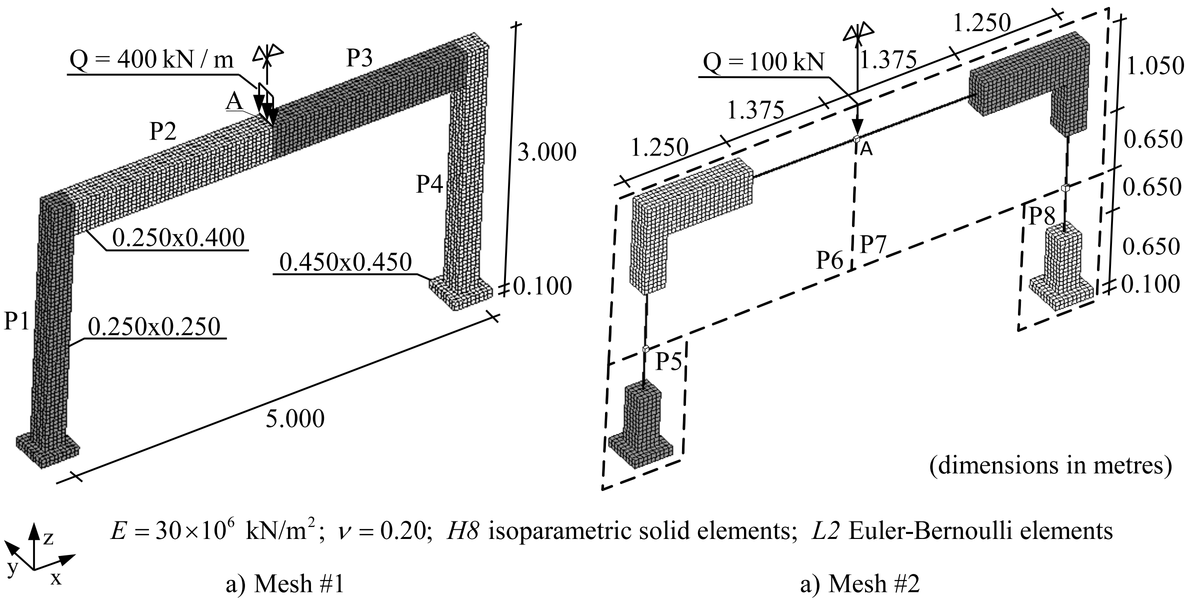


Figure 10: Example 2 - Definition of the problem.

Figure11

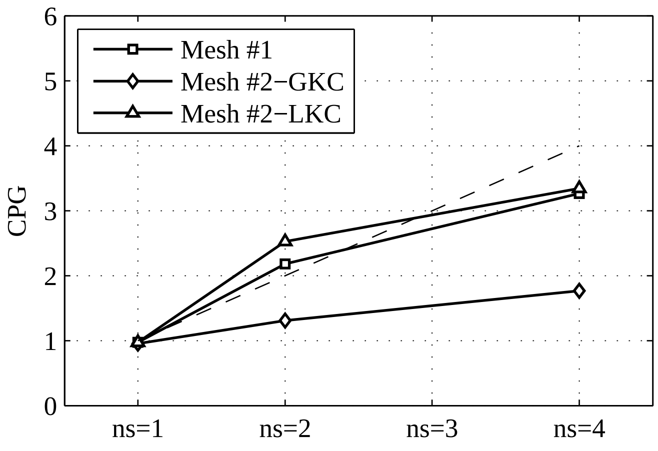


Figure 11: Example 2 - CPG values for different parallelization levels.

Figure12

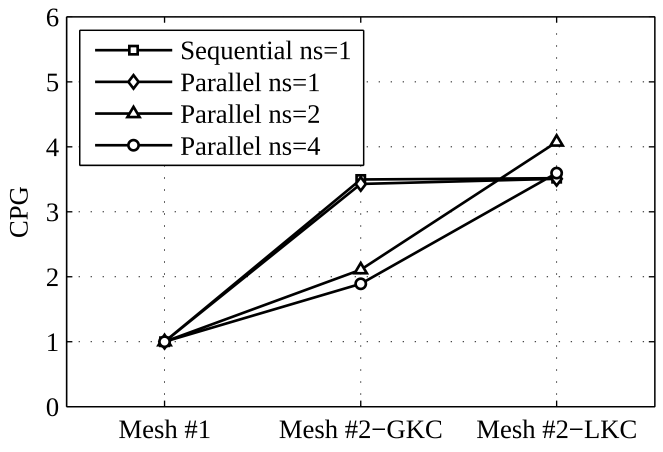


Figure 12: Example 2 - CPG values for different mesh types and KC-enforcing methods.

Figure13

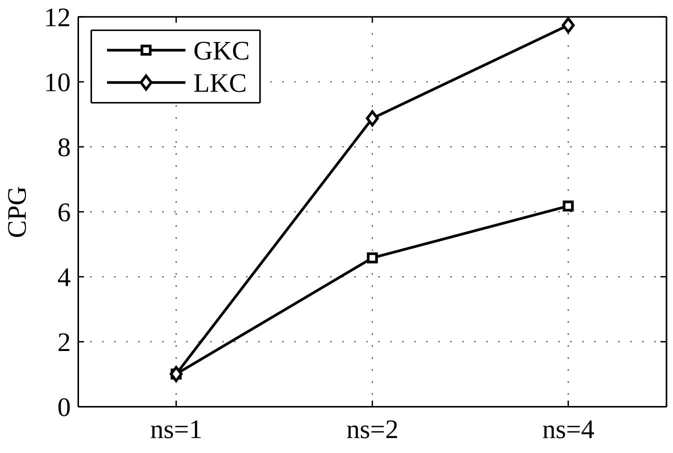


Figure 13: Example 2 - CPG values when combining all performance-enhancing techniques.

Figure14

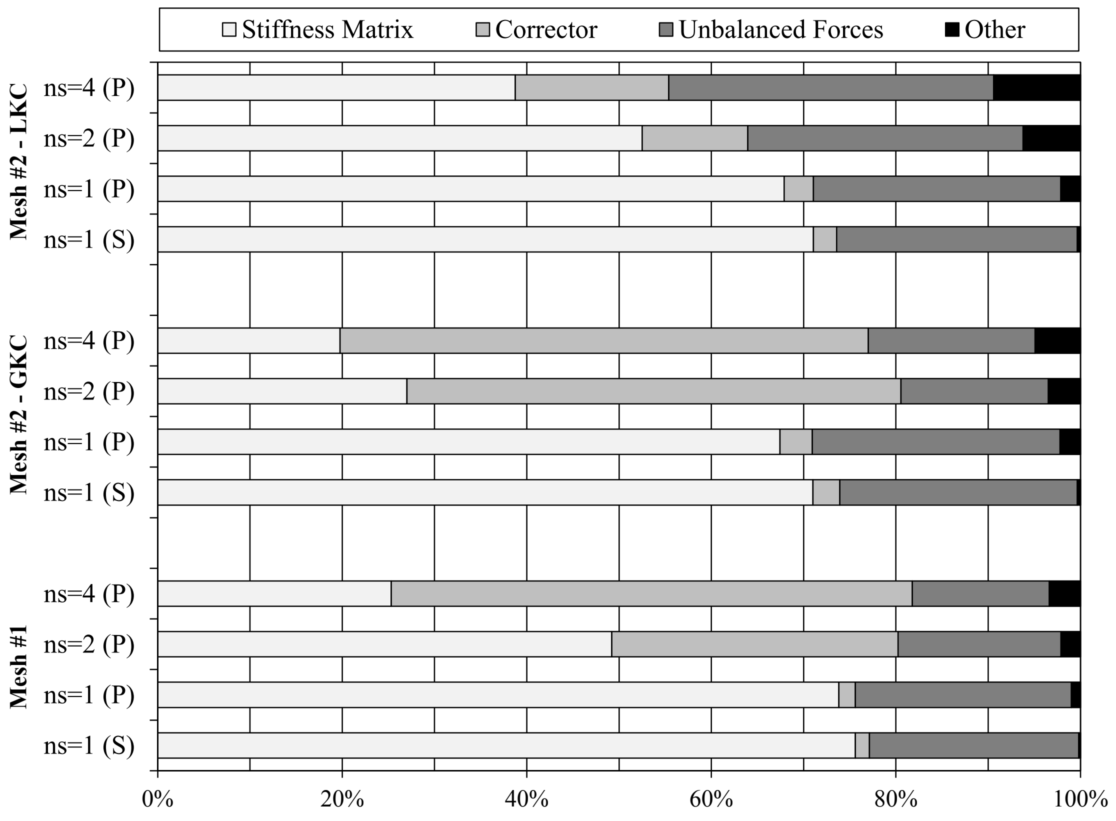


Figure 14: Example 2 - Percentage of computing time for the different code segments.

Figure15

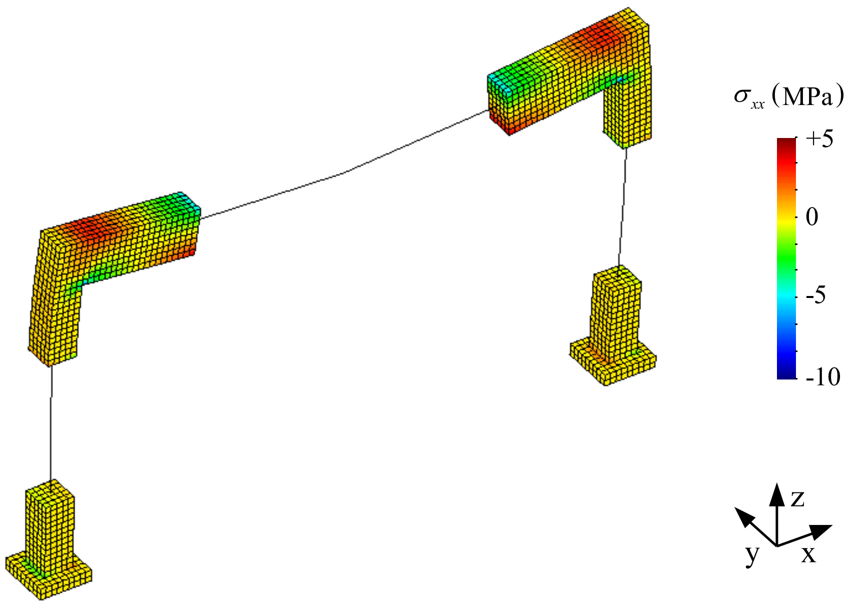


Figure 15: Example 2 - Deformed mesh (magnified by 50) and stress field σ_{xx} for analyses #09 to #12.

Table 1: Example 1 - Characteristics of the meshes and of the analyses.

Analysis	Mesh, n_d	Elements	*DD* Met., n_s	K. Constraints
#1	#1, 540	80 (*H8*)	-, 1	-
#2	#2, 540	80 (*H8*)	*PS*, 4	-
#3	#3, 315	40 (*H8*), 2 (*EBB*)	-, 1	Beam2Solid (*GKC*)
#4	#4, 315	40 (*H8*), 2 (*EBB*)	*PS*, 4	Beam2Solid (*GKC*)

Table2

Table 2: Example 2 - Characteristics of the discretizations.

Analysis	Mesh, n_d	Subdomains (n_s)	Boundary n_d	Interior n_d
#01	#1, 29388	P1+P2+P3+P4 (1)	-	29338
#02	#1, 29388	P1+P2+P3+P4 (1)	-	29338
#03	#1, 29388	P1+P2; P3+P4 (2)	162	14694; 14532 (50%; 50%)
#04	#1, 29388	P1; P2; P3; P4 (4)	486	6918; 7614; 7452; 6918 (24%; 26%; 26%; 24%)
#05	#2, 15294	P5+P6+P7+P8 (1)	-	15294
#06	#2, 15294	P5+P6+P7+P8 (1)	-	15294
#07	#2, 15294	P5+P6; P7+P8 (2)	798	7650; 7650 (50%; 50%)
#08	#2, 15294	P5; P6; P7; P8 (4)	810	1896; 5346; 5346; 1896 (13%; 37%; 37%; 13%)
#09	#2, 15294	P5+P6+P7+P8 (1)	-	15294
#10	#2, 15294	P5+P6+P7+P8 (1)	-	15294
#11	#2, 15294	P5+P6; P7+P8 (2)	6	7644; 7644 (50%; 50%)
#12	#2, 15294	P5; P6; P7; P8 (4)	18	2010; 5628; 5628; 2010 (13%; 37%; 37%; 13%)

Table 3: Example 2 - Characteristics of the analyses.

Analysis	*DD* Met. (n_s)	Algorithm	K. Constraints (n_k)	Reduced Problem n_d
#01	- (1)	Sequential	-	29338
#02	*PS* (1)	Parallel	-	29338
#03	*PS* (2)	Parallel	-	162
#04	*PS* (4)	Parallel	-	486
#05	- (1)	Sequential	*GKC-LM* (756)	15294+756=16050
#06	*PS* (1)	Parallel	*GKC-LM* (756)	15294+756=16050
#07	*PS* (2)	Parallel	*GKC-LM* (756)	798+756=1554
#08	*PS* (4)	Parallel	*GKC-LM* (756)	810+756=1566
#09	- (1)	Sequential	*LKC-MSE* (756)	15294-756=14538
#10	*PS* (1)	Parallel	*LKC-MSE* (756)	15294-756=14538
#11	*PS* (2)	Parallel	*LKC-MSE* (756)	6
#12	*PS* (4)	Parallel	*LKC-MSE* (756)	18

Table 4: Example 2 - Selected results from the analyses.

Analysis	δ_Z^A (m) , e_r	$W_e = -U^e$ (kN.m) , e_r
Reference	-4.20086e-3, -	2.10112e-1, -
#01 to #04	-4.20086e-3, 0.00%	2.10112e-1, 0.00%
#05 to #12	-4.15165e-3, -1.17%	2.07583e-1, -1.20%

Table5

Table 5: Example 2 - CPG and efficiency values for different parallelization levels.

Analysis	Parallel, $n_s = 1$	Parallel, $n_s = 2$	Parallel, $n_s = 4$
Mesh #1	0.97 (97.5%)	2.18 (108.9%)	3.26 (81.6%)
Mesh #2-*GKC*	0.96 (95.5%)	1.31 (65.5%)	1.77 (44.1%)
Mesh #2-*LKC*	0.97 (97.2%)	2.52 (126.2%)	3.34 (83.5%)

Table 6: Example 2 - CPG values for different mesh types and KC-enforcing methods.

Analysis	Mesh #1	Mesh #2-*GKC*	Mesh #2-*LKC*
Sequential, $n_s = 1$	1.00	3.50	3.51
Parallel, $n_s = 1$	1.00	3.43	3.51
Parallel, $n_s = 2$	1.00	2.10	4.07
Parallel, $n_s = 4$	1.00	1.89	3.60

Table7

Table 7: Example 2 - CPG values when combining all performance-enhancing techniques.

Analysis	Mesh #1	Mesh #2-*GKC*	Mesh #2-*LKC*
Sequential, $n_s = 1$	1.00	-	-
Parallel, $n_s = 2$	-	4.58	8.87
Parallel, $n_s = 4$	-	6.17	11.74