Solveurs performants pour l’optimisation sous contraintes en identification de paramètres

Naoufal Nifa

NNT : 2017SACLC066

Thèse de doctorat de l'Université Paris-Saclay préparée à CentraleSupélec

École Doctorale n°579
Sciences mécaniques et énergétiques, matériaux et géosciences
Spécialité de doctorat : Génie Mécanique

par

M. Naoufal NIFA

Solveurs performants pour l'optimisation sous contraintes en identification de paramètres

Thèse présentée et soutenue le 24 novembre 2017, à Gif-sur-Yvette, devant le jury composé de :

M.	**Yousef SAAD**	Professeur, University of Minnesota	Président
M.	**Etienne BALMÈS**	Professeur, ENSAM	Rapporteur
M.	**François-Xavier ROUX**	Professeur, UPMC - Paris 6	Rapporteur
M.	**Denis AUBRY**	Professeur, CentraleSupélec	Directeur de thèse
M.	**Didier CLOUTEAU**	Professeur, CentraleSupélec	Examinateur
M.	**Mathieu CORUS**	Ingénieur de recherche, EDF R&D	Examinateur

ÉCOLE DOCTORALE

Sciences mécaniques et énergétiques, matériaux et géosciences (SMEMAG)

Titre : Solveurs performants pour l'optimisation sous contraintes en identification de paramètres

Keywords : Optimisation sous contraintes, Systèmes point-selle, Préconditionneurs par blocs, Problèmes inverses

Résumé : Cette thèse vise à concevoir des solveurs efficaces pour résoudre des systèmes linéaires, résultant des problèmes d'optimisation sous contraintes dans certaines applications de dynamique des structures et vibration (la corrélation calcul-essai, la localisation d'erreur, le modèle hybride, l'évaluation des dommages, etc.). Ces applications reposent sur la résolution de problèmes inverses, exprimés sous la forme de la minimisation d'une fonctionnelle en énergie. Cette fonctionelle implique à la fois, des données issues d'un modèle numérique éléments finis, et des essais expérimentaux. Ceci conduit à des modèles de haute qualité, mais les systèmes linéaires point-selle associés, sont coûteux à résoudre. Nous proposons deux classes différentes de méthodes pour traiter le système. La première classe repose sur une méthode de factorisation directe profitant de la topologie et des propriétés spéciales de la matrice point-selle. Après une première renumérotation pour regrouper les pivots en blocs d'ordre 2. L'élimination de Gauss est conduite à partir de ces pivots et en utilisant un ordre spécial d'élimination réduisant le remplissage. Les résultats numériques confirment des gains significatifs en terme de remplissage, jusqu'à deux fois meilleurs que la littérature pour la topologie étudiée. La seconde classe de solveurs propose une approche à double projection du système étudié sur le noyau des contraintes, en faisant une distinction entre les contraintes cinématiques et celles reliées aux capteurs sur la structure. La première projection est explicite en utilisant une base creuse du noyau. La deuxième est implicite. Elle est basée sur l'emploi d'un préconditionneur contraint avec des méthodes itératives de type Krylov. Différentes approximations des blocs du préconditionneur sont proposées. L'approche est implémentée dans un environnement distribué parallèle utilisant la bibliothèque PETSc. Des gains significatifs en terme de coût de calcul et de mémoire sont illustrés sur plusieurs applications industrielles.

ÉCOLE DOCTORALE
Sciences mécaniques et énergétiques, matériaux et géosciences (SMEMAG)

Title : Efficient solvers for constrained optimization in parameter identification problems

Keywords : Constrained optimization, Saddle point systems, Block preconditioners, Inverse problems

Abstract : This thesis aims at designing efficient numerical solution methods to solve linear systems, arising in constrained optimization problems in some structural dynamics and vibration applications (test-analysis correlation, model error localization, hybrid model, damage assessment, etc.). These applications rely on solving inverse problems, by means of minimization of an energy-based functional. This latter involves both data from a numerical finite element model and from experimental tests, which leads to high quality models, but the associated linear systems, that have a saddle-point coefficient matrices, are long and costly to solve. We propose two different classes of methods to deal with these problems. First, a direct factorization method that takes advantage of the special structures and properties of these saddle point matrices. The Gaussian elimination factorization is implemented in order to factorize the saddle point matrices block-wise with small blocks of orders 2 and using a fill-in reducing topological ordering. We obtain significant gains in memory cost (up to 50%) due to enhanced factors sparsity in comparison to literature. The second class is based on a double projection of the generated saddle point system onto the nullspace of the constraints. The first projection onto the kinematic constraints is proposed as an explicit process through the computation of a sparse null basis. Then, we detail the application of a constraint preconditioner within a Krylov subspace solver, as an implicit second projection of the system onto the nullspace of the sensors constraints. We further present and compare different approximations of the constraint preconditioner. The approach is implemented in a parallel distributed environment using the PETSc library. Significant gains in computational cost and memory are illustrated on several industrial applications.

Remerciements

Cette thèse a été une véritable aventure semée de défis et de difficultés. Une expérience humaine de premier plan qui m'a emmené au-delà de ce que je pouvais imaginer en l'entamant. Au début de chaque aventure, il faut prévoir toutes les mesures pour être paré aux éventualités du voyage. Mais l'aventure de thèse est certainement celle où l'on est le plus diminu. Pour arriver au bout, il fallait que je sois entouré par des personnes d'exception qui ont cru en moi. J'ai donc cette envie irrépressible de leur rendre hommage et de leur exprimer ma plus profonde reconnaissance.

Je tiens à exprimer mes plus vifs remerciements à mon Directeur de thèse, le Professeur **Denis Aubry**. Son écoute, sa confiance, et ses conseils précieux à la hauteur de ses compétences, m'ont aidé à mener ce voyage jusqu'au bout. Denis est aussi un encadrant avec de réelles qualités humaines, chaleureux, toujours souriant et bien veillant. Je n'oublierai jamais ses anecdotes croustillantes lors de nos réunions fréquentes. Je suis vraiment fier de faire partie de ses doctorants.

Toute ma gratitude s'adresse à Monsieur **Mathieu Corus**, ingénieur chercheur à EDF R&D et enseignant à CentraleSupélec. Sans son encadrement hors du commun, ce travail n'aurait probablement pas vu le jour. Mathieu est un expert sur qui on peut compter, il est non seulement quelqu'un de très compétant, il est aussi un fin pédagogue. Sa grande confiance en mes capacités m'a toujours fait progresser sur tous les niveaux. Le voir chaque jour était un réel plaisir et était une opportunité pour moi de pouvoir voir clair lorsque les routes de la thèse semblaient bloquées. Je n'ai vraiment pas les mots pour lui exprimer toute mon admiration et ma reconnaissance.

J'exprime tous mes remerciements aux Professeurs **Etienne Balmes** (ENSAM) et **François-Xavier Roux** (Université Pierre et Marie Curie – Paris 6), pour l'honneur qu'ils m'ont fait en prenant la charge de rapporteurs et en siégeant à ce jury. Je les remercie également pour leurs questions, leurs commentaires, et leurs retours sur ce travail de thèse.

Je tiens tout particulièrement à témoigner ma vive reconnaissance au Professeur **Yousef Saad**, pour m'avoir gentiment accueilli au sein de son laboratoire de calcul scientifique à l'Université du Minnesota. Cette expérience m'a permis de partager des discussions riches et inspirantes avec lui. Je suis honoré par sa présidence de mon jury de thèse.

Je remercie le Professeur **Didier Clouteau** pour avoir accepté de faire partie de mon jury de thèse. Ses remarques, ses questions et ses retours étaient pertinents pour faire évoluer le travail effectué.

Durant mon travail de thèse, j'ai passé le plus grand de mon temps au sein du groupe vibrations des structures du département Analyses Mécaniques Avancées d'EDF R&D, où j'ai adoré travailler. Je voudrais donc remercier sincèrement **Anika Razakanaivo**, **Sebastien Caillaud** et **Thomas papaconstantinou**, non seulement pour m'avoir intégré à leurs équipes, mais aussi pour leur confiance permanente qui m'a permis de travailler en de bonnes conditions. J'étends mes remerciements à **Alexandre Foucault**, mon chef de groupe, pour son écoute, son aide et sa bonne humeur.

Je voudrais insister sur la chaleur de l'accueil, la disponibilité et la gentillesse de l'ensemble des acteurs que j'ai pu côtoyer tout au long de ce travail à EDF R&D. Je voudrais tout spécialement adresser mes remerciements à **Nicolas Tardieu**, **Olivier Boiteau** et **Natacha Bereux** qui ont accepté de partager avec moi leur temps, leurs idées et leurs motivations sur ce projet de recherche.

Aussi, je tiens à remercier chaleureusement tous ceux qui ont été là : collègues, nouveaux et anciens amis, et toutes les personnes que j'aime. La liste est trop longue, mais permettez-moi d'en mentionner au moins quelques-uns : **Fabien Banci**, **Vincent**, **Thibault**, **Zouhair**, **Fabien Grange**, **Aurélien**, **Hassan**, **Vinicius**, **Oana**, **Karima**, **Harinaivo**, **Alexandre Patrice**, **Pierre Badel**, **Nicolas**, **Zakariae**, **Mahmoud**, **Amine**, **Sara**, **Yannick**, **Pascal**. Mention spéciale à **Pierre Moussou** dont la capacité à faire rire inconditionnellement dépasse largement les attentes. Je tiens également à remercier sincèrement tous les collègues d'EDF et de CentraleSupélec sans exception, pour leur bonne humeur.

Enfin, j'adresse toute mon affection aux membres de ma famille. A mon jumeau **Iliass**, j'espère te voir docteur dans quelques mois. Je souhaite remercier mon grand frère **Anass**, qui m'a appris tant de choses, et qui a su m'inculquer l'amour des maths dès mon jeune âge. Je remercie ma sœur ainée **Hanae** pour son soutien pendant les périodes difficiles. Finalement, je dédie cette thèse à mes parents, **Noura** et **Mostafa**, en témoignage de l'amour, de l'affection et du soutien qu'ils m'ont offert depuis ma naissance. Pour toutes les peines, les sacrifices qu'ils ont consentis pour mon éducation, aucun mot ne saurait exprimer ma gratitude, mon amour, et mon profond respect.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction and general overview

Contents

1.1. Industrial context

EDF, as an operator of electric power production, needs to understand the mechanical behavior of ageing equipments of which it is not the manufacturer, in the purpose to ensure their proper functioning and to optimize their availability. Its R&D division rely on its expertise in the field of structural mechanics through a wide-scope of research activity (vibrations, fluid- structure interaction, seismic, rotor-dynamic machines, etc.) in order to guarantee the safety of the generation plants and maintaining high efficiency.

Some structures may be exposed to high levels of vibration due to poor initial design and ageing installations. As soon as the power plant starts up, a number of problems appear and no definitive solution is adopted. These issues can result in reduced structural integrity and could therefore be detrimental to both engine reliability and performance with a potentially serious impact on safety.

The detection of high vibration levels leads to the production shutdown, in order to avoid equipments damage. And this shutdown has a significant impact on the quality of service and performance. The industrial context emphasizes the need to consider the condition of ageing nuclear assets and manage their performance.

1.2. Considered methods and goals

This issue requires a deep understanding of the physical phenomena involved and the realization of numerical models to assess the corrective solutions. In order to diagnose the origin of the problem, test campaign is first and foremost performed on structures. A numerical model is then built to reproduce the nominal behavior and evaluate proposed solutions. Therefore, the designed numerical models must be of good quality in order to accurately predict the behavior of analyzed structures. Experimental data is then used for model-updating or field reconstruction purposes through inverse and namely identification problems.

Up to now, least-square's type methods were used to solve these identification problems. Indeed, the representativeness of numerical models is quantified during the stages of verification and validation and experimental information is then combined with numerical simulations to complete the a priori knowledge of structural behavior to propose industrial solutions. Hence, we seek to give the best state and parameter estimation in structural dynamics problems from both a finite element based mathematical model and a set of available experimental data. This work arises from EDF's need to improve solution methodologies for these inverse problems and their associated constrained optimization problems in structural dynamics.

Among the different existing approaches that enable these steps, one is based on energy functionals. This approach has shown its efficiency and has appeared to be an appropriate indicator of the quality of a model with respect to measured data [38][94]. Adopting the above-mentioned approach in a finite element framework leads to a sparse and large linear system of equations equivalent to a saddle-point system or Karush-Kuhn-Tucker (KKT) system. It arises too in many applications of scientific computing, e.g. constrained optimization and incompressible fluid flow [17].

The implementation of energy-based functional approach within the work of Kuczkowiak [71] shows that direct solvers used in mechanical softwares fail to solve efficiently the inverse problem associated to an industrial structure model with more than 10^6 dofs and few hundreds of measurement points and provide a huge computation cost for a single calculation. But the numerical solution of large-scale industrial scientific problems generally requires large computational times and large memory needs. Many efforts are acknowledged with respect to both computer architecture aspects and the design of efficient algorithms, in order to provide effective simulation tools. In this thesis, we are rather interested in the second aspect.

This research work focuses on the solution of problems issued from large-scale identification problems. More specifically, the main goal is to provide efficient solution methods to speedup the solution of constrained optimization problems within the framework of the open-source software Code_Aster ® [1], which is a general purpose finite element code developed at EDF. The choice of the considered methods directly is guided by from the main target of memory and computational time efficiency.

In this work, direct solution methods have been investigated as a way to address the sequence of saddle point systems to solve. Direct solvers consist of a first factorization of the coefficient matrix into triangular factors, then successive forward and backward substitutions. They are usually designed as a preprocessing step is applied before the factorization. This includes scaling, pivoting and ordering. The preprocessing step makes the numerical factorization in many cases easier and cheaper, which influences by the way the memory and the time of the factorization step[3]. In that sense, direct solvers have been widely and successfully used in the past decades in structural dynamics problems, particularly in the area of vibrations.

Direct solution methods, however, require a huge memory when dealing with a large and sparse linear system. In structural mechanics, as in other applications, these methods have proved to be efficient in the two-dimensional case, but a significant fill-in phenomenon usually occurs when factorizing large-scale three-dimensional problems [103]. Hence, the memory requirement generally penalizes their use. Furthermore, the a priori saddle point structure of the systems appears to be problematic.

The second class of methods is the iterative methods which provide a sequence of approximations of the solution of the linear system. Contrary to the direct solvers, these methods only require the knowledge of the action of the matrix on a vector. Furthermore, they are particularly well adapted to parallel computing [97].

We try here by means of a hybrid approach to reduce the needed memory and computation time. We mix up direct and iterative methods so that they contribute together to empower each other. Therefore, the factorization which is the key element of direct methods is used in preconditioned iterative methods.

We finally implements the developed methods in the mechanical code *CodeAster*®. The contribution of this thesis will therefore be useful for industrial applications such as updating structural finite element models from test data, or identifying unknown parameters[34].

1.3. Overview of the thesis

In light of the above topics and issues, this dissertation is divided into four chapters besides of this short chapter of introduction. Chapter 2 proposes an introduction to important information that is used along the different chapters. We explain the identification problems through energy based functionals. We detail the formulation of their associated constrained optimization problems and how it leads to the solution of a sequence of symmetric saddle point linear systems. Next, we give a brief overview of existing methods for the solution of those systems, where we we give relevant information about direct solvers and Krylov subspace methods. The chapter's main goal is to provide a detailed explanation of the purpose of this thesis.

In Chapter 3, two different strategies using direct solution methods are proposed. The first strategy is devoted to building a variant direct solution method that uses a dynamic process handling factorization and ordering in the same step. This process enables to avoid pivoting and to gain some fill-in especially in the case of indefinite symmetric matrices. While this first strategy is of general purpose, the second one takes more advantage of the special structure and properties of the studied saddle point system. The Gaussian elimination factorization is implemented in order to factorize the saddle point matrices block-wise with small blocks of orders 2 and using a fill-in reducing topological ordering. We will notice through numerical experiments that those strategies remain less efficient in term of memory.

Then, we develop another class of solution methods in Chapter 4. We propose a double projection of the generated saddle point system onto the nullspace of constraints. The first projection onto the kinematic constraints is proposed as an explicit process through the computation of a null basis. Then, we detail the application of a constraint preconditioner as an implicit second projection of the system onto the nullspace of the sensors constraints. We characterize the spectrum of the preconditioned matrix, and we further carry out a comparison of different approximations of the constraint preconditioner.

Chapter 5 puts the latter strategy at work on two problems of industrial relevance, namely on a nuclear power plant alternator and a cooling water pump. Firstly, we illustrate numerical efficiency of the double projection approach when applied to the complex industrial application of the alternator, we solve a sequence of the saddle point systems generated for a set of experimental eigenfrequencies and we evaluate the parallel performance. The second application investigates the use of the solution method when applying the energy-based approach in order to update the pump numerical model.

Finally, we draw final remarks and future research plans in Chapter 6.

Chapter 2

Large-scale identification problems

Contents

2.1. Identification methods

As seen in the industrial context section, it is essential in industry to understand the mechanical behavior of equipments to ensure their structural performance and optimize their availability. Some structures may be exposed to high levels of vibration which requires a deep understanding of the physical phenomena involved and the realization of numerical models to assess the corrective solutions. In many cases, experimental information is combined with numerical simulations to complete the *a priori* knowledge of structural behavior in an effort to address industrial issues. Consequently, numerical models must be of good quality in order to accurately predict the behavior of analyzed structures. In order to achieve a good model prediction, we use the identification of adequate model parameters.

2.1.1. Model parameter identification as an inverse problem

Model parameter identification is a very important and challenging matter in science and technology. This kind of inverse problem arises in many applications.

In direct problems, it is often considered as a general rule to impose boundary conditions either on the primal variable (temperature, displacement, electric potential, ...) as a Dirichlet problem or on the dual one (temperature flux, surface force density, current vector) as a Neumann problem. In the most complicated cases, one can have a combination of conditions as a mixed boundary conditions problem but the number of independent conditions must always be the same. These conditions generally ensure that the direct problem is solvable. Basically, we consider that the solid geometry and its physical characteristics (conductivity, elastic moduli,...) are also known. Conversely and from a physical point of view, an inverse problem is a situation in which we want to evaluate some physical quantities θ that are inaccessible through experimental setting. In order to identify these unknown quantities called parameters, we need to exploit experimental information from another measurable physical quantities d. And using a mathematical model of the direct problem, we get θ explicitly from d (which is symbolically denoted $d = G(\theta)$). The principle of identification methods consists in establishing a mathematical relation based on physical laws, also known as the model, linking both measurable and non-measurable quantities in a way that the sought-after quantities can be found from the available measurements.

The solution of inverse problems may encounter, mathematically speaking, problems of existence, uniqueness and continuity of the solutions [21].The reader may find a general overview on these methods describing general theory and inversion techniques in [106] and [22].

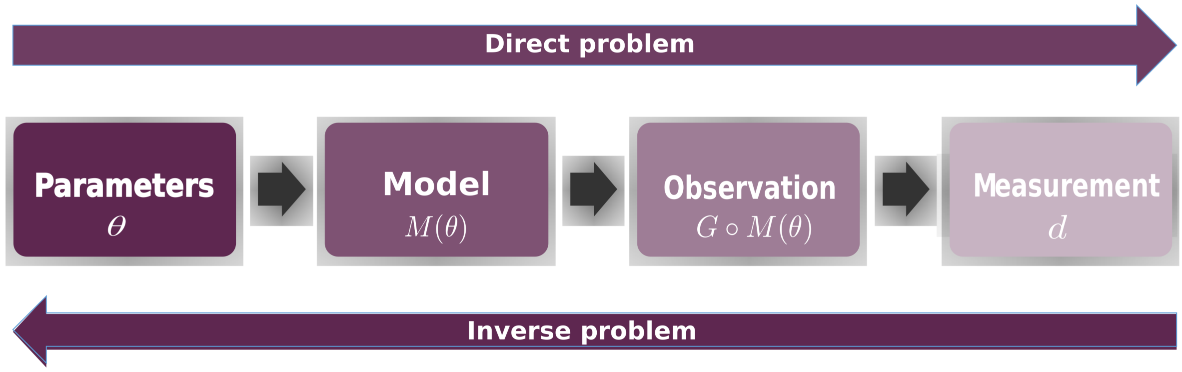


Figure 2.1.: An inverse problem and its associated direct problem

It is usually possible to access in-service structures information and their structural behavior like their frequencies or their modal displacements. These modal data can be calculated by means of a continuous or discrete model, if the structures are perfectly known. Conversely, this experimental modal data makes it possible to cover up a certain degree of ignorance of the system. More concretely, they are used in some industrial applications like

- Model-updating, if the used model is suspected. Due to aging and evolving structures, their functioning is affected and does not comply with the initial design;
- Identification of imperfectly known boundary conditions;
- Monitoring, detection or quantification of defects.

Solving model parameters identification problems depends on the nature of the problem (statics, dynamics, etc.). Parameters identification problems ends up generally being formulated as an optimization problem, namely seeking the minimum of a cost function that quantifies in a certain metrics the difference between a model prediction and the available data. The cost function is built in the literature in different ways. Among the different approaches, there are

- Approaches based on least-squares [107] where the metrics is a L^2 norm.
- An approach based on auxiliary fields, which are fields whose equations of motion admit a single solution. Generally, cost functions are built upon the overdetermined data over the boundary domain [6].
- Approaches consisting on energy-based functionals. An interesting example using this approach within the framework of the Error in Constitutive Relation (ECR) [72] can be found in [34].

In the next section, a more in-depth description of energy-based identification methods is provided. For more information on identification problems for structural mechanics, the reader can find a general overview in [23] and [22], and in particular for industrial applications like modal-updating or fields reconstruction in [75], [5], [14] and [15].

2.1.2. Energy-based functional approaches

This approach has first been introduced by P. Ladevèze [72] in the 1970s as a method that uses an error indicator to check the quality of the solutions obtained by finite element (FE) model. Unlike in the case of the auxiliary fields or the least-squares approaches, where the quality of the model is measured by the distance between the solution of the direct problem to the measured data, the energy-based functionals measure the model error.
As mentioned before, many applications and different versions have been proposed, for instance for model updating [29, 40] and identification problems [55, 85], or model quality assessment [73, 76]. Besides, several studies have successfully applied this principle under linear or nonlinear conditions [14, 29], either in the frequency domain [40] or the time domain [55, 54].
Therefore, the energy-based functional approach appears to be an appropriate indicator of the quality of a model with respect to measured data and some particularly good properties deserve to be highlighted. Indeed, it is able to locate erroneously modeled regions in space [13, 67]. It is robust even in presence of noisy data and provides good convexity properties of cost functions[62].

Thanks to energy based functional approach, we are able to build a model that predict the structural behavior of structures [71, 34]. It is called the hybrid model, and is the combination of the numerical and experimental ones. We integrate the identified experimental data into the numerical model instead of looking at both models separately. An expansion operator is then constructed introducing an additional approximation. More precisely, we seek to extend the specific solutions identified experimentally on the numerical model. This step makes it possible to eliminate the model calibration phase and to ensure that the finite element model has an overall good inertia and stiffness properties.

2.1.3. The constrained optimization problem

In most of the industrial and application cases, and in the particular scope of interest of this work, the study of structural dynamic behavior is performed by means of Finite Element models. Besides, for industrial application cases energy-based functional approach aims to reconstruct fields on a numerical model from experimentally obtained data. In structural dynamics, the kind of data we seek to reconstruct on a FE model are often eigenmodes or displacement fields. The objectives are various as seen in above sections reconstruction for visualization, test assessment, model calibration, damage identification and extrapolation of the delicate degrees of freedom (dofs) that can not be obtained by experiments.

Let us consider a structure and its FE model with n degrees of freedom (dofs), where $[M] \in \mathbb{R}^{n\times n}$ and $[K] \in \mathbb{R}^{n\times n}$ are the so-called mass and stiffness matrices respectively. We know that each couple of eigenvalue and eigenvector (ω, φ) of the finite element numerical model satisfies

$$\left([K] - \omega^2[M]\right)\{\varphi\} = 0, \{\varphi\} \neq 0. \tag{2.1}$$

Nevertheless, the model equations are not correct nor complete to represent the physical equations, due to modeling assumptions, simplifications, misconceptions and possible model errors. Besides there are some model parameters errors, which include the uncertainty on boundary conditions, and inaccuracy on model parameters. These factors imply that the numerical eigencouples may not correspond to the real dynamic behavior of the structure.
We use a set of unknown model parameters θ that parametrize the mass matrix $[M] = [M(\theta)] \in \mathbb{R}^{n\times n}$ and the stiffness matrix $[K] = [K(\theta)] \in \mathbb{R}^{n\times n}$, and consequently couples of eigenvalue and eigenvector $(\omega_\theta, \varphi_\theta)$, in order to modelize these uncertainties and mis-knowledge.
The identification problem aims to find this set of parameters θ such that each couple of numerical eigenvalue and eigenvector $(\omega_\theta, \varphi_\theta)$ is close to the correspondent experimental one $(\omega_{exp}, \phi_{exp})$ where ϕ_{exp} is only defined on $s << n$ sensors.
The energy-based functional is constructed using two fields

- Let $\{\varphi\}$ be the solution field and interpreted as the best estimation of the eigenmode φ_θ, minimizing the distance with the measured eigenmode ϕ_{exp} at the pulsation ω_{exp}, while maintaining the regularity properties of eigenmodes.
- Let $\{\psi\}$ be an error field that expresses the error in stiffness in the model which facilitates identifying the best set of parameters θ that enables a satisfactory reproduction of the measurements through successive iterations. It satisfies

$$[K(\theta)]\{\psi\} = \left([K(\theta)] - \omega_{exp}^2[M(\theta)]\right)\{\varphi\}. \tag{2.2}$$

The energy functional consists of the elastic potential energy of the error term ψ and augmented by the distance between measured and computed eigenmodes

$$e_\omega(\{\varphi\},\{\psi\},\{\theta\}) = \frac{1}{2}\{\psi\}^T[K(\theta)]\{\psi\}+\frac{r}{2(1-r)}(\Pi\{\varphi\}-\{\phi_{exp}\})^T[K_r](\Pi\{\varphi\}-\{\phi_{exp}\}) \quad (2.3)$$

where $r \in [0,1]$ is a weighting scalar, Π is a projection operator from the space of the numerical finite element model to the observation space and $K_r \in \mathbb{R}^{s\times s}$ is a symmetric positive definite scaling matrix. Although the choice of K_r is not *a priori* defined, it is usually chosen to be dimensionally consistent with the induced energy norm and can be obtained, for instance, by using the Guyan reduction on the observation space. More details about K_r are available in [34].

In addition to the constraint (2.2), there are kinematic linear constraints which are described as follows

$$[C]\{\varphi\} = 0, \quad [C]\{\psi\} = 0, \quad (2.4)$$

where $[C] \in \mathbb{R}^{m\times n}$ represents m linear relations coming from the kinematic boundary conditions and modeling constraints as will be detailed in section 2.2.1. It is supposed to be of full row rank m. If it is not the case, we find either that the problem is inconsistent or that some of the constraints are redundant and can be deleted without affecting the solution of the problem. Moreover, the matrix $[K]$ is supposed to be positive definite on $\ker([C])$, which ensures that the constraints lock the rigid body motions of the structure.

An important question to ask is what role experimental measurements play in the energy-based functional in equation (2.3) ?

Before answering this question, let us present some helpful concepts. As seen before, energy-based functional approach is generally based on the minimization of a cost function. In general, this kind of problems leads to the resolution of a sub-determined linear system, and whose solution may be in many cases ill-posed in the sense of Hadamard [30] as it may not respect one or more of these listed conditions

- a solution exists,
- the solution is unique,
- the solution's behavior changes continuously with the initial conditions.

The ill-posedness of the identification problem will generally lead to instability and sensitivity of the solutions with respect to noisy data. To overcome this issue, we use Tikhonov regularization techniques [108, 22], which are widely used when solving inverse problems [106]. It consists on adding, among the set of admissible solutions, what we call regularization conditions. It thus makes possible to introduce an *a priori* knowledge about the sought-after solution, which has the property to stabilize the solution with respect to noise in data.

Hence, to answer the above question, the role of experimental measurements here is to be Tikhonov regularization parameters, which facilitates the reconstruction of unobserved fields. The parameter r makes it possible to control the importance of regularization in the cost function. It represents the confidence that we have in the identified eigenvectors. Actually, the more the coefficient r tends to 1, the more the motion of identified solution degrees of freedom corresponds to the motion of the experimental degrees of freedom. On the contrary, the more r tends to 0, the more the motion of the identified solution degrees of freedom tends towards the motion of the numerical model degrees of freedom. In the many publications dealing with energy-based approaches for identification problems, few provide a real justification for the value of r, and choosing its suitable value is not trivial. Nevertheless, the value of $r = 0.5$ makes the cost function robust with respect to the noise as mentioned in [90, 74]. Here, we choose this value, so that both terms in the cost function (2.3) fulfill a different role during the identification. The term corresponding to the error in stiffness try to highlight the admissible forms while the second term try to obtain the best solution among all admissible ones, in the sense that it minimizes the distance between experimental and numerical model data.

Remark 2.1. *It is also possible to choose a varying scheme for r, as done in [12] where the energy-based approach is used as a cost function for the identification of elastodynamic parameters from experimental measurements. Since the minimization is done in an iterative way, the paper proposes an increasing iterative scheme for r in order to reduce the number of iterations to obtain the optimal identified parameters. This approach may facilitate the solution of the inverse problem due to the effect of the regularization terms that constrain the cost function more and more.*

In order to evaluate the discrepancy of a FE model with respect to a set of measurements, we adopt the following method. Given a set of model parameters θ that parametrize $[M] = [M(\theta)] \in \mathbb{R}^{n\times n}$ and $[K] = [K(\theta)] \in \mathbb{R}^{n\times n}$, we obtain the admissible fields

$$\mathcal{S}_\omega = (\{\hat{\varphi}\}, \{\hat{\psi}\}), \tag{2.5}$$

that minimizes (2.3). This minimization leads to the resolution of linear systems as developed in next section. Finally, we evaluate the model error by computing $e_\omega(\mathcal{S}_\omega, \theta)$.
To find the best set of model parameters θ, since the admissible fields solution of (2.3) depends on θ ($\mathcal{S}_\omega = (\mathcal{S}_\omega(\theta))$, we solve the identification problem over a number of iterations.

The mathematical problem is formally written as

$$\hat{\theta} = arg\min_{\theta\in\Theta} e_\omega(\mathcal{S}_\omega, \theta). \tag{2.6}$$

2.2. Toward a saddle point system

We search a solution that minimize the energy functional (2.3) under the constraints (2.2) and (2.4). In order to solve this constrained optimization problem, a Lagrangian functional is introduced using the following Lagrange multipliers λ, λ_1, and λ_2 and is presented in the following

$$F_\omega\left(\{\varphi\},\{\psi\},\{\lambda\},\{\lambda_1\},\{\lambda_2\},\theta\right) = e_\omega\left(\{\varphi\},\{\psi\},\theta\right) + c_\omega\left(\{\varphi\},\{\psi\},\{\lambda\},\{\lambda_1\},\{\lambda_2\},\theta\right), \tag{2.7}$$

where

$$\begin{aligned} c_\omega(\{\varphi\},\{\psi\},\{\lambda\},\{\lambda_1\},\{\lambda_2\},\theta) = \{\lambda\}^T &\left(\left([K(\theta)] - \omega_{exp}^2[M(\theta)]\right)\{\varphi\} - [K(\theta)]\{\psi\}\right) \\ &- \{\lambda_1\}^T[C]\{\psi\} + \{\lambda_2\}^T\left([C]\{\psi\} - [C]\{\varphi\}\right). \end{aligned} \tag{2.8}$$

The variable θ is considered as implicit here and it will be used afterwards for model updating and robust expansion purposes. The optimal value of $\{\varphi,\psi,\lambda,\lambda_1,\lambda_2\}$ satisfies the stationarity condition

$$dF_\omega\left(\{\varphi\},\{\psi\},\{\lambda\},\{\lambda_1\},\{\lambda_2\},\theta\right) = 0, \qquad \forall\{\varphi,\psi,\lambda,\lambda_1,\lambda_2\} \tag{2.9}$$

$$\frac{\partial F_\omega}{\partial \varphi}d\varphi + \frac{\partial F_\omega}{\partial \psi}d\psi + \frac{\partial F_\omega}{\partial \lambda}d\lambda + \frac{\partial F_\omega}{\partial \lambda_1}d\lambda_1 + \frac{\partial F_\omega}{\partial \lambda_2}d\lambda_2 = 0, \qquad \forall\{\varphi,\psi,\lambda,\lambda_1,\lambda_2\} \tag{2.10}$$

which yields the following system of equations using the symmetry property of $[M]$ and $[K]$

$$\left\{\begin{array}{l} \frac{r}{1-r}\Pi^T[K_r](\Pi\{\varphi\} - \{\phi_{exp}\}) + ([K(\theta)] - \omega_{exp}^2[M(\theta)])\{\lambda\} - [C]^T\{\lambda_2\} = 0 \\ [K(\theta)]\{\psi\} - [K(\theta)]\{\lambda\} + [C]^T\{\lambda_2\} - [C]^T\{\lambda_1\} = 0 \\ -[K(\theta)]\{\psi\} + ([K(\theta)] - \omega_{exp}^2[M(\theta)])\{\varphi\} = 0 \\ [C]\{\psi\} = 0 \\ [C]\{\psi\} - [C]\{\varphi\} = 0 \end{array}\right. \tag{2.11}$$

The third equation of (2.11) allows us to write

$$[K(\theta)]\{\psi\} = ([K(\theta)] - \omega_{exp}^2[M(\theta)])\{\varphi\} \tag{2.12}$$

consequently transforming the second equation in (2.11)

$$([K(\theta)] - \omega_{exp}^2[M(\theta)])\{\varphi\} - [K(\theta)]\{\lambda\} + [C]^T\{\lambda_2\} - [C]^T\{\lambda_1\} = 0 \tag{2.13}$$

Using first, third and fourth equations in (2.11) in addition to (2.13), yields the following system of linear equations

$$\begin{pmatrix} -[K(\theta)] & -[C]^T & [K(\theta)] - \omega_{exp}^2[M(\theta)] & [C]^T \\ -[C] & 0 & [C] & 0 \\ [K(\theta)] - \omega_{exp}^2[M(\theta)] & [C]^T & \frac{r}{1-r}\Pi^T[K_r]\Pi & 0 \\ [C] & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \{\psi\} \\ \{\lambda_1\} \\ \{\varphi\} \\ \{\lambda_2\} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \frac{r}{1-r}\Pi^T[K_r]\phi_{exp} \\ 0 \end{pmatrix} \tag{2.14}$$

This system arises as the first-order optimality conditions for an equality-constrained quadratic programming problem.
The stiffness and mass matrices considered here have different structures from usual ones

$$[\widetilde{K}] = \begin{pmatrix} [K(\theta)] & [C]^T \\ [C] & 0 \end{pmatrix}, \quad [\widetilde{M}] = \begin{pmatrix} [M(\theta)] & 0 \\ 0 & 0 \end{pmatrix} \tag{2.15}$$

with $[K] \in \mathbb{R}^{n\times n}$, $[M] \in \mathbb{R}^{n\times n}$ and $[C] \in \mathbb{R}^{m\times n}$, $m \leq n$. In this case, the matrices $[K]$ and $[M]$ are the stiffness and mass matrices respectively without any constraints. $[K]$ is sparse and symmetric positive semidefinite. The dimension of its nullspace $\ker([K])$ corresponds to the number of rigid body motions of the studied body.
Through this formulation, the constrained mass $[\widetilde{M}]$ and stiffness $[\widetilde{K}]$ operators enable to describe the studied system (2.14) as follows

$$\begin{pmatrix} -[\widetilde{K}(\theta)] & [\widetilde{K}(\theta)] - \omega_{exp}^2[\widetilde{M}(\theta)] \\ [\widetilde{K}(\theta)] - \omega_{exp}^2[\widetilde{M}(\theta)] & \frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\Pi} \end{pmatrix} \begin{pmatrix} \{\widetilde{\psi}\} \\ \{\widetilde{\varphi}\} \end{pmatrix} = \begin{pmatrix} \widetilde{0} \\ \frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\phi}_{exp} \end{pmatrix} \tag{2.16}$$

where the symbol $\sim$ describes the augmented form of each operator and each variable.

Remark 2.2. *The constraint equations are introduced via Lagrange multipliers $\{\lambda\}$, $\{\lambda_1\}$ and $\{\lambda_2\}$. Here, we choose a specific form to describe the kinematic constraints through the Lagrange multipliers which is $-\lambda_1^T C\psi + \lambda_2^T(C\psi - C\varphi)$ instead of the classical form $\lambda_1^T C\psi + \lambda_2^T C\varphi$, in order to get the same structure of the constrained matrix (2.16) as the one generated by the industrial mechanical software Code_Aster ®.*

2.2.1. Different approaches for introducing kinematic constraints

As we study linearly constrained structures, the constraint matrix $[C] \in \mathbb{R}^{m\times n}$ has m linear equations coming from two kinds of constraints

- *single-freedom constraints* (SFCs) they are essential boundary conditions on individual degrees of freedom. For example $u_i = 0$ means that only the node i is constrained, where $\{u\} = (u_i)$ is the vector of physical degrees of freedom (dofs).
- *multifreedom constraints* (MFCs) These are functional equations that connect two or more displacement components. In our case of study only linear constraints will be studied. As usually done in industrial solvers, these relations are introduced directly into the discretized problem. They can model for instance a non-deformable part of the structure or connecting conditions between modelings [83].

The stiffness matrix is assembled ignoring all constraints. these constraints is imposed by changing the assembled stiffness matrix to produce a modified one. The modification process is not unique because there are many constraint imposition methods. These ones offer trade-offs in implementation, computational effort, numerical accuracy and stability. Among these methods, we find in the mechanical software *Code_Aster*® two main ones : elimination and dualization of linear and affine kinematic constraints.

Remark 2.3. *In Code_Aster®, the SFCs can be either directly eliminated or dualized. The MFCs can not be easily eliminated in the matrix $[\widetilde{K}]$. They are added to the discretized problem in Code_Aster®, as in many industrial software. Nevertheless, the imposition of MFCs directly on the discretized unknowns is much easier to implement and much more versatile (it easily mixes displacement and rotation) so that it is real common and powerful modelization tool for the engineers.*

We discuss in the following sections both methods for introducing the linear kinematic constraints and their application within the developed solutions methods in Chapters 3 and 4.

Dualization of the boundary conditions

The dualization method is the one used in section 2.2 by introducing Lagrange multipliers, and also the one that is considered in Chapter 3 to generate the saddle point systems.

This approach is used in *Code_Aster*® and in many other general-purpose finite element programs that it is supposed to work as a black-box by minimizing guesses from its users. It increases the size of the problem by introducing new unknowns through Lagrange multipliers. Physically this set of unknowns represent constraint forces that would enforce the kinematic constraints applied to the unconstrained system. This approach is exact aside from finite precision arithmetic problems, it is extended to nonlinear constraints and we could obtain the constraint forces directly.

The approach has two drawbacks. On the one hand, the adjunction of Lagrange multipliers increases the number of degree of freedom of the whole problem, requiring expansion of the original stiffness and mass matrices, which means more complicated storage allocation procedures. This may be disadvantageous when the number of boundary conditions increases.

On the other hand, it leads to a loss of the positivity property of the stiffness matrix. We recall that the resulting augmented stiffness and mass matrices have different structures from usual ones as mentioned in (2.15). The resulting stiffness matrix becomes indefinite which restrains the use of many factorization methods without permutation especially LDL^T that rely on positive definiteness. In fact, breakdown can occur due to zeros on the diagonal of the (2,2) block in (2.16).

Even though, we will consider this approach when solving the system directly in Chapter 3, as done in *Code_Aster*® software. In this case, the studied linear system is the system 2.16.

Remark 2.4. *To overcome the stiffness matrix indefiniteness issue in Code_Aster®, the stiffness matrix in 2.15 is transformed into an equivalent one, using double Lagrange multipliers [88] as follows*

$$[\widetilde{\widetilde{K}}] = \begin{pmatrix} [K] & \alpha[C]^T & \alpha[C]^T \\ \alpha[C] & -\alpha\mathbb{I} & \alpha\mathbb{I} \\ \alpha[C] & \alpha\mathbb{I} & -\alpha\mathbb{I} \end{pmatrix}, \tag{2.17}$$

where $\alpha > 0$ is a scaling coefficient, chosen in order to obtain coefficients in the matrix $\alpha[C]$ of similar magnitude with the ones of $[K]$. And similarly

$$[\widetilde{\widetilde{M}}] = \begin{pmatrix} [M] & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.18}$$

where $[M]$ is the mass matrix of the unconstrained system.

The saddle point linear system 2.16 becomes as follows

$$\begin{pmatrix} -[\widetilde{\widetilde{K}}(\theta)] & [\widetilde{\widetilde{K}}(\theta)] - \omega_{exp}^2[\widetilde{\widetilde{M}}(\theta)] \\ [\widetilde{\widetilde{K}}(\theta)] - \omega_{exp}^2[\widetilde{\widetilde{M}}(\theta)] & \frac{r}{1-r}\widetilde{\widetilde{\Pi}}^T[\widetilde{\widetilde{K}}_r]\widetilde{\widetilde{\Pi}} \end{pmatrix} \begin{pmatrix} \{\widetilde{\widetilde{\psi}}\} \\ \{\widetilde{\widetilde{\varphi}}\} \end{pmatrix} = \begin{pmatrix} \widetilde{\widetilde{0}} \\ \frac{r}{1-r}\widetilde{\widetilde{\Pi}}^T[\widetilde{\widetilde{K}}_r]\widetilde{\widetilde{\phi}}_{exp} \end{pmatrix}, \qquad (2.19)$$

where the symbol $\approx$ describes the new augmented form of each operator and each variable, and

$$\begin{cases} \frac{r}{1-r}\widetilde{\widetilde{\Pi}}^T[\widetilde{\widetilde{K}}_r]\widetilde{\widetilde{\Pi}} = \begin{pmatrix} \frac{r}{1-r}\Pi^T[K_r]\Pi & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \\ \frac{r}{1-r}\widetilde{\widetilde{\Pi}}^T[\widetilde{\widetilde{K}}_r]\widetilde{\widetilde{\phi}}_{exp} = \begin{pmatrix} \frac{r}{1-r}\Pi^T[K_r]\phi_{exp} \\ 0 \\ 0 \end{pmatrix}, \\ \widetilde{\widetilde{\psi}} = \begin{pmatrix} \{\psi\} \\ \{\lambda_{11}\} \\ \{\lambda_{12}\} \end{pmatrix}, \\ \widetilde{\widetilde{\varphi}} = \begin{pmatrix} \{\varphi\} \\ \{\lambda_{21}\} \\ \{\lambda_{22}\} \end{pmatrix}, \end{cases} \qquad (2.20)$$

where λ_{11}, λ_{12}, λ_{21} and λ_{22} are the associated Lagrange multipliers. In [88], we prove the equivalence of both linear systems 2.16 and 2.19.

Remark 2.5. *If we seek to find the eigencouples*

$$\left(\begin{pmatrix} [K] & \alpha[C]^T & \alpha[C]^T \\ \alpha[C] & -\alpha\mathbb{I} & \alpha\mathbb{I} \\ \alpha[C] & \alpha\mathbb{I} & -\alpha\mathbb{I} \end{pmatrix} - \omega^2 \begin{pmatrix} [M] & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\right) \begin{pmatrix} \varphi \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \qquad (2.21)$$

searching eigenvalues will be more costly because of the loss of the properties of positivity of the operator. Moreover, the spectrum of the problem widens due to the rise in degrees of freedom, which implies different numerical problems to solve [81].

Elimination of boundary conditions

There is a more suitable approach when dealing with fixed and affine boundary conditions. Instead of using the dual form through the Lagrange multipliers to constraint the problem with affine boundary conditions, we eliminate some of the variables from the problem, to obtain a simpler problem with fewer degrees of freedom. We use this technique as a first step in the iterative method developed in Chapter 4 in order to eliminate the kinematic constraints.

We do so by projecting the problem in a more suitable space. Actually, we choose to describe the subspace of feasible points that are solutions of $[C]\{u\} = d$.
Let $Z \in \mathbb{R}^{n\times(n-m)}$ be a matrix such that $\text{span}(Z) = \ker(C)$ and $Y \in \mathbb{R}^{n\times m}$ be a matrix such that $\text{span}(Y) = \text{range}(C^T)$. We now express any solution of the linear constraints as $u = Yu_Y + Zu_Z$ for some vectors $u_Y \in \mathbb{R}^m$ and $u_Z \in \mathbb{R}^{n-m}$. By substitution, we obtain $Cu = (CY)u_Y = d$, hence by nonsingularity of CY we conclude that any vector u of the form $u = Y(CY)^{-1}d + Zu_Z$ satisfies the constraints $Cu = d$. We say that the solution is the sum of a particular one $u_p = Y(CY)^{-1}d$ and a general one $u_g = Zu_Z$. In other words, the elimination technique expresses feasible points as the sum of a particular solution plus a displacement along the null space of the constraints.
If we consider the problem $[K]\{u\} = \{f\}$ under these affine boundary conditions, we obtain a new equivalent problem which is

$$Z^T KZ = Z^T(f - Y(CY)^{-1}d) \tag{2.22}$$

Where $[K_c] = Z^T KZ \in \mathbb{R}^{(n-m)\times(n-m)}$ is the reduced stiffness matrix.

Remark 2.6. *Searching for eigencouples is also simplified in the nullspace of the constraints. The mass and the stiffness matrices* $([M], [K])$ *are assembled in a first step without taking into account the boundary conditions, they describe an unconstrained structure. Then, We project the eigenmodes of the unconstrained problem* $([K] - \omega^2[M])\{\varphi\} = 0$ *to obtain the eigenmodes of the constrained one* $\left(Z^T KZ - \omega^2 Z^T MZ\right)\psi = 0$. *Therefore, the elimination approach seems more natural, and suits better the eigenvalue problem in presence of affine constraints.*

Remark 2.7. *In [88], we prove that the dualized system is equivalent to the reduced, in the condition if we dualize the stiffness matrix and not the mass matrix, which is used within the mechanical code Code_Aster®.*

2.2.2. Sensors constraints and observability of shape modes

Let us recall in the following the remaining constraints applied to the optimization problem. As mentioned above, they are the equality constraints that represent imposed equations at the sensors degrees of freedom

$$[K(\theta)]\{\psi\} = \left([K(\theta)] - \omega^2_{exp}[M(\theta)]\right)\{\varphi\}, \tag{2.23}$$

where $\{\psi\}$ is the error field that expresses the error in stiffness in the model, and $\{\varphi\}$ is the solution field and interpreted as the best estimation of the numerical eigenmode φ_θ, which is minimizing the distance with the measured eigenmode ϕ_{exp} at the experimental pulsation ω_{exp}.

We call the *sensors constraints* here the constraints described by the third line of the system 2.14, which represent sensors degrees of freedom that are constrained through the equation 2.23. We notice that they are dependent of the experimental frequency ω_{exp}, therefore they slowly change for each linear system along the sequence. In section 4.1.2, we present an approach that enables us to eliminate them implicitly.

Here, there is an interest in a better understanding of the structural dynamic behavior, and identifying the modal parameters, namely the natural frequencies, damping ratios and mode shapes. In this context, observability is a notion that plays a major role in the reconstruction of states from inputs and outputs. Together with reachability, observability is central to the understanding of feedback control systems [104].
Let us consider ϕ an eigenmode of the structure. We present in the following a condition on ϕ such that

$$\Pi\phi \neq 0 \tag{2.24}$$

where Π is the projection operator from the space of numerical finite element model to the observation space. If the condition 2.24 holds then the eigenmode ϕ is called *observable*. The observability notion enables by using only information from the output measurements to learn everything about the dynamical behavior.

This notion is mainly used here to describe the experimental configuration of the structure. We see later in Chapter 4, that the studied linear system is invertible if and only if every eigenmode is observable *i.e.* can be seen in at least one output channel.

2.2.3. Saddle-point linear systems

Adopting the above approach in a FE framework leads to a large and sparse linear system which, as recommended by industrial guidelines of this work, is formulated symmetrically. This symmetric formulation generates a sparse and large linear system of equations equivalent to a saddle-point or Karush-Kuhn-Tucker (KKT) system. Such systems arise typically in many applications of scientific computing, including for instance constrained optimization [51], electromagnetism [89], incompressible fluid flow [52] and contact mechanics [83]. Hence, there has been in recent years a growth of interest in saddle point problems, and many solution techniques have been developed. A review of the most known resolution techniques is

found in [17].
In solid mechanics, the most known saddle point linear system is as follows

$$\mathcal{K}x = f \Longleftrightarrow \begin{pmatrix} G & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ \lambda \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix} \tag{2.25}$$

where $G \in \mathbb{R}^{n\times n}$ is symmetric positive semidefinite, $B \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^n$, $d \in \mathbb{R}^m$ and $m \le n$. This symmetric indefinite problem of dimension $N = n + m$ is assumed to belong to the class of large and sparse saddle point problems.
Let us state a theorem related to the nonsingularity of $\mathcal{K}$ which makes sure the existence and uniqueness of the solution.

Theorem 2.8. *Let $\mathcal{K} \in \mathbb{R}^{N\times N}$ be the coefficient matrix in (2.25). Assume that G is symmetric positive semidefinite, B has full row rank m and* $\ker(G) \cap \ker(B) = \{0\}$*. Then $\mathcal{K}$ is nonsingular.*

The proof can be found in [17]. In this manuscript, we focus on the case where the (1,1) block G is also an indefinite saddle point matrix. As we will see later in this section, this property can imply some restrictions about the nonsingularity and the choice of the method used to solve the saddle point system.
According to the previous Section, the initial linear problem to solve (2.16) has a saddle point structure and is described as follows

$$\mathcal{A} = \begin{pmatrix} -[\widetilde{K}(\theta)] & [\widetilde{K}(\theta)] - \omega_{exp}^2[\widetilde{M}(\theta)] \\ [\widetilde{K}(\theta)] - \omega_{exp}^2[\widetilde{M}(\theta)] & \frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\Pi} \end{pmatrix} \tag{2.26}$$

where $\mathcal{A}$ is a $2N \times 2N$ matrix partitioned into 2-by-2 block structure of dimension $N \times N$ each. First, there is the constrained stiffness matrix $[\widetilde{K}(\theta)] = \begin{pmatrix} [K(\theta)] & [C]^T \\ [C] & 0 \end{pmatrix}$, then the constrained impedance $[\widetilde{K}(\theta)] - \omega_{exp}^2[\widetilde{M}(\theta)] = \begin{pmatrix} [K(\theta)] - \omega_{exp}^2[M(\theta)] & [C]^T \\ [C] & 0 \end{pmatrix}$ that shares the same structure as $[\widetilde{K}(\theta)]$.
Finally $\frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\Pi} = \begin{pmatrix} \frac{r}{1-r}\Pi^T[K_r]\Pi & 0 \\ 0 & 0 \end{pmatrix}$ is a very sparse symmetric matrix, it is composed of a dense $c \times c$ sub-block scattered into a $N \times N$ matrix.
In the following and throughout the remainder of the manuscript, we consider the following abbreviations $A = [K(\theta)]$, $B = [K(\theta)] - \omega_{exp}^2[M(\theta)]$ and $T = \frac{r}{1-r}\Pi^T[K_r]\Pi$. The coefficient

matrix $\mathcal{A}$ is then described as follows

$$\mathcal{A} = \begin{pmatrix} -A & -C^T & B^T & C^T \\ -C & 0 & C & 0 \\ B & C^T & T & 0 \\ C & 0 & 0 & 0 \end{pmatrix} \tag{2.27}$$

An other alternative structure is presented below, it is more appealing than (2.27) and is more suited for the developed solution method in Chapter 4 as will be shown later. It represents a saddle point coefficient matrix with a null $(2, 2)$ block. Using the permutation matrix

$$\mathcal{P} = \begin{pmatrix} \mathbb{I}_n & 0 & 0 & 0 \\ 0 & 0 & \mathbb{I}_m & 0 \\ 0 & \mathbb{I}_n & 0 & 0 \\ 0 & 0 & 0 & \mathbb{I}_m \end{pmatrix} \tag{2.28}$$

we get the equivalent saddle-point structure

$$\widetilde{\mathcal{A}} = \mathcal{P}^T \mathcal{A} \mathcal{P} = \begin{pmatrix} -A & B^T & -C^T & C^T \\ B & T & C^T & 0 \\ -C & C & 0 & 0 \\ C & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \widetilde{E} & \widetilde{C}^T \\ \widetilde{C} & 0 \end{pmatrix} \tag{2.29}$$

where $\widetilde{E} = \begin{pmatrix} -A & B^T \\ B & T \end{pmatrix} \in \mathbb{R}^{2n \times 2n}$ and $\widetilde{C} = \begin{pmatrix} -C & C \\ C & 0 \end{pmatrix} \in \mathbb{R}^{2m \times 2n}$. The constraint matrix $\widetilde{C}$ is an augmented incidence matrix that is created from the contraint matrix C of the numerical model.

Remark 2.9. *We present in* (2.30) *a partitioning of the saddle point coefficient matrix* (2.29)*, that takes into account the sensors degrees of freedom in order to highlight their influence*

$$\widetilde{\mathcal{A}} = \begin{pmatrix} -A_{ss} & -A_{st} & B_{ss}^T & B_{ts}^T & -C_{ss}^T & -C_{ts}^T & C_{ss}^T & C_{ts}^T \\ -A_{ts} & -A_{tt} & B_{st}^T & B_{tt}^T & -C_{st}^T & -C_{tt}^T & C_{st}^T & C_{tt}^T \\ B_{ss} & B_{st} & T_{ss} & 0 & C_{ss}^T & C_{ts}^T & 0 & 0 \\ B_{ts} & B_{tt} & 0 & 0 & C_{st}^T & C_{tt}^T & 0 & 0 \\ -C_{ss} & -C_{st} & C_{ss} & C_{st} & 0 & 0 & 0 & 0 \\ -C_{ts} & -C_{tt} & C_{ts} & C_{tt} & 0 & 0 & 0 & 0 \\ C_{ss} & C_{st} & 0 & 0 & 0 & 0 & 0 & 0 \\ C_{ts} & C_{tt} & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{2.30}$$

where

$$C = \begin{pmatrix} C_{ss} & C_{st} \\ C_{ts} & C_{tt} \end{pmatrix}, T = \frac{r}{1-r}\Pi^T[K_r]\Pi = \begin{pmatrix} T_{ss} & 0 \\ 0 & 0 \end{pmatrix},$$

$$A = [K(\theta)] = \begin{pmatrix} A_{ss} & A_{st} \\ A_{ts} & A_{tt} \end{pmatrix}, B = [K(\theta)] - \omega_{exp}^2[M(\theta)] = \begin{pmatrix} B_{ss} & B_{st} \\ B_{ts} & B_{tt} \end{pmatrix}.$$

Subscript s indicates the set of sensors degrees of freedom in the numerical model. Since there are s sensors then $T_{ss} \in \mathbb{R}^{s\times s}$ is a small symmetric positive definite matrix. The equality constraint 2.23 is described through the third block line/column in the coefficient matrix 2.30.

In the next section, we state some results and properties related to the coefficient matrix $\mathcal{A}$.

2.2.4. The coefficient matrix properties

First, here are some important assumptions considered in this manuscript and imposed in the mechanical code *Code_Aster* ®

- $C \in \mathbb{R}^{m\times n}$ is supposed to be of full row rank m, if such is not the case, we find either that the problem is inconsistent or that some of the constraints are redundant and can be deleted without affecting the solution of the problem. Also, if C is of full row rank it is obvious that the augmented constraint matrix $\widetilde{C} \in \mathbb{R}^{2m\times 2n}$ described above in (2.29), is of full row rank $2m$.
- The matrix A in (2.29) is supposed to be positive definite on $\ker(C)$, which ensures that the constraints forbid the rigid body motions of the structure. This hypothesis is verified also if A is only symmetric positive semidefinite and $\ker(A) \cap \ker(C) = \{0\}$.

These assumptions will be supposed to be satisfied from now on.

Solvability conditions of the coefficient matrix block (1,1) $\widetilde{E}$

Let us consider the coefficient matrix (2.29), in the following we present the conditions that guarantee the nonsingularity of $\widetilde{\mathcal{A}}$. We begin by proving the nonsingularity of $\widetilde{E}$ which is the $(1,1)$ block in (2.29).

Theorem 2.10. *Let the matrix* $\widetilde{E} = \begin{pmatrix} -A & B^T \\ B & T \end{pmatrix} \in \mathbb{R}^{2n\times 2n}$ *be as defined in* (2.29)*.* $\widetilde{E}$ *is invertible whenever the intersection of* $\ker(B)$ *and* $\ker(T)$ *is reduced to the null vector.*

Proof. Assume A and B admit the following spectral decomposition

$$\widehat{A} = \Lambda^T A \Lambda = \underset{1\leq j\leq p}{\mathrm{Diag}}(\omega_j^2), \;\; \widehat{B} = \Lambda^T B \Lambda = \underset{1\leq j\leq p}{\mathrm{Diag}}(\omega_j^2 - \omega^2) \tag{2.31}$$

Where $0 \leq \omega_1^2 \leq ... \leq \omega_p^2$ are the p eigenvalues (counting possible multiplicities), and $\Lambda = (\Lambda_1, \Lambda_2, ..., \Lambda_p)$ their correspondent eigenvectors..
Since A is semidefinite then $rank(A) = n - r$ with $0 \leq r \leq 6$ the number of dofs of rigid body motions, which implies the following decomposition

$$\widehat{A} = \begin{pmatrix} 0_{r,r} & 0 \\ 0 & \widehat{A}_{n-r,n-r} \end{pmatrix}, \; \widehat{B} = \begin{pmatrix} -\omega^2 I_{r,r} & 0 \\ 0 & \widehat{B}_{n-r,n-r} \end{pmatrix}, \; \widehat{T} = \begin{pmatrix} \widehat{T}_{r,r} & \widehat{T}_{r,n-r} \\ \widehat{T}_{n-r,r} & \widehat{T}_{n-r,n-r} \end{pmatrix} \tag{2.32}$$

In the following, we will use the symbol $\equiv$ to describe similarity between two matrices. Hence, if $X = P^{-1} Y P$ for some invertible matrix P, then $X \equiv Y$ with P being the change of basis matrix. Using the spectral decomposition on the whole matrix leads to the similar matrix $\overline{E}$

$$\widetilde{E} \equiv \overline{E} = \begin{pmatrix} \Lambda^T & 0 \\ 0 & \Lambda^T \end{pmatrix} \begin{pmatrix} -A & B^T \\ B & T \end{pmatrix} \begin{pmatrix} \Lambda & 0 \\ 0 & \Lambda \end{pmatrix} = \begin{pmatrix} -\Lambda^T A \Lambda & \Lambda^T B^T \Lambda \\ \Lambda^T B \Lambda & \Lambda^T T \Lambda \end{pmatrix} \tag{2.33}$$

$$\overline{E} = \begin{pmatrix} 0_{r.r} & 0 & -\omega^2 I_{r,r} & 0 \\ 0 & \widehat{A}_{n-r,n-r} & 0 & \widehat{B}_{n-r,n-r} \\ -\omega^2 I_{r,r} & 0 & \widehat{T}_{r,r} & \widehat{T}_{r,n-r} \\ 0 & \widehat{B}_{n-r,n-r} & \widehat{T}_{n-r,r} & \widehat{T}_{n-r,n-r} \end{pmatrix} \tag{2.34}$$

Permuting the above matrix leads to the following similar form

$$\overline{E} \equiv \left(\begin{array}{c|ccc} -\widehat{A}_{n-r,n-r} & 0 & \widehat{B}_{n-r,n-r} & 0 \\ \hline 0 & 0_{r,r} & 0 & -\omega^2 I_{r,r} \\ \widehat{B}_{n-r,n-r} & 0 & \widehat{T}_{n-r,n-r} & \widehat{T}_{n-r,r} \\ 0 & -\omega^2 I_{r,r} & \widehat{T}_{r,n-r} & \widehat{T}_{r,r} \end{array}\right) \tag{2.35}$$

This decomposition leads to a saddle point matrix with a $(1,1)$ invertible block. We use here the following LDL^T block decomposition

$$\begin{pmatrix} -A & B_1^T \\ B_2 & C \end{pmatrix} = \begin{pmatrix} \mathbb{I} & 0 \\ B_2 A^{-1} & \mathbb{I} \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & C + B_2 A^{-1} B_1^T \end{pmatrix} \begin{pmatrix} \mathbb{I} & A^{-1} B_1^T \\ 0 & \mathbb{I} \end{pmatrix} \tag{2.36}$$

which enables us to prove the following similarity

$$\begin{pmatrix} -A & B_1^T \\ B_2 & C \end{pmatrix} \equiv \begin{pmatrix} A & 0 \\ 0 & C + B_2 A^{-1} B_1^T \end{pmatrix} \tag{2.37}$$

From this similarity, it is clear that $\overline{A}$ is congruent to the matrix

$$\overline{E} \equiv \left(\begin{array}{c|ccc} -\widehat{A}_{n-r,n-r} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & -\omega^2 I_{r,r} \\ 0 & 0 & \widehat{T}_{n-r,n-r} + \widehat{B}_{n-r,n-r}(\widehat{A}_{n-r,n-r})^{-1}\widehat{B}_{n-r,n-r} & \widehat{T}_{n-r,r} \\ 0 & -\omega^2 I_{r,r} & \widehat{T}_{r,n-r} & \widehat{T}_{r,r} \end{array}\right) \tag{2.38}$$

Permuting this matrix so that the block $(2, 2)$ gets triangular leads to

$$\overline{E} \equiv \left(\begin{array}{c|ccc} -\widehat{A}_{n-r,n-r} & 0 & 0 & 0 \\ \hline 0 & -\omega^2 I_{r,r} & 0 & 0 \\ 0 & \widehat{T}_{n-r,r} & \widehat{T}_{n-r,n-r} + \widehat{B}_{n-r,n-r}(\widehat{A}_{n-r,n-r})^{-1}\widehat{B}_{n-r,n-r} & 0 \\ 0 & \widehat{T}_{r,r} & \widehat{T}_{r,n-r} & -\omega^2 I_{r,r} \end{array}\right) \tag{2.39}$$

Since $-\widehat{A}_{n-r,n-r}$ is symmetric and negative definite, it follows that $\overline{E}$ is invertible if and only if $S = \widehat{T}_{n-r,n-r} + \widehat{B}_{n-r,n-r}(\widehat{A}_{n-r,n-r})^{-1}\widehat{B}_{n-r,n-r}$ is invertible. As $\widehat{B}_{n-r,n-r}(\widehat{A}_{n-r,n-r})^{-1}\widehat{B}_{n-r,n-r}$ and $\widehat{T}_{n-r,n-r}$ are symmetric positive semi-definite matrices, it is obvious that S is a symmetric positive semi-definite matrix. S is singular if and only if $\exists V \neq 0$ such that $V^T S V = 0$. Since for any W, $W^T \widehat{T}_{n-r,n-r} W \geq 0$ and $W^T \widehat{B}_{n-r,n-r}(\widehat{A}_{n-r,n-r})^{-1}\widehat{B}_{n-r,n-r} W \geq 0$ then S is singular if and only if

$$V^T \widehat{T}_{n-r,n-r} V = 0 \text{ and } V^T \widehat{B}_{n-r,n-r}(\widehat{A}_{n-r,n-r})^{-1}\widehat{B}_{n-r,n-r} V = 0 \tag{2.40}$$

which implies that $\widehat{T}_{n-r,n-r} V = 0$ and $(\widehat{A}_{n-r,n-r})^{-1}\widehat{B}_{n-r,n-r} V = 0$ since $\widehat{T}_{n-r,n-r}$ and $(\widehat{A}_{n-r,n-r})^{-1}$ are symmetric positive, then $\widehat{T}_{n-r,n-r} V = 0$ and $\widehat{B}_{n-r,n-r} V = 0$ as $(\widehat{A}_{n-r,n-r})^{-1}$ is invertible. Finally, S is singular if and only if $V \in \ker(\widehat{B}_{n-r,n-r}) \cap \ker(\widehat{T}_{n-r,n-r})$. □

We notice that it suffices that A being positive semidefinite without being invertible to prove the nonsingularity of $\widetilde{E}$.
Thanks to the congruence (2.37) and from Sylvester's Law of Inertia [66, p. 224] we conclude that $\widetilde{E}$ is highly indefinite, with $n - r$ positive and $n + r = n - r + 2s$ negative eigenvalues, where $r = dim(\ker(A))$. The same is of course true if B is rank deficient, as long as S remains positive definite which is true as long as $\ker(B) \cap \ker(T) = 0$.

Solvability conditions of the coefficient matrix $\widetilde{\mathcal{A}}$

In the following theorem, we prove the condition of invertibility of the whole matrix (2.29).

Theorem 2.11. *Let the matrix* $\widetilde{\mathcal{A}} = \begin{pmatrix} \widetilde{E} & \widetilde{C}^T \\ \widetilde{C} & 0 \end{pmatrix} \in R^{2(n+m)\times 2(n+m)}$ *be as defined in* (2.29). *if* $\widetilde{E}$ *is positive definite then* $\widetilde{\mathcal{A}}$ *is invertible. If* $\widetilde{E}$ *is invertible, the nonsingularity of* $\widetilde{\mathcal{A}}$ *is guaranteed iff* $\widetilde{C}\widetilde{E}^{-1}\widetilde{C}^T$ *is nonsingular.*

Proof. $\widetilde{E}$ is invertible, it yields that the saddle point matrix $\widetilde{\mathcal{A}}$ admits the following block triangular factorization

$$\widetilde{\mathcal{A}} = \begin{pmatrix} \mathbb{I} & 0 \\ \widetilde{C}\widetilde{E}^{-1} & \mathbb{I} \end{pmatrix} \begin{pmatrix} \widetilde{E} & 0 \\ 0 & \widetilde{S} \end{pmatrix} \begin{pmatrix} \mathbb{I} & \widetilde{E}^{-1}\widetilde{C}^T \\ 0 & \mathbb{I} \end{pmatrix} \tag{2.41}$$

where $\widetilde{S} = -\widetilde{C}\widetilde{E}^{-1}\widetilde{C}^T$ is the the *Schur complement* of $\widetilde{E}$ in $\widetilde{\mathcal{A}}$. Since the triangular blocks are nonsingular, it follows that $\widetilde{\mathcal{A}}$ is nonsingular iff $\widetilde{S}$ is nonsingular.
Now if $\widetilde{E}$ is positive definite, it is easy to see that $\widetilde{E}^{-1}$ is also positive definite. For a nonzero $x \neq 0$, we have

$$x^T(\widetilde{C}\widetilde{E}^{-1}\widetilde{C}^T)x = (\widetilde{C}^T x)^T \widetilde{E}^{-1}(\widetilde{C}^T x) > 0 \tag{2.42}$$

we recall that $\widetilde{C}^T x \neq 0$ since $\widetilde{C}$ has full row rank. Hence, $\widetilde{S}$ is negative definite and $\widetilde{\mathcal{A}}$ is nonsingular.
It is not sufficient that $\widetilde{E}$ is nonsingular and $\widetilde{C}$ has full rank for $\widetilde{S}$ being nonsingular. Let us consider

$$\widetilde{E} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \widetilde{C} = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

Clearly $\widetilde{S} = 0$ so that $\widetilde{\mathcal{A}}$ is singular.
Besides, as $\widetilde{\mathcal{A}}$ is congruent to $\begin{pmatrix} \widetilde{E} & 0 \\ 0 & \widetilde{S} \end{pmatrix}$, it readily follows that $\widetilde{\mathcal{A}}$ is nonsingular iff $\widetilde{S} = -\widetilde{C}\widetilde{E}^{-1}\widetilde{C}^T$ is invertible. □

The condition considered here to prove the nonsingularity of $\widetilde{\mathcal{A}}$ is algebraic and not practical. We present in section 4.1.1 an alternative way to prove the nonsingularity of $\widetilde{\mathcal{A}}$ by using the nullspace projection, and we prove that the nonsingularity condition depends on the experimental set up, and precisely on the fact that each eigenmode need to be observable.

Coefficient matrix pattern and numerical properties

An important care must be taken when dealing with saddle point systems, as these systems can be poorly conditioned. Their special structure makes them vulnerable to many numerical difficulties. Besides, it can be used to avoid or attenuate the ill-conditioning. Actually, the special structure of the resulting saddle point linear system (2.16) is a difficult challenge especially for mechanical softwares which are more developed for FE-like matrices. The coefficient matrix $\widetilde{\mathcal{A}}$ is real and symmetric and presents several difficulties for mechanical solvers

- It is not positive definite;
- It is poorly conditioned;
- it has a large bandwidth;

Figure 2.2 shows the structure of the non-zero elements of a 60×60 matrix $\widetilde{\mathcal{A}}$ and illustrates its mathematical properties, namely the very large bandwidth besides the very sparse block $(2, 2)$. The second figure (right) presents a similar size finite element matrix.

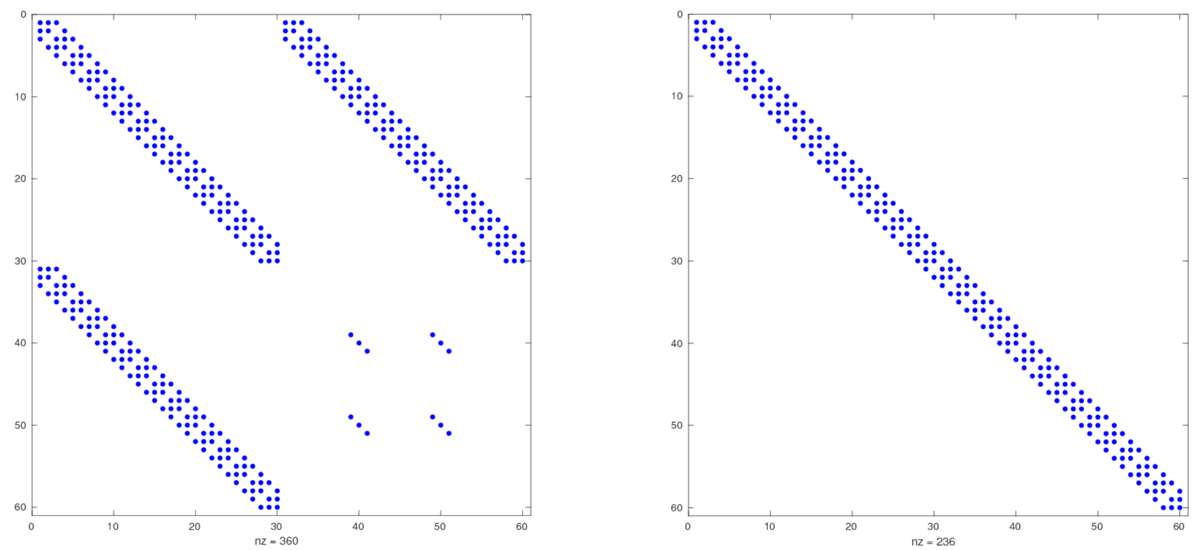


Figure 2.2.: The pattern of non-zero elements of the coefficient matrix (2.27) (left) and a finite element stiffness matrix (right)

Moreover, the structure of the right-hand side in (2.16) also plays a role

$$rhs = \begin{pmatrix} \widetilde{0} \\ \frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\phi}_{exp} \end{pmatrix} \tag{2.43}$$

Here, the right hand side is composed of two sub-blocks. The first block is null and the second one is sparse and its non-zero elements correspond to sensors degrees of freedom. If we consider that

$$\mathcal{A}^{-1} = \begin{pmatrix} G & F^T \\ F & D \end{pmatrix}^{-1} = \begin{pmatrix} G^{-1}(I + F^T S^{-1} F G^{-1}) & -G^{-1}F^T S^{-1} \\ -S^{-1}FG^{-1} & S^{-1} \end{pmatrix} \tag{2.44}$$

where $S = D - FG^{-1}F^T$. Then, it is clear that the $(1,1)$ and $(2,1)$ blocks in $\mathcal{A}^{-1}$ have no influence on the solution $u = \mathcal{A}^{-1}rhs$, so that any ill-conditioning that may be present in these blocks will not affect the solution. For details, see [43, 56, 111, 17].

2.3. Solution methods of saddle point problems

Many existing resolution methods are used to solve the saddle-point problems like (2.16), a review of the most known resolution techniques is found in [17]. The most used solvers are coupled (or global) ones which enables to solve the whole system at once and then to compute the unkowns simultaneously.

2.3.1. Solving symmetric indefinite systems with sparse direct solvers

Direct methods are a tightly-coupled combination of techniques from numerical linear algebra, combinatorics, graph theory and numerical analysis. The direct approach is used as a black box in simulation software. The solutions obtained by these solvers are precise and robust. Direct methods are thus often used in industrial codes where reliability is paramount or for difficult problems. If the efficiency of direct methods is now difficult to surpass for 2D problems, the unknowns of 3D problems are more coupled.

Depending on the topology of the matrix processed, a specific factorization is used. For a positive definite symmetric matrix A, we will use a Cholesky decomposition $A = LL^T$ where L is a lower triangular matrix. For a symmetric indefinite matrix, we prefer a decomposition $A = LDL^T$ with D a block diagonal matrix. And more generally, we will use a decomposition $A = LU$ where U is an upper triangular matrix. For dense and structured matrices, an in-depth coverage of these methods could be found in [33, 35].

Direct solvers are usually implemented with a preprocessing step before the factorization. This includes scaling, pivoting and ordering. The preprocessing step makes the numerical factorization in many cases easier and cheaper, which influences the memory and the CPU time of the factorization step [3]. The preprocessing step highlights two important parameters in the factorization process. First, a low fill-in is sought by using ordering methods on the matrix. Then, stability by preventing division by zero or by small quantities through pivoting strategies. In fact, after the application of an ordering method to reduce fill-in, it may be necessary to use pivoting in order to prevent a numerical breakdown, since pivots may become very small or vanish which impacts stability [105]. We present a simple example of this issue in the following

$$\begin{pmatrix} \epsilon & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{1}{\epsilon} & 1 \end{pmatrix} \begin{pmatrix} \epsilon & 0 \\ 0 & -\frac{1}{\epsilon} \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{\epsilon} \\ 0 & 1 \end{pmatrix}. \tag{2.45}$$

Rows and/or columns are permuted to first have pivotal elements with a large magnitude and then to reorder null or small pivots last in the hope that these entries will be filled before they are chosen as pivots[63]. Many pivot selection techniques have been proposed for symmetric indefinite matrices. Bunch-Parlett [26] is based on complete pivoting and is the most stable pivoting technique. Bunch-Kaufman [25] is based on partial pivoting and may be instable whenever its factor matrix L got unbounded. In addition, there are two others pivoting strategies that are usually referred to as rook's pivoting : bounded Bunch-Kaufman method and fast Bunch-Kaufman [8]. Nevertheless, it is difficult to achieve simultaneously a good fill-in and stability because of complex interplays between ordering (for sparsity) and pivoting (for stability)[10].

In the case of sparse matrices, numerical pivoting restrains a full static prediction of factors pattern: it forces the use of dynamic data structures because it dynamically modifies the structure of the factors, which have a significant impact on the fill-in and on the amount of floating-point operations. To limit the amount of numerical pivoting, and stick better to the sparsity predictions done during the symbolic factorization, partial pivoting can be relaxed, leading to the partial threshold pivoting strategy [3].

The studied linear system (2.16) is symmetric, indefinite and very sparse saddle point system. As Cholesky decomposition requires symmetric positive semidefiniteness, it is not used. Instead, there are only two possible factorization techniques in this case, which are LU and LDL^T decompositions [57]. LU decomposition leads to ignore the matrix symmetry by doing Gaussian elimination with partial or complete pivoting. This decomposition is stable and recommended for the solution of such systems [26]. It is considered in the first direct approach of section 3.2. For symmetric indefinite factorization and in the purpose of preserving symmetry, we use LDL^T where L is lower unit triangular matrix, and D is block diagonal with 1-by-1 and 2-by-2 diagonal blocks. This latter is used partially in the second direct approach developed in section 3.3 to maintain the numerical stability.

Let us consider $A = -[\widetilde{K}(\theta)]$, $B = [\widetilde{K}(\theta)] - \omega_{exp}^2[\widetilde{M}(\theta)]$ and $T = \frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\Pi}$. Since A is an invertible matrix, the saddle point matrix can be factorized in the following block diagonal triangular LDL^T factorization

$$\mathcal{E} = \begin{pmatrix} A & B^T \\ B & T \end{pmatrix} = \begin{pmatrix} \mathbb{I} & 0 \\ BA^{-1} & \mathbb{I} \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & T - BA^{-1}B^T \end{pmatrix} \begin{pmatrix} \mathbb{I} & A^{-1}B^T \\ 0 & \mathbb{I} \end{pmatrix} \tag{2.46}$$

As usually found in classical saddle point systems, if A is symmetric negative (or positive) definite, if B has a full column rank and if T is symmetric positive (or negative) semi-definite, the saddle-point matrix admits the LDL^T factorization with no pivoting is needed [17], and with D diagonal. For instance, using the Cholesky decomposition of $A = -L_A L_A^T$ and $T - BA^{-1}B^T = L_C L_C^T$ in this case

$$\begin{pmatrix} A & B^T \\ B & T \end{pmatrix} = \begin{pmatrix} L_A & 0 \\ BA^{-1}L_A & L_C \end{pmatrix} \begin{pmatrix} -\mathbb{I} & 0 \\ 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} L_A^T & L_A^T A^{-1}B^T \\ 0 & L_C^T \end{pmatrix} = \widetilde{\mathcal{L}}\widetilde{\mathcal{D}}\widetilde{\mathcal{L}}^T \tag{2.47}$$

The singular values of the factor $\widetilde{\mathcal{L}}$, in this case, can be then bounded with the following inequality [95]

$$\kappa(\widetilde{\mathcal{L}}) \leq ||\mathcal{E}||(3||\mathcal{E}^{-1}|| + 2||A^{-1}||). \tag{2.48}$$

Here A is augmented and equal to

$$\begin{pmatrix} -K & -C^T \\ -C & 0 \end{pmatrix} \tag{2.49}$$

which means that A and $T - BA^{-1}B^T$ are not definite. Consequently, they only admit a symmetric LDL^T factorization, and in such factorizations, the inequality 2.48 do not hold. We conclude that the numerical stability comes at the expense of pivoting.

In Chapter 3, we present a global approach that combines factorization and ordering and that avoids pivoting. We present in the same chapter a more in-depth method that takes advantage of the saddle point structure of the coefficient matrix of the linear system 2.16.

Remark 2.12. *Two main direct methods are available in the mechanical code Code_Aster ®[20]. First, The inhouse multifrontal method MULTFRONT which is parallelized in shared memory (OpenMP). This method does not use any pivoting, and a breakdown can occur due to zeros on the diagonal of the (2,2) block in* (2.27). *To overcome this situation, the* *original system is transformed into an equivalent one, using the notion of "double Lagrange" multipliers as described in section 2.2.1. The second one is MUMPS [3] which is a package with a multifrontal approach for solving systems of linear equations. it is designed for square sparse matrix that can be either unsymmetric, symmetric positive definite, or general symmetric. The direct factorization is performed depending on the symmetry of the matrix.*

2.3.2. block factorization approach

This approach is based on the *implicit factorization block preconditioners* which is used to figure out a better application of block preconditioners for saddle point systems by considering the decomposition of the block preconditioner $\mathcal{P} = \mathcal{F}\mathcal{E}\mathcal{F}^T$ where solutions with each of the matrices $\mathcal{E}$ and $\mathcal{F}$ are easily obtained [41]. Then the preconditioner $\mathcal{P}$ is derived implicitly from specially chosen $\mathcal{E}$ and $\mathcal{F}$. Using this same idea in order to produce block factorization can be considered to solve large saddle point systems. We present in this section two different block factorization methods for the studied coefficient matrix.

Before that, let us recall here the second variant of the coefficient matrix structure presented in equation (2.29)

$$\widetilde{\mathbb{A}} = \mathcal{P}\widetilde{\mathcal{A}}\mathcal{P} = \begin{pmatrix} -A & B^T & -C^T & C^T \\ B & T & C^T & 0 \\ -C & C & 0 & 0 \\ C & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \widetilde{E} & \widetilde{C}^T \\ \widetilde{C} & 0 \end{pmatrix}, \tag{2.50}$$

where $\widetilde{E} = \begin{pmatrix} -A & B^T \\ B & T \end{pmatrix} \in \mathbb{R}^{2n\times 2n}$ and $\widetilde{C} = \begin{pmatrix} -C & C \\ C & 0 \end{pmatrix} \in \mathbb{R}^{2m\times 2n}$.
This structure of the coefficient matrix $\widetilde{\mathbb{A}}$ is useful in the framework of the block factorization approach. Actually, it is mainly used with a specific partitioning such that

$$\widetilde{\mathbb{A}} = \begin{pmatrix} \widetilde{E}_{11} & \widetilde{E}_{21}^T & \widetilde{C}_1^T \\ \widetilde{E}_{21} & \widetilde{E}_{22} & \widetilde{C}_2^T \\ \widetilde{C}_1 & \widetilde{C}_2 & 0 \end{pmatrix}, \tag{2.51}$$

where $\widetilde{E}_{11} \in \mathbb{R}^{2m\times 2m}$, $\widetilde{E}_{21} \in \mathbb{R}^{2(n-m)\times 2m}$, $\widetilde{E}_{22} \in \mathbb{R}^{2(n-m)\times 2(n-m)}$, $\widetilde{C}_1 \in \mathbb{R}^{2m\times 2m}$ and $\widetilde{C}_1 \in \mathbb{R}^{2m\times 2(n-m)}$. In this configuration, we assume that $\widetilde{C}_1$ is invertible. This structure enables us to find triangular block factorizations. This is achievable by many existing methods.
The first one uses an algebraic description of the nullspace method in order to factor the

coefficient matrix. It is used in the fields of optimization and known as reduced Hessian methods, structural mechanics and known as "direct elimination"/"force method", electrical engineering and known as "loop analysis", and fluid mechanics and known as the "dual variable" method. Even if these different interpretations of the nullspace method have never been used in the context of matrix factorization, the nullspace method remains a good approach if we want to guarantee the stability of numerical factors and to predetermine the elimination ordering [92].

In this method, we use a new basis, called the fundamental basis relative to the nullspace $\widetilde{Z}$ of the constraint matrix $\widetilde{C}$. We use

- $\widetilde{Z} \in \mathbb{R}^{2n\times 2(n-m)}$ the nullbasis such that $\text{span}(\widetilde{Z}) = \ker(\widetilde{C})$,
- $\widetilde{Y} \in \mathbb{R}^{2n\times 2m}$ such that $\text{span}(\widetilde{Y}) = \text{range}(\widetilde{C}^T)$,

we write the fundamental nullspace basis $\widetilde{\mathbb{N}}$ as :

$$\widetilde{\mathbb{N}} = \begin{pmatrix} \widetilde{Y} & \widetilde{Z} & 0 \\ 0 & 0 & \mathbb{I}_{2m} \end{pmatrix} \tag{2.52}$$

It turns out that

$$\widetilde{\mathbb{N}}^T \widetilde{\mathbb{A}} \widetilde{\mathbb{N}} = \begin{pmatrix} \widetilde{Y}^T \widetilde{E} \widetilde{Y} & \widetilde{Y}^T \widetilde{E} \widetilde{Z} & \widetilde{Y}^T \widetilde{C}^T \\ \widetilde{Z}^T \widetilde{E} \widetilde{Y} & \widetilde{Z}^T \widetilde{E} \widetilde{Z} & 0 \\ \widetilde{C} \widetilde{Y} & 0 & 0 \end{pmatrix} \tag{2.53}$$

then the coefficient matrix $\widetilde{\mathbb{A}}$ is as follows

$$\widetilde{\mathbb{A}} = \widetilde{\mathbb{N}}^{-T} \begin{pmatrix} \widetilde{Y}^T \widetilde{E} \widetilde{Y} & \widetilde{Y}^T \widetilde{E} \widetilde{Z} & \widetilde{Y}^T \widetilde{C}^T \\ \widetilde{Z}^T \widetilde{E} \widetilde{Y} & \widetilde{Z}^T \widetilde{E} \widetilde{Z} & 0 \\ \widetilde{C} \widetilde{Y} & 0 & 0 \end{pmatrix} \widetilde{\mathbb{N}}^{-1} \tag{2.54}$$

From the decomposition 2.54, it is sufficient to build $\widetilde{\mathbb{N}}$ as a triangular matrix, which is for instance achievable by taking $\widetilde{Z}^T = \begin{pmatrix} V & \mathbb{I}_{2(n-m)} \end{pmatrix}$, where $V = -\widetilde{C}_1^{-1} \widetilde{C}_2$ and $\widetilde{Y} = \begin{pmatrix} \mathbb{I} & 0 \end{pmatrix}$ [17]. The factors can be expressed as

$$\begin{pmatrix} \widetilde{Y} & \widetilde{Z} & 0 \\ 0 & 0 & \mathbb{I}_{2m} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbb{I}_{2m} & V & 0 \\ 0 & \mathbb{I}_{2(n-m)} & 0 \\ 0 & 0 & \mathbb{I}_{2m} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbb{I}_{2m} & -V & 0 \\ 0 & \mathbb{I}_{2(n-m)} & 0 \\ 0 & 0 & \mathbb{I}_{2m} \end{pmatrix} \tag{2.55}$$

The second method is the Schilders factorization [80]. It was originally derived by considering models for electronic circuits. This decomposition is given by :

$$\widetilde{\mathbb{A}} = \begin{pmatrix} \widetilde{E}_{11} & \widetilde{E}_{21}^T & \widetilde{C}_1^T \\ \widetilde{E}_{21} & \widetilde{E}_{22} & \widetilde{C}_2^T \\ \widetilde{C}_1 & \widetilde{C}_2 & 0 \end{pmatrix} = \begin{pmatrix} \widetilde{C}_1^T & 0 & \widetilde{J}_1 \\ \widetilde{C}_2^T & \widetilde{J}_2 & \widetilde{U} \\ 0 & 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} \widetilde{L}_1 & 0 & \mathbb{I} \\ 0 & \widetilde{L}_2 & 0 \\ \mathbb{I} & 0 & 0 \end{pmatrix} \begin{pmatrix} \widetilde{C}_1 & \widetilde{C}_2 & 0 \\ 0 & \widetilde{J}_2^T & 0 \\ \widetilde{J}_1^T & \widetilde{U}^T & \mathbb{I} \end{pmatrix} \tag{2.56}$$

where $\widetilde{L}_1 \in \mathbb{R}^{2m\times 2m}$, $\widetilde{L}_2 \in \mathbb{R}^{2(n-m)\times 2(n-m)}$ nonsingular, $\widetilde{J}_1 \in \mathbb{R}^{2m\times 2m}$, $\widetilde{J}_2 \in \mathbb{R}^{2(n-m)\times 2(n-m)}$ nonsingular, $\widetilde{U} \in \mathbb{R}^{2(n-m)\times 2m}$ and

$$\begin{aligned} \widetilde{E}_{11} &= \widetilde{J}_1\widetilde{C}_1 + \widetilde{C}_1^T\widetilde{J}_1^T + \widetilde{C}_1^T\widetilde{J}_1\widetilde{C}_1 \\ \widetilde{E}_{21} &= \widetilde{U}\widetilde{C}_1 + \widetilde{C}_2^T\widetilde{J}_1^T + \widetilde{C}_2^T\widetilde{L}_1\widetilde{C}_2 \\ \widetilde{E}_{22} &= \widetilde{J}_2\widetilde{L}_2\widetilde{J}_2^T + \widetilde{U}\widetilde{C}_2 + \widetilde{C}_2^T\widetilde{U}^T + \widetilde{C}_2^T\widetilde{L}_1\widetilde{C}_2 \end{aligned}$$

Since we search for an exact decomposition, we need to define implicitly $\widetilde{\mathbb{A}}$ by the choices of $\widetilde{L}_i$ and $\widetilde{J}_i$. One possible choice as mentioned in [92] is :

$$\widetilde{L}_1 = -\widetilde{C}_1^{-T}\widetilde{D}_A\widetilde{C}_1^{-1}, \qquad \widetilde{J}_1 = \widetilde{L}_A\widetilde{C}_1^{-1}$$

where $\widetilde{L}_A$ and $\widetilde{D}_A$ the strictly lower triangular part and the diagonal part of $\widetilde{E}_{11}$. Through this and precedent equations, we get $\widetilde{U}$, $\widetilde{L}_2$ and $\widetilde{J}_2$.

We present in section 3.3 a direct solution method that exploits the special saddle point structure of the coefficient matrix using a sparse 2-by-2 block factorization. We then compare it numerically to existing direct solvers.

2.3.3. Krylov subspace methods

The iterative methods are the most used when we treat a large and sparse problem. They use an initial guess to generate successive approximations to a solution. There are many iterative methods in the literature like stationary iterations or Krylov subspace methods. we present in this section those latter for solving saddle point problems. Rather than discussing all existing methods and implementations, we will describe the main properties of the most commonly used methods.

Krylov subspace Theory

Suppose we have the following system to solve

$$\mathcal{A}x = b \tag{2.57}$$

If we consider x_0 the initial guess for the solution x and define the initial residual to be $r_0 = b - \mathcal{A}x_0$. Krylov subspace is constructed as an iterative process whose kth iterate x_k satisfies

$$x_k \in x_0 + \mathcal{K}_k(\mathcal{A}, r_0), \quad \forall k, 1 \leq k \leq n \tag{2.58}$$

where $\mathcal{K}_k(\mathcal{A}, r_0)$ is the kth Krylov subspace generated by $\mathcal{A}$ and r_0 and equals to

$$\mathcal{K}_k(\mathcal{A}, x_0) \equiv \text{span}\{r_0, \mathcal{A}r_0, ..., \mathcal{A}^{k-1}r_0\} \tag{2.59}$$

The starting idea of the Krylov subspace methods comes from the Cayley-Hamilton theorem that proves that the inverse of a matrix can be expressed as a linear combination of its powers [97]. Krylov subspace methods involve finding an "optimal" solution in a given space, augmenting the space, and repeating the procedure.

Conjugate gradient method

The conjugate gradient (CG) method is one of the well known iterative techniques for solving sparse symmetric positive definite linear systems. The method converges to the solution via the minimization of the $\mathcal{A}$-norm of the error as the Krylov subspace is increased at each step [65].

Theoretically the method could take at most n steps to calculate the exact solution if $A \in \mathbb{R}^{n \times n}$. However, in practice, convergence to acceptable accuracy often occurs after only a few steps [93]. The conjugate gradient method uses a 3-term recurrence relation, so as we increase the subspace from which we seek a solution, we need only recall the approximations from the two most recent subspaces to produce the approximation x_k that minimizes the norm of the error at the kth step $e_k = u - u_k$. We present in the following the CG algorithm

Algorithm 2.1: Conjugate Gradient Method

Choose x_0.
Set $r_0 = b - \mathcal{A}x_0$ and $p_0 = r_0$.
for $k \leftarrow 0$ **to** .. **do**
    $\alpha_k = \frac{\langle r_k, r_k \rangle}{\langle \mathcal{A}p_k, p_k \rangle}$,
    $u_{k+1} = u_k + \alpha_k p_k$,
    $r_{k+1} = r_k - \alpha_k \mathcal{A} p_k$,
    $\beta_k = \frac{\langle r_{k+1}, r_{k+1} \rangle}{\langle r_k, r_k \rangle}$,
    $p_{k+1} = r_{k+1} + \beta_k p_k$.

In order to prevent any possible breakdown in the calculation of α_k and β_k in Algorithm 2.1, we need that the matrices $\mathcal{A}$ be symmetric positive definite. Indeed, the vector sequences in the Conjugate Gradient method correspond to a factorization of a tridiagonal matrix similar to the coefficient matrix. Therefore, a breakdown of the algorithm can occur corresponding to a zero pivot if the matrix is indefinite. Furthermore, for indefinite matrices the minimization property of the Conjugate Gradient method is no longer well-defined. If the coefficient matrix is symmetric but indefinite, then we could use the MINRES or SYMMLQ algorithms [87].

The MINRES and SYMMLQ methods are variants of the CG method that avoid the *LU* factorization and do not suffer from breakdown. MINRES minimizes the 2-norm of the residual in Krylov subspaces of increasing dimension instead of minimizing the $\mathcal{A}$-norm of the error. SYMMLQ solves the projected system, but does not minimize anything (it keeps the residual orthogonal to all previous ones). It is based on the LQ factorization of the tridiagonal matrices formed in the Lanczos method. Though, those methods can be less robust and more vulnerable to rounding errors, in this case we can use GMRES. Actually, when the problem is symmetric, GMRES and MINRES do the same calculations in exact arithmetic, but GMRES tends to suffer less from loss of orthogonality. We can describe GMRES as the best implementation of MINRES [97].

Generalized Minimum Residual Method (GMRES)

When the coefficient matrix is symmetric but indefinite, it is possible to find an approximation in a particular subspace which minimizes the 2-norm of the residual. The Generalized Minimum Residual (GMRES) Method [98] is a robust algorithm that do that. It generates an orthogonal basis for the Krylov subspace via the Arnoldi method mentioned in Algorithm 2.2.

Algorithm 2.2: Arnoldi Method

Given v_1 such that $||v_1|| = 1$.
for $i \leftarrow 1$ **to** .. **do**
  $\widetilde{v}_{i+1} = \mathcal{A} v_i$,
  for $j \leftarrow 1$ **to** i **do**
    $h_{ij} = \langle \widetilde{v}_{i+1}, v_j \rangle$,
    $\widetilde{v}_{i+1} = \widetilde{v}_{i+1} - h_{ij} v_j$,
  $h_{i+1,i} = ||\widetilde{v}_{i+1}||$,
  $v_{i+1} = \frac{\widetilde{v}_{i+1}}{h_{i+1,i}}$.

After making use of the Arnoldi process, we construct the GMRES method presented in

Algorithm 2.3.

Algorithm 2.3: The GMRES algorithm

Choose x_0.
Set $r_0 = b - \mathcal{A}x_0$.
Set $v_1 = \frac{r_0}{||r_0||}$.
for $k \leftarrow 0$ **to** .. **do**
  ◇ Compute v_{k+1} and $h_{i,k}$, $\forall i = 1, 2, ..., k+1$ using Arnoldi method,
    ie. Compute V_{k+1} *and* $H_{k+1,k}$ *such that* $AV_k = V_{k+1}H_{k+1,k}$,
  ◇ Solve the least squares problem $y_k = \min\limits_{y \in R^k} ||\beta e_1 - H_{k+1,k}y||$,
  ◇ Set $x_k = x_0 + V_k y_k$

In term of convergence, we notice that if $k < n$ exists such that $\mathcal{K}_k(\mathcal{A}, r_0) = \mathcal{K}_{k+1}(\mathcal{A}, r_0)$, then $x_k = x = \mathcal{A}^{-1}b$ in exact real arithmetic [61], which stops the process. Otherwise, it yields $\mathcal{K}_k(\mathcal{A}, r_0) = \mathbb{R}^n$ which means that x_n is obviously the expected solution. Moreover in the case of the GMRES method we have a bound for the residual of the kth iteration given by

$$||r_k||_2 \leq ||r_0||_2 \min_{q \in \mathbb{P}_k} ||q(\mathcal{A})||_2 \tag{2.60}$$

where $\mathbb{P}_k = \{q \mid q \text{ is a polynomial of degree at most } k \text{ with } q(0) = 1\}$.

In the GMRES method, the convergence behavior is described by the spectrum in the symmetric case. Actually it is influenced namely by the minimization of a polynomial over the set of eigenvalues of the matrix, which is proved for instance in [98]. In the unsymmetric case, although the spectrum is not a sufficient parameter to characterize the convergence behavior of GMRES [60], nevertheless the existence of clustered eigenvalues contributes to speedup the convergence as shown in [17, 27, 101].

When solving large-scale linear systems of size n, it is possible that a large number of iterations may be necessary to obtain an acceptable solution, which is prohibitive in terms of memory requirement. Indeed, at each iteration, an additional basis vector of the Krylov subspace need to be stored. Besides when the number of iterations increases, then the dimension of the the least-squares problem goes up as well. There is a restarted version of GMRES, in which we choose to restart the method after each m steps [98]. This method, called GMRES(m), restarts with a vector x_0 equal to the last computed iterate x_m, which limits the the memory requirements.

2.3.4. Preconditioning

It is usually conceivable to transform the original system $\mathcal{A}x = b$ in a new one, maintaining the same solution but getting more favorable properties for the con vergence of iterative methods. Generally, the rate of convergence is accelerated when many clusters appear away from 0 [27]. There is different ways to transform the original problem using a nonsingular preconditioner $\mathcal{M}$

- left preconditioning

$$\mathcal{M}\mathcal{A}x = \mathcal{M}b \tag{2.61}$$

- right preconditioning

$$\begin{cases} \mathcal{A}\mathcal{M}\hat{x} = & b \\ \quad x & = \mathcal{M}\hat{x} \end{cases} \tag{2.62}$$

- split preconditioning using $\mathcal{M} = \mathcal{M}_1\mathcal{M}_2$

$$\begin{cases} \mathcal{M}_1\mathcal{A}\mathcal{M}_2\hat{x} = & \mathcal{M}_1 b \\ \quad x & = \mathcal{M}_2\hat{x} \end{cases} \tag{2.63}$$

It is worth mentioning that the number of iterations of the Krylov subspace method is generally different if $\mathcal{M}$ is used as a left, right or split preconditioner, even though the spectrum of the associated preconditioned matrices are identical. In particular, the stopping criterion is evaluated with $\mathcal{A}$ replaced by the left-preconditioned operator $\mathcal{M}^{-1}\mathcal{A}$ and the preconditioned residual $\mathcal{M}^{-1}(\mathcal{A}x_k - b)$ or with the right-preconditioned operator $\mathcal{A}\mathcal{M}^{-1}$ and the error $\mathcal{M}(x_k - x)$. If $\mathcal{M}^{-1}$ is a good preconditioner, the preconditioned operator will be well-conditioned. For left-preconditioning, this means the preconditioned residual can be made small, but the true residual may not be. For right preconditioning, $||\mathcal{M}(x_k - x)||_2$ is easily made small, but the true error $||x_k - x||$ may not be. This explains why left-preconditioning is better for making error small while right-preconditioning is better for making the residual small and for debugging unstable preconditioners. Besides, right preconditioning can be attractive, since the preconditioned residual is equal to the original one.

General-purpose preconditioners

Since preconditioners play a very important role in the convergence of iterative methods, it is an active domain of research (see e.g. [17, 16, 109]). Many different preconditioning strategies can be applied to a system. A trivial example of a very simple preconditioner can be obtained using, for instance, $\mathcal{M} = \text{diag}(\mathcal{A})$. The following classes of algebraic preconditioners can be distinguished

- **Stationary iterative methods** (Chapters 4 and 10 in [97]) the oldest methods employed to solve linear systems. Even if they are supplanted by more sophisticated methods, they remain in action as simple preconditioners. The main ones are the Richardson method, the Jacobi method, the Gauss-Seidel method, the Successive Overrelaxation (SOR) method and the Symmetric Successive Overrelaxation (SSOR) method.

- **Incomplete factorization (ILU)** (Sections 10.3 and 10.4 in [97]) They are first Introduced separately around 1960 [24]. The theoretical existence of incomplete factorization can be verified for some classes of matrices [82], nevertheless it may fail for other ones. The principle of an incomplete factorization is to limit the fill-in during the factorization by ignoring some factors entries, in order to get a good approximation of the matrix, at least cost in terms of time and memory. Such a factorization can be used as a preconditioner of an iterative method, in the form $\mathcal{M} = \widetilde{L}\widetilde{U}$ where $\widetilde{L}$ and $\widetilde{L}$ are the approximate factors of the factorization of $\mathcal{A}$. The incomplete factorization method generally use two kinds of criteria

  - The position of different entries in the matrix,
  - The numerical values of the different entries.

  It is of course possible to combine these two criteria respectively lower and upper triangular matrices. The main issue with this method, concerns the fill-in of the factorization, as the factors can be much more dense than the original matrix. There is then recommended to use ordering techniques in the same way as in direct methods.

- **Sparse approximate inverses** (Section 10.5 in [97]) These methods focus on finding a sparse matrix $\mathcal{M}$ that approximates the inverse $\mathcal{A}^{-1}$ under the condition

$$\mathcal{M} = \underset{\mathcal{M}\in\mathcal{T}}{\arg\min} \, ||\mathbb{I} - \mathcal{A}\mathcal{M}||_F \tag{2.64}$$

  where $||.||_F$ is the Frobenius norm and $\mathcal{T}$ is a sparsity pattern to impose. The advantage of this type of preconditioners compared to incomplete factorizations is to be more stable numerically and easier to parallelize. Nevertheless, it remains very expansive in terms of computational time. The study of these preconditioners is beyond the scope of this state of the art.

- **Algebraic Multigrid (AMG)** (Section 13.6 in [97]) Relaxation schemes, such as the Gauss-Seidel or the Jacobi method, efficiently damp high frequency errors, however they make no progress towards reducing low frequency errors. The main idea of multi-

grid methods is to move the problem to a coarser grid so that previously low frequency errors turn now into high frequency errors and can be damped efficiently. If we apply this procedure recursively, we obtain a method with a computational cost that depends only linearly on the problem size.
Contrary to physics-based geometric multigrid approach, where the geometry of the problem is used to define the various multigrid components, the algebraic multigrid (AMG) methods use only the information available in the linear system of equations and are therefore suitable to solve problems on more complicated domains and unstructured grids.

Remark 2.13. *Different techniques are accessible in Code_Aster ®. We can find an in-house preconditioned conjugate gradient method (PCG), used for both symmetric positive definite and indefinite linear systems. Besides several Krylov subspace methods such as GMRES, are performed with the PETSc library [11].*

Some preconditioning techniques for saddle point problems

Recently, a large amount of work has been devoted to developing effective preconditioners to enhance iterative solution methods for large symmetric linear systems in saddle point forms which are mostly special cases of :

$$\begin{pmatrix} A & B^T \\ B & -D \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{2.65}$$

where $A \in \mathbb{R}^{n\times n}$ nonsingular, $B \in \mathbb{R}^{m\times n}, m \leq n$ and $D \in \mathbb{R}^{n\times n}$. We present mainly two important classes : block diagonal/triangular preconditioners and constraint preconditioners.

- **Block preconditioners** They are based explicitly on the block factorization

$$\begin{pmatrix} A & B^T \\ B & -D \end{pmatrix} = \begin{pmatrix} \mathbb{I} & 0 \\ BA^{-1} & \mathbb{I} \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} \mathbb{I} & A^{-1}B^T \\ 0 & \mathbb{I} \end{pmatrix} \tag{2.66}$$

where $S = -(D + BA^{-1}B^T)$ is the Shur complement. Their performance depends on the existence of efficient approximations to A and S [96].
Assuming that A and $-S$ are both symmetric positive definite, the essential diagonal preconditioner is

$$\mathcal{P}_{diag} = \begin{pmatrix} A & 0 \\ 0 & -S \end{pmatrix} \tag{2.67}$$

More details about this preconditioner can be found in [84].
Similarly, the essentially block triangular preconditioner is

$$\mathcal{P}_{triang} = \begin{pmatrix} A & B^T \\ 0 & \pm S \end{pmatrix} \tag{2.68}$$

Choosing the minus sign in $\mathcal{P}_{triang}$ results in a diagonalizable preconditioned matrix with only two distinct eigenvalues equal to 1. Choosing the plus sign yields a preconditioned matrix with all the eigenvalues equal to 1. For either choice of the sign, GMRES is guaranteed to converge in at most two steps in exact arithmetic [102]. In practice, A and S are replaced by some appropriate approximations.

- **Constraint preconditioners** ([28, 79]) They have the general form

$$\mathcal{P}_{constr} = \begin{pmatrix} G & B^T \\ B & 0 \end{pmatrix} \tag{2.69}$$

where $G \in \mathbb{R}^{n \times n}$. Their structure is also of saddle point form with the same constraints as the original problem. The constraint preconditioner projects the problem onto the null space of the constraint matrix B which explains why this preconditioning technique is closely related to the null space method. Using an iterative method with constraint preconditioning or using the nullspace method are, in fact, mathematically equivalent [58].
Keller et al. [28] investigated the use of the constraint preconditioner on saddle point problems without $(2, 2)$ block, while Dollar [42] extended the constraint preconditioner to regularized symmetric saddle-point problems. We shall adapt these results to suit our case in section 4.2.

For a more extensive survey of these and other techniques, see [17].

2.3.5. Segregated solvers

Both direct solvers based on triangular factorizations of the global matrix, and iterative algorithms like Krylov subspace methods applied to the entire system, typically with some form of preconditioning, are entitled coupled methods. These solvers deal with the system (2.16) as a whole, computing $\{\widetilde{\psi}\}$ and $\{\widetilde{\varphi}\}$ simultaneously and without making explicit use of reduced systems. Besides coupled solvers, there are segregated ones. We present here the

Schur complement reduction. We recall again the linear system

$$\begin{pmatrix} \widetilde{E} & \widetilde{C}^T \\ \widetilde{C} & 0 \end{pmatrix} \begin{pmatrix} x \\ w \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{2.70}$$

where $\widetilde{E} \in \mathbb{R}^{2n \times 2n}$ is symmetric positive definite and $\widetilde{C} \in \mathbb{R}^{2m \times 2m}$ is of full row rank $2m$, $f \in \mathbb{R}^{2n}$. This saddle point system can also be written as

$$\begin{cases} \widetilde{E}x + \widetilde{C}^T w = f \\ \widetilde{C}x = 0 \end{cases} \tag{2.71}$$

The idea is to multiply the first equation in (2.71) by $\widetilde{C}\widetilde{E}^{-1}$ and to subtract the second relation to deduce the equation satisfied by w

$$\widetilde{C}\widetilde{E}^{-1}\widetilde{C}^T x = \widetilde{C}\widetilde{E}^{-1} f \tag{2.72}$$

The matrix $\widetilde{S} = -\widetilde{C}\widetilde{E}^{-1}\widetilde{C}^T$ is known as the Schur complement of the saddle point system. Once the solution w^* has been computed, x will finally satisfy the equation

$$\widetilde{E}x = f - \widetilde{C}^T w^* \tag{2.73}$$

There is an other well known segregated solver which is the nullspace method. Considering the fundamental basis 2.52 by taking $\widetilde{Y} = \widetilde{C}^T$, we transform the initial system (2.71) to a new one

$$\begin{pmatrix} \widetilde{C}\widetilde{E}\widetilde{C}^T & \widetilde{C}\widetilde{E}\widetilde{Z} & \widetilde{C}\widetilde{C}^T \\ \widetilde{Z}^T\widetilde{E}\widetilde{C}^T & \widetilde{Z}^T\widetilde{E}\widetilde{Z} & 0 \\ \widetilde{C}\widetilde{C}^T & 0 & 0 \end{pmatrix} \begin{pmatrix} x_Y \\ x_Z \\ w \end{pmatrix} = \begin{pmatrix} \widetilde{C}f \\ \widetilde{Z}^T f \\ 0 \end{pmatrix} \tag{2.74}$$

It is obvious that x_Y is determined by the following $2m \times 2m$ system

$$\widetilde{C}\widetilde{C}^T x_Y = 0 \tag{2.75}$$

As recalled above, it is clear here that $x_Y = 0$ since $\widetilde{C}$ is of full row rank so that $\widetilde{C}\widetilde{C}^T$ is symmetric and positive definite. In case if the right hand side of (2.75) is not null, we could solve the $2m \times 2m$ system by Cholesky factorization if $2m$ is small enough otherwise by using an effective iterative method the conjugate gradient (CG).
Since $x_Y = 0$, we could find x_Z by solving the following $2(n-m) \times 2(n-m)$ system

$$\widetilde{Z}^T\widetilde{E}\widetilde{Z}x_Z = \widetilde{Z}^T f \tag{2.76}$$

It is possible to solve the system by computing a factorization when $2(n-m)$ is small. For problems where $2(n-m)$ is large, we use suitable iterative methods to find an approximate solution to (2.76). Once the solution x_Z is found, the solution set is then described by $x = \widetilde{Z}x_Z$.
Finally w can be obtained by solving

$$\widetilde{C}\widetilde{C}^T w = \widetilde{C}(f - \widetilde{E}\widetilde{Z}x_Z) \tag{2.77}$$

which is a reduced system of order $2m$ with the same symmetric positive definite coefficient matrix $\widetilde{C}\widetilde{C}^T$ as (2.75). It is the normal equation for the overdetermined system $\widetilde{C}^T w = f - \widetilde{E}\widetilde{Z}x_Z$. It is possible to use the same Cholesky factorization of the (2.75).
This method will be used in the chapter 4 and enhanced in order to be an adequate strategy to solve the presented saddle point problem.

Remark 2.14. *Even when $\widetilde{E}$ is singular, the nullspace method is applicable which not the case for Shur complement method. More generally, if $\widetilde{H}$ is the symmetric part of $\widetilde{E}$, the method is applicable as long as the condition* $\ker(\widetilde{H}) \cap \ker(\widetilde{C}) = 0$ *is satisfied.*

2.4. Conclusion

In this chapter, we have carried out a presentation of the physical and mathematical formulation coming from parameters identification problems. Energy-based functional approaches present specific features that make them particularly attractive when dealing with an identification problem in structural mechanics good convexity of cost functionals, ability to localize model errors in space, and robustness in the presence of of noisy data. For these reasons, they are adopted in this work as a tool to enhance the a priori model error knowledge.

We have also provided a review of sparse direct methods dedicated to symmetric indefinite matrix in order to highlight the difficulties we can encounter when solving our problem. We have focused then on the solution of saddle point systems by iterative methods based on Krylov subspace, and some attractive preconditioners have been detailed. As a matter of fact, the preconditioning tends to blur the distinction between direct and iterative solvers, and also that between segregated and coupled schemes. Many direct methods are used as preconditioners, and also many preconditioners used with coupled iterative schemes are frequently based on segregated methods.

The next chapter investigates the direct solution methods to solve the saddle point problem by considering it firstly as a general symmetric indefinite matrix and then by taking advantage of its special structure.

Chapter 3

Sparse block-wise factorization for saddle point matrices

Contents

3.1. Direct solvers for large saddle point systems generated by the energy functional approach

One of the main computational challenges that arises in energy-based approach is obtaining a solution for the generated coupled system :

$$\begin{pmatrix} -[\widetilde{K}(\theta)] & [\widetilde{K}(\theta)] - \omega_{exp}^2[\widetilde{M}(\theta)] \\ [\widetilde{K}(\theta)] - \omega_{exp}^2[\widetilde{M}(\theta)] & \frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\Pi} \end{pmatrix} \begin{pmatrix} \{\widetilde{\psi}\} \\ \{\widetilde{\varphi}\} \end{pmatrix} = \begin{pmatrix} \widetilde{0} \\ \frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\phi}_{exp} \end{pmatrix} \tag{3.1}$$

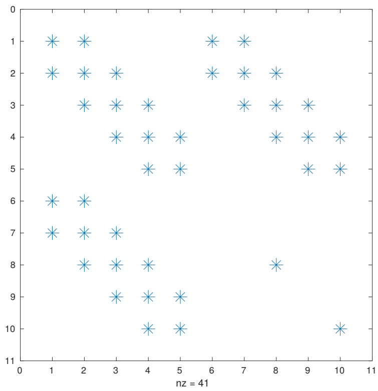
This is achievable employing direct solvers on the whole system or using a segregated strategy. Applying direct linear solvers to (3.1) for large scale problems may be prohibitive. We use the symbol $\widetilde{\mathcal{A}}$ to describe the coefficient matrix of the studied linear system throughout this chapter.
We present in the following some mechanical direct solvers for saddle point linear systems generated by the energy-based approach.

3.1.1. Mechanical direct solvers for the energy functional approach

The energy-based functional approach is rarely implemented, mainly because the repeated use of this approach for model updating or robust expansion applications, leads then to a huge computational cost [53].

This computational cost is due to the special structure of the resulting linear system (3.1). Indeed, this is a difficult challenge especially for mechanical softwares which are more than often used for FE-like matrices. These softwares using direct methods and factorization, are usually preferred for their robustness and the moderate storage requirement of mechanical problems. Though, when applied on these symmetric indefinite matrices, they are inefficient due to a significant growth factor and a high fill-in. The implementation of the energy-based functional approach within the framework of robust expansion shows that direct solvers used in mechanical softwares fail to solve efficiently the inverse problem [71]. In fact, it is shown that for an industrial structure model with more than 10^6 degrees of freedom and few hundreds of measurement points, *MD Nastran®* provides a huge computation cost for a single calculation.

Most of mechanical solvers are suitable for large, sparse, hermitian matrices with a small bandwidth of non-zero terms around the diagonal. Indeed, as the finite element matrices are generally symmetric, only the terms on and above the diagonal need to be stored. This reduces the memory requirements and number of operations to solve the matrix. However,



$$\left(\begin{array}{ccccc|ccccc}
-3 & 2 & 0 & 0 & 0 & 1 & -2 & 0 & 0 & 0 \\
2 & -4 & 2 & 0 & 0 & -2 & 3 & -2 & 0 & 0 \\
0 & 2 & -5 & 2 & 0 & 0 & -2 & 3 & -2 & 0 \\
0 & 0 & 2 & -5 & 3 & 0 & 0 & -2 & 4 & -3 \\
0 & 0 & 0 & 3 & -5 & 0 & 0 & 0 & -3 & 5 \\
\hline
1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2 & 3 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -2 & 3 & -2 & 0 & 0 & 0 & 5 & 0 & 0 \\
0 & 0 & -2 & 4 & -3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -3 & 5 & 0 & 0 & 0 & 0 & 5
\end{array}\right)$$

Figure 3.1.: A 10×10 matrix describing the same structure of the coefficient matrix in (3.1)

here, the saddle-point matrices have large bandwidth of non-zero entries around the diagonal, which precisely makes the factorization operation prohibitive. An example of the structure of such matrices is presented in Figure 3.1, it will be used throughout this chapter.

One way to remedy that matter is the model reduction. Actually, the reduction may be performed on two different types of data : the shapes in the different domains, namely here eigenmodes and the structural matrices (stiffness and mass).

Reduction of shapes is easy since only the measured partition of the degrees of freedom is retained, and the reduction of the system matrices is, unavoidably, a simplifying process [68]. The price to pay is the limited validity of the reduced model since it is only able to represent a partial subset of the modes of the structure. A numerical force connectivity pattern appears between the retained degrees of freedom of the reduced structural matrices. A clear advantage of this method in model updating is that there is no need to use the information from a numerical model (containing errors) to extrapolate experimental results, as it is the case for expansion. The reader can refer to these references [91, 86] about parametric reduced models.

Although many techniques based on model reduction have accelerated calculations, nevertheless this action downgrades in the same time the error localization properties as shown in [19, 39]. Here, wishing to keep these localization properties, the problem is solved without further reduction.
We present in the following a new direct method to solve the saddle point systems.

3.2. A dynamic dual factorization and ordering strategy to reduce fill-in

Sparse symmetric indefinite direct solvers encounter some numerical difficulties to handle the fill-in issue, as the ordering method effect is deteriorated due to some dynamic pivoting techniques. Hence, it may be interesting to figure out a method that enables factorization without a need to do pivoting. We propose in this section a new direct approach that enables a fill-in reduction. It is implemented as a dynamic factorization approach that combines factorization and ordering in the same step.

3.2.1. Factorization and ordering dual process

Direct solvers are usually implemented with a preprocessing step before the factorization. This includes scaling, pivoting and ordering. The preprocessing step makes the numerical factorization in many cases easier and cheaper, which influences the memory and the CPU time of the factorization step. However, for the kind of matrices envisaged here, this process is not the best one. To explain in details this issue, let us take the example of the Figure 3.1. It is about a small 10×10 matrix $\widetilde{\mathcal{A}}$ that describes the same structure of the coefficient matrix in 3.1. We modelize this matrix using three colors for each block : blue for block $(1, 1)$, green for blocks $(2, 1)$ and $(1, 2)$ and red for block $(2, 2)$, as presented in Figure 3.2.

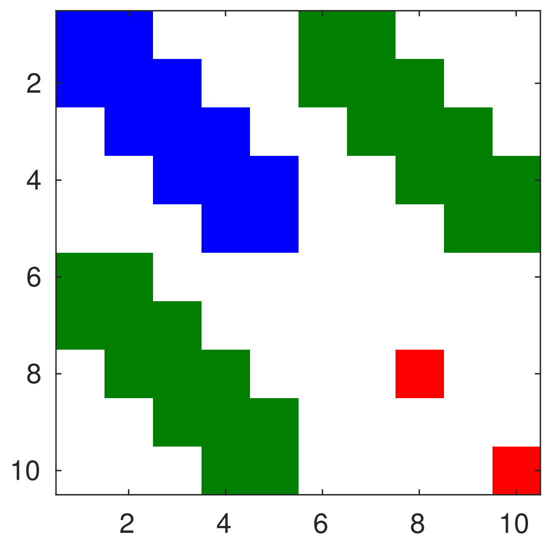


Figure 3.2.: A colorful structure of the matrix $\widetilde{\mathcal{A}}$ of the Figure 3.1
blue (stiffness), green (constraint), red (measures)

We apply the standard LU algorithm on the matrix $\widetilde{\mathcal{A}}$ without any ordering method. The number of nonzero elements is equal to 70 as mentioned in the Figure 3.3. We note that we cannot use the same colors as in Figure 3.2 for each block, because in the process of the factorization, we do not keep track of the initial elements of the coefficient matrix, the structure of the updated matrix $\mathcal{L}+\mathcal{U}$ changes in an unpredictable way. We notice that the nonzero elements of $\mathcal{L}+\mathcal{U}$ fills the positions of zero elements that are localized in the center of the global block matrix and also in the (2,2) block. This is predictable because of the specific structure of the global matrix and the sparsity of the (2,2) block.

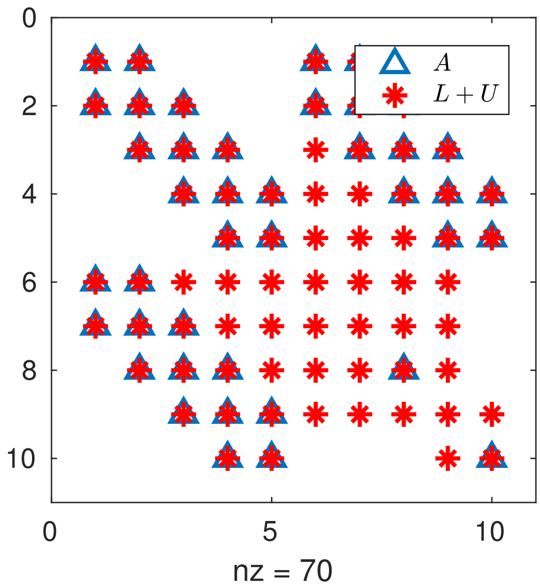


Figure 3.3.: The structures of $\widetilde{\mathcal{A}}$ and the factor matrices $\mathcal{L}+\mathcal{U}$ with standard LU and implied large fill-in

Now, we would like to analyze the impact of application of a reordering algorithm. Many heuristics have been developed in order to find good fill-reducing ordering. Here, we will be focusing on minimum degree heuristics, since they are the most efficient for non structured matrices and because of their incremental process.

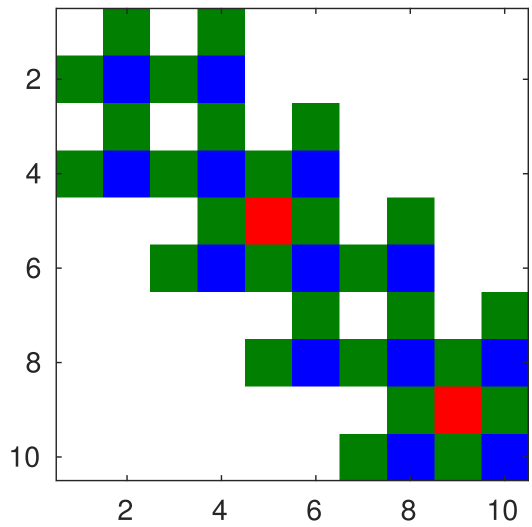


Figure 3.4.: The structure of the reordered matrix $\mathcal{P}^{\mathcal{T}}\widetilde{\mathcal{A}}\mathcal{P}$ using the approximate minimum degree algorithm
blue (stiffness), green (constraint), red (measures)

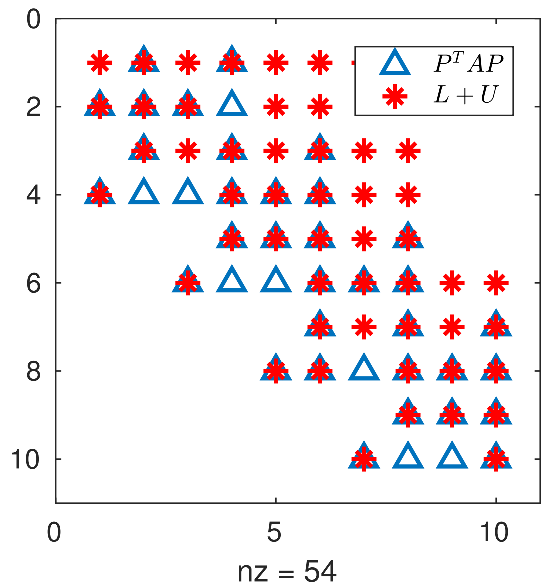


Figure 3.5.: The structures of $\mathcal{P}^{\mathcal{T}}\widetilde{\mathcal{A}}\mathcal{P}$ and the factor matrices $\mathcal{L}+\mathcal{U}$ with standard LU using the approximate minimum degree algorithm and consequent reduced fill-in

We apply the approximate minimum degree algorithm (AMD) on the matrix $\widetilde{\mathcal{A}}$. It consists in the application of the permutation $\mathcal{P}$ on the working matrix $\widetilde{\mathcal{A}}$.
We notice in Figure 3.4 that the element $(1,1)$ vanishes, which implies the necessity of ap-

plying some pivoting technique in order to continue the elimination process. The pivoting technique then modifies the initial ordering of the matrix $\mathcal{P}^{\mathcal{T}}\widetilde{\mathcal{A}}\mathcal{P}$, which explains the unsymmetric structure of the resulting factors. We obtain in the Figure 3.5 a number of nonzero elements of 54. This fill-in can be improved if the AMD algorithm does not yield null pivots.

To address this concern, the following idea is proposed. Instead of doing successive and disjoint ordering and factorization steps, these ones are performed in conjunction, so that, when a small pivot is detected, the efficient ordering for sparsity preservation is overridden. Thus, each pivot used in the factorization process is well chosen and no pivoting is needed. We present in the following an algorithm that mixes the AMD ordering with the LU factorization with a specific strategy for choosing pivots.

The AMD algorithm is modified in such a way we can merge it with the LU factorization algorithm. First, instead of ordering all graph nodes, the modified algorithm gives an order to a node dynamically at each iteration. Thanks to this, the line of the factor matrix $\mathcal{U}$ and the column of the second factor matrix $\mathcal{L}$ associated to the chosen node are filled at each iteration. This process goes on until $\mathcal{L}$ and $\mathcal{U}$ are totally computed. The described process is sequential.

In the described process, it happens that in some iterations the AMD algorithm generates many nodes with the same minimum degree. Actually, this can be found in the usual AMD algorithm also and when this situation occurs, the first node encountered among them is usually taken. Here, the proposed algorithm 3.1 chooses among them the node that gives the better pivot. It is worth mentioning that this observation is useful in order to improve the algorithm. In fact, three-dimensional large structures in mechanics involve large regular meshes which are huge graphs with millions of nodes. Away from borders, each node has the same degree. Hence, at each iteration and in order to choose the suitable node, the classical AMD ordering algorithm may find thousands of nodes with the same minimum degree.

It may occur that none of these minimum degree nodes is suitable because of the singularity of their associated pivot elements. In the regular AMD ordering algorithms, this implies the requirement to use pivoting techniques to avoid breakdown. Here, the Algorithm 3.1 performs a search among the nodes with a degree just above the minimum. It is obvious that this choice downgrades the initial fill-in reduction of the AMD algorithm. Nevertheless, it prevents factorization from breakdown, preserves a high level of sparsity and avoids in the same time the use of pivoting techniques. This algorithm 3.1 of dual factorization and ordering based on the approximate minimum degree algorithm is shown.

Algorithm 3.1: Direct solution method based on a dual ordering-factorization step and using a modified version of the AMD algorithm

Data: a sparse Matrix $\widetilde{\mathcal{A}} = (a_{ij}) \in \mathbb{R}^{n \times n}$
Result: factor matrices $\mathcal{L}$,$\mathcal{U}$ such that $LU = \widetilde{\mathcal{A}}$ and the ordering vector $Perm$

Initialization of different parameters;
$threshold$ (a threshold for numerical robustness);
$Perm = \emptyset$ (the permutation vector);
$ZeroPivots = \emptyset$ (a list of nodes with zero pivots);
$iterations = 0$;
while $iterations \leq$ *number of nodes* **do**
    Pick a node p with an approximate minimum degree;
    if $\widetilde{\mathcal{A}}(p,p) > threshold$ *& node p unlabelled &* $p \notin ZeroPivots$ **then**
        Number the node p;
        Put the node p in the list $Perm$;
        Do $AlgParLU(\widetilde{\mathcal{A}}, p)$;
        $iterations = iterations + 1$;
        if $ZeroPivots \neq \emptyset$ **then**
            while $\exists q \in ZeroPivots$ *such that* $\widetilde{\mathcal{A}}(q,q) > threshold$ **do**
                Number the node q ;
                Put the node q in the list $Perm$;
                Remove the node q from the list $ZeroPivots$;
                Do $AlgParLU(\widetilde{\mathcal{A}}, q)$;
                $iterations = iterations + 1$;
    else
        if $p \notin J$ **then**
            Put the node p in the list $ZeroPivots$;
if $ZeroPivots \neq \emptyset$ **then**
    Add the list J at the end of $Perm$;

We add below the description of the routine *AlgParLU* for line-by-line factorization.

Algorithm 3.2: A routine describing a line-by-line factorization process

---

Data: a sparse Matrix $\widetilde{\mathcal{A}} = (a_{ij}) \in \mathbb{R}^{n \times n}$, the selected node p, and the factor matrices $\mathcal{L}$ and $\mathcal{U}$

Result: factor matrices $\mathcal{L}$,$\mathcal{U}$

Do a Gauss elimination on the p^{th} pivot of the matrix $\widetilde{\mathcal{A}}$;

Update the p^{th} line in $\mathcal{U}$ and column in $\mathcal{L}$;

We apply the algorithm 3.1 on the example of Figure 3.1. We present in Figures 3.6 and 3.7 the new structure of the ordered matrix $\mathcal{Q}^{\mathcal{T}}\widetilde{\mathcal{A}}\mathcal{Q}$ and the fill-in generated by its factorization.

We notice that the algorithm 3.1 gives an approximate ordering to that of classical AMD algorithm. Clearly, if no singular pivot is found in the factorization process, it would give the exact same ordering of the AMD algorithm. Besides, in comparison to Figure 3.4, the first element $(1, 1)$ is not zero, which means that there is no need to apply pivoting techniques. The number of nonzero element is equal to 49 in this case. It is less than both previous cases which highlights the fill-in reduction gained thanks to the algorithm 3.1 by avoiding pivoting.

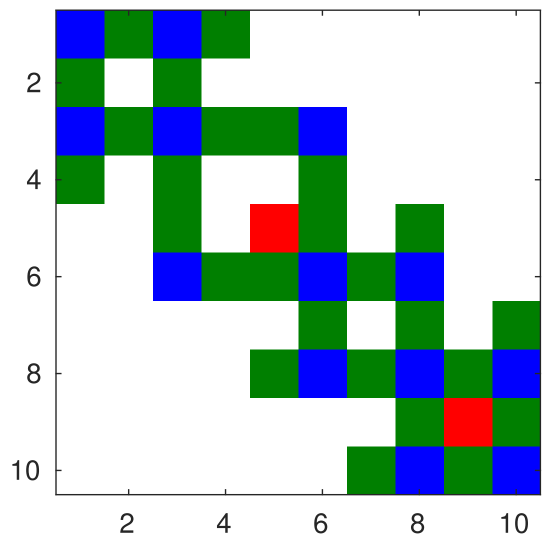


Figure 3.6.: The structure of the reordered matrix $\mathcal{Q}^{\mathcal{T}}\widetilde{\mathcal{A}}\mathcal{Q}$ with the dual factorization and ordering method (Algorithm 3.1)
blue (stiffness), green (constraint), red (measures)

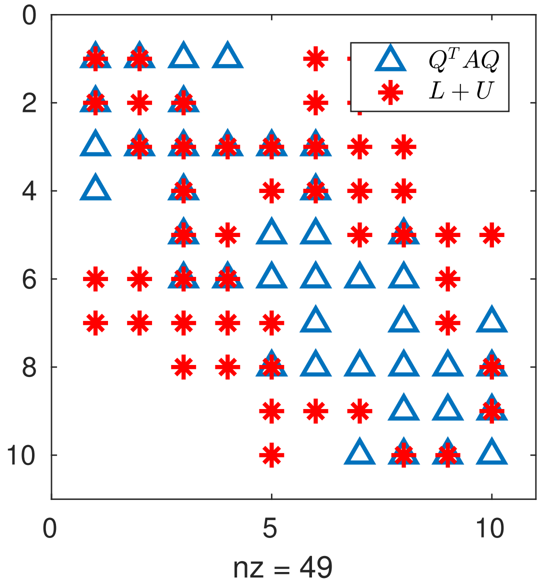


Figure 3.7.: The structures of $\mathcal{Q}^T\widetilde{\mathcal{A}}\mathcal{Q}$ and the factor matrices $\mathcal{L}+\mathcal{U}$ with the dual factorization and ordering method (Algorithm 3.1)

Actually, Algorithm 3.1 downgrades the usual ordering yield by AMD to bring more robustness and efficiency to the factorization. It avoids singular pivots by putting them in a list (called $ZeroPivots$ in the algorithm) then retest them each time the factorized matrix is updated. This technique helps keep a level of optimality with respect to the fill-in. The factorization ends up when all the remaining (uneliminated) diagonal entries are null.

In the following section, Algorithm 3.1 is applied on some matrices with medium size and compared with unsymmetric direct solvers. From now on, we choose the name Minimum Degree Factorization (MDF) for Algorithm 3.1.

3.2.2. Comparison with off-the-shelf general direct solvers on medium size test-structures

A prototype code of MDF has been implemented in Matlab®. Numerical experiments focus on producing meaningful comparisons according to sparsity and stability. The test matrices share the same properties of the system (3.1). The numerical experiments were conducted with three different sparse direct solvers : $MUMPS$[3], $SuperLU$[77] and $UMFPACK$[36].

Setting equivalent parameters

Many technical differences prevent to achieve a fair comparison. As many parameters in these codes can be set by the user and any modification of their values can lead to different performances [4]. Consequently, preprocessing parameters are set similarly so that the codes perform equivalently.
Some codes can use any sparsity ordering provided by the user, which is the case for both $MUMPS$ and $SuperLU$, other codes can only use their own ordering algorithms which are sometimes buried deeply within the code. This will cause different amounts of fill-in, whose difference is not intrinsic to the factorization algorithms. Here, only minimum degree ordering variants are used for each solver. $MUMPS$ uses an approximate minimum degree, $SuperLU$ uses the multiple minimum degree and both compute this symmetric permutation on the structure of $A + A^T$. $UMFPACK$ uses a column approximate minimum degree to compute a fill-in reducing preordering[36].
Furthermore, The smallest pivoting threshold for $MUMPS$, $UMFPACK$, $SuperLU$ and MDF has been chosen so that it provides a trade-off between preserving sparsity and ensuring numerical stability during factorization. The relative threshold for numerical pivoting is chosen equal to 0.01.

In all test cases, an artificial right-hand side b is used in the runs, so that the system $Ax = b$ had the known solution $x = (x_i)$ with $x_i = 1$, $1 \leq i \leq n$. The iterative refinement is used with all different solvers and is stopped when the componentwise sparse relative backward error ($Berr$) defined by $Berr = max_i(\frac{|Ax-b|_i}{(|A|.|x|+|b|)_i})$, is close to machine precision [7]. The number of steps of iterative refinement and the error are also reported. For each test matrix, the fill-in $nnz(\mathcal{L} + \mathcal{U})$ of LU factorization is given with a ratio of sparsity. This ratio is computed as $\frac{nnz(L+U)}{nnz(A)}$. The number of steps ($Steps$), the error (Err_{ref}) and the backward error ($Berr_{ref}$) after the iterative refinement process are also reported.

The following sections present numerical results to measure how MDF interacts with randomized finite element test matrices and an industrial test matrix that share the structure of the coefficient matrix of the linear system given in equation (3.1). It is reminded that only storage and numerical stability outcomes are considered within this study.

Results for randomized One-Dimensional finite element test matrices

First, a finite classical one-dimensional mass and spring system composed of n identical springs and masses is considered, see Figure 3.8. Its stiffness matrices are discrete Laplacians. c sensors are used to produce the experimental finite element mesh.

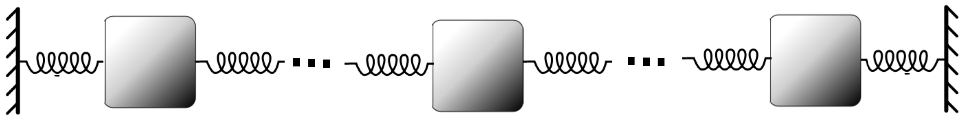


Figure 3.8.: One-dimensional system composed of N masses and springs in series

The fill-in behavior of the different solvers is illustrated in Table 3.1.

	Matrix	$FE1$	$FE2$	$FE3$	$FE4$
	n	200	400	600	800
	c	10	20	30	40
	nnz	904	1814	2724	3634
MDF	*Fill-in*	1,229	2,387	3,643	4,805
	Ratio(%)	1.35	1.31	1.33	1.32
$UMFPACK$	*Fill-in*	1,316	2,646	4,021	5,249
	Ratio(%)	1.45	1.45	1.47	1.44
$MUMPS$	*Fill-in*	**996**	**2,242**	**3,444**	**4,064**
	Ratio(%)	**1.10**	**1.23**	**1.26**	**1.11**
SuperLU	*Fill-in*	1,346	2,663	4,014	5,383
	Ratio(%)	1.48	1.46	1.47	1.48

Table 3.1.: Fill-in results for one-dimensional randomized finite element test matrices using MDF, $UMFPACK$, $MUMPS$ and *SuperLU* respectively.

Some observations that can be drawn from Table 3.1 are twofold. Firstly, $MUMPS$ has the best reducing fill-in factorization and MDF has the second best one. The reason is that in such matrices and by construction, the pivots are chosen among a large set of candidates that present all the same numerical value which decreases the efficiency of MDF.
Secondly, as the test matrices are one-dimensional finite element ones, they keep a simple pattern in *LU* factors which explains that $1 \leq Ratio \leq 2$. In addition, for each solver the *Ratio* of sparsity remains in an average range, even though the size of problem gets bigger.

In the following, Table 3.2 is shown to contrast the numerical behavior of MDF with other well-known solvers. Clearly, for such one-dimensional finite element test matrices all solvers yields comparable stable solutions. The backward error $Berr$ almost reaches the machine precision in double precision. With one step of iterative refinement process solvers manage to reach it, which explains that the error Err_{ref} and the backward error $Berr_{ref}$ decrease by one or two orders of magnitude.

	Matrix	$FE1$	$FE2$	$FE3$	$FE4$
	n	200	400	600	800
	c	10	20	30	40
	nnz	904	1814	2724	3634
MDF					
	Err	1.5E-12	2.9E-12	3.1E-11	1.8E-12
	$Berr$	4.1E-15	7.5E-15	7.3E-14	3.8E-15
	$Steps$	1	1	1	1
	Err_{ref}	7.2E-14	9.6E-14	1.1E-13	1.4E-13
	$Berr_{ref}$	1.3E-16	1.3E-16	1.2E-16	1.7E-16
$UMFPACK$					
	Err	**6.3E-14**	**1.0E-13**	**1.2E-13**	**1.4E-13**
	$Berr$	**2.4E-16**	**3.0E-16**	**6.7E-16**	**3.9E-16**
	$Steps$	1	1	1	1
	Err_{ref}	8.0E-14	9.7E-14	1.2E-13	1.5E-13
	$Berr_{ref}$	1.3E-16	1.6E-16	1.5E-16	1.3E-16
$MUMPS$					
	Err	4.8E-13	1.0E-12	1.1E-12	1.4E-12
	$Berr$	8.7E-16	3.8E-15	2.6E-15	2.3E-15
	$Steps$	1	1	1	1
	Err_{ref}	7.1E-14	1.0E-13	1.2E-13	1.4E-13
	$Berr_{ref}$	1.2E-16	1.8E-16	1.4E-16	1.5E-16
$SuperLU$					
	Err	1.7E-12	3.4E-12	8.1E-11	2.8E-12
	$Berr$	5.1E-15	8.2E-15	9.4E-14	5.4E-15
	$Steps$	1	1	2	1
	Err_{ref}	8.2E-14	2.8E-14	7.1E-12	2.5E-13
	$Berr_{ref}$	1.6E-16	1.5E-16	1.7E-16	1.4E-16

Table 3.2.: The numerical behavior of MDF, $UMFPACK$, $MUMPS$ and $SuperLU$ when solving one-dimensional randomized finite element test matrices before and after using an iterative refinement process

Results for randomized Three-Dimensional finite element test matrices

We generate a randomized three dimensional unstructured numerical model. Using its stiffness and mass matrices, the coefficient matrix of the linear system (3.1) is constructed. Building the experimental mesh is done such as the $(2,2)$ block in the coefficient matrix of the system (3.1) keeps the same pattern, even though the problem gets bigger.

Based on the results, the following conclusions are derived. Not surprisingly, MDF and $MUMPS$ perform much better than other solvers regarding the fill-in. As the test matrices are three dimensional finite element ones with irregular pattern, MDF achieves a smaller fill-in than $MUMPS$. For all solvers, the *Ratio* of sparsity is growing as the size of the test matrices increases.

	Matrix	$FE1_{3D}$	$FE2_{3D}$	$FE3_{3D}$	$FE4_{3D}$
	n	200	400	600	800
	c	10	20	30	40
	nnz	1238	2588	3732	4910
MDF	*Fill-in*	**6,628**	**28,039**	**48,372**	**84,654**
	Ratio(%)	**5.35**	**10.38**	**12.96**	**17.24**
$UMFPACK$	*Fill-in*	7,956	30,386	60,577	91,028
	Ratio(%)	6.42	11.74	16.23	18.53
$MUMPS$	*Fill-in*	7,294	28,282	52,226	92,558
	Ratio(%)	5.89	10.92	13.91	18.85
SuperLU	*Fill-in*	8,014	29,307	53,554	95,821
	Ratio(%)	6.47	11.32	14.34	19.51

Table 3.3.: Fill-in results for 3D randomized finite element test matrices using MDF, $UMFPACK$, $MUMPS$ and *SuperLU* respectively

Regarding the stability issue, MDF gets inaccurate when dealing with this type of matrices so that it needs in some cases two steps of iterative refinement. The number of $zerosPivots$ in MDF has an important consequence on the accuracy of the solution and the number of iterative refinement steps. Here, a higher number of these pivots is being recorded which result, in general, in inaccurate but sparser factors L and U. Furthermore, $MUMPS$ is less impacted so that the backward error needs only two or three orders of magnitude to reach the machine precision through iterative refinement.

	Matrix	$FE1_{3D}$	$FE2_{3D}$	$FE3_{3D}$	$FE4_{3D}$
	n	200	400	600	800
	c	10	20	30	40
	nnz	1238	2588	3732	4910
MDF					
	Err	3.8E-08	8.5E-06	1.7E-05	7.1E-06
	$Berr$	1.0E-10	2.9E-08	5.4E-08	1.6E-08
	$Steps$	1	2	2	2
	Err_{ref}	1.5E-13	2.2E-13	2.5E-13	3.0E-13
	$Berr_{ref}$	1.9E-16	2.0E-16	1.9E-16	2.1E-16
$UMFPACK$					
	Err	**4.0E-13**	**9.5E-13**	**1.7E-12**	**2.5E-12**
	$Berr$	**1.9E-15**	**4.1E-15**	**7.7E-15**	**9.9E-15**
	$Steps$	1	1	1	1
	Err_{ref}	1.4E-13	2.5E-13	2.9E-13	2.9E-13
	$Berr_{ref}$	1.4E-16	2.0E-16	2.0E-16	1.9E-16
$MUMPS$					
	Err	1.5E-11	2.0E-10	1.4E-10	3.2E-10
	$Berr$	3.7E-14	3.5E-13	5.6E-13	1.7E-12
	$Steps$	1	1	1	1
	Err_{ref}	1.2E-13	2.0E-13	2.6E-13	2.7E-13
	$Berr_{ref}$	1.3E-16	1.7E-16	1.8E-16	1.8E-16
$SuperLU$					
	Err	1.8E-10	2.9E-08	7.5E-08	5.1E-08
	$Berr$	3.1E-13	3.6E-12	9.2E-11	8.4E-11
	$Steps$	2	2	2	2
	Err_{ref}	1.2E-13	3.1E-13	3.7E-13	2.8E-13
	$Berr_{ref}$	1.3E-16	1.4E-16	1.7E-16	1.4E-16

Table 3.4.: The numerical behavior of MDF, $UMFPACK$, $MUMPS$ and $SuperLU$ when solving 3D randomized finite element test matrices before and after using an iterative refinement process

Results for an industrial Three-Dimensional test case

Finally we consider the coefficient matrix generated from a small version of the numerical FE model of the industrial cooling pump presented in Chapter 5 and from its experimental measurements. Its corresponding saddle-point matrix pattern is provided in Figure 3.9. The size of the system is 2006×2006 and the number of nonzero elements is $nnz = 169,538$.

Some structures of the reordered matrix are shown in Figure 3.10. The structure of the matrices reordered by MDF looks like $UMFPACK$ built-in AMD reordering algorithm, due to the same node selection process as explained in the previous section but it gives a more scattered matrix. Each ordering influences the structure of the initial matrix and at the same time the structure of the factor matrices $\mathcal{L}$ and $\mathcal{U}$ as shown in Figure 3.11.

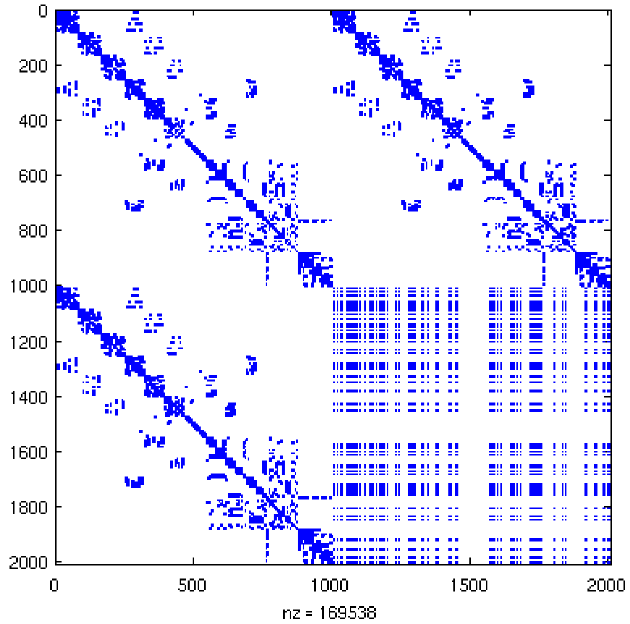


Figure 3.9.: The initial structure of the industrial test case coefficient matrix (size 2006×2006, $nnz = 169,538$)

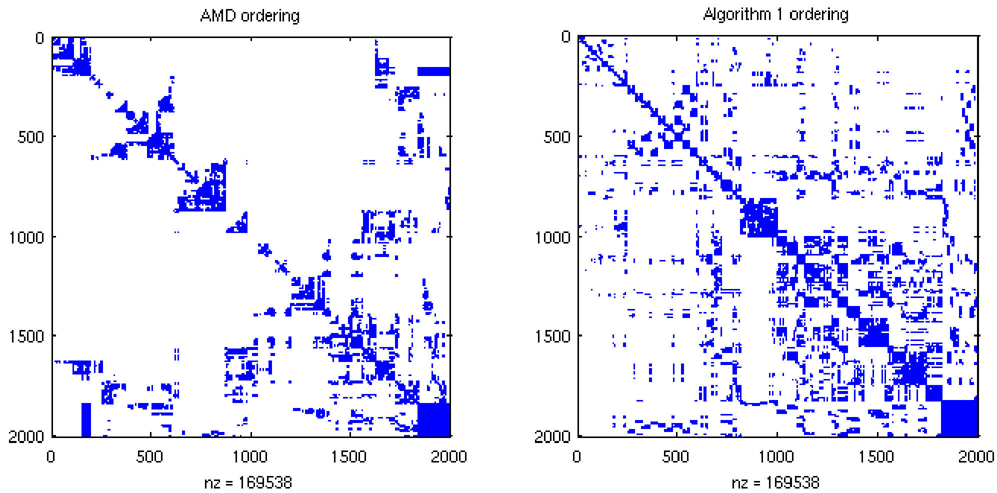


Figure 3.10.: The structure of the reordered matrix using AMD (left) and MDF (right)

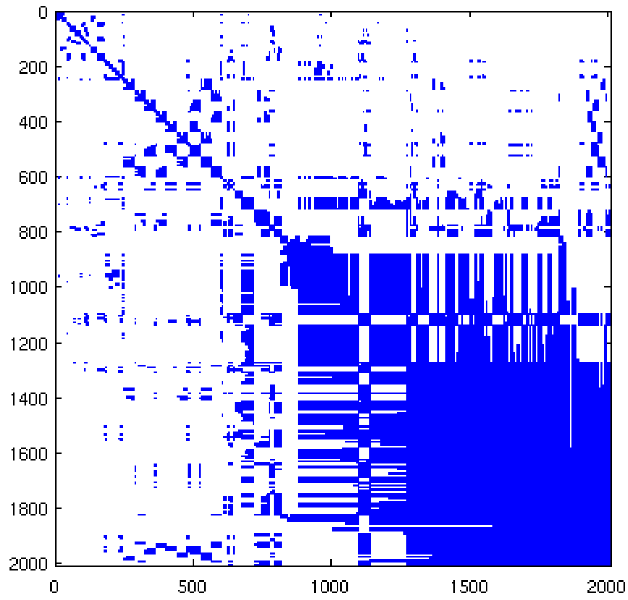


Figure 3.11.: The structure of the factor matrix $\mathcal{L}+\mathcal{U}$ after a LU factorization of the industrial test case coefficient matrix reordered by MDF ($nnz_{LU} = 850,645$)

When comparing the fill-in of LU factorization, MDF seems to perform slightly better than $MUMPS$ and is significantly better than other solvers. Even if each solver uses a minimum degree ordering variant they do not perform the same way. Actually, the pivoting is an important parameter that influences how each solver works. Threshold partial pivoting is used in all solvers here. It is a usual technique in sparse Gaussian elimination and helps to avoid excessive growth in the size of entries during the matrix factorization.

Matrix A	MDF	$UMFPACK$	$MUMPS$	$SuperLU$
Fill-in	**850,645**	914,271	878,848	1,002,364
Ratio (%)	**5.01**	5.39	5.18	5.91
Err	8.7E-05	1.8E-08	2.1E-06	5.1E-06
Berr	1.1E-08	5.4E-10	1.6E-10	2.7E-09
Steps	2	2	1	2
Err_{ref}	4.6E-09	4.1E-09	1.3E-09	2.7E-09
$Berr_{ref}$	1.3E-16	1.6E-16	1.7E-16	1.5E-16

Table 3.5.: FIll-in results and numerical behavior of MDF, $UMFPACK$, $MUMPS$ and $SuperLU$ when solving the industrial matrix A

This pivoting strategy may differ depending on the matrix structure. Here, a strong preference is given to diagonal entries. Actually, in *SuperLU* and $UMFPACK$ if the pivot does not satisfy the threshold (0.01), diagonal elements are tested. If neither of the above satisfies the threshold, the maximum magnitude element in the column will be used as the pivot. For $MUMPS$, even if it can choose pivots from off the diagonal, the largest entry in the column might be unavailable for pivoting at this stage if all entries in its row are not fully summed. In this case, it is kept in the Schur complement and is passed to the parent node where all rows will be available for pivoting so that a pivot can be chosen from the column. It is interesting to note that for $MUMPS$ the number of delayed pivots has an impact on the fill-in in the same way that the $zerosPivots$ have in MDF. This matches with the fact that this strategy is used at the cost of increasing the size of the frontal matrices and causing more work and fill-in than were forecast [4].
The same remarks as for randomized three dimensional finite element matrices are formulated regarding stability. Solutions obtained by MDF, $MUMPS$, $UMFPACK$ and *SuperLU* need to be refined through some steps of iterative refinement process. This behavior steps up with the matrix enlargement.

Limitations and performance inhibitors

In this section, we just presented a variant direct solution method that uses a dynamic process handling factorization and ordering in the same step. This variant enables us to avoid pivoting and gains some fill-in especially in the case of indefinite symmetric matrices. The performance of standard solvers is presented. We limited our study to one specific kind of ordering algorithms based on minimum degree concept. The goal was to compare these solvers using the same basic ordering concept as MDF does.

This approach has several limitations. Firstly, although we study a very particular saddle point matrices. Their specific structure is though not exploited within MDF since this latter could be applied to any invertible matrix.
Secondly, MDF requires an implementation with dynamic memory management of the LU factors. Actually, this algorithm is proposed to handle the problem of the fill-in generated by pivoting, that involves tracking it dynamically and thus impacts dramatically the computation time. Hence, the proposed algorithm is intrinsically dynamic and is not suitable with a symbolic factorization.
Thirdly, the numerical results are evaluated measuring only the fill-in in the LU factors as well as the numerical stability. There are other aspects that influence the efficiency of the numerical factorization, for example the sparsity pattern. It influences the number of FLOPS required and, more importantly, to which extent it is possible to exploit dense linear

algebra kernels. However, these latter elements need a proper symbolic factorization that exploits an elimination tree. However, as said before, this algorithm is intrinsically dynamic and then not suited for those strategies.

That is why, we present in the next section a more suitable approach that takes into account the specific structure of our problem. Since symmetric indefinite matrices arising from saddle point problems can be congruently transformed into special form of a block structure as will be discussed, the transformed matrices can be exploited efficiently by taking advantage of the structure and properties of their blocks. We then compare this approach numerically with the standard direct solvers.

3.3. Sparse 2-by-2 block factorization of saddle point matrices

A general sparse symmetric indefinite solver, as it employs numerical pivoting for stability, has rarely the capability of predicting the fill-in rising in the generated factors, which is emphasized in the previous section. In the saddle-point systems literature, there is another approach that predetermines the elimination ordering. We call it the block factorization approach. However it may or may not produce sparser factors than a general sparse solver.

3.3.1. Determination of the transformation matrix and partitioning

As seen in section 2.3.2, a comparative study of different null-space block factorizations for symmetric saddle point systems has demonstrated the possibility of exploiting the block structure of those matrices in a direct solver [92].

We present here an approach that exploits the specific structure of our saddle point matrix. This new factorization considers doing micro-factorizations. In this formulation, the coefficient matrix is reordered by pairing every entry on the diagonal of the $(1, 1)$ block with a corresponding entry in the constraint block $(2, 1)$ so that the entries on the diagonal of the permuted matrix form micro 2-by-2 block saddle point systems.

To reach this goal, we propose in this section a new transformation $\pi^T \mathcal{A} \pi = \hat{\mathcal{A}}$, followed by a 2-by-2 block Gaussian elimination factorization $\mathcal{P}^T \hat{\mathcal{A}} \mathcal{P} = \mathcal{L}\mathcal{D}\mathcal{L}^T$, where :

- $\mathcal{L}$ is block lower triangular with blocks of order 2, and $\mathcal{D}$ is a block diagonal matrix with block of order 2.

- π is a $2(n+m)\times 2(n+m)$ transformation matrix, which describes a new matrix whose entries are 2-by-2 block saddle point matrices. We choose it such that the $\mathcal{LDL}^T$ factorization is stable and has a sparse factors $\mathcal{L}$ and $\mathcal{D}$.
- $\mathcal{P}$ is a predefined $2(n+m)\times 2(n+m)$ permutation matrix for *a priori* pivoting of $\hat{\mathcal{A}}$.

We define first a permutation $\tau : \mathcal{N}^{2(n+m)} \rightarrow \mathcal{N}^{2(n+m)}$ by

$$\tau = \begin{pmatrix} 1 & 2 & 3 & \cdots & 2(n+m)-1 & 2(n+m) \\ 1 & n+m+1 & 2 & \cdots & n+m & 2(n+m) \end{pmatrix}. \tag{3.2}$$

The $2(n+m)\times 2(n+m)$ permutation matrix $\mathcal{P}_\tau$ related to τ is given by

$$\mathcal{P}_\tau = \begin{bmatrix} e_1 & , & e_{n+m+1} & , & e_2 & , & \cdots & , & e_{2(n+m)} \end{bmatrix}, \tag{3.3}$$

where e_i is the ith unit vector of length $2(n+m)$.

In order to illustrate this transformation, we consider here a 10×10 saddle point matrix $\widetilde{\mathcal{A}}$ with the same pattern as the coefficient matrix of the studied linear system (3.1), see Figure 3.12. We choose the same color for each block of the matrix. We recall that $(1,1)$ block refers to the constrained stiffness and is in blue, the blocks $(2,1)$ and $(1,2)$ describes the impedance and are in red, and the block $(2,2)$ is a matrix describing the projection of experimental sensors degrees of freedom on the numerical model degrees of freedom and is in green.

We notice that by applying this natural order symmetrically to the initial matrix $\widetilde{\mathcal{A}}$, the entries having different color and index get coupled, see Figure 3.13. The new matrix is formed of 2-by-2 block entries that inherit the same structure of $\widetilde{\mathcal{A}}$.

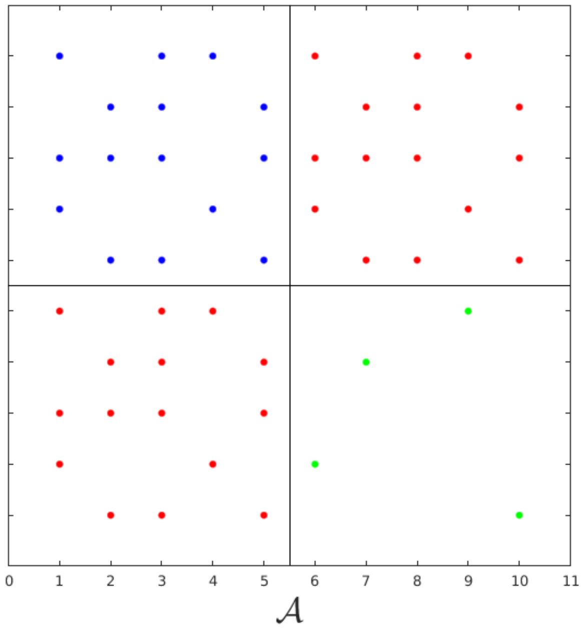


Figure 3.12.: The matrix $\widetilde{\mathcal{A}}$

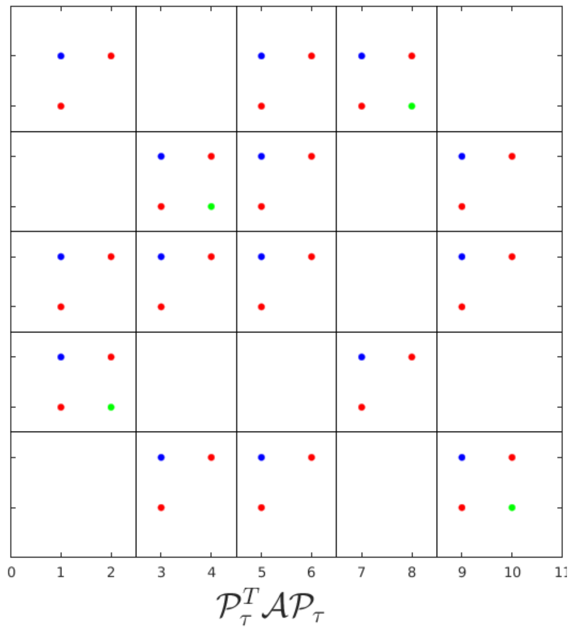


Figure 3.13.: The matrix $\mathcal{P}_\tau^T \mathcal{A} \mathcal{P}_\tau$

An interesting observation to report, is to see these 2-by-2 elemental block of the $2(n+m) \times 2(n+m)$ matrix as entries of an augmented $(n+m) \times (n+m)$ matrix.

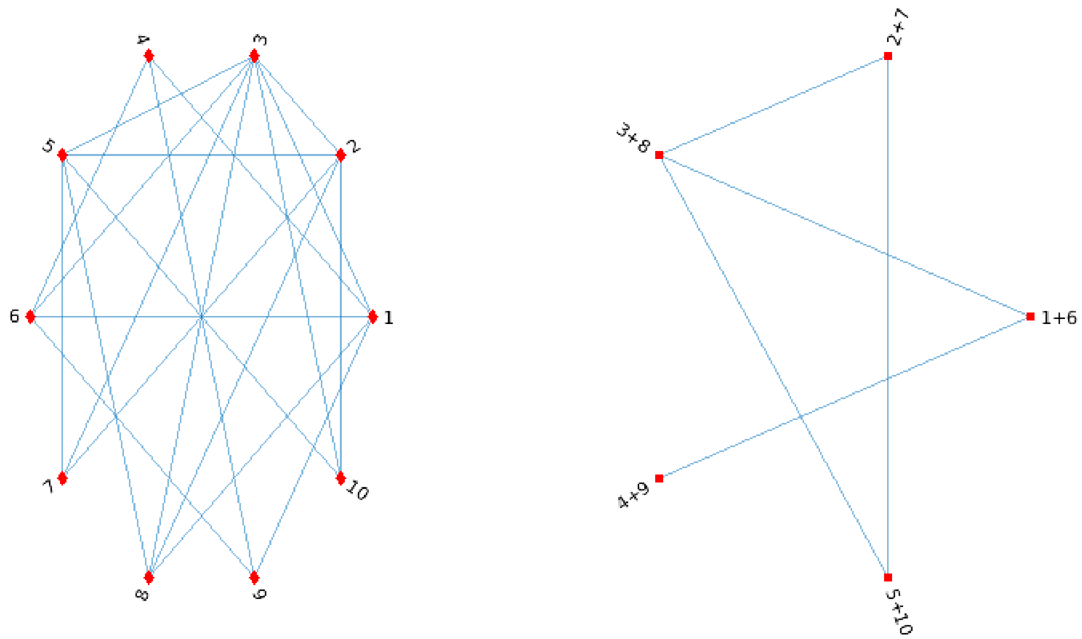


Figure 3.14.: The graph associated with matrix $\widetilde{\mathcal{A}}$ (left) and the reduced graph associated with matrix $\mathcal{P}_\tau^T \mathcal{A} \mathcal{P}_\tau$ (right)

In term of graph terminology, this could be seen as a graph with supernodes of order 2. This is illustrated in Figure 3.14.

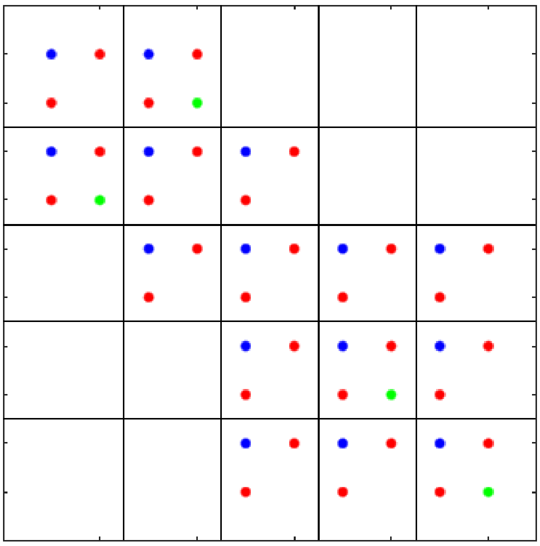
The main idea we propose here to maintain sparsity is to reorder those supernodes using approximate minimum degree algorithm. This approach enables getting a more comprehensive sparse ordering regarding to 2-by-2 block Gaussian elimination.
This yields a new transformation r applied on a half smaller set $\mathcal{N}^{2(n+m)} \to \mathcal{N}^{2(n+m)}$ representing supernodes labelled form 1 to $n+m$. This permutation reorders increasingly the different nodes of the set $\{i, i+1, \cdots, n+m\}$ with respect to their approximate minimum degree.
Finally we can describe the final permutation $\pi : \mathcal{N}^{2(n+m)} \to \mathcal{N}^{2(n+m)}$ used on the matrix $\widetilde{\mathcal{A}}$ by

$$\pi = \begin{pmatrix} 1 & 2 & 3 & \cdots & 2(n+m)-1 & 2(n+m) \\ r(1) & n+m+r(1) & r(2) & \cdots & r(n+m) & (n+m)+r(n+m) \end{pmatrix}. \tag{3.4}$$

The $2(n+m) \times 2(n+m)$ permutation matrix $\mathcal{P}_\pi$ related to π is given by

$$\mathcal{P}_\pi = \left[e_{r(1)} \quad , \quad e_{n+m+r(1)} \quad , \quad e_{r(2)} \quad , \quad \cdots \quad , \quad e_{(n+m)+r(n+m)} \right], \tag{3.5}$$

where e_i is the ith unit vector of length $2(n+m)$. We illustrate the action of this permutation matrix on Figure 3.15.



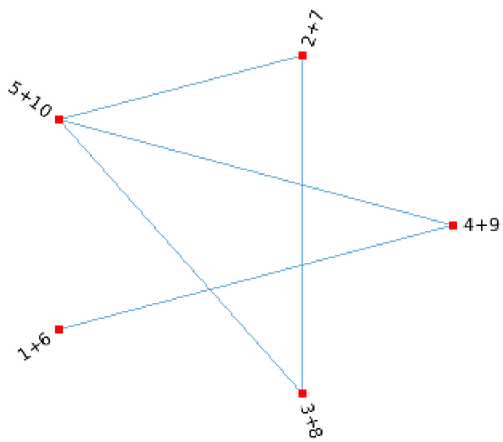


Figure 3.15.: The matrix $\mathcal{P}_\pi^T \mathcal{A} \mathcal{P}_\pi$ and its associated compressed graph

The matrix $\mathcal{G} = \mathcal{P}_\pi^T \mathcal{A} \mathcal{P}_\pi$ has block entries of order 2, they are given for $1 \leq i, j \leq (n+m)$

by :

$$\mathcal{G}_{ij} = \begin{bmatrix} -a_{ii} & b_{ii} \\ b_{ii} & t_{ii} \end{bmatrix} \tag{3.6}$$

where $A = [a_{ij}] = -[\widetilde{K}(\theta)]$, $B = [b_{ij}] = [\widetilde{K}(\theta)] - \omega_{exp}^2[\widetilde{M}(\theta)]$, $T = [t_{ij}] = \frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\Pi}$, $1 \leq i, j \leq (n+m)$.

Note that a suitable pairing that preserves sparsity and is numerically stable should be chosen, see Section 3.3.2 for further discussion.

3.3.2. Sparse 2-by-2 block factorization and numerical stability

We use the following LDL^T block decomposition at each stage of elimination, as already discussed in [26]

$$\begin{pmatrix} -A & B_1^T \\ B_2 & C \end{pmatrix} = \begin{pmatrix} \mathbb{I} & 0 \\ B_2A^{-1} & \mathbb{I} \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & C + B_2A^{-1}B_1^T \end{pmatrix} \begin{pmatrix} \mathbb{I} & A^{-1}B_1^T \\ 0 & \mathbb{I} \end{pmatrix}. \tag{3.7}$$

That means we need to have a $(1,1)$ nonsingular block in $\hat{\mathcal{A}} = \mathcal{P}_\pi^T \mathcal{A} \mathcal{P}_\pi$.

We recall that at the kth stage of Gaussian elimination, the updated matrix $\hat{\mathcal{A}}^{(k)}$ has the following partitioning

$$\hat{\mathcal{A}}^{(k)} = \begin{pmatrix} \hat{\mathcal{A}}_{11}^{(k)} & \hat{\mathcal{A}}_{21}^{(k)} \\ \hat{\mathcal{A}}_{21}^{(k)} & \hat{\mathcal{A}}_{22}^{(k)} \end{pmatrix}, \tag{3.8}$$

where $\hat{\mathcal{A}}_{11}^{(k)}$ is 2×2 pivot and $\hat{\mathcal{A}}_{21}^{(k)}$ contains $2(n+m) - 2$ rows and two columns

$$\hat{\mathcal{A}}_{11}^{(k)} = \begin{pmatrix} -a_{k,k}^{(k)} & b_{k,k}^{(k)} \\ b_{k,k}^{(k)} & t_{k,k}^{(k)} \end{pmatrix} \text{ and } \hat{\mathcal{A}}_{21}^{(k)} = \begin{pmatrix} -a_{(k+1),k}^{(k)} & b_{(k+1),k}^{(k)} \\ b_{(k+1),k}^{(k)} & t_{(k+1),k}^{(k)} \\ \vdots & \vdots \\ -a_{(n+m),k}^{(k)} & b_{(n+m),k}^{(k)} \\ b_{(n+m),k}^{(k)} & t_{(n+m),k}^{(k)} \end{pmatrix}, \tag{3.9}$$

$$\hat{\mathcal{A}}_{22}^{(k)} = \begin{pmatrix} -a_{(k+1),(k+1)}^{(k)} & b_{(k+1),(k+1)}^{(k)} & \cdots & -a_{(n+m),(k+1)}^{(k)} & b_{(n+m),(k+1)}^{(k)} \\ b_{(k+1),(k+1)}^{(k)} & t_{(k+1),(k+1)}^{(k)} & \cdots & b_{(n+m),(k+1)}^{(k)} & t_{(n+m),(k+1)}^{(k)} \\ \vdots & & \ddots & & \vdots \\ -a_{(n+m),(k+1)}^{(k)} & b_{(n+m),(k+1)}^{(k)} & \cdots & -a_{(n+m),(n+m)}^{(k)} & b_{(n+m),(n+m)}^{(k)} \\ b_{(n+m),(k+1)}^{(k)} & t_{(n+m),(k+1)}^{(k)} & \cdots & b_{(n+m),(n+m)}^{(k)} & t_{(n+m),(n+m)}^{(k)} \end{pmatrix}. \tag{3.10}$$

Through this matrix, we determine a 2×2 block diagonal element $D_{k,k}$, and the kth two columns of a unit lower triangular matrix L, which are partitioned into $L_{k,k}$ and $L_{k+1,k}$ in the same way as $\hat{\mathcal{A}}$. It follows that

$$D_{k,k} = \hat{\mathcal{A}}_{11}^{(k)}, \qquad L_{k,k} = \mathbb{I}, \qquad L_{k+1,k} = \hat{\mathcal{A}}_{21}^{(k)} D_{k,k}^{-1}, \tag{3.11}$$

At the kth stage of elimination, the size of $\hat{\mathcal{A}}^{(k)}$ decreases by 2 and it yields

$$\hat{\mathcal{A}}^{(k+1)} = \hat{\mathcal{A}}^{(k)} - \begin{pmatrix} \mathbb{I} \\ L_{k+1,k} \end{pmatrix} D_{k,k} \begin{pmatrix} \mathbb{I} & L_{k+1,k}^T \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & \hat{\mathcal{A}}_{22}^{(k)} - \hat{\mathcal{A}}_{21}^{(k)} D_{k,k}^{-1} \hat{\mathcal{A}}_{21}^{(k)} \end{pmatrix}. \tag{3.12}$$

Let us call $\widetilde{g}_{i,j}^{(k)} = \begin{pmatrix} -a_{i,j}^{(k)} & b_{i,j}^{(k)} \\ b_{i,j}^{(k)} & t_{i,j}^{(k)} \end{pmatrix}$, we explicit in the following the expression

$$\hat{\mathcal{A}}_{22}^{(k)} - \hat{\mathcal{A}}_{21}^{(k)} D_{k,k}^{-1} \hat{\mathcal{A}}_{21}^{(k)} = \begin{pmatrix} g_{(k+1),(k+1)}^{(k)} & \cdots & g_{(n+m),(k+1)}^{(k)} \\ \vdots & \ddots & \vdots \\ g_{(n+m),(k+1)}^{(k)} & \cdots & g_{(n+m),(n+m)}^{(k)} \end{pmatrix} - \begin{pmatrix} g_{(k+1),k}^{(k)} \\ \vdots \\ g_{(n+m),k}^{(k)} \end{pmatrix} (g_{k,k}^{(k)})^{-1} \begin{pmatrix} g_{(k+1),k}^{(k)} & \cdots & g_{(n+m),k}^{(k)} \end{pmatrix}. \tag{3.13}$$

Thus,

$$\hat{\mathcal{A}}_{22}^{(k)} - \hat{\mathcal{A}}_{21}^{(k)} D_{k,k}^{-1} \hat{\mathcal{A}}_{21}^{(k)} = \begin{pmatrix} g_{(k+1),(k+1)}^{(k)} - g_{(k+1),k}^{(k)} (g_{k,k}^{(k)})^{-1} g_{(k+1),k}^{(k)} & \cdots & g_{(n+m),(k+1)}^{(k)} - g_{(k+1),k}^{(k)} (g_{k,k}^{(k)})^{-1} g_{(n+m),k}^{(k)} \\ \vdots & \ddots & \vdots \\ g_{(n+m),(k+1)}^{(k)} - g_{(n+m),k}^{(k)} (g_{k,k}^{(k)})^{-1} g_{(k+1),k}^{(k)} & \cdots & g_{(n+m),(n+m)}^{(k)} - g_{(n+m),1}^{(k)} (g_{k,k}^{(k)})^{-1} g_{(n+m),1}^{(k)} \end{pmatrix}. \tag{3.14}$$

Therefore the $(k+1)$th 2×2 pivot is described by

$$pivot_{k+1} = g_{(k+1),(k+1)}^{(k)} - g_{(k+1),k}^{(k)} (pivot_k)^{-1} g_{(k+1),k}^{(k)}. \tag{3.15}$$

Using the different expressions of each quantity, we obtain that

$$pivot_{k+1} = \begin{pmatrix} -a_{(k+1),(k+1)}^{(k)} & b_{(k+1),(k+1)}^{(k)} \\ b_{(k+1),(k+1)}^{(k)} & t_{(k+1),(k+1)}^{(k)} \end{pmatrix} - \begin{pmatrix} -a_{(k+1),k}^{(k)} & b_{(k+1),k}^{(k)} \\ b_{(k+1),k}^{(k)} & t_{(k+1),k}^{(k)} \end{pmatrix} \frac{1}{\delta} \begin{pmatrix} t_{k,k}^{(k)} & -b_{k,k}^{(k)} \\ -b_{k,k}^{(k)} & -a_{k,k}^{(k)} \end{pmatrix} \begin{pmatrix} -a_{(k+1),k}^{(k)} & b_{(k+1),k}^{(k)} \\ b_{(k+1),k}^{(k)} & t_{(k+1),k}^{(k)} \end{pmatrix}, \tag{3.16}$$

where $\delta = -t_{k,k}^{(k)} a_{k,k}^{(k)} - (b_{k,k}^{(k)})^2$.
Repeating the 2×2 Gaussian elimination recursively on the transformed matrix leads to a substantial reduction of the fill-in especially in the studied sparse system.

However, to solve directly sparse linear systems, an efficient factorization need to be stable. The numerical stability is controlled by both the growth factor defined $\rho = \frac{\max_{i,j,k} a_{ij}^{(k)}}{\max_{i,j} a_{ij}}$ and the elements of L. The pivoting strategies of BunchâParlett and BunchâKaufman guarantee the bound $\rho \leq (2.57)^{2(n+m)-1}$ [26, 8]. In many cases, the growth factor stays far away from the bound, but it appears to be hard to find a substantially smaller upper bound for a general problem. For saddle point systems, a smaller bound for $\mathcal{F}$-matrices is given in [37].

Here all the pivots are of order 2 and are of the form

$$\begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix}. \tag{3.17}$$

Those pivots need to be invertible and their inverse should be properly bounded. In order to guarantee the numerical stability, we choose the best 2×2 pivots. We develop a strategy for selecting those pivots. Firstly, we test the minimum degree pivots in the transformed matrix and its compressed graph and we compute their determinants. If pivots do not match numerical requirements, then we alter the ordering of the transformed matrix in such a way we could maintain the same minimum degree ordering on the compressed graph. We do that by only permuting the second row/column in the actual pivot. Using this technique we track the best diagonal 2×2 pivots during the Gauss elimination. If there is no possibility of maintaining the same initial order of the compressed graph to guarantee numerical stability, then we alter this order in order to get go back to the classical LDL^T factorization algorithm with pivots of order 1 and 2.
It is clear that we pay much more attention on the construction of a suitable ordering that enables numerical stability as well as sparsity of the factor matrices, than focusing on the construction of the LDL^T factorization. Indeed, this latter is already extensively developed in many packages and solvers, for instance in MA47 [46].

3.3.3. Comparison with symmetric direct solvers

We consider the same set-up as in 3.2.2. In this section, we present results using the above-detailed factorization method, that we call $SBlock$ here, as the basis of a direct method. Our intention is to demonstrate the stability and sparsity of these factorizations when applied to randomized three dimensional test cases like in 3.2.2. Our main goal is to compare its performance with that of a general sparse symmetric indefinite solvers such as $MUMPS$. We concentrate on the solution of a single saddle point system and do not exploit the potential advantages of $SBlock$ within a sequence of saddle point systems.

We perform tests using MATLAB ®. For comparison, we consider results for the MATLAB command *ldl* and $MUMPS$. Both solvers compute an $\mathcal{L}\mathcal{D}\mathcal{L}^T$ factorization, where $\mathcal{L}$ is unit lower triangular and $\mathcal{D}$ is block diagonal with 1×1 and 2×2 blocks. They use $MA57$ code from the HSL mathematical software library [45]. It computes a sparsity-preserving ordering of the coefficient matrix $\widetilde{\mathcal{A}}$ and a scaling to ensure numerical stability. This code ois designed to robustly solve general symmetric indefinite sparse linear systems. It thus does not exploit the block structure of the matrix $\widetilde{\mathcal{A}}$.

	Matrix	$FE1$	$FE2$	$FE3$	$FE4$
	$2(n+m)$ nnz	2,410 14,427	4,104 24,608	6,504 39,660	10,000 60,054
$SBlock$	*Fill-in* Err	100,112 1.94e-10	266,800 1.75e-10	612,522 2.48e-10	1,216,652 1.22e-09
$UMFPACK-LU\ Matlab$	*Fill-in* Err	774,108 9.90e-13	2,817,520 1.75e-12	7,165,751 2.48e-12	15,218,709 2.83e-12
$MA57-LDL^T\ Matlab$	*Fill-in* Err	217,136 5.55e-09	624,756 1.62e-08	1,701,536 5.54e-09	3,583,097 1.69e-08
$MUMPS-LU$	*Fill-in* Err	574,624 2.94e-12	1,823,314 4.18e-12	4,800,112 3.48e-12	10,365,088 8.60e-12
$MUMPS-LDL$	*Fill-in* Err	303,299 1.51e-09	1,016,383 2.54e-09	2,860,465 2.48e-09	6,602,797 4.18e-08

Table 3.6.: Fill-in and numerical stability results for three-dimensional randomized finite element test matrices using $SBlock$, $UMFPACK$ and $MUMPS$ with AMD ordering

We note that many parameters in these codes can be set by the user and any modification of their values can lead to different performances. Consequently, preprocessing parameters are set similarly so that the codes perform equivalently. We recall that for each test matrix, the fill-in $nnz(\mathcal{L}+\mathcal{U})$ of LU factorization is given with a ratio of sparsity. This ratio is computed as $\frac{nnz(L+U)}{nnz(A)}$.

Table 3.6 is shown to contrast the numerical behavior of $SBlock$ with other direct solvers. We notice that the gain in memory can reach up to 50% to 70%, which is very significant from an industrial point of view. Beyond the efficiency of the proposed factorization in term of fill reduction, we have also obtained good results with regard to numerical stability. Indeed $SBlock$ is better than both Matlab's *ldl* and $MUMPS$ even if the size increases.

3.4. Conclusion

We achieved significant results in comparison to global direct solvers by using a new block factorization approach. This latter takes the advantage of the specific structure of the saddle point systems generated by the energy-based functional approach.

Much more attention was paid to construct a suitable sparse and stable direct method, than to focus on building an efficient LDL^T factorization. Through numerical experiments, this approach led to significant gain in term of fill-in reduction up to 50% in comparison with symmetric sparse solvers, and up to 90% in comparison with unsymmetric sparse solvers for the studied sequence of saddle point linear systems. In term of numerical stability, we outperformed symmetric sparse solvers, but we were less stable than unsymmetric ones. Hence, this approach showed how important it can be to take the structure of the coefficient matrix into account to build an associated compressed graph.

Even with this amount of fill reduction, the memory cost remains expansive which prohibits using direct approaches for large scale real life problems. Investigating those direct approaches has been conducted with the main goal of reducing the memory cost. Up to now, and a part of the interesting results, the proposed strategies have not achieved completely this objective. Those conclusions have motivated the research work presented in the next chapter. Indeed, we will use iterative methods instead of direct ones, and search to design efficient block preconditioners to solve the sequence of saddle point systems, using the knowledge we gain about factorization strategies.

Chapter 4

Constraint preconditioners for the projected saddle point system

Contents

4.1. A double explicit-implicit projections onto the constraints nullspace

When dealing with constrained optimization problems, it is natural to try to use the constraints to eliminate some of the variables from the problem, to obtain a simpler problem with fewer degrees of freedom.
Up to now, we did not set up a distinction between the several constraints of the studied constrained optimization problem. In Chapter 3, we considered the whole constraints and we solve the linear system globally. In this chapter, we make the distinction between them, and we recall that the existing constraints are twofold. Firstly, fixed linear or affine constraints, that are easy to eliminate explicitly as will be shown in section 4.1.1. Secondly, fixed sensors constraints, that we enforce in order to describe a quantifiable distance between the numerical and experimental models. Their elimination may cause an additional computation cost if done explicitly as will be demonstrated in section 4.1.2. To remedy to that, we present an implicit elimination approach later in this chapter.

4.1.1. First explicit projection of the saddle point system onto the nullspace of kinematic constraints

As mentioned in section 2.2.1, we recall that in order to practically introduce kinematic linear constraints, we usually choose a dual form to describe them through the Lagrange multipliers. This approach is used in many general-purpose finite element programs that are supposed to work as a black-box by minimizing guesses from its users. It increases the size of the problem by introducing new unknowns through Lagrange multipliers. Physically this set of unknowns represent constraint forces that would enforce the kinematic constraints applied to the unconstrained system.
We also recall that the approach drawbacks are twofold. On the one hand, the adjunction of Lagrange multipliers increases the number of degree of freedom of the whole problem, requiring expansion of the original stiffness and mass matrices, which means more complicated storage allocation procedures. This may be disadvantageous when the number of boundary conditions increases. On the other hand, it leads to a loss of the positivity property of the stiffness matrix. It becomes indefinite which restrains the use of many factorization methods and preconditioning schemes that rely on positive definiteness.
This is why we use nullspace projection in order to get a more suitable description of constrained structures when dealing with fixed and affine boundary conditions. Hence, instead of using the dual form through the Lagrange multipliers to constraint the problem with

affine constraints, we eliminate some of the variables from the problem, to obtain a simpler problem with fewer degrees of freedom. We do so by projecting the problem in the nullspace of the kinematic constraints as will be shown later in this section.

Let us recall here the equivalent variant of the coefficient matrix $\widetilde{\mathcal{A}}$ of the studied system presented before in section 2.2.3

$$\widetilde{\mathcal{A}} = \begin{pmatrix} -A & B^T & -C^T & C^T \\ B & T & C^T & 0 \\ -C & C & 0 & 0 \\ C & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \widetilde{E} & \widetilde{C}^T \\ \widetilde{C} & 0 \end{pmatrix}, \tag{4.1}$$

described using the following abbreviations

$$A = [K(\theta)],\ B = [K(\theta)] - \omega_{exp}^2 [M(\theta)],\ T = \frac{r}{1-r} \Pi^T [K_r] \Pi,$$

and where

$$\widetilde{E} = \begin{pmatrix} -A & B^T \\ B & T \end{pmatrix} \in \mathbb{R}^{2n \times 2n} \text{ and } \widetilde{C} = \begin{pmatrix} -C & C \\ C & 0 \end{pmatrix} \in \mathbb{R}^{2m \times 2n}.$$

The constraint matrix $\widetilde{C}$ is an augmented incidence matrix that is created from the constraint matrix C of the numerical model. This alternative structure is more appealing than the initial one (2.27), it represents a saddle point coefficient matrix with a null $(2,2)$ block, which is more suitable for a projection of $\widetilde{\mathcal{A}}$ onto the nullspace of the kinematic constraints $\widetilde{C}$.

In the following, we describe the coefficient matrix $\widetilde{\mathcal{A}}$ in the nullspace of $\widetilde{C}$. We need then to compute a nullspace basis

$$\widetilde{Z} \in \mathbb{R}^{2n \times 2(n-m)} \text{ such that } \operatorname{span}(\widetilde{Z}) = \ker(\widetilde{C}).$$

In order to build the nullspace basis $\widetilde{Z}$, we take advantage of the block structure of $\widetilde{C}$, in such a way that the computation cost is cut by half, as presented in the lemma below.

Lemma 4.1. *Let the matrix* $\widetilde{C} = \begin{pmatrix} -C & C \\ C & 0 \end{pmatrix} \in \mathbb{R}^{2m \times 2n}$ *be as defined above.*

If $Z \in \mathbb{R}^{n \times (n-m)}$ *is a nullspace basis of* C *then* $\widetilde{Z} = \begin{pmatrix} Z & 0 \\ 0 & Z \end{pmatrix}$ *is a nullspace basis of* $\widetilde{C}$.

Proof. The following equation proves the lemma

$$\widetilde{C}\widetilde{Z} = \begin{pmatrix} -C & C \\ C & 0 \end{pmatrix} \begin{pmatrix} Z & 0 \\ 0 & Z \end{pmatrix} = \begin{pmatrix} -CZ & CZ \\ CZ & 0 \end{pmatrix}.$$

□

It is then sufficient to compute the nullspace basis of the constraint matrix C in order to get the nullspace basis of $\widetilde{C}$. We recall that the matrix C is a full row rank matrix and if such is not the case, we find either that the problem is inconsistent or that some of the constraints are redundant and can be deleted without affecting the solution of the problem.

One standard method for computing a null space basis for C is known as the *fundamental basis* method [17]. In this method and under the assumption of full rank of C, it is possible to find a subset of m columns of C that are linearly independent. We then arrange those columns as the first m columns of C. Let P be a permutation matrix such that $CP = (C_m, C_{n-m})$, where, C_m is the first m linearly independent columns of C. Then, the columns of

$$Z = P \begin{bmatrix} -C_m^{-1}C_{n-m} \\ I_{n-m} \end{bmatrix}, \tag{4.2}$$

form a basis of the null space of C. This concept is applied in other forms using QR, LU or the singular value decomposition (SVD) of the matrix C [32]. All these techniques have a trade-off between sparsity and numerical stability. In the QR and SVD methods, we get an orthogonal basis of the nullspace that is dense and computationally expensive [70].

The nullspace basis Z needs to be sparse, well-conditioned, easy to apply. Those criteria are difficult to match together, actually Z can be rather ill-conditioned or dense. Paramount among them is sparsity. The problem of finding a sparse nullspace basis is shown to be NP-hard [31] and even If C is sparse that do not mean the sparsity of Z.
There is a technique that suits those purposes, which is the sparse Gaussian elimination approach that attempts to preserve sparsity while keeping rounding errors under control. We compute the nullspace basis Z by performing LU on the matrix C^T. This latter is called a *skinny* matrix, since its rows outnumbers its columns. As done above, there exists two permutation matrices, P used for stability, and Q used for sparsity such that

$$PC^TQ = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} U_1, \tag{4.3}$$

where $L_1 \in \mathbb{R}^{m\times m}$ is invertible. We define the nullspace basis such as

$$Z = P^T \begin{bmatrix} -L_1^{-T} L_2^T \\ \mathbb{I} \end{bmatrix}. \tag{4.4}$$

In Section 2.3.5, we saw that the saddle point coefficient matrix 4.1 can be reduced to a block triangular form using the basis

$$\begin{pmatrix} \widetilde{Y} & \widetilde{Z} & 0 \\ 0 & 0 & \mathbb{I}_{2m} \end{pmatrix}, \tag{4.5}$$

where

$$\widetilde{Y} \in \mathbb{R}^{2n\times 2m} \text{ such that } \operatorname{span}(\widetilde{Y}) = \operatorname{range}(\widetilde{C}^T) \text{ i.e. } \operatorname{span}\left(\begin{bmatrix}\widetilde{Y} & \widetilde{Z}\end{bmatrix}\right) = \mathbb{R}^{2n}.$$

Indeed, when taking $\widetilde{Y} = \widetilde{C}^T$, which is the description of range$(\widetilde{C}^T)$ in canonical basis, and when premultiplying 4.1 at right and left, it turns out that

$$\begin{pmatrix} \widetilde{C}\widetilde{E}\widetilde{C}^T & \widetilde{C}\widetilde{E}\widetilde{Z} & \widetilde{C}\widetilde{C}^T \\ \widetilde{Z}^T\widetilde{E}\widetilde{C}^T & \widetilde{Z}^T\widetilde{E}\widetilde{Z} & 0 \\ \widetilde{C}\widetilde{C}^T & 0 & 0 \end{pmatrix}. \tag{4.6}$$

(4.6) is an anti-triangular system. It is invertible iff $\widetilde{Z}^T\widetilde{E}\widetilde{Z}$ and $\widetilde{C}\widetilde{C}^T$ are invertible. Since $\widetilde{C}$ is of full row rank then $\widetilde{C}\widetilde{C}^T$ is symmetric and positive definite. Therefore, we need to prove that $\widetilde{Z}^T\widetilde{E}\widetilde{Z}$ is nonsingular, so that the coefficient matrix 4.6 is proved invertible.
Theorem 4.2 gives a necessary and sufficient condition to prove the nonsingularity of matrix $\widetilde{Z}^T\widetilde{E}\widetilde{Z}$.

Theorem 4.2. *The matrix $\widetilde{Z}^T\widetilde{E}\widetilde{Z}$ is invertible whenever*

$$Ker(Z^T B Z) \cap Ker(Z^T T Z) = \{0\} \tag{4.7}$$

Proof. Using the fact that $\widetilde{Z} = \begin{pmatrix} Z & 0 \\ 0 & Z \end{pmatrix}$, we obtain :

$$\widetilde{Z}^T\widetilde{E}\widetilde{Z} = \begin{pmatrix} Z & 0 \\ 0 & Z \end{pmatrix}^T \begin{pmatrix} -A & B \\ B & T \end{pmatrix} \begin{pmatrix} Z & 0 \\ 0 & Z \end{pmatrix} = \begin{pmatrix} -Z^T A Z & Z^T B Z \\ Z^T B Z & Z^T T Z \end{pmatrix} \tag{4.8}$$

We use the following LDL^T block decomposition

$$\begin{pmatrix} -A & B_1^T \\ B_2 & C \end{pmatrix} = \begin{pmatrix} \mathbb{I} & 0 \\ B_2 A^{-1} & \mathbb{I} \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & C + B_2 A^{-1} B_1^T \end{pmatrix} \begin{pmatrix} \mathbb{I} & A^{-1} B_1^T \\ 0 & \mathbb{I} \end{pmatrix} \tag{4.9}$$

The matrix $\widetilde{Z}^T \widetilde{E} \widetilde{Z}$ is congruent to the matrix

$$\begin{pmatrix} -Z^T A Z & 0 \\ 0 & Z^T T Z + (Z^T B Z)(Z^T A Z)^{-1}(Z^T B Z) \end{pmatrix}. \tag{4.10}$$

We know that the matrix $A = K$ is supposed to be positive definite on $\ker(C)$, this implies that $Z^T A Z$ is symmetric positive definite. Also the matrix

$$S = Z^T T Z + (Z^T B Z)(Z^T A Z)^{-1}(Z^T B Z)$$

is symmetric positive semidefinite matrix. Thus, the invertibility of $\widetilde{Z}^T \widetilde{E} \widetilde{Z}$ depends of the definiteness of S. Suppose that S is singular then there exists $V \neq 0$, such that $V^T S V = 0$ which means

$$V^T (Z^T T Z) V = 0 \text{ and } V^T [(Z^T B Z)(Z^T A Z)^{-1}(Z^T B Z)] V = 0.$$

As T is symmetric positive semidefinite then $V \in \ker(Z^T T Z)$. Similarly it is clear that

$$V \in \ker((Z^T A Z)^{-1} Z^T B Z) = \ker(Z^T B Z).$$

Finally $\widetilde{Z}^T \widetilde{E} \widetilde{Z}$ is nonsingular iff $\ker(Z^T B Z) \cap \ker(Z^T T Z) = \{0\}$. □

This condition is naturally fulfilled in mechanical measurements. Actually, let us consider the numerical finite element model of the constrained structure with ($K_c = Z^T K Z, M_c = Z^T M Z$) its stiffness and mass matrices. if ω is not an eigenfrequency, then $\ker(Z^T B Z) = \ker(K_c(\theta) - \omega^2 M_c(\theta)) = 0$, it follows that the condition below is fulfilled. On the other hand, if ω is an eigenfrequency, then there exists $u \neq 0$ such that $Z^T B Z u = (K_c(\theta) - \omega^2 M_c(\theta)) u = 0$. Since it is possible to choose a measurement configuration such that any eigenmode is observable, it implies that $Z^T T Z u \neq 0$, this satisfies the condition of invertibility.

Remark 4.3. *As mentioned in 2.30, we can produce the same process by considering the* *sensors degrees of freedom partitioning. let us take*

$$\widetilde{E}_s = \begin{pmatrix} -A_{ss} & -A_{st} & B_{ss}^T & B_{ts}^T \\ -A_{ts} & -A_{tt} & B_{st}^T & B_{tt}^T \\ B_{ss} & B_{st} & T_{ss} & 0 \\ B_{ts} & B_{tt} & 0 & 0 \end{pmatrix}. \tag{4.11}$$

When projecting onto the kinematic constraints as described in section 4.1.1, it yields that the projected coefficient matrix is such that

$$\mathcal{A}_{Z_s} = \widetilde{Z}_s^T \widetilde{E}_s \widetilde{Z}_s = \begin{pmatrix} -A_{Z_{ss}} & -A_{Z_{st}} & B_{Z_{ss}}^T & B_{Z_{ts}}^T \\ -A_{Z_{ts}} & -A_{Z_{tt}} & B_{Z_{st}}^T & B_{Z_{tt}}^T \\ B_{Z_{ss}} & B_{Z_{st}} & T_{Z_{ss}} & 0 \\ B_{Z_{ts}} & B_{Z_{tt}} & 0 & 0 \end{pmatrix}, \tag{4.12}$$

where

$$\widetilde{Z}_s = \begin{pmatrix} Z_s & 0 \\ 0 & Z_s \end{pmatrix},$$

is the nullspace basis of $\widetilde{C}_s$ and Z_s is the nullspace basis of C_s partitioned in the same way as in matrix 2.30. The subscript Z indicates the projected form of each block matrix.

Since the kinematic constraints are fixed and do not depend of the experimental frequency, the nullspace basis $\widetilde{Z}$ is computed once for the sequence of saddle point systems to solve.
In the case of the sensors constraints, we need to compute a different nullspace basis of $\begin{pmatrix} B_{Z_{ss}} & B_{Z_{st}} \end{pmatrix}$ for each system along the sequence of saddle point linear systems, which is presented in the next subsection.

The equivalent linear system to solve is then

$$\begin{pmatrix} \widetilde{C}\widetilde{E}\widetilde{C}^T & \widetilde{C}\widetilde{E}\widetilde{Z} & \widetilde{C}\widetilde{C}^T \\ \widetilde{Z}^T\widetilde{E}\widetilde{C}^T & \widetilde{Z}^T\widetilde{E}\widetilde{Z} & 0 \\ \widetilde{C}\widetilde{C}^T & 0 & 0 \end{pmatrix} \begin{pmatrix} x_Y \\ x_Z \\ w \end{pmatrix} = \begin{pmatrix} \widetilde{Y}^T f \\ \widetilde{Z}^T f \\ 0 \end{pmatrix}, \tag{4.13}$$

where

$$f = \begin{pmatrix} 0 \\ \frac{r}{1-r}\Pi^T[K_r]\phi_{exp} \end{pmatrix}, \begin{pmatrix} x_Y \\ x_Z \end{pmatrix} = x = \begin{pmatrix} \psi \\ \varphi \end{pmatrix}, w = \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix},$$

and the solution set of $\widetilde{C}x = 0$ is described by $x = \widetilde{Y}x_Y + \widetilde{Z}x_Z$.
The unknown w in equation 4.13 enables us to get reaction forces. Actually, any kinematic constraint may be replaced by a system of forces. Very often, reaction forces can be identified

with Lagrange multipliers, here w is one of these latter. Here, we are not interested by computing reaction forces, we thus focus our attention on computing the unknown x only. The third equation in the linear system 4.13 is $\widetilde{C}\widetilde{C}^T x_Y = 0$, since $\widetilde{C}$ is of full row rank then the matrix $\widetilde{C}\widetilde{C}^T$ is symmetric positive definite, which implies that $x_Y = 0$. Since $x = \widetilde{Y}x_Y + \widetilde{Z}x_Z = \widetilde{Z}x_Z$ then we need to obtain x_Z. To fulfill this purpose, equation 2 in the system 4.13 is the only linear equation to solve as will be discussed in upcoming sections. It is presented here such as

$$\begin{pmatrix} -Z^T AZ & Z^T BZ \\ Z^T BZ & Z^T TZ \end{pmatrix} \begin{pmatrix} x_{Z_1} \\ x_{Z_2} \end{pmatrix} = \begin{pmatrix} -A_Z & B_Z \\ B_Z & T_Z \end{pmatrix} \begin{pmatrix} x_{Z_1} \\ x_{Z_2} \end{pmatrix} = \begin{pmatrix} 0 \\ Z^T f \end{pmatrix}, \tag{4.14}$$

where $x_Z^T = \begin{pmatrix} x_{Z_1}^T & x_{Z_2}^T \end{pmatrix}$.

The findings of this section are twofold. Firstly, we describe the studied saddle point system in an equivalent form by using the nullspace method. This method is already used in the literature of saddle point systems [17], however it is enhanced here by taking advantage of the augmented structure of the studied system. We use a skinny LU technique in order to construct the nullspace basis, as already done in [70], then we adapt it to our case. Thanks to the specific structure we have, we cut by half the computational cost for obtaining the nullspace basis.

Secondly, we give an adequate proof of the invertibility of our system, using this new equivalent form. We also generate the new projected linear system to solve. This one have a saddle point structure.

4.1.2. Second projection of the projected system onto the nullspace of sensors constraints

After eliminating the kinematic constraints in the previous section, we eliminate the remaining constraints. Those latter are associated with the sensors degrees of freedom. We focus in this section on developing the second projection through the same explicit approach used in section 4.1.1. Then, due to its expansive computational cost, we propose an alternative implicit projection approach through solving the first projected system using constraint preconditioners.

Explicit projection onto the nullspace of sensors constraints

Let us develop the second explicit projection, in order to show how difficult it is, to compute the nullspace basis of the sensors constraints. We recall the projected saddle point system to solve

$$\begin{pmatrix} -A_Z & B_Z \\ B_Z & T_Z \end{pmatrix} \begin{pmatrix} x_{Z_1} \\ x_{Z_2} \end{pmatrix} = \begin{pmatrix} 0 \\ Z^T f \end{pmatrix}, \tag{4.15}$$

As shown in [41], we use a LDL^T factorization with pivoting on $T_Z = Z^T T Z$ so that

$$Q^T T_Z Q = JDJ^T, \tag{4.16}$$

where $J \in \mathbb{R}^{(n-m)\times s}$, $Q \in \mathbb{R}^{(n-m)\times(n-m)}$ a permutation matrix and $D \in \mathbb{R}^{s\times s}$ invertible such that $s_Z = rank(T_Z)$. We then add the variable $p = DJ^T Q^T x_{Z_2}$ so that we may write

$$\begin{pmatrix} -A_Z & 0 & B_Z Q \\ 0 & -D^{-1} & J^T \\ Q^T B_Z & J & 0 \end{pmatrix} \begin{pmatrix} x_{Z_1} \\ p \\ Q^T x_{Z_2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ Q^T(Z^T f) \end{pmatrix}. \tag{4.17}$$

Finally, we obtain a non regularized saddle-point as follows

$$\begin{pmatrix} H & F^T \\ F & 0 \end{pmatrix} \begin{pmatrix} t \\ Q^T x_{Z_2} \end{pmatrix} = \begin{pmatrix} 0 \\ -Q^T(Z^T f) \end{pmatrix}, \tag{4.18}$$

where

$$H = \begin{pmatrix} A_Z & 0 \\ 0 & D^{-1} \end{pmatrix} \in \mathbb{R}^{(n-m+s)\times(n-m+s)},$$

is the main matrix and

$$F = \begin{pmatrix} Q^T B_Z & J \end{pmatrix} \in \mathbb{R}^{(n-m)\times(n-m+s)},$$

is the constraint one, and $t = \begin{pmatrix} -x_{Z_1} \\ -p \end{pmatrix}$.

The rank of F is independent from B_Z one. Let us prove this in the lemma below.

Lemma 4.4. *Let $F = \begin{pmatrix} Q^T B_Z & J \end{pmatrix} \in \mathbb{R}^{(n-m)\times(n-m+s)}$ be as described above. Then F is of full row rank independently from the rank of B_Z.*

Proof. We recall that $B_Z = K_Z - \omega_{exp}^2 M_Z$.
If ω_{exp}^2 is an eigenvalue of the generalized eigenproblem

$$(K_Z - \lambda M_Z)V = 0$$

then B_Z is singular, its nullspace is then the corresponding modal subspace. From equation 4.16, we note that $\ker(J) \subset \ker(T_Z)$. Also as shown in Theorem 4.2, the nonsingularity property implies $\ker(B_Z) \cap \ker(T_Z) = \{0\}$. Thus, $\ker(B_Z) \cap \ker(J) = \{0\}$, which yields the linear independence of F rows.
If ω_{exp}^2 is not an eigenvalue then F is directly of full row rank. □

The solution set of $Ft = -Q^T(Z^T f)$ is described by $t = Ut_U + Vt_V$ where

- $V \in \mathbb{R}^{(n-m+s)\times s}$ a matrix such that $\text{span}(V) = \ker(F)$,
- and $U \in \mathbb{R}^{(n-m+s)\times(n-m)}$ a matrix such that $\text{span}(U) = \text{range}(F^T)$.

Using the following basis transformation

$$\begin{pmatrix} t \\ Q^T x_{Z_2} \end{pmatrix} = \begin{pmatrix} U & V & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} t_U \\ t_V \\ Q^T x_{Z_2} \end{pmatrix}, \tag{4.19}$$

and as done in section 4.1.1, we obtain the following system

$$\begin{pmatrix} FHF^T & FHV & FF^T \\ V^T HF^T & V^T HV & 0 \\ FF^T & 0 & 0 \end{pmatrix} \begin{pmatrix} t_U \\ t_V \\ Q^T x_{Z_2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -Q^T(Z^T f) \end{pmatrix}, \tag{4.20}$$

which yields to 3 reduced linear systems

$$\begin{cases} FF^T t_U = -Q^T(Z^T f), \\ \\ V^T HV t_V = -V^T HF^T t_U, \\ \\ FF^T Q^T x_{Z_2} = -F(HF^T t_U + HV t_V). \end{cases} \tag{4.21}$$

t_U is determined by the the first $(n-m)\times(n-m)$ system. Since F is of full row rank as assumed below then FF^T is symmetric and positive definite so that we can solve the first and third linear systems in equation 4.21 by Cholesky factorization or by conjugate gradient (CG).

Let us prove in the following that the coefficient matrix $V^T HV$ of the second equation is also symmetric positive definite.

Theorem 4.5. *Let the matrix $V^T HV \in \mathbb{R}^{s\times s}$ be as described above, then it is symmetric positive definite.*

Proof. Let $x \in \mathbb{R}^{s\times s}$ be a nonzero, we have $Vx \neq 0$ since V has full column rank. It yields

$$x^T(V^THV)x = (Vx)^T \begin{pmatrix} Z^TAZ & 0 \\ 0 & D^{-1} \end{pmatrix} (Vx) > 0. \tag{4.22}$$

□

Hence, it is also possible to solve the second linear system in 4.21 by Cholesky factorization or conjugate gradient method.
Once the solution t_V is found, the solution set of $Ft = -Q^T(Z^Tf_2)$ is then described by $t = Ut_U + Vt_V$. Then, x_{Z_2} can be obtained using the third reduced system . It is possible to use the same Cholesky factorization for both first and third equations.

This approach details the explicit use of the basis V of the nullspace

$$\ker\left(\begin{bmatrix} Q^TB_Z & J \end{bmatrix}\right)$$

in order to solve the sequence of saddle point systems. We highlight the need to compute an explicit nullspace basis V for each saddle point system because it changes at each new experimental frequency. Even if it is possible to maintain the same symbolic structure while computing V through skinny factorization of F^T, it remains a costly process.
In the next subsection, we introduce an alternative implicit approach that avoids computing an explicit basis of the nullspace of sensors constraints.

Replacing the explicit projection by an implicit approach

In order to avoid the expensive cost to find the nullspace basis explicitly, it seems more judicious to use the implicit variant of the nullspace method.
A modern version of the null-space method is to implicitly projecting onto the null space through the class of projected Krylov methods [99]. Since those methods are proved to be mathematically equivalent to applying a Krylov subspace method preconditioned with a constraint preconditioner [58, 59], we use and adapt this preconditioner to replace the explicit projection.
We consider in the following the saddle point coefficient matrix $\mathcal{A}_Z$ and its associated constraint preconditioner $\mathcal{G}_Z$ as follows

$$\mathcal{A}_Z = \begin{pmatrix} -A_Z & B_Z \\ B_Z & T_Z \end{pmatrix} \in \mathbb{R}^{2(n-m)\times 2(n-m)} \text{ and } \mathcal{G}_Z = \begin{pmatrix} -G_Z & B_Z \\ B_Z & T_Z \end{pmatrix} \in \mathbb{R}^{2(n-m)\times 2(n-m)}. \tag{4.23}$$

Unlike the problems studied in the saddle point literature, and especially in constraint preconditioners one, we note here that the coefficient matrix of the projected system $\mathcal{A}_Z$ has a square constraint block B_Z.

In the next section, we study the properties of the preconditioned coefficient matrix $\mathcal{G}_Z^{-1}\mathcal{A}_Z$ for the studied case. Then we adapt the constraint preconditioner $\mathcal{G}_Z$ to be used with the special structure of the projected saddle point system 4.14, by proposing better fit approximations of the Schur complement.

4.2. Constraint preconditioners approximations for the projected saddle point system

We recall that we are using the constraint preconditioner in order to eliminate implicitly the sensors constraints as mentioned in previous section. We propose here to adapt the constraint preconditioner results in [28, 42] to suit the case of the studied problem. Then, we propose a physics-based approximation of the constraint preconditioner.

4.2.1. Spectral characterization of the preconditioned matrix

We recall the projected linear system to solve

$$\begin{pmatrix} -A_Z & B_Z \\ B_Z & T_Z \end{pmatrix} \begin{pmatrix} x_{Z1} \\ x_{Z2} \end{pmatrix} = \begin{pmatrix} 0 \\ Z^T f \end{pmatrix}. \tag{4.24}$$

Here, the saddle point coefficient matrix $\mathcal{A}_Z$ and the constraint preconditioner $\mathcal{G}_Z$ satisfy all the following assumptions that are valid during the whole chapter

- the constraint matrix $B_Z = K_Z - \omega_{exp}^2 M_Z \in \mathbb{R}^{(n-m)\times(n-m)}$ is square and maybe rank deficient if $\omega_{exp}^2 \in Sp(K_Z, M_Z)$,
- $T_Z \in \mathbb{R}^{(n-m)\times(n-m)}$ is symmetric and positive semidefinite, and has rank s equal to the number of sensors, where $0 < s < n-m$.

In light of those assumptions, we bring out the following results that we use later in this section

- Since $T_Z = Z^T T Z = Z^T(\frac{r}{1-r}\Pi^T K_r \Pi)Z$, it is possible to consider a factorization of T_Z such as $T_Z = LDL^T$ where $L \in \mathbb{R}^{(n-m)\times s}$, and $D \in \mathbb{R}^{s\times s}$ is symmetric and positive definite.

- If we consider $Q \in \mathbb{R}^{(n-m)\times(n-m-s)}$ such that $\text{range}(Q) = \ker(T_Z)$, then from the condition of invertibility $\ker(B_Z) \cap \ker(T_Z) = \{0\}$, we conclude that $Q^T B_Z \in \mathbb{R}^{(n-m-s)\times(n-m)}$ has full row rank $n-m-s$. We consider $N \in \mathbb{R}^{(n-m)\times s}$ such that $\text{range}(N) = \ker(Q^T B_Z)$.
- Since $\text{range}(L) = \text{range}(C)$, we can describe $\mathbb{R}^{(n-m)\times(n-m)}$ using the basis $\begin{pmatrix} L & Q \end{pmatrix}$.

Let us characterize the spectrum of the constraint preconditioner $\mathcal{G}_Z$ for the studied problem. Using the fundamental nullspace basis of T_Z which is described as follows

$$N_f = \begin{pmatrix} \mathbb{I}_{n-m} & 0 & 0 \\ 0 & L & Q \end{pmatrix}.$$

It turns out that the equivalent coefficient matrix is

$$\mathcal{A}_{ZN} = N_f^T \mathcal{A}_Z N_f = \begin{pmatrix} -A_Z & B_Z L & B_Z Q \\ L^T B_Z & L^T T_Z L & 0 \\ Q^T B_Z & 0 & 0 \end{pmatrix} \tag{4.25}$$

This approach has been used in interior point optimization problems [42] when the constraint matrix is a rectangular one. But it remains true when B_Z is square and singular like here. We obtain then

$$\mathcal{G}_{ZN}^{-1} \mathcal{A}_{ZN} = \begin{pmatrix} -A_Z & B_Z L & B_Z Q \\ L^T B_Z & L^T T_Z L & 0 \\ Q^T B_Z & 0 & 0 \end{pmatrix} \tag{4.26}$$

Let us take

$$R = \begin{pmatrix} -A_Z & B_Z L \\ L^T B_Z & L^T T_Z L \end{pmatrix} \text{ and } P = \begin{pmatrix} Q^T B_Z & 0 \end{pmatrix}, \tag{4.27}$$

such that

$$\mathcal{G}_{ZN}^{-1} \mathcal{A}_{ZN} = \begin{pmatrix} R & P^T \\ P & 0 \end{pmatrix} \tag{4.28}$$

Using the assumptions in the beginning of this section, we have that R is invertible and P is of full row rank. This is the case already studied in the literature [28]. We present in the following some theorems that describe the same results we got when B_Z is singular.

Theorem 4.6. *Let $\mathcal{G}_Z$ and $\mathcal{A}_Z$ be as defined before. We suppose that B_Z is singular. The matrix $\mathcal{G}_Z^{-1} \mathcal{A}_Z$ has*

- *an eigenvalue at 1 with multiplicity $2(n-m)-s$,*

- *s eigenvalues which are defined by the generalized eigenvalue problem*

$$N^T(A_Z + B_Z^T LD^{-1}L^T B_Z)Nv = \lambda N^T(G_Z + B_Z^T LD^{-1}L^T B_Z)Nv. \tag{4.29}$$

The general proof of this theorem can be found in [42]. Theoretically, applying the constraint preconditioner enables to cluster nearly the whole spectrum to 1. In industrial applications, the number of sensors s is negligible in comparison to the number of physical degree of freedom n.

Using this theorem, it is possible to conclude some conditions in order to guarantee that the eigenvalues are real, which are

- either $N^T(A_Z + B_Z^T LD^{-1}L^T B_Z)N$ or $N^T(G_Z + B_Z^T LD^{-1}L^T B_Z)N$ is positive definite
- both $N^T(A_Z + B_Z^T LD^{-1}L^T B_Z)N$ and $N^T(G_Z + B_Z^T LD^{-1}L^T B_Z)N$ are positive definite

Theorem 4.7. *Let $\mathcal{G}_Z$ and $\mathcal{A}_Z$ be as defined before. We suppose that B_Z is singular. If $G_Z + B_Z^T LD^{-1}L^T B_Z$ is positive definite, then the matrix $\mathcal{G}^{-1}\mathcal{A}$ has $2(n-m)$ eigenvalues as defined in the above theorem and (n-m)+i+j linearly independent eigenvectors. There are :*

- *$n-m$ eigenvectors of the form $\begin{pmatrix} 0^T & y^T \end{pmatrix}$ that corresponds to the case $\lambda = 1$,*
- *$i(0 \leq i \leq (n-m))$ eigenvectors of the form $\begin{pmatrix} x^T & y^T \end{pmatrix}$ arising from $A_Z x = G_Z x$ for which the i vectors x are linearly independent and $\lambda = 1$.*
- *$j(0 \leq i \leq (n-m))$ eigenvectors of the form $\begin{pmatrix} x^T & y^T \end{pmatrix}$ that correspond to the case $\lambda \neq 1$.*

We now consider the degree of the minimal polynomial of the preconditioned matrix which determines the convergence behavior of a Krylov subspace method such as GMRES. If we use the results cited before when $G_Z + B_Z^T LD^{-1}L^T B_Z$ is positive definite, then the dimension of Krylov subspace $\mathcal{K}(\mathcal{G}^{-1}\mathcal{A}, b)$ is at most $\min\{s+2, 2(n-m)\}$.

Now from the spectral characterization presented here it is possible to conclude that the proposed approach is efficient theoretically. But, we need to find and build in practice suitable approximations for the constraint preconditioner blocks, especially for the (1,1) block A_Z and the Schur complement S_Z. We propose next some possible factorizations and approximations for the constraint preconditioner.

4.2.2. Factorization of constraint preconditioners and Schur complement approximations

The coefficient matrix $\mathcal{A}_Z$ of the projected system 4.24 as well as the constraint preconditioner $\mathcal{G}_Z$ are not positive definite, which means we cannot apply the PCG algorithm. However we can treat the problem using a general nonsymmetric and preconditioned GMRES. We recall that preconditioning is typically used with Krylov projection methods to alter the spectrum and hence accelerate the convergence rate of iterative techniques, here it is also used to eliminate the sensors constraints.

Since we solve the preconditioned system in each iteration of the iterative method, we need a suitable direct factorization that can be reused symbolically for several systems. We present a possible factorization in the following

$$\mathcal{G}_Z = \begin{pmatrix} \mathbb{I}_{n-m} & 0 \\ -B_Z G_Z^{-1} & S_Z \end{pmatrix} \begin{pmatrix} -G_Z & B_Z \\ 0 & \mathbb{I}_{n-m} \end{pmatrix}, \tag{4.30}$$

where $S_Z = T_Z + B_Z G_Z^{-1} B_Z$ is the Schur complement of $-G_Z$.

$$\mathcal{G}_Z^{-1} = \begin{pmatrix} -G_Z^{-1} & G_Z^{-1} B_Z \\ 0 & \mathbb{I}_{n-m} \end{pmatrix} \begin{pmatrix} \mathbb{I}_{n-m} & 0 \\ S_Z^{-1} B_Z G_Z^{-1} & S_Z^{-1} \end{pmatrix}. \tag{4.31}$$

Remark 4.8. *It is also possible to present the constraint preconditioner using this LDL^T factorization based on the Schur complement of G_Z*

$$\mathcal{P} = \begin{pmatrix} -G_Z & B_Z^T \\ B_Z & T_Z \end{pmatrix} = \begin{pmatrix} \mathbb{I} & 0 \\ -B_Z G_Z^{-1} & \mathbb{I} \end{pmatrix} \begin{pmatrix} -G_Z & 0 \\ 0 & T_Z + B_Z G_Z^{-1} B_Z^T \end{pmatrix} \begin{pmatrix} \mathbb{I} & -G_Z^{-1} B_Z^T \\ 0 & \mathbb{I} \end{pmatrix}. \tag{4.32}$$

The constraint preconditioner is implemented as

$$\begin{pmatrix} -G_Z & B_Z^T \\ B_Z & T_Z \end{pmatrix}^{-1} = \begin{pmatrix} -G_Z^{-1} & 0 \\ 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} \mathbb{I} & -B_Z^T \\ 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} \mathbb{I} & 0 \\ 0 & (T_Z + B_Z G_Z^{-1} B_Z^T)^{-1} \end{pmatrix} \begin{pmatrix} \mathbb{I} & 0 \\ B_Z G_Z^{-1} & \mathbb{I} \end{pmatrix} \tag{4.33}$$

Many algorithms for solving saddle point systems depend on the availability of good approximations for the (1,1) block $-A_Z$ and for the Schur complement S_Z. The construction of such approximations is a strongly problem-dependent matter.

Here we build a specific constraint preconditioner $\mathcal{P}_{Chol}$ that relies on an approximation of the Schur complement. This approximation is mostly based on structural dynamics.

The constraint preconditioner $\mathcal{P}_{Chol}$

Let us first propose an approximation of the Schur complement. Actually, we observe that

$$S_Z = T_Z + B_Z A_Z^{-1} B_Z = T_Z - A_Z - 2\omega_{exp}^2 M_Z + \omega_{exp}^4 M_Z K_Z^{-1} M_Z. \tag{4.34}$$

Let us consider again the spectral decomposition of K_Z and M_Z more explicitly with associated eigenvalues as done in 2.31

$$U^T K_Z U = \underset{1\leq j\leq N}{\mathrm{Diag}}\,(\omega_j^2), \;\; U^T M_Z U = \mathbb{I}, \tag{4.35}$$

where $0 \leq \omega_1^2 \leq ... \leq \omega_N^2$ are the N eigenvalues (counting possible multiplicities), and $U = (u_1, u_2, ..., u_N)$ their correspondent M-orthonormalized eigenvectors. We obtain :

$$U^T B_Z U = U^T (K_Z - \omega_{exp}^2 M_Z) U = \underset{1\leq j\leq N}{\mathrm{Diag}}\,(\omega_j^2 - \omega_{exp}^2). \tag{4.36}$$

Through above expressions and after some algebraic simplification we get

$$B_Z A_Z^{-1} B_Z = U^{-T} \left[\underset{1\leq j\leq N}{\mathrm{Diag}} \left(\frac{(\omega_j^2 - \omega_{exp}^2)^2}{\omega_j^2} \right) \right] U^{-1} \tag{4.37}$$

Since we are more interested by low experimental frequencies in industrial applications $\omega_{exp} \ll \omega_N$, we can approximate the dense matrix $B_Z A_Z^{-1} B_Z$ by the sparse matrix $K_Z = -A_Z$.

$$B_Z A_Z^{-1} B_Z \approx -A_Z. \tag{4.38}$$

Actually, we note that the equation 4.34 can be seen as a polynomial approximation of a function that has ω_{exp}^2 as an argument.

Let us then introduce the approximation of the Schur complement S_Z we consider here

$$\mathring{S}_Z = T_Z - A_Z \approx S_Z. \tag{4.39}$$

$\mathring{S}_Z$ is symmetric and positive definite and admits an exact Cholesky factorization $\mathring{S}_Z = \mathring{L}_S \mathring{L}_S^{\,T}$. Since A_Z is also positive definite, we can take $G_Z = L_A L_A^T$ an exact Cholesky decomposition as an approximation of this block. Consequently, we present the preconditioner

$\mathcal{P}_{Chol}$

$$\mathcal{P}_{Chol}^{-1} = \begin{pmatrix} -L_A L_A^T & B_Z \\ B_Z & T_Z \end{pmatrix}^{-1} \equiv \begin{pmatrix} -L_A^{-T} L_A^{-1} & L_A^{-T} L_A^{-1} B_Z \\ 0 & \mathbb{I}_{n-m} \end{pmatrix} \begin{pmatrix} \mathbb{I}_{n-m} & 0 \\ \mathring{L}_S^{-T} \mathring{L}_S^{-1} B_Z D^{-1} & \mathring{L}_S^{-T} \mathring{L}_S^{-1} \end{pmatrix}. \tag{4.40}$$

The constraint preconditioner has an approximation of the Schur complement that is more problem-dependent. It enables to replace the denser matrix $S_Z = T_Z + B_Z A_Z^{-1} B_Z$ by a sparse matrix, which reduces enormously the computational cost as will be shown later in this chapter and in Chapter 5. However, this preconditioner rely on a direct factorization based approximations, and direct factorization is known to be a less scalable approach in parallel framework.

The preconditioner $\mathcal{P}_{Chol}$ is incorporated in the iterative method $GMRES$. The default convergence test is based on the L_2-norm of the residual. Convergence (or divergence) is decided by three quantities : the decrease of the residual norm relative to the norm of the right hand side, $rtol$, the absolute size of the residual norm, $atol$, and the relative increase in the residual, $dtol$. Convergence is detected at iteration k if

$$||r_k||_2 = ||\mathcal{A}x_k - b||_2 < max(rtol.||b_2||_2, atol), \tag{4.41}$$

and divergence is detected whenever

$$||r_k||_2 = ||\mathcal{A}x_k - b||_2 > dtol.||b_2||_2. \tag{4.42}$$

When solving large scale real life problems, the constraint preconditioner may be unstable and ill-conditioned. To reach satisfying numerical results, we choose to use the constraint preconditioner at right. Actually, right-preconditioning is better for making the residual small and for debugging unstable preconditioners. It can be also attractive because the preconditioned residual $||\mathcal{P}(x_k - x)||_2$ is equal to the true error $||x_k - x||$.

Remark 4.9. *It is worth mentioning that the number of iterations of the Krylov subspace method is generally different if $\mathcal{P}$ is used as a left, right or split preconditioner, even though the spectra of the associated preconditioned matrices are identical. In particular, the stopping criterion is evaluated with $\mathcal{A}$ replaced by the left-preconditioned operator $\mathcal{P}^{-1}A$ and the preconditioned residual $\mathcal{P}^{-1}(\mathcal{A}x_k - b)$ or with the right-preconditioned operator $\mathcal{A}\mathcal{P}^{-1}$ and the error $\mathcal{P}(x_k - x)$.*

Let us present here the developed algorithm that we use later for our applications.

Algorithm 4.1: Double explicit-implicit projection iterative approach to solve the saddle point linear system 2.16

Data: The linear system 2.16

Result: Solution fields $\{\psi\}$ and $\{\varphi\}$

- Permuting the saddle point system to obtain the structure in 4.1
- **First explicit projection**
  - Computing the skinny LU factorization with pivoting of the constraint matrix C

$$PC^TQ = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} U_1,$$

  where $L_1 \in \mathbb{R}^{m\times m}$ is invertible.
  - Construct the nullspace basis Z of the constraint matrix C

$$Z = P^T \begin{bmatrix} -L_1^{-T}L_2^T \\ \mathbb{I} \end{bmatrix}.$$

  - Construct the nullspace basis $\widetilde{Z}$ of the constraint matrix $\widetilde{C}$ as

$$\widetilde{Z} = \begin{pmatrix} Z & 0 \\ 0 & Z \end{pmatrix}$$

  - Building the projected saddle point system

$$\begin{pmatrix} -A_Z & B_Z \\ B_Z & T_Z \end{pmatrix} \begin{pmatrix} x_{Z1} \\ x_{Z2} \end{pmatrix} = \begin{pmatrix} 0 \\ Z^T f \end{pmatrix}. \qquad (*)$$

- **Second implicit projection**
  - Build the constraint preconditioner $\mathcal{G}_Z$ used to eliminate sensors constraints

$$\mathcal{G}_Z^{-1} = \begin{pmatrix} Solve_A(-G_Z^{-1}, -\hat{G}_Z^{-1}) & 0 \\ 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} \mathbb{I} & -B_Z^T \\ 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} \mathbb{I} & 0 \\ 0 & Solve_S(S_Z^{-1}, \hat{S}_Z^{-1}) \end{pmatrix} \begin{pmatrix} \mathbb{I} & 0 \\ B_Z Solve_A(-G_Z^{-1}, -\hat{G}_Z^{-1}) & \mathbb{I} \end{pmatrix}$$

    - Choose G_Z the approximation of the (1,1) block A_Z.
    - Choose the solution method $Solve_A$ and its preconditioner $\hat{G}_Z$.
    - Choose S_Z the approximation of the Schur complement.
    - Choose the solution method $Solve_S$ and its preconditioner $\hat{S}_Z$.
  - Solve the projected system $(*)$ using the preconditioned GMRES with the constraint preconditioner $\mathcal{G}_Z$ in outer loop.
- Obtain

$$\begin{pmatrix} \psi \\ \varphi \end{pmatrix} = \begin{pmatrix} Z & 0 \\ 0 & Z \end{pmatrix} \begin{pmatrix} x_{Z1} \\ x_{Z2} \end{pmatrix}$$

We note that since G_Z and S_Z are both symmetric positive definite, then we use the conjugate gradient method in the inner solution methods $Solve_A$ and $Solve_S$ in the constraint preconditioner $\mathcal{P}_{Chol}$.

In the next section, we showcase the efficiency of the proposed algorithm 4.1 and we evaluate the performance of different approximations of the blocks A_Z and S_Z in the constraint preconditioner.

4.3. Academic application to structural mechanics

The iterative resolution process is illustrated first on an academic three-beam structural system. More numerical results are presented in Chapter 5 on large industrial applications. The geometry of the studied structure is depicted in Figure 4.1 (left) while its finite element model is presented in Figure 4.1 (right). The structure is composed of different types of finite elements.

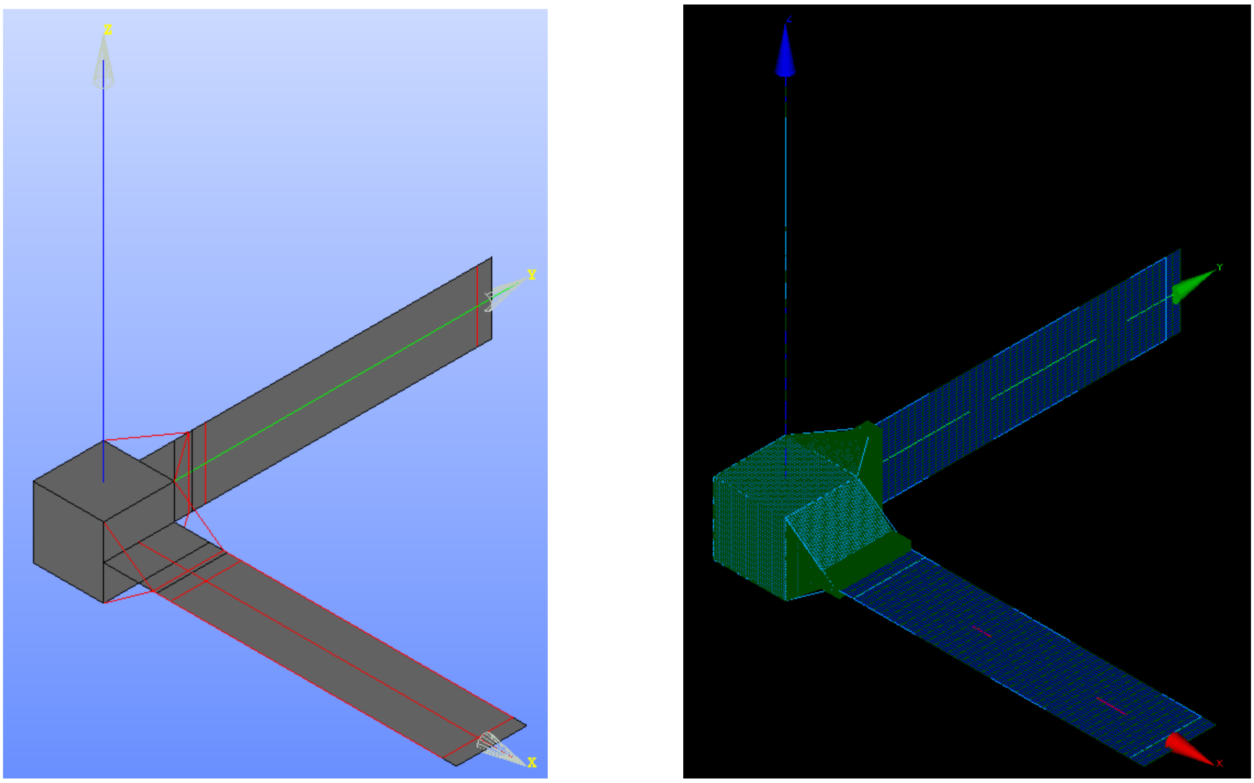


Figure 4.1.: The geometry and FE model of the tree-beam structure

We consider 4 different test cases generated from the finite element model of the three-beam structural system. Each one present different number of physical and Lagrange degrees of freedom. They are shown in Table 4.1.

Matrix	Physical dofs (n)	Lagrange dofs (m)	System size	System nnz	condition number
A_1	10,074	873	21,894	1,798,539	2.1e+08
A_2	13,497	1,089	29,172	2,503,134	2.5e+08
A_3	22,791	1,593	48,768	4,490,766	3.2e+08
A_4	64,506	3,273	135,558	13,778,775	7.1e+08

Table 4.1.: Presentation of the global coefficient matrices of each test case

Here the mechanical software *Code_Aster*® enables us to get a full row rank constraint matrix C for all test cases. Also, those constraints are integrated in order to forbid the rigid body motions of the structure. For all the test cases, we choose an experimental frequency near to the second eigenfrequency, which means that B_Z is ill-conditioned. This case is the most challenging since the experimental configuration needs to enable observing all model eigenmodes to ensure the nonsingularity of the generated saddle point system. The experimental mesh is constructed artificially to fulfill this requirement.

The condition number is estimated using MUMPS in double precision arithmetics as a direct solver of the linear system. Regarding this parameter (around 10^8), the solution may be challenging for iterative solution methods. The condition number can be explained by the quality of the mesh: we guess in Figure 4.1 that the mesh elements have a strong spatial variations. We also deduce from the condition number that all test cases have nonsingular coefficient matrices, which confirms that all solvability conditions mentioned in Chapter 2 are satisfied.

4.3.1. Implementation considerations

In this section, we succinctly describe the routines developed in both the mechanical software *Code_Aster*® [1], the direct solver SuperLU [77] and the PETSc library (Portable, Extensible Toolkit for Scientific Computation) [11], related to the developed iterative algorithm 4.1 studied in this manuscript. More precisely, we emphasize that the implementation has been done in a parallel framework. Finally, we notify the reader that this section refers to several routines specific to PETSc [11].

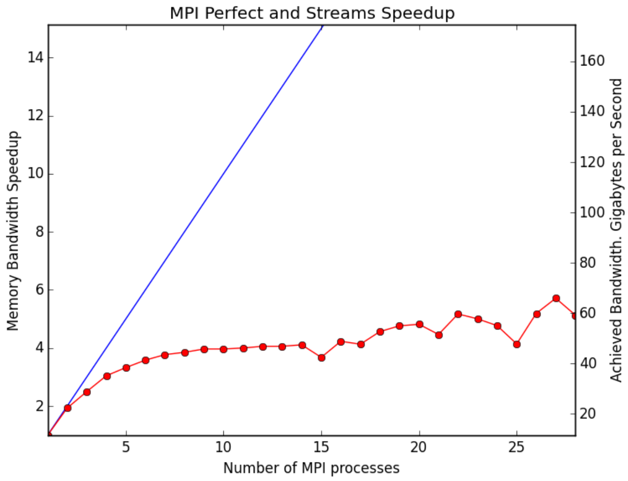
The developments are split in three main parts. The first one gives information on how the saddle point structure of the systems is taken into account in the code; this has been particularly developed in order to define the constraint preconditioner. Currently in *Code_Aster*®, the available preconditioning techniques treat the global system and unfortunately do not take into account the saddle point structure of the systems. We developed in the mechanical software a routine that enables to retrieve the stiffness matrix K, the mass matrix M, the constraint matrix, the observation matrix Π, the norm matrix K_r and the right hand side. We used *Code_Aster*® global ordering, which makes possible to identify the nature of each degree of freedom, and to extract the matrix blocks. Once the different blocks were retrieved, we used local PETSc ordering in order to identify unknowns on each process. For sake of flexibility of implementation, we chose to implement the system with the function `MATSHELL` in PETsc to define our own matrix type, then `PCSHELL` to construct our new preconditioner class. We emphasize that the implementation has been done in a parallel framework.

The second part illustrates the developments carried out both in standalone in order to get an explicit nullspace basis from the constraint matrix. The code uses an interface developed in Fortran 90 to the direct solver SuperLU, which is a a general purpose library written in C and callable from either C or Fortran for the direct solution of large, sparse, nonsymmetric systems of linear equations. Its main feature is that it enables getting access explicitly to the factor matrices. Those are then used to construct the nullspace basis as described in equation 4.4. We note that we used the routine `amd` that performs minimum degree ordering in order to get sparse factors.

The third part main idea is to use the PETSc library as an iterative solver for the solution of each linear system in the sequence, and more specifically the GMRES method, as justified in section 4.2.2. This approach is already available in *Code_Aster*® , thanks to an interface developed in Fortran 90. However, we made developments in standalone in C, since we need to first transform our system through explicit nullspace projection. We used the PETSc preconditioner framework called `PCFIELDSPLIT` to implement the block solvers in PETSc, with `PCFieldSplitSetIS` to indicate exactly which rows/columns of the matrix belong to a particular block.
Since GMRES in PETSc Library is implemented with left-preconditioning by default, we used FGMRES in which only right preconditioning is supported, up to an accuracy of 10^{-9} for the applications of this chapter. The program implemented works in parallel. A very close attention is paid to the ordering of unknowns especially in this parallel framework.

All experiments carried out on this chapter were performed on the Aster5 cluster, an IBM IDATAPLEX computer located at EDF R&D Data Center (each node of Aster5 is equipped with 2 Intel Xeon E5 â 2600, each running 12 cores at 2.7 Ghz). Physical memory available on a given node (24 cores) of Aster5 ranges from 64 GB to 1 TB. This code was compiled by the Intel compiler suite with the best optimization options and linked with the Intel

MKL BLAS and LAPACK subroutines. Since we use PETSc with this parallel system that supports MPI, the figure below presents a summary of the bandwidth received with different number of MPI processes and potential speedups.



We emphasize that when studying the scalability of the iterative approach, we use up to 16 processors of only one node. In practice, even if we do not consider parallel framework in the theoretical part of this chapter, this framework allows us to consider selected large-scale industrial problems, and to prove their performance in regards to a limited amount of computational time on a moderate number of cores.

4.3.2. Results of the first explicit nullspace projection

As shown in section 4.1.1, the projection onto the nullspace of kinematic constraints is performed through the computation of an explicit basis Z of the nullspace of the constraint matrix C. This enables us to study the reduced system (4.14) instead of the global system (2.16).

Matrix A	Physical dofs (n)	Lagrange dofs (m)	System size	System nnz	The constraint matrix C	The nullspace basis Z	CPU Time (sec)	Reduced system size	Reduced system nnz
A_1	10,074	873	21,894	1,798,539	$873 \times 10{,}074$ $nnz = 3{,}387$	$10{,}074 \times 9{,}201$ $nnz = 12{,}890$	1.3e-2	18,402	1,943,373
A_2	13,497	1,089	29,172	2,503,134	$1{,}089 \times 13{,}497$ $nnz = 4{,}146$	$13{,}497 \times 12{,}408$ $nnz = 16{,}977$	2.9e-2	24,816	2,751,990
A_3	22,791	1,593	48,768	4,490,766	$22{,}791 \times 22{,}791$ $nnz = 6{,}062$	$1{,}593 \times 21{,}198$ $nnz = 27{,}847$	6.4e-2	42,396	5,071,433
A_4	64,506	3,273	135,558	13,778,775	$64{,}506 \times 3{,}273$ $nnz = 12{,}555$	$64{,}506 \times 61233$ $nnz = 74{,}794$	18.8e-2	122,466	16,358,756

Table 4.2.: Presentation of the global and reduced coefficient matrices of each test case and description of the projection onto the nullspace of kinematic constraints

We note that the reduced system size is comparable to the global system size, which is due to low number of Lagrange degrees of freedom. Besides, the computation time relative to the nullspace basis remains low even for large test cases.

4.3.3. Comparing different approximations of the constraint preconditioner $\mathcal{P}_{Chol}$

Before comparing above approximations of the constraint preconditioner, let us first set up some important parameters.

Choosing the restart parameter

It is very difficult practically to choose an appropriate value of the restart number m. Traditionally, it has been assumed that the larger the value of restart number m, the fewer iterations are required for convergence. In fact, a large m improves the information in the GMRES residual polynomial (see, e.g., [69]). Furthermore, a large enough m for GMRES(m) can to some extent reduce the impediment to superlinear convergence [110] and may be required to avoid stalling [98]. Nevertheless, if m is too large, the goal of restarting as a means of reducing computational and storage costs is negated.

In practice, we generally attempt to choose a value for m that balances in the one hand the good convergence properties resulting from a large value with the other hand the reduction of computational work resulting from a smaller value. Here, we will use the numerical discrepancy between preconditioned and unpreconditioned residual, which helps us to pick an appropriate restart number enabling the convergence of both residual.

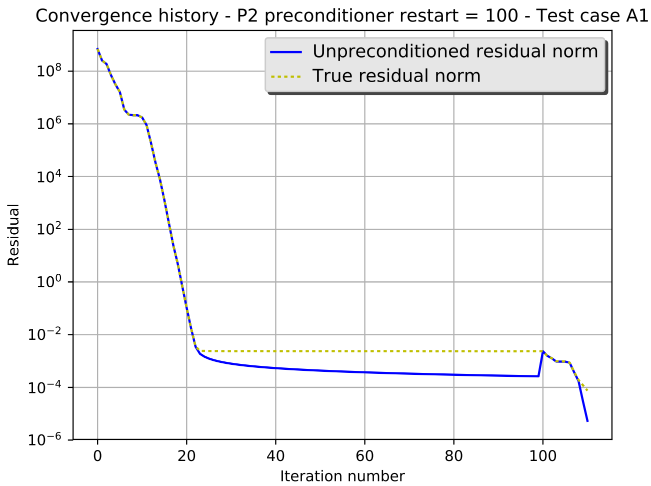


Figure 4.2.: Convergence history of the constraint preconditioner when setting up the restart number to 100

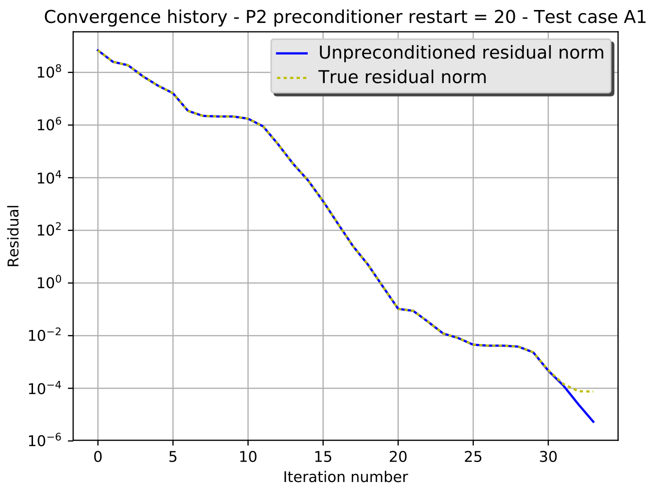


Figure 4.3.: Convergence history of the constraint preconditioner when setting up the restart number to 20

As shown in Figure 4.2 and Figure 4.3, if we run the preconditioner without a restart number, we notice that after some number of iterations, the preconditioned and unpreconditioned residuals, that are supposed be the same when preconditioning at right, move away from one another. We choose then this point of first disconnection as a restart number. Using this trick, we recover the convergence of both residuals and we accelerate hopefully the convergence time.

Choosing the direct solver

The approximation proposed above require an exact Cholesky decomposition for the block A_Z and the Schur complement approximation S_Z. In order to choose the best direct solver, we show the results of the direct solution of the linear system $\mathcal{A}x = b$ where $\mathcal{A}$ a symmetric positive definite matrix.

The direct solver enabling Cholesky factorization provided in PETSc [11] is tested together with the following matrix ordering methods: Nested Dissection (`ND`), One-way Dissection (`1WD`), Reverse Cuthill-McKee (`RCM`), Quotient Minimum Degree (`QMD`), and Approximate Minimum Degree (`AMD`). PETSc is used to call the MUMPS [3] and PaStiX [64] direct solvers, with the provided ordering methods. Further note that PETSc only provide sequential implementations of the Cholesky factorisation. Parallel implementations are made possible using MUMPS and PASTIX.

	Ordering	Set up				Apply	Solve	Fill-in	Fill Ratio
$size = 9,201$ $nnz = 602,603$		Get Ordering	Symbolic Facto.	Numeric Facto.	Total		Set up + Apply		
PETSc	`Natural`	0.00	49.01	27.70	76.70	0.03	76.73	16,980,101	28.17
	`ND`	0.00	1.00	0.48	1.49	0.00	1.49	1,943,372	3.22
	`1WD`	0.00	7.04	3.25	10.31	0.01	10.32	5,646,066	9.36
	`RCM`	0.00	5.42	3.20	8.63	0.01	8.64	5,718,474	9.48
	`QMD`	0.06	1.01	0.45	1.53	0.00	1.54	1,951,034	3.23
	`AMD`	0.00	0.91	0.42	1.34	0.00	**1.35**	1,856,094	**3.08**
MUMPS	`AMD`	0.00	0.06	0.34	0.41	0.00	**0.42**	1,920,557	3.18
	`AMF`	0.00	0.09	0.56	0.65	0.01	0.66	2,215,053	3.67
	`METIS`	0.00	0.14	0.40	0.55	0.01	0.56	1,871,817	**3.10**
	`PORD`	0.00	0.14	0.47	0.61	0.01	0.62	1,944,021	3.22
	`QAMD`	0.00	0.09	0.51	0.60	0.01	0.61	2,040,667	3.38
	`SCOTCH`	0.00	0.20	0.41	0.61	0.01	0.62	1,918,287	3.18
PASTIX	`SCOTCH`	0.14	0.00	0.57	0.57	0.02	0.59	3,343,588	5.46

Table 4.3.: Cholesky direct solve using different packages and ordering methods

Furthermore, we note "Fill-in" to mention the number of nonzeros in the matrix factor $\mathcal{L}$, and the term "fill ratio" to express the number of nonzeros in the factor divided by the number of nonzeros in the target matrix. We discuss mainly the computation time spent on the relevant PETSc functions.

PETSc with `AMD` ordering provides the best reordering in terms of fill-in. We reduce the fill factor from 28.17 to around 3.08. However, MUMPS with `AMD` ordering performs the best in term of computation time. Hence, we use this setting in the implementation of the exact direct factorization of the constraint preconditioner $\mathcal{P}_{Chol}$.

Testing constraint preconditioner $\mathcal{P}_{Chol}$

Before testing the constraint preconditioner $\mathcal{P}_{Chol}$, we present a very well-known constraint preconditioner in the saddle point literature that we call here $\mathcal{P}_{SIMPLE}$. This latter is the most used one in computational fluid mechanics applications known as SIMPLE schemes for 'Semi-Implicit Method for Pressure-Linked Equations' [18]. We want to evaluate its performance in comparison to the constraint preconditioner $\mathcal{P}_{Chol}$.

The preconditioner $\mathcal{P}_{SIMPLE}$ is as follows

$$\mathcal{P}_{SIMPLE}^{-1} = \begin{pmatrix} -L_A^{-T}L_A^{-1} & L_A^{-T}L_A^{-1}B_Z \\ 0 & \mathbb{I}_{n-m} \end{pmatrix} \begin{pmatrix} \mathbb{I}_{n-m} & 0 \\ L_S^{-T}L_S^{-1}B_Z D^{-1} & L_S^{-T}L_S^{-1} \end{pmatrix}, \tag{4.43}$$

where $\widetilde{S}_Z = T_Z + B_Z D^{-1} B_Z = L_S L_S^T$ the exact Cholesky factorization and $D = Diag(A_Z)$.

This preconditioner enables us to evaluate how the classical approximation of the Schur complement performs in comparison to the developed approximation in $\mathcal{P}_{Chol}$.

Let us apply the proposed constraint preconditioners $\mathcal{P}_{SIMPLE}$ and $\mathcal{P}_{Chol}$ on the test cases A_1, A_2, A_3 and A_4 described in Table 4.2.

Matrix A	System size	System nnz	The preconditioner	# Iterations	CPU Time (sec)	Flops	Memory (MB)
A_1	18,402	1,943,373	$\mathcal{P}_{SIMPLE}$	886	9.709e+01	1.958e+10	1.278e+02
			$\mathcal{P}_{Chol}$	20	1.917e+01	5.790e+08	6.670e+01
A_2	24,816	2,751,990	$\mathcal{P}_{SIMPLE}$	897	1.535e+02	2.772e+10	1.811e+02
			$\mathcal{P}_{Chol}$	19	5.361e+01	9.044e+08	1.140e+02
A_3	42,396	5,071,433	$\mathcal{P}_{SIMPLE}$	1213	4.030e+02	5.566e+10	2.420e+02
			$\mathcal{P}_{Chol}$	19	9.762e+01	1.622e+09	2.023e+02
A_4	122,466	16,358,756	$\mathcal{P}_{SIMPLE}$	3012	4.466e+04	5.998e+11	5.137e+02
			$\mathcal{P}_{Chol}$	20	1.158e+03	7.323e+09	9.218e+02

Table 4.4.: Iterative solution method of the test cases using FGMRES with the constraint preconditioners $\mathcal{P}_{SIMPLE}$ and $\mathcal{P}_{Chol}$

From numerical results, $\mathcal{P}_{Chol}$ seems efficient, compared to $\mathcal{P}_{SIMPLE}$. This latter approaches the dense Schur complement $S_Z = T_Z - B_Z A_Z^{-1} B_Z$ by a Cholesky decomposition of a dense matrix $\widetilde{S_Z} = T_Z - B_Z Diag(A_Z)^{-1} B_Z$, while $\mathcal{P}_{Chol}$ approaches the dense Schur complement by a sparse matrix. This enables a reduction of flops and CPU time with less iterations. However, the Cholesky direct decomposition is not scalable in time and memory and may be expansive for large systems. We investigate this point by running $\mathcal{P}_{Chol}$ within a parallel framework in a coming section.

Let us now analyze the computational time of each step of the iterative solution method when applied on the large test case A_4. In order to do that, we present in the following different steps of the application of the constraint preconditioner within the iterative method.

Let us begin with some definitions. Using the following decomposition of the constraint implementation

$$\begin{pmatrix} -G_Z & B_Z^T \\ B_Z & T_Z \end{pmatrix}^{-1} = \begin{pmatrix} \mathbb{I} & G_Z^{-1} B_Z^T \\ 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} -G_Z^{-1} & 0 \\ 0 & S_Z^{-1} \end{pmatrix} \begin{pmatrix} \mathbb{I} & 0 \\ B_Z G_Z^{-1} & \mathbb{I} \end{pmatrix}, \tag{4.44}$$

we define the following keywords as three solution steps

- **Solve_Low** : The solution method associated with the matrix G_Z^{-1} in those block matrices

$$\begin{pmatrix} \mathbb{I} & G_Z^{-1}B_Z^T \\ 0 & \mathbb{I} \end{pmatrix} \text{ and } \begin{pmatrix} \mathbb{I} & 0 \\ B_Z G_Z^{-1} & \mathbb{I} \end{pmatrix},$$

- **Solve_0** : The solution method associated with the coefficient matrix G_Z^{-1} in the block (1,1) of the matrix

$$\begin{pmatrix} -G_Z^{-1} & 0 \\ 0 & S_Z^{-1} \end{pmatrix},$$

- **Solve_Schur** : The solution method associated with the coefficient matrix S_Z^{-1} in the block (2,2) of the matrix

$$\begin{pmatrix} -G_Z^{-1} & 0 \\ 0 & S_Z^{-1} \end{pmatrix}.$$

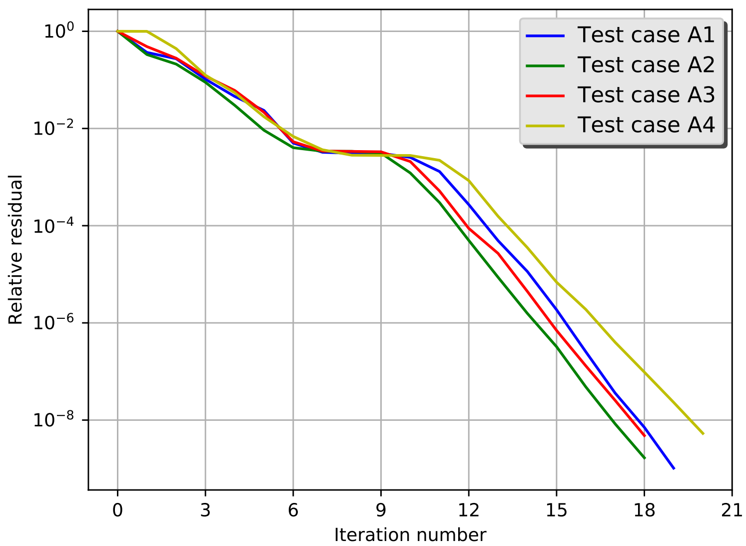


Figure 4.4.: Convergence history of the academic application problem for different system sizes when using the constraint preconditioner $\mathcal{P}_{Chol}$

Test case	A_4	Computational time for each Step of the iterative solution method								
		Set up				Apply				Solve
The preconditioner		Get ordering	Sym. Factorization	Num. Factorization	Total	Solve_Low	Solve_0	Solve_Schur	Total	
$\mathcal{P}_{SIMPLE}$	Time (sec)	0.09	4.46	853.62	866.22	1832.02	1632.10	40976.05	44475.21	44658.25
	percent time (%)	0	0	2	2	4	4	92	100	100
	Count	2	2	2	3	3012	3012	3012	3012	1
$\mathcal{P}_{Chol}$	Time (sec)	2.23	544.26	558.92	1104.01	317.68	17.72	812.42	1148.5	1152.40
	percent time (%)	0	47	48	95	27	2	70	99	100
	Count	2	2	2	3	20	20	20	20	1

Table 4.5.: Profiling table of test case A_4 with constraint preconditioners $\mathcal{P}_{SIMPLE}$ and $\mathcal{P}_{Chol}$ including computational time for each Step of the iterative solution method

From the Table 4.5, we confirm that the best Schur complement approximation is the one implemented within $\mathcal{P}_{Chol}$. Actually, the step associated with the Schur complement in $\mathcal{P}_{SIMPLE}$ takes 92% of the computational time. Moreover, we notice when using $\mathcal{P}_{Chol}$ that Cholesky decomposition takes 95% of the whole computational time, which emphasizes the need to replace these exact "approximations" of A_Z and S_Z by less memory consuming approaches or at least more scalable ones.

In Figure 4.4 we compare the performance (in terms of iteration count) of the constraint preconditioner $\mathcal{P}_{Chol}$ within a parallel framework and applied to the test case A_4. We note that the convergence does not depend on the problem size, all test cases have the same convergence pattern.

Comparing constraint preconditioner $\mathcal{P}_{Chol}$ and its block triangular equivalent preconditioner

We present the upper block preconditioner

$$P_{CholTrig} = \begin{pmatrix} -G_Z & B_Z^T \\ 0 & S_Z \end{pmatrix}. \tag{4.45}$$

We use the same approximations of the constraint preconditioner $\mathcal{P}_{Chol}$ for the block G_Z ans the Schur complement S_Z.
Let us compare both block preconditioners ($\mathcal{P}_{CholTrig}$ and $\mathcal{P}_{Chol}$) applied on the test case A_4. We show the results in Table 4.6.

Test case	A_4	Computational time for each Step of the iterative solution method								
		Set up				Apply				Solve
The preconditioner		Get ordering	Sym. Factorization	Num. Factorization	Total	Solve_Low	Solve_0	Solve_Schur	Total	
$\mathcal{P}_{Chol}$	Time (sec)	2.01	227.47	232.31	459.78	130.67	9.67	338.78	479.14	483.90
	percent time (%)	0	47	48	95	27	2	70	99	100
	Count	2	2	2	3	20	20	20	20	1
$\mathcal{P}_{CholTrig}$	Time (sec)	2.02	220.24	251.02	471.26	-	910.33	2308.3	3218.6	3251.20
	percent time (%)	0	7	8	14	-	29	70	99	100
	Count	2	2	2	3	-	120	120	120	1

Table 4.6.: Profiling table comparing the constraint preconditioner $\mathcal{P}_{Chol}$ and its equivalent triangular block preconditioner $\mathcal{P}_{CholTrig}$ including computational time for each step of the iterative solution method

We note that the steps **Solve_0** and **Solve_Low** are gathered in the same step **Solve_0** in the triangular block preconditioner. The number of iterations has grown to 120 using the triangular block preconditioner and CPU time increases nearly seven times as much $\mathcal{P}_{Chol}$ costs. Actually, applying the preconditioner $\mathcal{P}_{CholTrig}$ takes more time and more iterations than the constraint preconditioner $\mathcal{P}_{Chol}$. The application cost for this latter is expansive because of the direct Cholesky factorization (95%) included in the set up step. In comparison to the triangular block preconditioner, setting up the preconditioner costs only 14%.

Through these results, we conclude that it is much more advantageous to use $\mathcal{P}_{Chol}$ in its full form than using it in the triangular form. Thus, we continue using $\mathcal{P}_{Chol}$, however we intend to investigate its parallel performance and also some different approximations to A_Z in order to reduce the computational cost of the direct decomposition.

Parallel test of the constraint preconditioner $\mathcal{P}_{Chol}$

We use in the following the same preconditioner $\mathcal{P}_{Chol}$ but within a parallel context. This experiment is conducted on a cluster as mentioned in the implementation considerations 4.3.1. We mention that MUMPS is also parallel.
We study the scalability of the proposed preconditioner. To do so, we solve the test case A_4 by running on 1, 2, 4, 8 and 16 processes. The convergence of the iterative method for these different distributions is presented in Figure 4.5. It shows the independence of the convergence from the number of processes used.

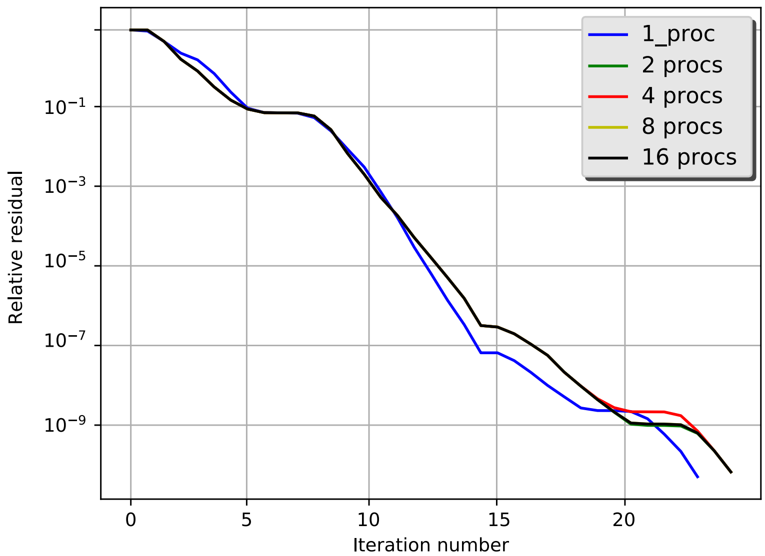


Figure 4.5.: Convergence history of the test case A_4 when using the constraint preconditioner $\mathcal{P}_{Chol}$ on 1, 2, 4, 8 and 16 processes

In Table 4.7, the parallel performance obtained, and in particular the parallel efficiency, are presented. This latter indicator is calculated as the ratio between the ideal acceleration between two runs with different number of processes and the actual acceleration measured. Compared to the reference execution with 1 processor, an efficiency of the order of 89% is observed for 2 processes. This figure indicates a good parallel efficiency. However, the efficiency drops as we run on more processes. This parallel efficiency can be improved by dealing with a larger problem as will be done in Chapter 5. Indeed, each processor must have a sufficient number of unknowns to solve.

A notable memory and Flops decrease is indeed obtained on the parallel results. Especially for the case of 2 processes, but they drop in efficiency when number of processes increase, which is explained by the moderate size of the problem.

		Functions & CPU time (sec)						Statistics			
		Set up	Apply				Total				
Preconditioner $\mathcal{P}_{Chol}$			Solve_Low	Solve_0	Solve_Schur	Total	CPU Time (sec)	Efficiency(%)	Iterations	Flops	Memory (MB)
	Time (sec)	459.78	130.67	9.67	338.78	479.14					
1 proc	percent (%)	95	27	2	70	99	4.839e+02	-	20	7.323e+09	9.218e+02
	Count	3	20	20	20	20					
	Time (sec)	238.21	184.67	10.42	67.73	263.20					
2 procs	percent (%)	88	68	4	25	97	2.719e+02	89	21	1.824e+09	1.977e+02
	Count	3	21	21	21	21					
	Time (sec)	141.23	111.64	6.88	38.91	157.66					
4 procs	percent (%)	84	67	4	23	94	1.677e+02	72	21	1.106e+09	1.198e+02
	Count	3	21	21	21	21					
	Time (sec)	84.51	66.13	5.21	25.14	96.63					
8 procs	percent (%)	82	64	5	24	94	1.030e+02	58	21	7.277e+08	7.573e+01
	Count	3	21	21	21	21					
	Time (sec)	58.50	49.50	4.05	14.66	68.31					
16 procs	percent (%)	76	65	5	19	89	7.657e+01	40	21	5.835e+08	5.743e+01
	Count	3	21	21	21	21					

Table 4.7.: Evaluation of the constraint preconditioner $\mathcal{P}_{Chol}$ on the test case A_4 running on 1, 2, 4, 8 and 16 processes

In previous experiments, it is shown that the convergence of the proposed block preconditioner $\mathcal{P}_{Chol}$ is independent of both the number of processes and the size of the problem. These properties are quite essential in parallel computing because they ensure to solve problems as large as desired as long as we have a sufficiently powerful machine. They also imply the optimality of the proposed approach. We intend to study large and real life industrial applications in Chapter 5 to prove the efficiency of the proposed approach.

Parallel test for different approximations of the block A_Z in $\mathcal{P}_{Chol}$

Let us consider in this section different approximations for A_Z in the constraint preconditioner $\mathcal{P}_{Chol}$, that are more scalable than Cholesky factorization. We take here the same exact Cholesky decomposition of $T_Z - A_Z - 2\omega_{exp}^2 M_Z = \mathring{L}_S \mathring{L}_S^T$ as an approximation for the

Schur complement S_Z, in order to analyze the impact of different approximations of A_Z on the convergence of the iterative method.

– Incomplete factorization preconditioners

We test the incomplete LU factorization with threshold approach which is quite similar to incomplete Cholesky factorization, however unlike this latter, it has a parallel implementation. The main inconvenient here, is using an unsymmetric preconditioner for a symmetric matrix, which imply using a general accelerator like GMRES instead of the conjugate gradient method in inner loops. This preconditioner rely on threshold dropping as well as a mechanism to control the maximum size of ILU factors, this approach can be expansive, but this cost can be offset by the gain in the acceleration part of the process.

We apply this preconditioner on the test case A_4 with different drop tolerances as shown in Figure 4.6. The motivation is to select a certain accuracy but at the same time limit the amount of memory and then of time required.

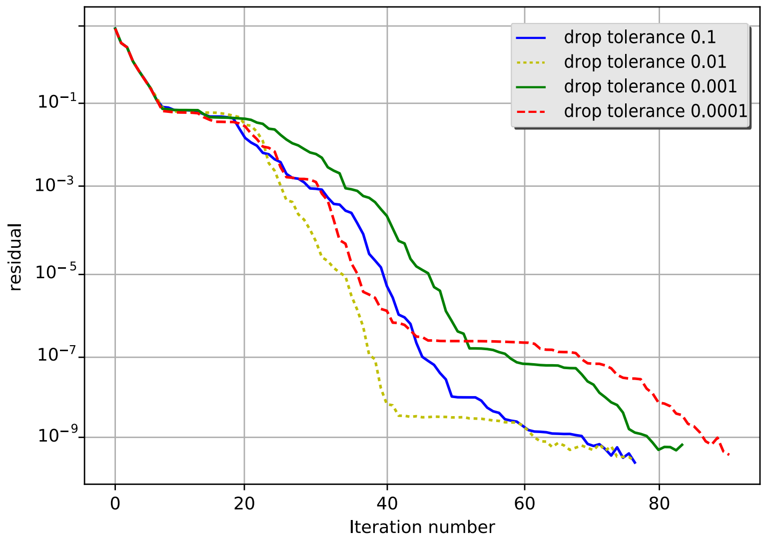


Figure 4.6.: Convergence history of the constraint preconditioner $\mathcal{P}_{Chol_{ILUT}}$ with different drop tolerances applied on the test case A_4

From Figure 4.6, we note that taking a drop tolerance equal to 0.01 gives the best trade-off between accuracy and speed. Using this threshold, we apply the constraint preconditioner $\mathcal{P}_{Chol_{ILUT}}$ for the test case A_4 in order to compare with the preconditioner $\mathcal{P}_{Chol}$. In addition to this sequential application of ILUT, we use a parallel implementation PILUT which is a parallel preconditioner based on Saad's dual-threshold incomplete factorization algorithm.

It uses the Schur-complement approach to generate parallelism. PILUT is proposed through Hypre library [2], it uses MPI and a coarse-grain parallelism.

Figure 4.7 shows the convergence history of the constraint preconditioner $\mathcal{P}_{Chol_{ILUT}}$ applied on the test case A_4 within a parallel implementation. Up to iteration 30, the convergence history of the running jobs on 2, 4, 8 and 16 processes keep the same shape, then each one converges in a different way than others.

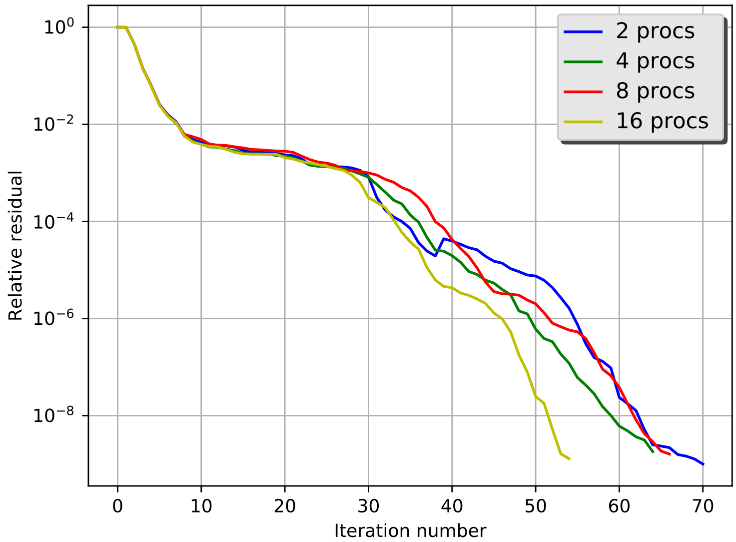


Figure 4.7.: Convergence history of the constraint preconditioner $\mathcal{P}_{Chol_{ILUT}}$ running on 2, 4, 8 and 16 processes applied on the test case A_4 - chosen drop tolerance = 0.01

	FGMRES ($rtol = 1e-9$, $MaxIt = 10,000$)				
The preconditioner $\mathcal{P}_{Chol_{ILUT}}$	1 proc	2 procs	4 procs	8 procs	16 procs
# Iterations	63	71	65	67	55
CPU Time (sec)	2.856e+03	1.742e+03	1.076e+03	8.587e+02	9.065e+02
Efficiency %	-	82	76	42	15
Flops	1.453e+12	8.396e+11	4.751e+11	3.199e+11	2.072e+11
Memory (MB)	4.561e+02	2.330e+02	1.335e+02	8.105e+01	5.644e+01

Table 4.8.: Evaluation of the constraint preconditioner $\mathcal{P}_{Chol_{ILUT}}$ applied on the test case A_4 running on 1, 2, 4, 8 and 16 processes

Table 4.8 collects the results for the five different simulations. We observe that the use of incomplete factorization instead of exact one reduces the memory cost by a half, nevertheless the CPU time and Flops increase dramatically. It is clear that using incomplete factorization approach sequentially is not the better choice, as they approximate poorly the three-dimensional structures finite element matrices. When using a parallel version on 4 processes, we divide the CPU time by three, which yields a parallel efficiency of 76.8 %. This number is acceptable in regards to the moderate dimension of the problem. However, the efficiency drops dramatically when increasing the number of processes.

– Algebraic Multigrid preconditioners (AMG)

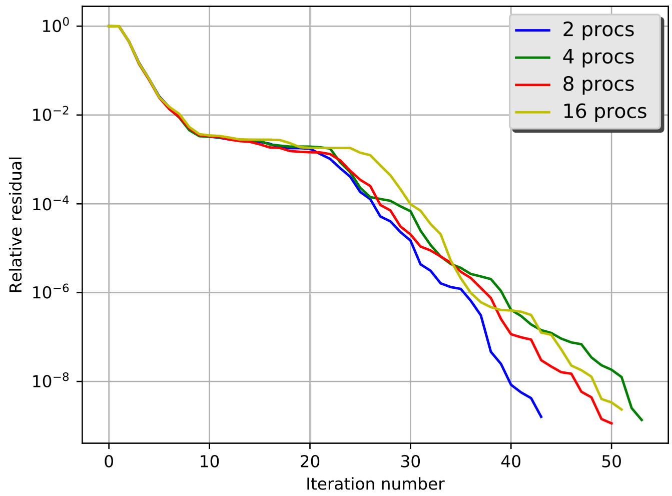
We introduce in this section another algebraic approximation of the block A_Z that is based on the multigrid approach. The main idea behind this approach came from vibration problems. Indeed, relaxation schemes, such as the Gauss-Seidel or the Jacobi method, efficiently damp high frequency errors, however they make no progress towards reducing low frequency errors. The multigrid methods aim to move the problem to a coarser grid so that previously low frequency errors turn now into high frequency errors and can be damped efficiently. If we apply this procedure recursively, we obtain a method with a computational cost that depends only linearly on the problem size.

Contrary to physics-based geometric multigrid approach, where the geometry of the problem is used to define the various multigrid components, the algebraic multigrid (AMG) methods use only the information available in the linear system of equations and are therefore suitable to solve problems on more complicated domains and unstructured grids.

Introduced in Hyper library, BoomerAMG is a parallel implementation of algebraic multigrid. It uses two different coarsening strategies, one suited to structured problems and the other more efficient for unstructured grids. Moreover, it provides some classical point-wise smoothers (Jacobi, Gauss-Seidel ...). We use the default symmetric relaxation (symmetric-SOR/Jacobi), which is required for conjugate gradient method that expects symmetry. We use this library to build the constraint preconditioner $\mathcal{P}_{Chol_{BoomerAMG}}$.

PETSc also provides a native algebraic multigrid preconditioner GAMG with different implementations : a smoothed/unsmoothed aggregation AMG, a classical AMG, a hybrid geometric AMG. Optimizing parameters for AMG is tricky; however we recall that we are using AMG methods as black box approximations. We use this approach to build the constraint preconditioner $\mathcal{P}_{Chol_{GAMG}}$.

Figure 4.8 presents the parallel performance, in terms of iteration count, of the constraint preconditioners $\mathcal{P}_{Chol_{BoomerAMG}}$ and $\mathcal{P}_{Chol_{GAMG}}$.



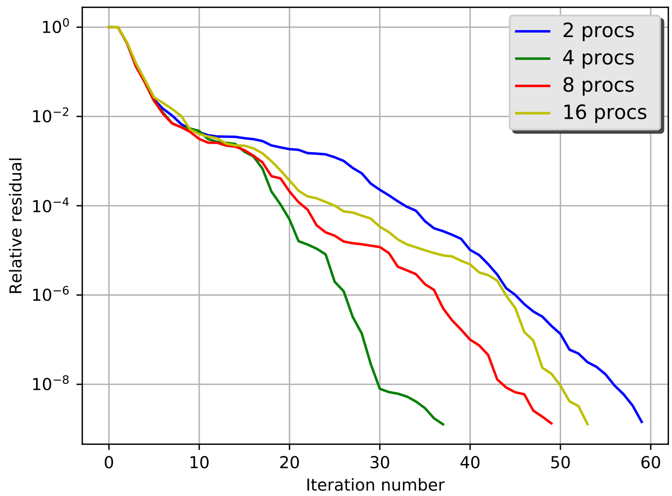


Figure 4.8.: Convergence history of the constraint preconditioners $\mathcal{P}_{Chol_{BoomerAMG}}$ (up) and $\mathcal{P}_{Chol_{GAMG}}$ (down) running on 2, 4, 8 and 16 processes applied on the test case A_4

We observe approximately the same pattern of convergence for each calculation. The iteration count is between 40 and 60 for both preconditioners.

	FGMRES ($rtol = 1e-9$, $MaxIt = 10,000$)				
The preconditioner $\mathcal{P}_{Chol_{BoomerAMG}}$	1 proc	2 procs	4 procs	8 procs	16 procs
# Iterations	51	44	54	51	52
CPU Time (sec)	4.527e+03	2.515e+03	1.894e+03	1.346e+03	1.341e+03
Efficiency %	-	90	60	42	21
Flops	7.473e+11	4.684e+11	3.343e+11	2.118e+11	1.693e+11
Memory (MB)	3.982e+02	2.133e+02	1.286e+02	7.608e+01	5.518e+01

Table 4.9.: Evaluation of the constraint preconditioner $\mathcal{P}_{Chol_{BoomerAMG}}$ applied on the test case A_4 running on 1, 2, 4, 8 and 16 processes

	FGMRES ($rtol = 1e-9$, $MaxIt = 10,000$)				
The preconditioner $\mathcal{P}_{Chol_{GAMG}}$	1 proc	2 procs	4 procs	8 procs	16 procs
# Iterations	53	60	38	50	54
CPU Time (sec)	1.255e+03	8.256e+02	7.471e+02	7.473e+02	9.272e+02
Efficiency %	-	76	42	21	9
Flops	7.473e+11	2.993e+12	3.757e+11	3.339e+11	1.693e+11
Memory (MB)	3.982e+02	6.752e+02	4.730e+02	3.246e+02	2.666e+02

Table 4.10.: Evaluation of the constraint preconditioner $\mathcal{P}_{Chol_{GAMG}}$ applied on the test case A_4 running on 1, 2, 4, 8 and 16 processes

We observe in Tables 4.9 and 4.10 that less memory needs are recorded than in $\mathcal{P}_{Chol}$ (by 57%) and in $\mathcal{P}_{Chol_{ILUT}}$ (by 13%). In the same time, the CPU time increases by 2.6 times for $\mathcal{P}_{Chol_{GAMG}}$ and by 9 times for $\mathcal{P}_{Chol_{BoomerAMG}}$ in the sequential computation in comparison to $\mathcal{P}_{Chol}$. Parallel computation helps to decrease the flops, the memory and the CPU time, however it is less efficient when the number of processes increases.

4.4. Conclusion

We have proposed in this chapter an iterative approach for the solution of linear systems with symmetric saddle point matrices generated through constrained optimization and arising from identification problems. Since the size of the industrial problems is large, the currently available solvers involving direct methods are too inefficient.

This approach based on a double projection onto the nullspace of the constraints can thus be used to replace the direct solvers to solve such linear systems in structural mechanics. It is based on a first projection onto the nullspace of kinematics constraints using an explicit sparse nullspace basis. This latter is computed using an augmented form of the nullspace basis of the kinematic constraint matrix associated with the finite element model of the structure.

The projected system has a saddle point structure too. The second projection of this system onto the nullspace of the sensor constraints is done using the implicit nullspace method. Actually, since the sensor constraints depend on each frequency, this means computing a nullspace basis for each linear system over the whole sequence of generated saddle point systems, which can be expensive.

The implicit nullspace projection method requires solving the saddle point system through constraint preconditioners. We have extended here the use of these preconditioners to solve the reduced saddle point system by proposing a new physical based approximation of the Schur complement block.

The double explicit implicit projection algorithm 4.1 has been implemented within different standalone codes for each step. In *Code_Aster*®, we developed a subroutine to retrieve essential structural matrices and information from test cases. Then, using a Fortran interface to *SuperLU*, we developed an implementation to build the nullspace basis. Finally, we developed the iterative solution method of the projected system in C using PETSc. In this latter, we developed a user method based on the Schur complement block factorization, that enables taking into account the block structure of the constraint preconditioners. Actually, instead of factoring those preconditioners with available global codes such as MUMPS, we implemented in PETSc a GMRES outer loop for the whole system, that incorporates two block linear systems using PCG inner loops.

We have tested the approach on a solid mechanics problem. we have compared the proposed variant of the constraint preconditioner with some other block preconditioners that are often used when solving saddle point systems arising in different applications.

This variant has proved to be efficient in terms of both preconditioner applications and computational operations. Numerical experiments have highlighted the relevance of the proposed preconditioner that leads to a significant decrease in terms of computational operations.

As reported in this chapter, the proposed iterative process is implemented in a parallel distributed memory environment through both the PETSc libraries. The scalability properties of the proposed preconditioner are illustrated, where the main focus is to allow us to consider a broader class of constraint preconditioners approximations using incomplete factorization and multigrid algorithms. Those latter, enhanced by the parallel framework, have demonstrated comparable results in regards to computational time with a limited amount of memory. We further emphasize that a thorough study on large scale real life applications is performed in Chapter 5.

Chapter 5

Evaluation of solvers performance on industrial applications

Contents

The goals of the industrial applications of this chapter is to illustrate the numerical behavior of the developed iterative solution methods from Chapter 4 on industrial finite element (FE) models. Two different cases are presented.

First, a large case built on an industrial FE model (roughly 1 million Degrees of freedom) and massive measurements (almost 700 sensors) is introduced. Direct solution methods are first used, providing a relevant reference point. Associated numerical difficulties are highlighted. Iterative solutions techniques are then introduced. Their robustness and precision are clearly demonstrated, and their performance both on speed and memory are emphasized.

A smaller test case, based on a light FE model (30 000 degrees of freedom) and restraint measurement set (80 sensors) is then presented. The same comparison methodology is derived, revealing the limits of the proposed techniques. Some ways of improvements are then proposed.

5.1. Illustration of performance and robustness of the double projection iterative approach – Large turbo generator

Turbo generators are the devices that converts mechanical energy of the steam turbine to electrical energy.

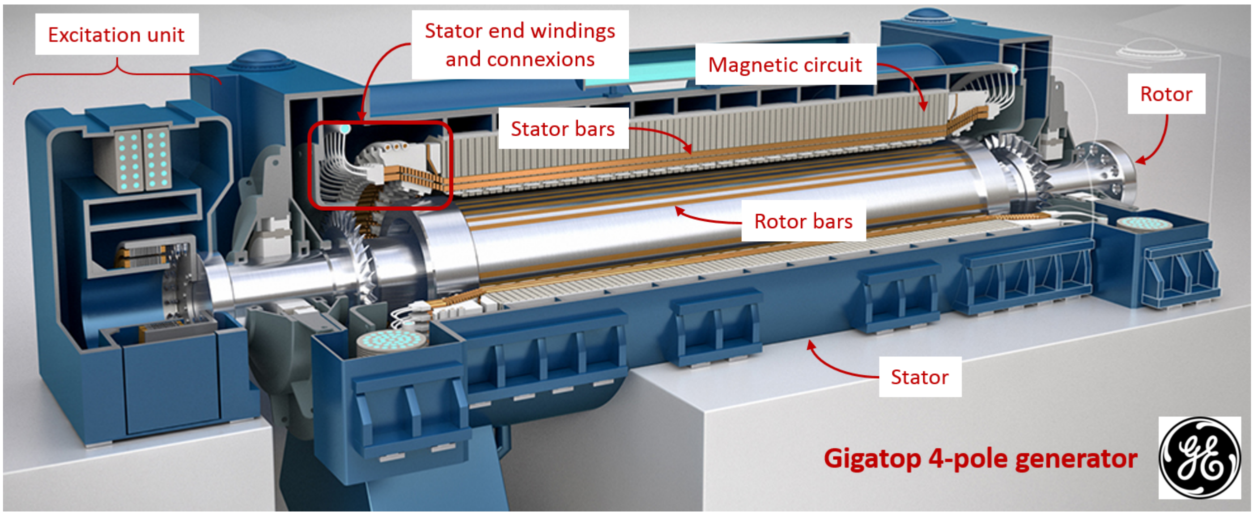


Figure 5.1.: A picture of the Gigatop 4-pole generator if General Electric used in nuclear power plant

They consist of a rotating part and a stationary part

- Rotor is the rotating part of an electric generator. The rotor has a wiring, fed in continuous current by the excitation unit. The rotation motion generates a controlled magnetic field,
- Stator is the stationary part of an electric generator, which surrounds the rotor. The stator has a wire winding in which the chaning electromagnetic field induces an electric current.

Manufacturers design their equipments to ensure proper operation, hence limiting vibration levels. However, in the particular case of the 900 MW power plant generators, the design choices resulted in significant vibrations levels. These abnormally high vibration levels on these equipments are due to the conjunction of two phenomena [9]

- The first phenomenon derives from the choice of design, in order to improve the efficiency and production capacity of the generators. This design induces the existence of a force component that has an ovalized shape in two lobes, rotating at 100 Hz. This shape is presented on Figure 5.2. These forces depend on the excitation current and the load applied to the machine.

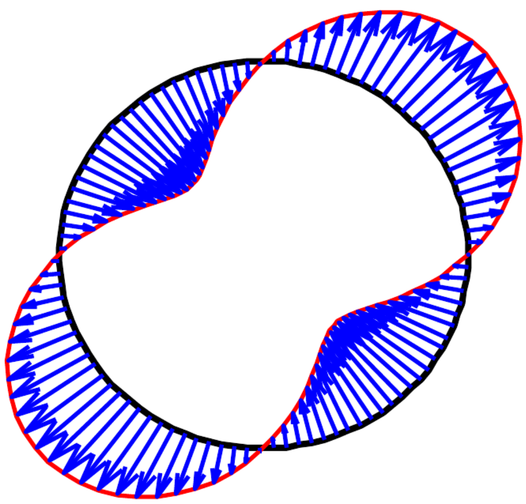


Figure 5.2.: A descriptive diagram of the force with an ovalized shape in two lobes

- The second phenomenon is the existence of modes shapes, around 100 Hz, whose forms coincide with ovalizations in two lobes. The modal density can be important, up to a mode every Hz on some equipments and with the presence of almost double modes.

The conjunction of these two phenomena is specific to the 900 MW generator technology and can lead to degradation, on the one hand, of stator elements and, on the other hand, of peripheral elements by the transmission of vibrations [50]. The presence of high vibration levels in the stator decreases mainly the performance of alternators, but can also result in material damage. In both cases, the unavailability of the equipment entails significant financial losses for EDF.

Figure 5.3.: A turbo generator in the engine room of a nuclear power plant

Figure 5.4.: A power plant turbo generator for the generation of electric power

Industrial constraints on power plant, and on turbo generators in particular, however, prohibit changes on the electro-technical design, so that vibration issues need to be addressed through structural modifications. These modifications are provided by the supplier, since EDF is only in charge of the production.
For years, FE models have been developed by the supplier, in order to asses the efficiency of the provided solutions. The confidence in the provided solutions, however, heavily relies on the representativeness of the FE models. Nevertheless, efficient solutions are hard to design. The manual manufacturing processors of large parts induce significant variability between the different equipments of a same type. Feedback also shows that these materials are very sensitive to operating conditions such as temperature.

As the recommended solutions do not always prove effective, EDF had to acquire numerical and experimental tools and models to anticipate the occurrence of these problems before restarting the machine. Many measurements have been performed on generator stator parts, and are available for model updating. Thanks to energy based functional approach, we are able to build a model that predict the structural behavior of structures [71, 34]. However, since each generator differ from each other, updating has to be performed for each structure, which is still a challenge, partly due to the large size of the model.

5.1.1. Experimental setup and numerical model details

Test campaigns and especially modal analysis, have been carried out on the not operating alternator. The excitation unit and the rotor are removed from the alternator, then accelerometers are positioned inside the magnetic circuit to record the vibration generated by an impact hammer, see Figure 5.5.

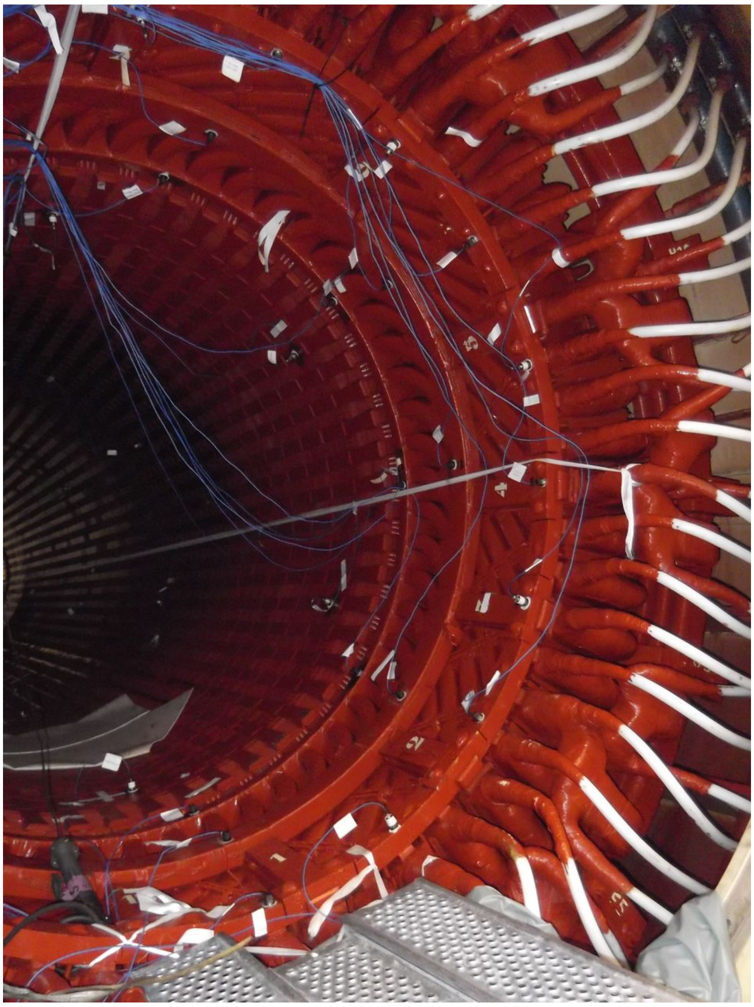


Figure 5.5.: The accelerometers are positioned inside the magnetic circuit to record vibration generated by an impact hammer

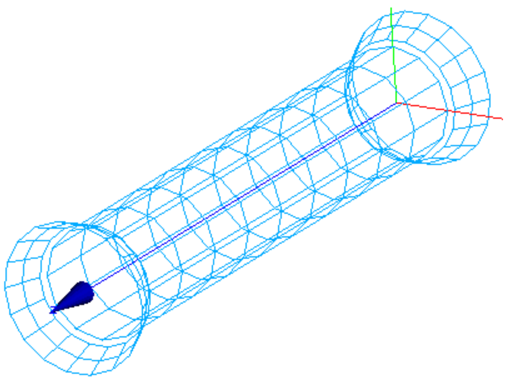


Figure 5.6.: The experimental mesh composed of a set of sensors on the 900 MW power plant alternator used for the study

Hence, experimental modal parameters are obtained by measuring its operating deflection shapes, and post-processoring the vibration data. We give here the experimental coarse mesh of the alternator and the first six eigenmodes of the experimental mesh.

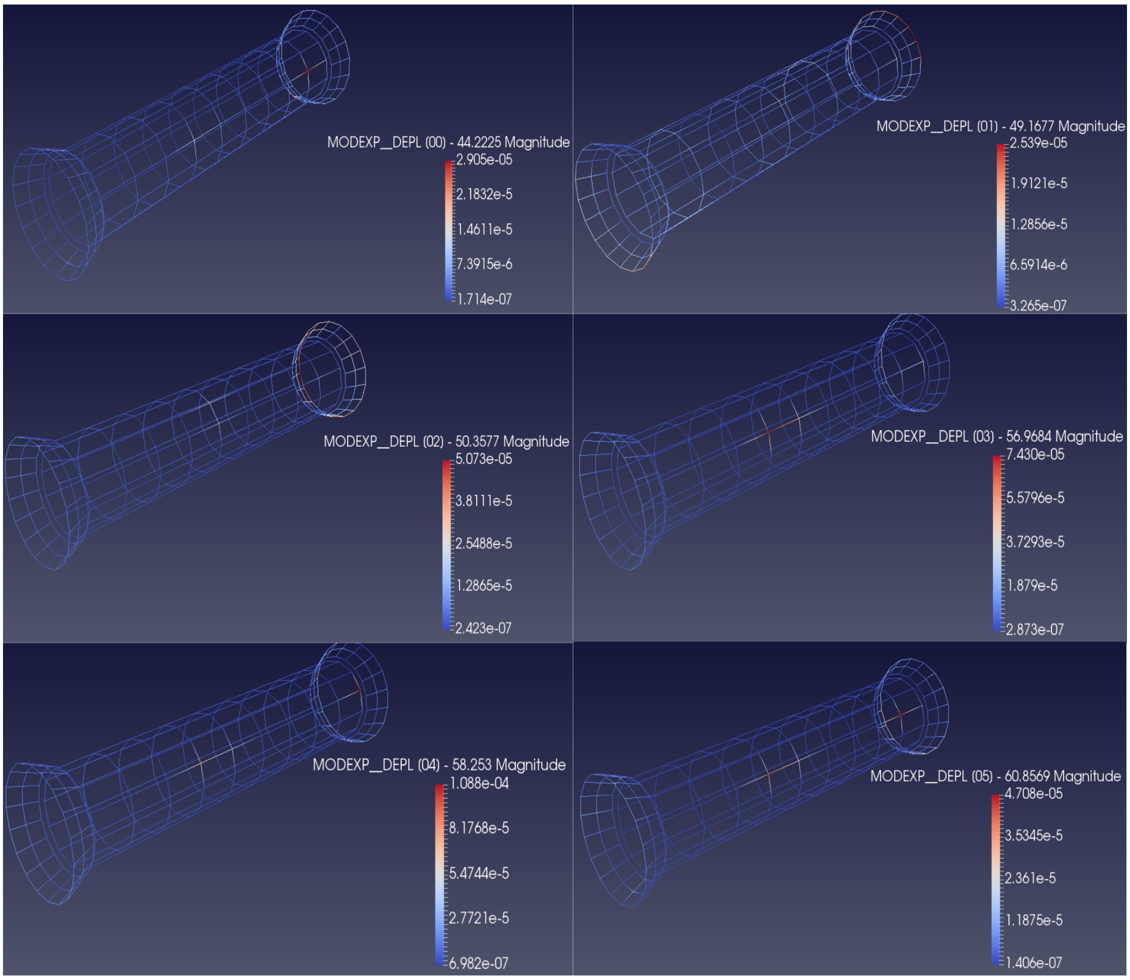


Figure 5.7.: The first six eigenmodes associated with the experimental mesh of the alternator

A generic numerical model, representing the average behavior resulting from the tests carried out on the various sites and equipments, has been built by EDF. This model is currently used to estimate operating response levels.
The numerical finite element model consists of a set of 3D elements, shell elements of type Discrete Kirchhoff Triangle, quadrangles and some Euler-Bernouilli straight beams as shown in Figure 5.8. It contains many kinematic relationships between several degrees of freedom, representing the assembly of components. Moreover, the materials are assumed to be isotropic while some components are considered orthotropic.

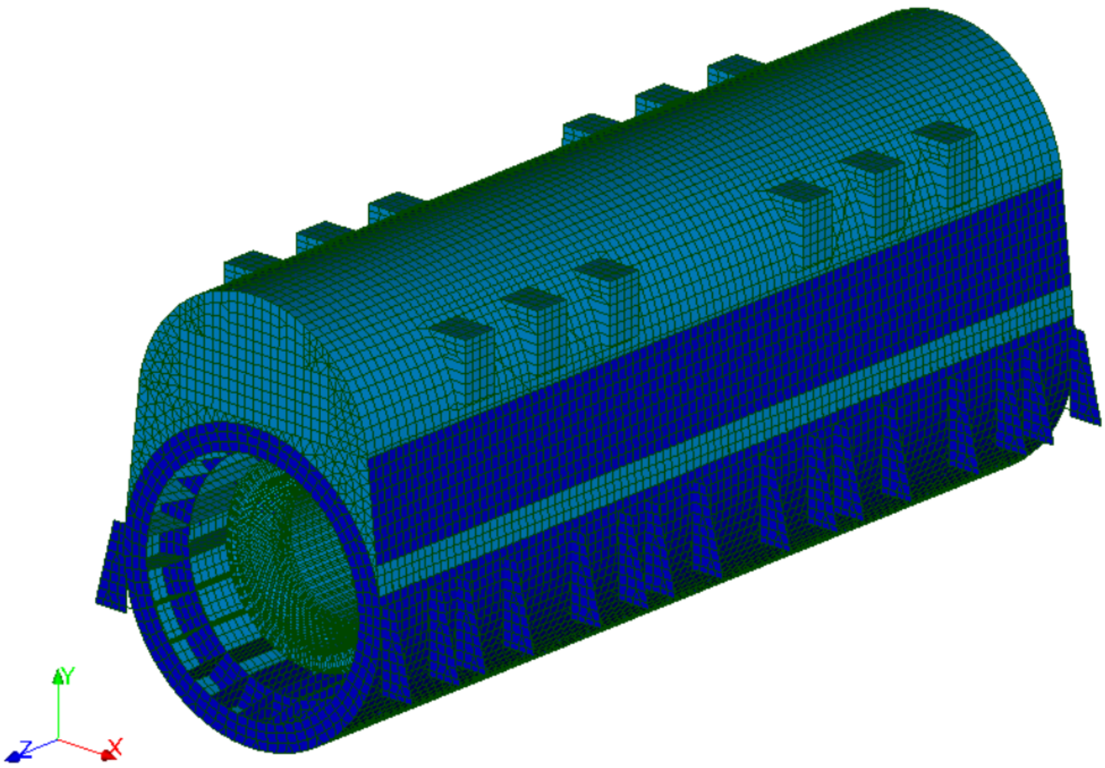


Figure 5.8.: The numerical mesh of the 900 MW power plant alternator used for the study

Number of nodes	Number of elements	Degrees of freedom	Degrees of observation
245,849	77,814 Hexa8 10,833 Beams 148,878 shells	887,601 (Physical) 44,292 (Lagrange)	684

Table 5.1.: The mesh characteristics of the numerical model

We recall that the saddle-point generated systems are described as follows

$$\begin{pmatrix} -[\widetilde{K}(\theta)] & [\widetilde{K}(\theta)] - \omega_{exp}^2[\widetilde{M}(\theta)] \\ [\widetilde{K}(\theta)] - \omega_{exp}^2[\widetilde{M}(\theta)] & \frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\Pi} \end{pmatrix} \begin{pmatrix} \{\widetilde{\psi}\} \\ \{\widetilde{\varphi}\} \end{pmatrix} = \begin{pmatrix} \widetilde{0} \\ \frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\phi}_{exp} \end{pmatrix} . \tag{5.1}$$

The matrix	Size	Nonzero elements number
The stiffness matrix $[\widetilde{K}(\theta)]$	909,747 × 909,747	25,169,423
The mass matrix $[\widetilde{M}(\theta)]$	909,747 × 909,747	10,621,566
The observation matrix $\widetilde{\Pi}$	684 × 909,747	4,815
The norm matrix $[\widetilde{K}_r]$	684 × 684	234,270
$\frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\Pi}$	909,747 × 909,747	12,487,542
The saddle point matrix	1,819,494 × 1,819,494	161,915,766

Table 5.2.: The size and sparsity of the generated coefficient matrix and its sub-blocks

Table 5.2 shows the size and the nonzero elements number of each block matrix in the coefficient matrix of the studied saddle point linear system 5.1. We note that the number of sensors used on the structure is equal to $s = 684$ and the number of Lagrange multipliers is equal to $m = 22,146$.

This system is challenging, not only from its dimension, but also from the condition number of the saddle point matrix which is of order of 10^{11}. We recall that the condition number is estimated using MUMPS in double precision arithmetics as a direct solver of the linear system. This value can be notably related to the properties of the mesh; in particular, when there are strong spatial variations of the size of the mesh elements, or when some of them are flattened, the condition number increases [100].

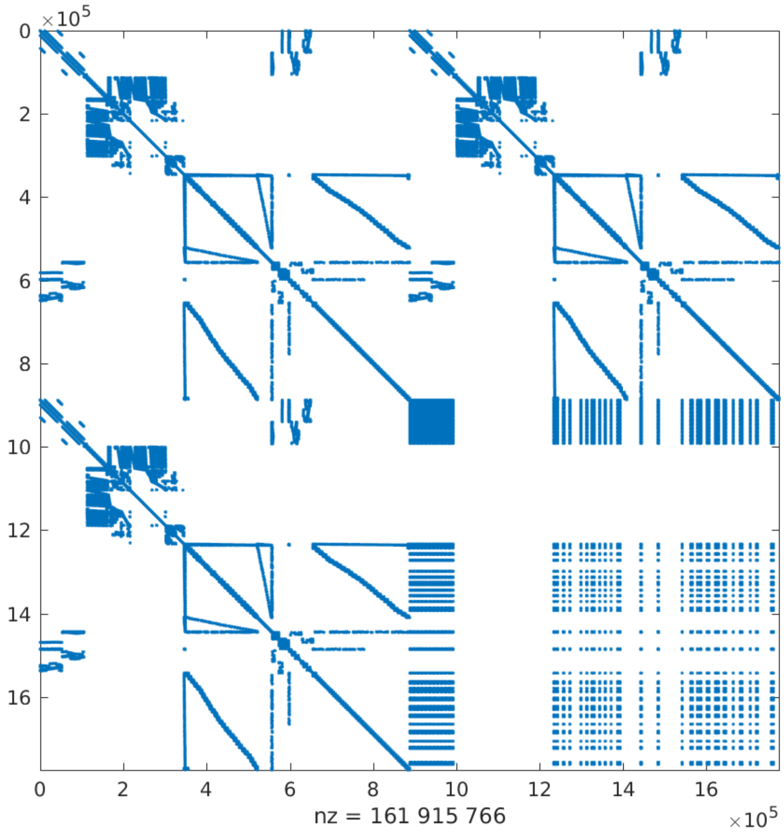


Figure 5.9.: The structure of the coefficient matrix of the saddle point linear system 5.1 associated with the alternator

Figure 5.9 shows the structure pattern of the coefficient matrix of the saddle point linear system 5.1. It is a $2(n+m) \times 2(n+m)$ saddle-point matrix partitioned into four $(n+m) \times (n+m)$ blocks. The $(1,1)$ block is a sparse symmetric finite element matrix. The blocks $(1,2)$ and $(2,1)$ share the same structure as the $(1,1)$ block. The $(2,2)$ block is a very sparse symmetric matrix, it is composed of a dense $s \times s$ sub-block scattered into a $(n+m) \times (n+m)$ matrix, where $s << n+m$ is the number of sensors.

5.1.2. Study of the performance of direct solvers

The goals of this section are twofold. Firstly, we intend to evaluate the application of direct solvers on a large industrial model, that could potentially pose problems related to the structure of the assembled saddle point matrix. Secondly, we compare this performance with the results of the next section, to prove the efficiency of the developed approach of Chapter 4.

In this part, we limit our computations to one frequency. It is chosen close to the model second eigenfrequency, in order to be more challenging for the solver.

The first attempt to solve the system generated for this model initially leads to a computation failure linked to a solver's message confronted with a quasi-singular matrix. However, the assembly of the problem is verified and proves to be quite correct. Moreover, we know that the matrix is invertible if all eigenmodes are observable (see Chapter 2), which is verified by the chosen measurement configuration. Theoretically, we should be able to invert the assembled saddle point matrix and obtain a unique solution. Nevertheless, MUMPS is misled due to the particular structure of the system, and it considers it as a a quasi-singular system. Hence, we force the computation by canceling some tests in the analysis phase of the solver.

This first calculation is initially launched on a one processoror. The computation time to solve the linear system for the requested frequency is 32,413 seconds with a memory consumption (Peak virtual memory usage) of more than 36 GB. The computation, although possible, is relatively time and memory consuming. Table 5.3 shows the different computation times observed for different parallel configurations of the same computation.

number of procs	1 proc	2 procs	4 procs	8 procs	16 procs
CPU time (sec)	32,413	16,180	14,741	9,210	8,718

Table 5.3.: The computation time observed for different parallel configurations for one frequency and with a default MUMPS setup

When using the parallel framework of MUMPS, we obtain a classical speed-up. However, the time consumed by the solver for the resolution of the large saddle point system seems important. Actually, an important part of the time is devoted to the preliminary analysis of the matrix. This is probably due to the particular structure of the saddle point matrix. Furthermore, the default setup of MUMPS choose AMF (see Chapter 3) as the ordering method. In the following, we present in Table 5.4 the results of the same calculation when choosing $METIS$ as the ordering method, since it is the best choice as found in Chapter 3.

number of procs	1 proc	2 procs	4 procs	8 procs	16 procs
CPU time (sec)	17,172	8,940	6,597	6,184	5,812

Table 5.4.: The computation time observed for different parallel configurations for one frequency using MUMPS (METIS ordering)

It is clear that there is an improvement due to the ordering method. Indeed, if we look at the time spent by MUMPS for the resolution of the system, we accelerate the calculation from 2 to 5 times compared to the default ordering. Nevertheless, the CPU time and the memory requirements are huge for one frequency. This result confirms our goal to develop less time consuming methods as done in chapter 4.

5.1.3. Application of the double projection approach

This numerical study is challenging and serves as a relevant realistic test case in structural dynamics to investigate the efficiency of preconditioners for Krylov subspace methods. As mentioned in Chapter 4, we solve the projected system instead of the full system as we eliminate kinematic constraints. We recall that we use the same implementation considerations mentioned in Section 4.3.1 throughout this chapter.

Application of the explicit nullspace projection onto the kinematic constraints

As expressed in Table 5.5, the difference between the full system generated from *Code_Aster*® and the one we generate in size is dependent of the number of Lagrange degrees of freedom. Since this number is not large in comparison to the number of physical degrees of freedom, the size of both systems is comparable. Also, it is usually known that matrix projection may generate some additional fill-in or a much denser matrix. Here, we limit this effect thanks to the skinny LU technique. We note that the generated linear system is also of saddle point structure, which is specific to the problems studied within this manuscript.

Physical dofs (n) Lagrange dofs (m)	887,601 22,146
The constraint matrix C	22,146×887,601 $nnz = 320,818$
The nullspace basis Z	887,601×865,455 $nnz = 1,239,034$
Full system size Full system nnz	1,819,494 161,915,766
Reduced system size Reduced system nnz	1,775,202 167,340,178

Table 5.5.: Information about the full saddle point system and the projected one

Steps	CPU time (sec)	Memory (MB)
LU factorization of C^T	8.8e–02	11.41
Building the inverse permutation	5.36e–01	0.00
Getting L_1 size $22,146 \times 22,146$ nonzero elements $64,049$	-	-
Getting L_2 size $865,455 \times 22,146$ nonzero elements $387,904$	-	-
Getting L_1^T and L_2^T	3.59e–02	2.02
$L_1^{-T} L_2^T$	2.65e+01	3.60
Stocking the nullbasis Z	2.49e+00	2.10
Total	2.94e+01	19,13

Table 5.6.: Steps of building the null basis of the kinematic constraint matrix

Table 5.6 details the steps to build the null basis of C. It is clear that the processor is not expensive in regards of CPU time and Memory. We note that computing $L_1^{-T} L_2^T$ is done by solving $865,455$ linear systems which have the same coefficient matrix L_1^{-T} and right hand sides form the columns of the matrix L_2^T. SuperLU solves using the same factorization of the coefficient matrix with multiple right hand sides.

Application of the implicit nullspace projection through the constraint preconditioner

As introduced in chapter 4, we begin testing the constraint preconditioner $\mathcal{P}_{Chol}$ on the projected saddle point system.

Figure 5.10 proves the independence of the convergence of the constraint preconditioner $\mathcal{P}_{Chol}$ from the number of processors, which confirms the findings of Chapter 4. We show in Table 5.7 the iteration count of the linear systems, the CPU time and the memory requirements provided by PETSc, respectively when using the constraint preconditioner $\mathcal{P}_{Chol}$ for 1, 2, 4, 8 and 16 processors.

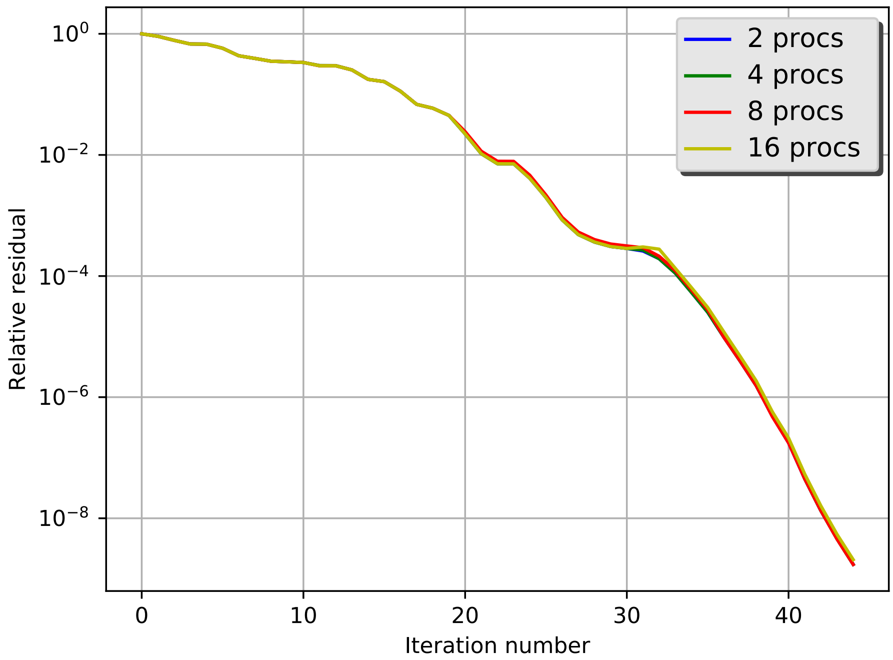


Figure 5.10.: Convergence history of the constraint preconditioner $\mathcal{P}_{Chol}$ applied on the alternator problem on 2, 4, 8 and 16 processors

In comparison to the reference execution on 1 processor, we note a good parallel efficiency, especially for 2 (95%) and 4 (94%) processors execution. From 1 processor to 16 processors, we have an efficiency of 81%, which is considered a good result. In term of iterations count, all executions yield the same number.

Nevertheless, the 1 processor execution takes nearly two times as much as MUMPS does (see Table 5.4) when this latter is conducted using METIS ordering. From 8 processors, the execution of the constraint preconditioner $\mathcal{P}_{Chol}$ yields better results in terms of CPU time compared to the direct solver MUMPS, thanks to the parallel efficiency.

From the standpoint of memory consumption, the developed approach of Chapter 4, is definitely better than direct solvers. Indeed for the 1 processor execution, we need less than 1 GB to conduct the solution method.

Preconditioner $\mathcal{P}_{Chol}$	CPU Time (sec)	Efficiency(%)	Iterations	Flops	Memory (MB)
1 proc	3.261e+04	-	45	9.124e+10	5.182e+03
2 procs	1.716e+04	95	45	4.578e+10	2.662e+03
4 procs	8.674e+03	94	45	2.537e+10	1.448e+03
8 procs	4.971e+03	82	45	1.314e+10	7.793e+02
16 procs	2.516e+03	81	45	6.687e+09	3.934e+02

Table 5.7.: Evaluation of the constraint preconditioner $\mathcal{P}_{Chol}$ applied on the alternator test case and running on 1, 2, 4, 8 and 16 processors for a precision of 10^{-9}

We show in Table 5.8 the profiling time table of all executions. We observe that the main time consuming step is when setting up the constraint preconditioner. Actually, considering the use of Cholesky factorization of the block matrices A_Z and S_Z as an approximations in the constraint preconditioner $\mathcal{P}_{Chol}$, it is a somehow predictable result.

Preconditioner $\mathcal{P}_{Chol}$		Functions & CPU time (sec)					
		Set up	Apply				Total
			Solve_Low	Solve_0	Solve_Schur	Total	CPU Time (sec)
1 proc	percent (%)	95	74	2	24	100	3.261e+04
	Count	3	45	45	45	45	
2 procs	percent (%)	95	75	2	23	100	1.716e+04
	Count	3	45	45	45	45	
4 procs	percent (%)	94	74	2	23	100	8.674e+03
	Count	3	45	45	45	45	
8 procs	percent (%)	94	78	3	19	100	4.971e+03
	Count	3	45	45	45	45	
16 procs	percent (%)	91	71	4	25	99	2.516e+03
	Count	3	45	45	45	45	

Table 5.8.: Profiling CPU time table of the constraint preconditioner $\mathcal{P}_{Chol}$ applied on the alternator test case and running on 1, 2, 4, 8 and 16 processors for a precision of 10^{-9}

Table 5.8 emphasizes the fact that $\mathcal{P}_{Chol}$ is a preconditioner that depends mostly on direct factorization. This preconditioner takes approximately 95% of the CPU time to be set up than to be applied, which means that there is no need to test the precision 10^{-4} in this case.

Now, let us compare the constraint preconditioner $\mathcal{P}_{Chol}$ to $\mathcal{P}_{Chol_{ILUT}}$ introduced in Chapter 4.

We evaluate the performance of $\mathcal{P}_{Chol_{ILUT}}$ by running the executions on 4, 8 and 16 processors. Table 5.9 summarizes the performance data for each execution. We observe that $\mathcal{P}_{Chol_{ILUT}}$ is less efficient than $\mathcal{P}_{Chol}$ in term of iteration count. The CPU time of the ILUT approach is ten times higher than the exact Cholesky approach. Besides, in term of flops, $\mathcal{P}_{Chol_{ILUT}}$ is applied more than 500 hundred time which increases directly the number of operations to approximately 10^4 billions. We conclude that the constraint preconditioner $\mathcal{P}_{Chol_{ILUT}}$ is not suited to industrial applications we aim to solve for the precision 10^{-9}.

Preconditioner $\mathcal{P}_{Chol_{ILUT}}$	CPU Time (sec)	Iterations	Flops	Memory (MB)
4 procs	1.471e+05	541	4.712e+13	1.448e+03
8 procs	8.216e+04	502	2.311e+13	7.793e+02
16 procs	4.986e+04	549	1.419e+13	4.779e+02

Table 5.9.: Evaluation of the constraint preconditioner $\mathcal{P}_{Chol_{ILUT}}$ applied on the alternator test case and running on 4, 8 and 16 processors for a precision of 10^{-9}

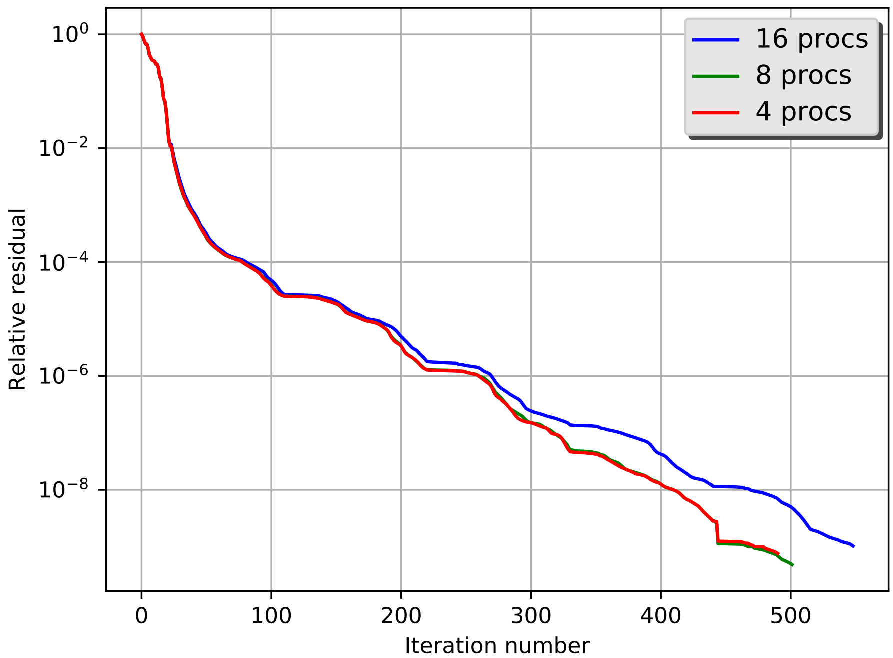


Figure 5.11.: Convergence history of the constraint preconditioner $\mathcal{P}_{Chol_{ILUT}}$ applied on the alternator problem on 4, 8 and 16 processors for a precision of 10^{-9}

Using the same preconditioner $\mathcal{P}_{Chol_{ILUT}}$, we restart the computation with a precision of 10^{-4}, see Table 5.10. We notice decreasing figures in comparison to Table 5.9 by a ratio of approximately 5. Indeed, this preconditioner takes only 3% of the CPU time to be set up, which means that the CPU time is almost proportional to the number of iterations. Though, the computation for 16 processors is two times CPU time consuming than the direct solver MUMPS.

Preconditioner $\mathcal{P}_{Chol_{ILUT}}$	CPU Time (sec)	Iterations	Flops	Memory (MB)
4 procs	2.851e+04	87	8.425e+12	1.372e+03
8 procs	1.583e+04	86	4.712e+12	7.688e+02
16 procs	9.875e+03	87	2.257e+12	4.581e+02

Table 5.10.: Evaluation of the constraint preconditioner $\mathcal{P}_{Chol_{ILUT}}$ applied on the alternator test case and running on 4, 8 and 16 processors for a precision of 10^{-4}

Finally, we compare the constraint preconditioner $\mathcal{P}_{Chol}$ to $\mathcal{P}_{Chol_{BoomerAMG}}$ introduced in Chapter 4.

Preconditioner $\mathcal{P}_{Chol_{BoomerAMG}}$	CPU Time (sec)	Iterations	Flops	Memory (MB)
4 procs	2.457e+04	97	8.326e+12	4.963e+02
8 procs	1.464e+04	97	4.361e+12	2.450e+02
16 procs	7.579e+03	101	2.353e+12	1.623e+02

Table 5.11.: Evaluation of the constraint preconditioner $\mathcal{P}_{Chol_{BoomerAMG}}$ applied on the alternator test case and running on 4, 8 and 16 processors for a precision of 10^{-9}

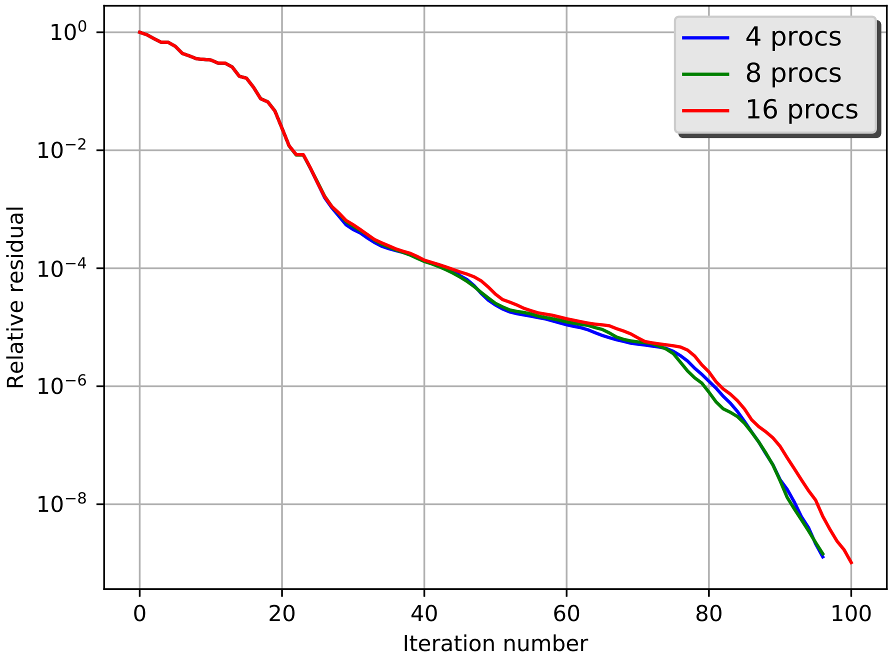


Figure 5.12.: Convergence history of the constraint preconditioner $\mathcal{P}_{Chol_{BoomerAMG}}$ applied on the alternator problem on 4, 8 and 16 processors for a precision of 10^{-9}

From Table 5.11 and Figure 5.12, we observe that the iteration count is stable for different number of processors. Globally, algebraic multigrid approximation is one time and a half as much CPU time consuming as the Cholesky direct factorization for 16 processors and with a precision of 10^{-9}. Nevertheless, it requires less memory and is more scalable.

Let us conduct the same computation for $\mathcal{P}_{Chol_{BoomerAMG}}$ with a precision of 10^{-4} this time. Here, as in the case of $\mathcal{P}_{Chol_{ILUT}}$, the preconditioner takes only 3% of the CPU time to be set up, which means that the computation time is almost proportional to the number of iterations, which is confirmed by the numerical results in Table 5.12.

Preconditioner $\mathcal{P}_{Chol_{BoomerAMG}}$	CPU Time (sec)	Iterations	Flops	Memory (MB)
4 procs	1.059e+04	43	3.678e+12	3.766e+02
8 procs	6.345e+03	42	2.251e+12	1.845e+02
16 procs	3.272e+03	43	1.242e+12	8.421e+01

Table 5.12.: Evaluation of the constraint preconditioner $\mathcal{P}_{Chol_{BoomerAMG}}$ applied on the alternator test case and running on 4, 8 and 16 processors for a precision of 10^{-4}

We conclude that $\mathcal{P}_{Chol_{BoomerAMG}}$ can perform 40% better than direct solver MUMPS for 16 processors for a negligible memory.

In the above numerical results of the alternator case, we tested the double projection approach, on a large real life problem. The approach proved to be efficient in terms of both preconditioner applications and computational operations.
Those numerical experiments have highlighted the relevance of the proposed preconditioner $\mathcal{P}_{Chol}$ that leads to a significant decrease in terms of computational operations. A saving of up to 80% in terms of Memory and to 50% of CPU time is obtained with respect to the classical direct method on the alternator large-scale application. We have also presented some alternatives to preconditioner $\mathcal{P}_{Chol}$, like incomplete factorization based and multigrid based which can achieve comparable results when reducing the sought precision.

5.2. Demonstration of the computational limitations of the double projection iterative approach – Industrial cooling water pump

This section is devoted to study the computational limitations of the proposed approach in Chapter 4 when dealing with a medium or small test case. We demonstrate those limitations on an industrial cooling water pump. The cooling water pump is used for supplying heat exchangers with cooling water. Their flow rate varies depending on the heat flow to be dissipated.

5.2.1. Experimental setup and problem description

Many difficulties could be encountered during the test campaign (very noisy measures due to other nearby pumps, difficulties in identifying modes) and during numerical studies (problem of representativeness of the numerical model).

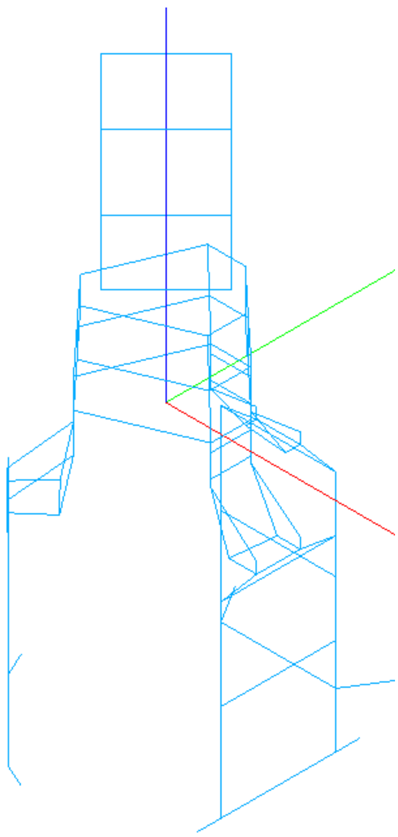


Figure 5.13.: The experimental mesh of the cooling water pump used for the study

Figure 5.13 shows the experimental mesh used to study the pump. 78 measuring points were instrumented. These measuring points are distributed as follows :

- Motor : 24 measurements points spread over 4 crowns of 6 points each;
- Pump : 18 measurements points spread over 3 crowns of 6 points each;
- Suction pipe: 3 measurements points;
- Discharge pipe: 3 measurements points
- Pump / pillar interface: for each of the 3 interfaces : 4 measurements points on the pump feet, 2 measurements points on the metal mounting plate, 3 measurements points on the pillar.

The modal analysis is carried out by successively moving the tridimensional axis sensors.

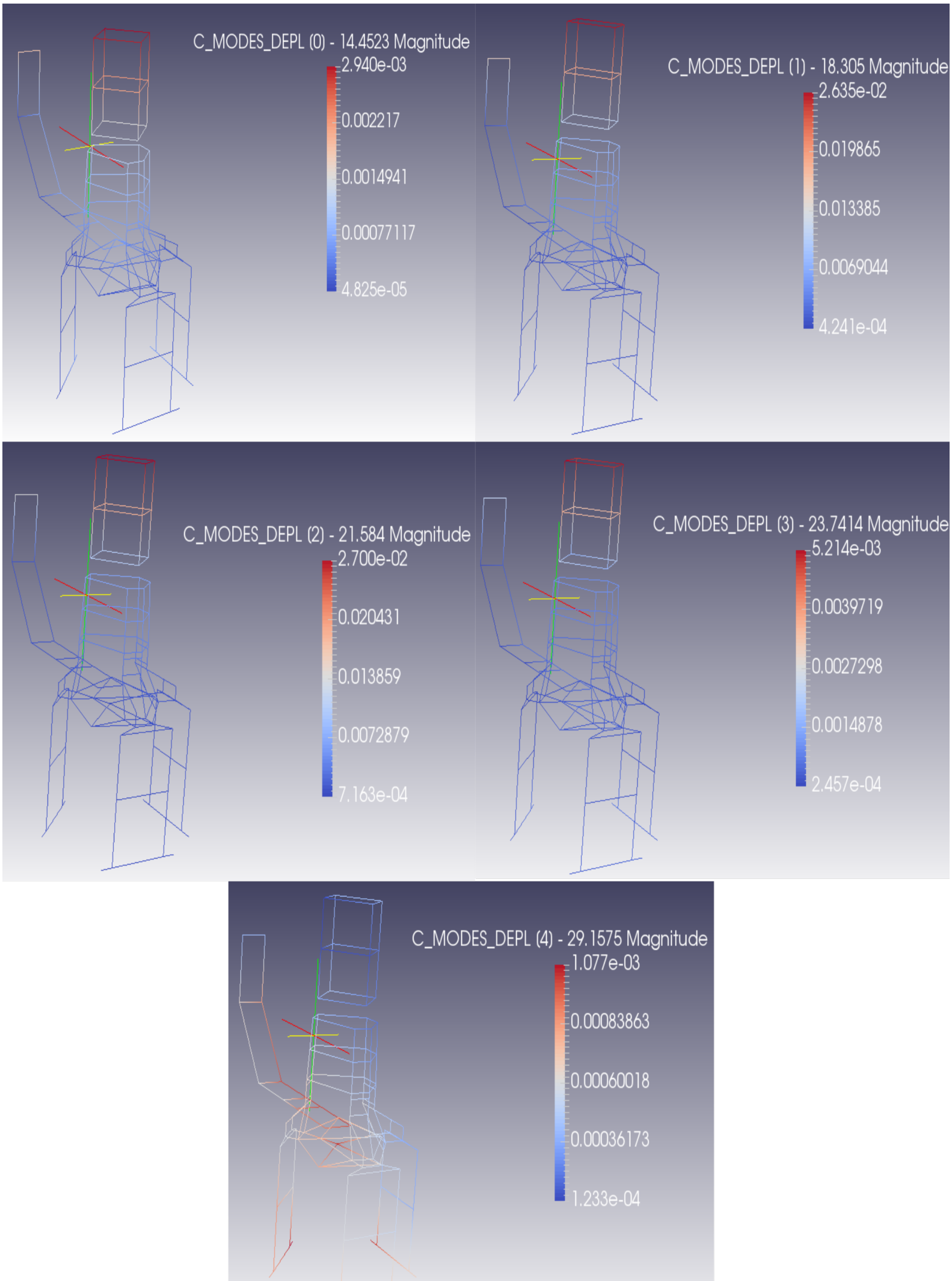


Figure 5.14.: The first five eigenmodes of the experimental mesh of the cooling water pump

On the other hand, a FE model, presented in Figure 5.15, is created to study the structure's behavior. This model is not meant to accurately represent the geometry and details of the pump. It simply serves as a support for the construction of a finite element model which must reproduce faithfully the first modes of the pump.

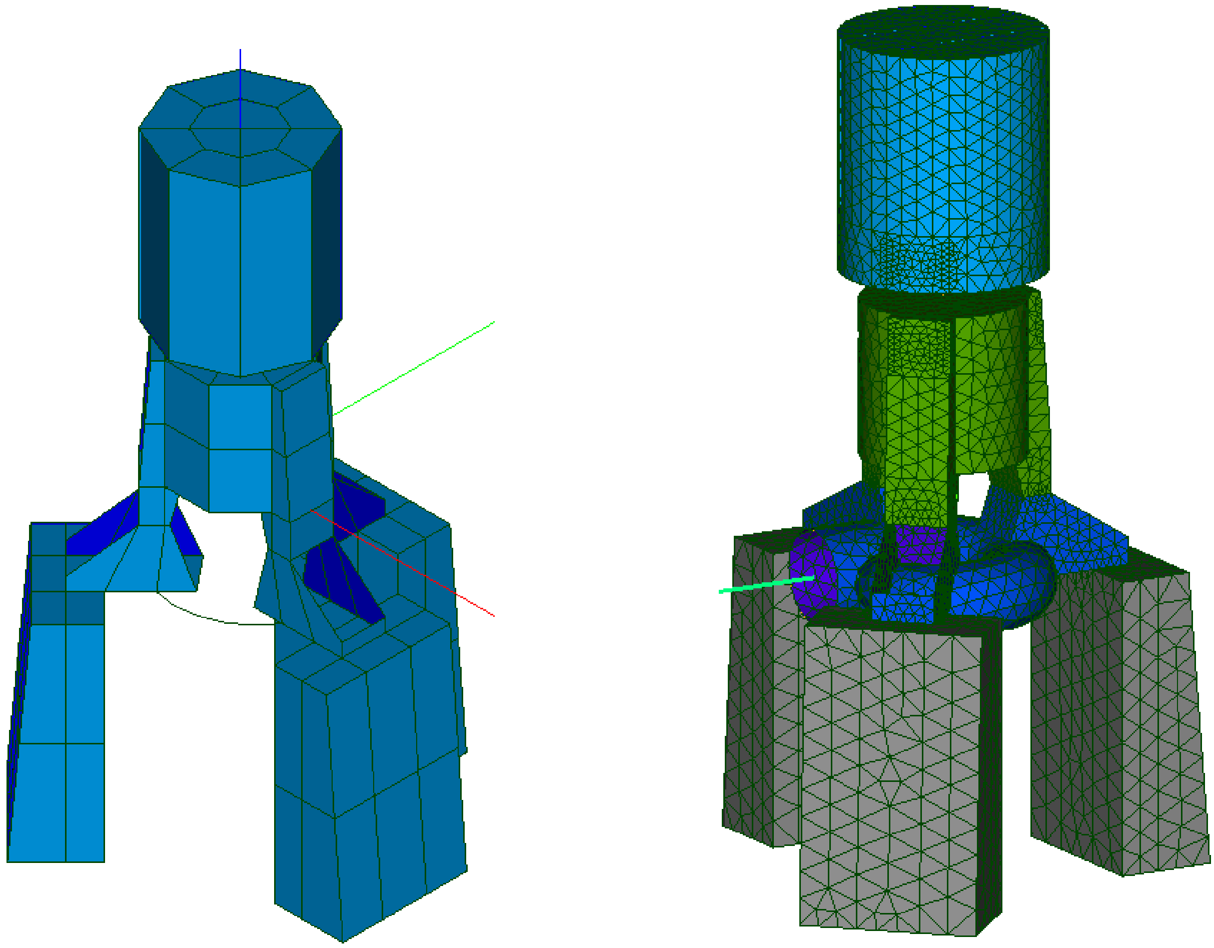


Figure 5.15.: The 3D model and the numerical mesh of the cooling water pump

Only the studs are meshed with 3D elements. The motor, the bearing support and the pump body are modeled using shell elements. The piping is modeled using Euler straight beam elements. The thicknesses attributed to the main components contribute to obtaining a mass similar to that of the real components.

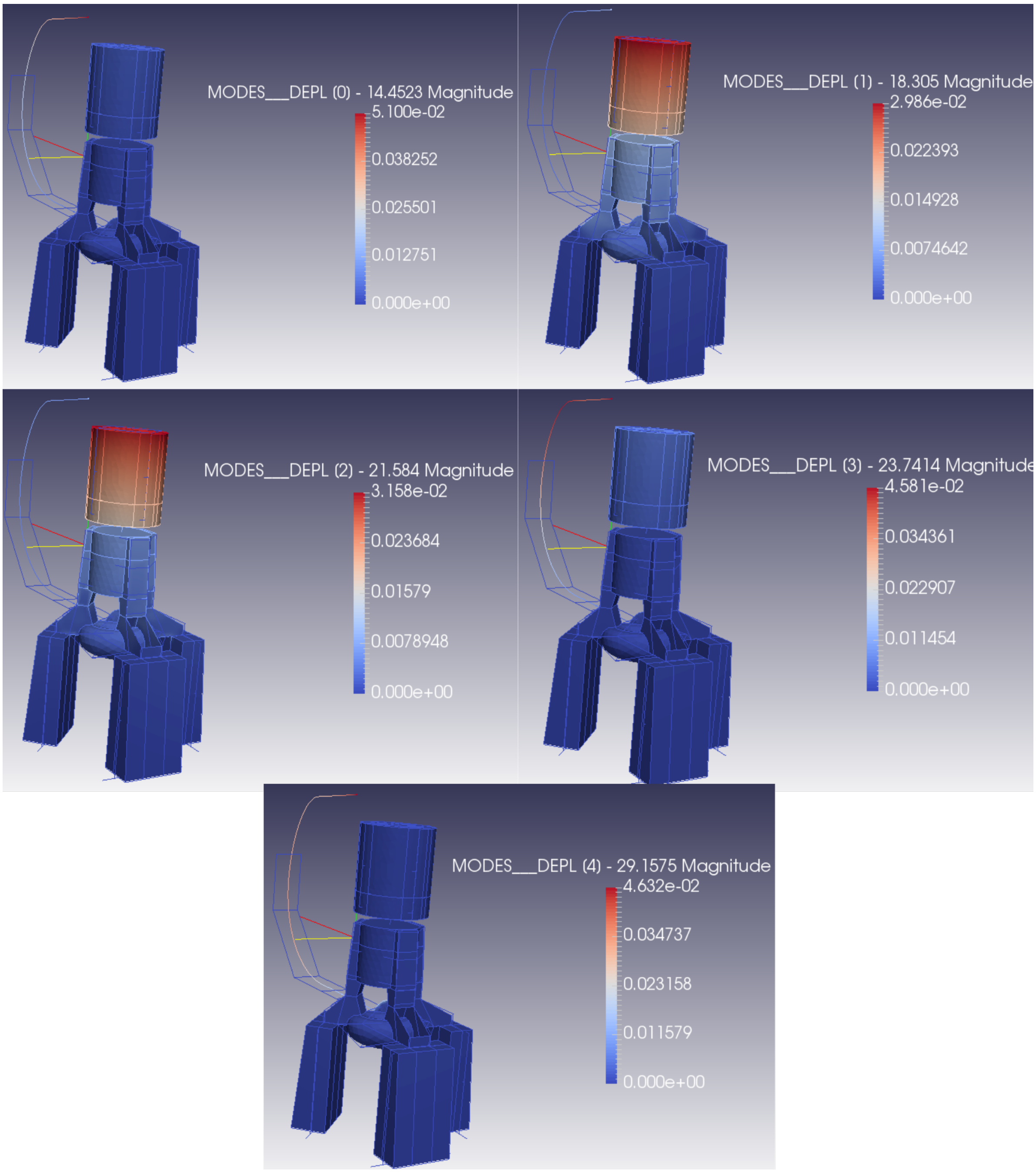


Figure 5.16.: The first five eigenmodes of the numerical mesh of the cooling water pump

We use the standard nominal parameters for concrete ($E = 1.92\ e10$ Pa and $\rho = 2300kg/m^3$) and steels ($E = 2.1\ e11$ Pa and $rho = 2300kg/m^3$). The thicknesses of the motor, the bearing bracket and the pump body were calculated to obtain the nominal masses of the components.

Table 5.13 shows the size and the nonzero elements number of each block matrix in the coefficient matrix of the studied saddle point linear system 5.1. We note that the number of sensors used on the structure is equal to $s = 324$ and the number of Lagrange multipliers is equal to $m = 2,046$.

The matrix	Size	Nonzero elements number
The stiffness matrix $[\widetilde{K}(\theta)]$	37,005 × 37,005	1,198,533
The mass matrix $[\widetilde{M}(\theta)]$	37,005 × 37,005	846,419
The observation matrix $\widetilde{\Pi}$	324 × 37,005	714
The norm matrix $[\widetilde{K}_r]$	324 × 324	52,650
$\frac{r}{1-r}\widetilde{\Pi}^T[\widetilde{K}_r]\widetilde{\Pi}$	37,005 × 37,005	254,898
The saddle point matrix	74,010 × 74,010	4,286,083

Table 5.13.: The size and sparsity of the generated coefficient matrix and its sub-blocks

Figure 5.17 shows the structure pattern of the coefficient matrix of the saddle point linear system 5.1.

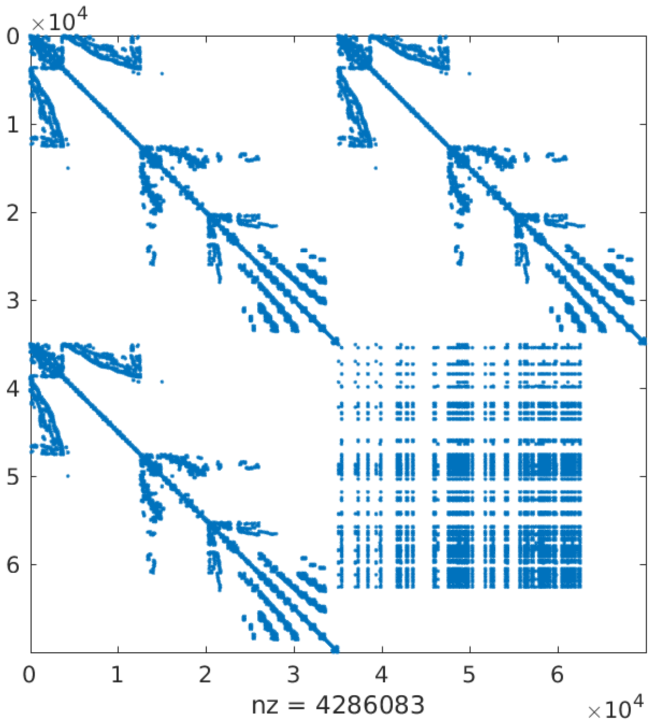


Figure 5.17.: The structure of the coefficient matrix of the saddle point linear system 5.1 from the pump application

This system have a condition number of order of 10^{11} even if it is smaller then the turbogenertor test case. This notably related to the strong spatial variations of the size of the mesh elements. As will be shown later in this chapter, this implies an ill-conditioned constraint preconditioner, which may increase the number of iterations of the GMRES solution method.

5.2.2. Application of the double projection approach

As done in previous test case, we begin by computing the null basis, then we generate the projected saddle point system.

Application of the explicit nullspace projection onto the kinematics constraints

Table 5.14 details the steps to build the null basis of C. Clearly, computing a nullbasis Z from the constraint matrix C of the pump numerical model is not expansive.

Steps	CPU time (sec)	Memory (MB)
LU factorization of C^T	1.20e–02	1.85
Building the inverse permutation	1.60e–02	0.00
Getting L_1 size $2,046 \times 2,046$ nonzero elements $3,942$	-	-
Getting L_2 size $32,913 \times 2,046$ nonzero elements 454	-	-
Getting L_1^T and L_2^T	3.59e–02	0.10
$L_1^{-T} L_2^T$	1.19e–02	0.46
Stocking the nullbasis Z	7.20e–02	2.54
Total	1.47z–01	19,13

Table 5.14.: Steps of building the null basis of the kinematic constraint matrix

Finally, we show in Table 5.15 the numerical properties of the global saddle point system and its equivalent projected one. Here, the number of Lagrange multipliers is limited, which explains the small difference between both sizes of the full and reduced systems. We recall that the reduced system has also a saddle point structure.

Physical dofs (n) Lagrange dofs (m)	34,959 2,046
The constraint matrix C	2,046×34,959 $nnz = 3,692$
The nullspace basis Z	34,959×32,913 $nnz = 36,592$
Full system size Full system nnz	74,010 4,286,083
Reduced system size Reduced system nnz	65,826 4,549,550

Table 5.15.: Information about the full saddle point system and the projected one

In the following, we study the performance of the constraint preconditioner $\mathcal{P}_{Chol}$, then we compare it to the preconditioner $\mathcal{P}_{Chol_{BoomerAMG}}$. From the alternator test case, we observe the inefficiency of the preconditioner $\mathcal{P}_{Chol_{ILUT}}$, that is why we limit our study to the multigrid approximation in the pump test case. We also limit our parallel performance test to 4 processors because of the moderate size of the application.

Demonstration of the computational limitation of the developed approach

Table 5.16 collects the results for the different simulations.

Whatever the number of processors is, we observe that the use of the constraint preconditioners leads to a significant decrease in terms of memory needs. We also notice an increasing number of iterations over the simulations using the preconditioner $\mathcal{P}_{Chol}$ with a moderate number of computational operations. A parallel efficiency of 77% in terms of computational time at a price of using 4 processors is obtained on the $\mathcal{P}_{Chol}$ calculation which is a rather satisfactory result.

Direct solver MUMPS	CPU Time (sec)		Memory (MB)	
1 proc	3.932e+01		1.445+03	
Preconditioner $\mathcal{P}_{Chol}$	CPU Time (sec)	Iterations	Flops	Memory (MB)
1 proc	1.514e+02	647	5.213e+09	8.105e+01
2 procs	7.725e+01	639	2.724e+09	4.521e+01
4 procs	3.921e+01	649	1.631e+09	2.705e+01
Preconditioner $\mathcal{P}_{Chol_{BoomerAMG}}$	CPU Time (sec)	Iterations	Flops	Memory (MB)
1 proc	2.554e+02	154	8.458e+10	6.782e+01
2 procs	1.671e+02	159	4.684e+10	3.652e+01
4 procs	9.170e+01	154	2.847e+10	2.014e+01

Table 5.16.: Evaluation of the constraint preconditioner $\mathcal{P}_{Chol}$ and $\mathcal{P}_{Chol_{BoomerAMG}}$ applied on the pump test case and running on 1, 2 and 4 processors

Table 5.16 emphasized the limitations of the proposed approach. In comparison to direct solver and apart from the memory cost, no gain in CPU time is obtained for the simulations using $\mathcal{P}_{Chol}$ in parallel. We reach comparable performance to direct solver using 4 processors. On this application, $\mathcal{P}_{Chol_{BoomerAMG}}$ is also less efficient than $\mathcal{P}_{Chol}$ in terms of CPU time and number of operations, even if this latter needs slightly a larger cost in memory. Besides, we note a high number of iterations, this is probably due to the poor condition number of the used constraint preconditioners and inherited from the coefficient matrix.

Thus, for medium and small size applications, direct solvers may be preferable since they give a satisfying result in term of computational time, memory and stability using a sequential calculation.

5.3. Conclusion

The chapter deals with real-life applications leading to sequences of symmetric saddle point linear systems. As reported in Chapter 4, the proposed iterative process is implemented in a parallel distributed memory environment through the PETSc libraries. In practice, the parallel framework allows us to consider selected large-scale industrial problems in a limited amount of computational time on a moderate number of cores.

This chapter highlights many findings. First, numerical experiments show that the explicit projection onto the nullspace of the kinematic constraints is the less consuming step in the iterative solution method developed in Chapter 4. Then, results emphasize the efficiency of the class of constraint preconditioners on a large-scale industrial application. A saving of up to 57% in terms of computational time is obtained with respect to the direct solver MUMPS on this application running on 16 processors.

Simultaneously, we have compared the performance of incomplete factorization and multigrid based approximations of the constraint preconditioner ($\mathcal{P}_{Chol_{ILUT}}$ and $\mathcal{P}_{Chol_{BoomerAMG}}$). Their performance is also tested in parallel setup. In terms of CPU time, they are less efficient, even if there is a substantial gain in terms of memory needs. When taking a less important relative tolerance, those preconditioners get a better gain, even if this direct factorization based constraint preconditioner gives slightly better results on the large-scale system illustrated here.

The second application showed us some limitations of the developed approach when dealing with medium and small size test cases. Actually, direct solver MUMPS performs better than the developed iterative method even when running on 1 processor. This latter method may involve an ill-conditioned constraint preconditioner and then increases the number of iterations, which makes it less competitive in terms of computational time.

Chapter 6

Conclusions and future research

Conclusions

This work aims at improving the solution of identification problems in structural dynamics within the industrial context of EDF's research and development needs. The main objective is to provide efficient methods in term of memory cost to solve the sequences of linear systems, arising from those problems. This guarantee the application of energy based functional approach for model updating or data expansion within the open source *Code_Aster*® software, a finite element code developed at EDF, which is used as a simulation tool by the engineering departments to produce notably safety analysis.

Chapter 3 subsequently studied different approaches to design efficient direct sparse solvers. The investigations carried out in this chapter were not designed to be very thorough, the purpose was mainly to design a direct solution method that was cheap to apply and reasonably effective at reducing the fill-in of the factor matrices required in solving the saddle point system.

The first strategy that has been investigated studied the adequacy of mixing the steps of factorization and ordering to address additional fill-in due to pivoting strategies. This approach was motivated for its purpose to get an ordering based on approximate minimum degree algorithm applied to a weighted graph. Taking numerical values into account has helped to avoid destroying the initial fill-in reducing ordering. Broadly speaking, this strategy showed a good ability to compete other unsymmetric direct solvers in term of fill-in reduction, when applied to the studied saddle point systems. Though, many difficulties appeared. Indeed, the approach requires an implementation with dynamic memory management of the LU factor which prohibits the exploitation of an elimination tree in a proper symbolic factorization.

Besides, although we study a very particular saddle point systems, their specific structure is not exploited.
To do so, a second strategy combining the specific structure of the systems within a block factorization has been developed. This approach allows to take advantage of the individual feature of the studied system in such a way we gain inherent fill-in reduction. This new factorization considers doing micro-factorizations after permuting the saddle point system. The proposed permutation is based on pairing every entry on the diagonal of the $(1, 1)$ block with a corresponding entry in the constraint block $(2, 1)$ so that the entries on the diagonal of the permuted matrix form micro 2-by-2 block saddle point systems. Then, considering the compressed graph formed by those supernodes of order 2 and associated with the coefficient matrix and reordering it. A comprehensive effort has been considered to maintain the numerical stability during the factorization process to avoid factors growth. Much more attention was paid to construct a suitable sparse and stable direct method, than to focus on building an efficient LDL^T factorization. Thus, we implemented and performed different tests on MATLAB® which implied choosing medium size test cases for this chapter. Through numerical experiments, this approach led to significant gain in term of fill-in reduction up to 50% in comparison with symmetric sparse solvers, and up to 90% in comparison with unsymmetric sparse solvers for the studied sequence of saddle point linear systems. In term of numerical stability, we outperformed symmetric sparse solvers, but we were less stable than unsymmetric ones. Hence, this approach showed how important it can be to take the structure of the coefficient matrix into account to build an associated compressed graph. However, The memory cost of the developed algorithm remained expensive for medium size test cases, which confirmed that the proposed approach is not adapted for large scale real life applications.

The challenge of proposing another class of solvers, able to handle the studied sequence of saddle point linear systems with regard of negligible memory cost, has been addressed in Chapter 4. In this framework, the thesis has contributed to the research area related to algebraic block preconditioning for the GMRES Krylov subspace method. This central part of the research subsequently studied the adequacy of a projection onto the nullspace of the constraints to address problems of industrial relevance. This approach has been made possible by making a distinction in the existing constraints. While kinematic constraints are fixed linear and affine conditions, the constraints related to sensors degrees of freedom are varying along the sequence of saddle point linear systems. This distinction has dictated the type of projection of each kind of constraints. We project the system onto the nullspace of the kinematic constraints by computing an explicit nullspace basis. This latter step is generally computationally too expensive in practice, but reveals that in our case is relevant due the nested special structure of the studied saddle point systems. The second projection onto the nullspace of remaining varying constraints is performed through applying constraint preconditioners with the action of the GMRES method. In this sense, we have contributed in studying several approximations of the blocks of those preconditioners, especially for the Schur complement block.

The double explicit implicit projection algorithm 4.1 has been implemented within different standalone codes for each step. In *Code_Aster*®, we developed a subroutine to retrieve essential structural matrices and information from test cases. Then, using a Fortran interface to *SuperLU*, we developed an implementation to build the nullspace basis. Finally, we developed the iterative solution method of the projected system in C using PETSc. In this latter, we developed a user method based on the Schur complement block factorization, that enables taking into account the block structure of the constraint preconditioners. Actually, instead of factoring those preconditioners with available global codes such as MUMPS, we implemented in PETSc a GMRES outer loop for the whole system, that incorporates two block linear systems using PCG inner loops.

Thanks to this code, the efficiency of this approach has been illustrated on several problems in structural mechanics. In fact, the studied approach is particularly well suited to improve the convergence of the iterative solver. A first medium-scale problem has revealed a huge gain in terms of GMRES iteration count and has strengthened our belief in selecting the proposed problem-dependent approximation of the Schur complement. Then, two large-scale real-life problems have been studied in Chapter 5. The first application is a turbogenerator of a nuclear power plant, and leads to significant gains in terms of computational time. The second application is a medium size cooling water pump, and proves that the developed approach may be less appealing for medium and small size applications, since direct solvers give better results in regards to computational time. Finally, all these results have been obtained at a negligible memory requirement. A relevant point has been illustrated in different applications, concerning the scalability of this class of preconditioners proposed by replacing the exact factorization by incomplete one or multigrid method as possible approximations. On a large-scale problem, this technique enables us to preserve the degree of scalability of the constraint preconditioner.

We provided here a new efficient solution approach to speedup the solution of constrained optimization problems arising in large-scale identification applications. The contribution of this thesis is therefore useful for industrial applications such as updating structural finite element models from test data, or identifying unknown parameters [34].

Perspectives

All these contributions let us address interesting perspectives and mention several extensions to the present research. We can distinguish

- In the second direct approach developed in Chapter 3, the Minimum degree algorithm is just one possibility for obtaining suitable permutation of the compressed graph. There are other graph based methods and nested dissection algorithms that need exploring.
  For the approach developed in Chapter 4, an interesting approach would be to test another way to apply those preconditioners : *implicit factorization constraint preconditioners*, which is demonstrated to be very effective as well as being cheap to apply for classical constraint preconditioners [42], and it may be adapted here for our specific structure.

- In terms of implementation, we may in near future implement the developed direct symmetric approach of Chapter 3 in a solver that support the LDL^T factorization, for instance in MA47 [46].
  Also, in a future work, the implementation of different steps of Algorithm 4.1 in *Code_Aster*® environment will increase their numerical performance. Actually, this software uses PETSc as an iterative solver, we may then develop our approach directly within it. For the explicit step, in which we need to compute a sparse nullspace basis, it is possible to integrate *SuperLU* as an external package of PETSc, in order to gather all developments in the same platform. This future action aims an integration of the developed approach into a future distribution of the open-source computational platform.

- Once all developments are integrated in the same version of the mechanical software *Code_Aster*®. This latter can be used to deal with real life industrial applications of the energy-based functional approach. For instance, we may update large scale structural finite element models from test data, or identify unknown parameters.

Appendix A

Sparse direct solution methods

The basic idea of direct methods is to decompose the coefficient matrix of the linear system into a product of particular matrices (triangular lower and upper, diagonal) that are easier to reverse. This is called factorization or decomposition. One of the most widely known direct methods is that of Gaussian Elimination. It transforms a linear system into an upper triangular one by introducing zeros below the diagonal, first in column 1, then column 2, and so on. To do this we subtract multiples of each row with subsequent rows. We can think of this as being equivalent to premultiplying the matrix by a sequence of lower triangular matrices.

A.1. Sparse matrix factorization

A sparse matrix is a matrix with a high proportion of zero coefficients. Practically, sparse matrices correspond to systems of equations in which each equation involves a very small number of unknowns. When a matrix A is sparse, its inverse A^{-1} is generally much denser A^{-1} contains many non-zero terms. Moreover, the triangular matrices L and U resulting from factorization $A = LU$ may be reasonably sparse when using suitable techniques. Hence, many algorithms and data structures are designed in order to take advantage of this sparse structure. The goal is to reduce the memory and avoid computation costs that would be induced by a dense computation. Conventional matrix operations can be performed economically without storing the null elements of the matrix and without performing unnecessary computation between null elements.

The algorithm A.1 presents LU factorization algorithm of a sparse matrix A. The matrix A

is factorized in-place by replacing its terms with those of the two triangular matrices L and U. When the matrix is sparse, only the non-zero entries are stored and the operations on the null entries are then deleted. We notice if the entry a_{ij} is null before the factorization, a new non-zero entry is introduced into the factorized matrices (Line 6 in algorithm A.1). It is then clear that during the factorization process, new non-zero entries are created and the factors are denser than the initial matrix. This phenomenon is called **fill-in**.

Algorithm A.1: kij in-place version of LU factorization algorithm

Data: A sparse Matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$

Result: Matrices L, U such that $LU = A$

```
1 initialization;
2 for k ← 1 to n do
3     for i ← k + 1 to n − 1 do
4         a_ik ← a_ik / a_kk
5         for j ← k + 1 to n do
6             a_ij ← a_ij − a_ik a_kj
```

Fill-in is the main challenge for sparse direct methods. Indeed, the problems to be solved have sparse matrices, but the decomposition algorithms (Gauss, LU, Cholesky, etc.) generally lead to the construction of very full matrices, which implies the emergence of storage problems And computation time costs. Let us take the famous example illustrated in Figure A.1. This matrix entries are null except those of the first row, the first column and the diagonal, it is pretty clear that the matrix becomes full after the first step of factorization.

$$\begin{bmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & 0 & 0 & 0 \\ \blacksquare & 0 & \blacksquare & 0 & 0 \\ \blacksquare & 0 & 0 & \blacksquare & 0 \\ \blacksquare & 0 & 0 & 0 & \blacksquare \end{bmatrix}$$

Figure A.1.: An example of a sparse matrix that has a full fill-in

Many techniques have been developed to enable efficient factorization of sparse matrices. It is then possible to reduce the fill-in and predict the structure of the factors. Indeed, direct solvers are usually implemented with a preprocessing step before the factorization. This includes scaling, pivoting and ordering. The preprocessing step makes the numerical

factorization in many cases easier and cheaper, which influences the memory and the CPU time of the factorization step.
Solving a large sparse linear system $Ax = b$ is generally composed of four major phases [47] as shown in Figure A.2

1. The analysis phase (also called the preprocessing step) It consists in applying a permutation P on the working matrix A in order to change the order of elimination of the unknowns of the system, in order to reduce the phenomenon of fill-in of the new matrix PAP^T.

2. The symbolic factorization phase its purpose is to determine the structure of the factor matrices before the numerical factorization step. It avoids the use of dynamic data structures. The potential drawback to using dynamic data structures, though, is that because allocation of memory is not fixed, there is the possibility for the structure to overflow if it exceeds the maximum allowed memory limit or underflow if the data structure becomes empty.

3. The numerical factorization phase it is in this stage that the computation of numerical values of factors L and U, taking into account the changes made during the analysis phase.

4. The "solve" phase It enables the computation of the numerical solution by forward and backward substitution. Indeed, it solves the triangular systems $y = Ux$ and $Ly = b$.

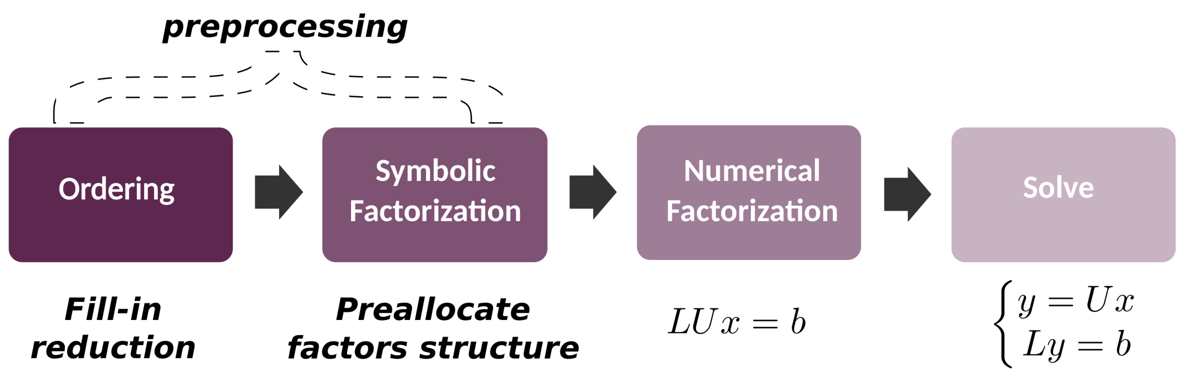


Figure A.2.: Different steps of sparse direct methods

The main issue in sparse matrices comes from the fill-in in the matrix factorization. As the choice of pivots affects the sparsity of the resulting factors, a fill-in reducing ordering is usually applied. Nevertheless, a very large number of ordering alterations is needed in order to choose a stable pivot and that implies generating the fill-in in a somewhat unpredictable way due to this pivoting which involves tracking it dynamically. This impacts dramatically the computation time and yields to excessive fill-in[17]. As shown in [44], preserving sparsity may conflict with pivoting, so there is a need to reach a compromise. In [49], it is introduced a method that used the minimum degree algorithm [78] with relative pivot tolerance for stability for symmetric indefinite factorization. In [48], we find the multifrontal approach used in many solvers like MUMPS (Multifrontal Massively Parallel Solver) [3].

Bibliography

[1] Code_aster ® open source - general finite element analysis software. (cite p. 19 et 108)

[2] Hyper : scalable linear solvers and multigrid methods. https://computation.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods. (cite p. 122)

[3] Amestoy, P. R., Duff, I. S., Koster, J., and L'Excellent, J.-Y. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications 23*, 1 (2001), 15–41. (cite p. 19, 45, 47, 70, 113 et 164)

[4] Amestoy, P. R., Duff, I. S., L'excellent, J. Y., and Li, X. S. Analysis and comparison of two general sparse solvers for distributed memory computers. *ACM Trans* (2000). (cite p. 71 et 78)

[5] Andriambololona, H. Optimisation des essais et recalage de modeles structuraux. (cite p. 26)

[6] Andrieux, S., Abda, A. B., and Bui, H. D. Sur l'identification de fissures planes via le concept d'écart à la réciprocité en élasticité. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics 324*, 12 (1997), 1431–1438. (cite p. 25)

[7] Arioli, M., Demmel, J. W., and Duff, I. S. Solving sparse linear systems with sparse backward error. *SIAM Journal on Matrix Analysis and Applications 10*, 2 (1989), 165–190. (cite p. 71)

[8] Ashcraft, C., Grimes, R. G., and Lewis, J. G. Accurate symmetric indefinite linear equation solvers. *SIAM Journal on Matrix Analysis and Applications 20*, 2 (1998), 513–561. (cite p. 45 et 84)

[9] Astier, J. G., Dutrech, L., Lebailly, P., and Playe, G. note ht-61/05/006/a

projet vital : Impact du suivi de réseau sur le vieillissement des alternateurs 900 mw. rapport technique. *EDF R&D*. (cite p. 131)

[10] Baboulin, M., Li, X., and Rouet, F. Using random butterfly transformations to avoid pivoting in sparse direct methods. *High Performance Computing for Computational Science – VECPAR 2014* (2015), 135–144. (cite p. 45)

[11] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Rupp, K., Smith, B. F., Zampini, S., Zhang, H., and Zhang, H. PETSc Web page. http://www.mcs.anl.gov/petsc, 2016. (cite p. 55, 108 et 113)

[12] Banerjee, B., Walsh, T. F., Aquino, W., and Bonnet, M. Large scale parameter estimation problems in frequency-domain elastodynamics using an error in constitutive equation functional. *Computer methods in applied mechanics and engineering 253* (2013), 60–72. (cite p. 29)

[13] Barthe, D., Ladevèze, P., Deraemaecker, A., and Le Loch, S. Validation and updating of industrial models based on the constitutive relation error. *AIAA 42* (2004), 1427–1434. (cite p. 26)

[14] Ben Abdallah, J. Inversion gaussienne appliquée à la correction paramétrique de modèles structuraux. (cite p. 26)

[15] Ben-Haim, Y. Identification of certain polynomial nonlinear structures by adaptive selectively-sensitive excitation. *Transactions - American Society of Mechanical Engineers journal of vibration and acoustics 115* (1993), 246–246. (cite p. 26)

[16] Benzi, M. Preconditioning techniques for large linear systems: a survey. *Journal of computational Physics 182*, 2 (2002), 418–477. (cite p. 53)

[17] Benzi, M., Golub, G., and Liesen, J. Numerical solution of saddle point problems. *Acta Numerica* (2005), 1–137. (cite p. 19, 37, 44, 46, 48, 52, 53, 56, 92, 96 et 164)

[18] Benzi, M., and Liu, J. Block preconditioning for saddle point systems with indefinite (1, 1) block. *International Journal of Computer Mathematics 84*, 8 (2007), 1117–1129. (cite p. 114)

[19] Bobillot, A., and Balmès, E. Solving minimum dynamic residual expansion and using results for error localization. *In Proceeding of IMAC XIX 4359* (2001), 179–185. (cite p. 63)

[20] Boiteau, O. Mot-clé solveur. technical report u4.03.05, edf r&d, ama. (cite p. 46)

[21] Bonnet, M. Problèmes inverses. *Cours de DEA DSSC, Ecole Centrale Paris* (2004). (cite p. 24)

[22] Bonnet, M., and Constantinescu, A. Inverse problems in elasticity. *Inverse problems 21*, 2 (2005), R1. (cite p. 24, 26 et 28)

[23] Boutayeb, M., and Aubry, D. A strong tracking extended kalman observer for non-linear discrete-time systems. *IEEE Transactions on Automatic Control 44*, 8 (1999), 1550–1556. (cite p. 26)

[24] Buleev, N. Numerical method for solving two-and three-dimensional diffusion equations. *Matematicheskii Sbornik 93*, 2 (1960), 227–238. (cite p. 54)

[25] Bunch, J. R., and Kaufman, L. Some stable methods for calculating inertia and solving symmetric linear systems. *Mathematics of computation* (1977), 163–179. (cite p. 45)

[26] Bunch, J. R., and Parlett, B. N. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM Journal on Numerical Analysis 8*, 4 (1971), 639–655. (cite p. 45, 83 et 84)

[27] Campbell, S., Ipsen, I., Kelley, C., and Meyer, C. Gmres and the minimal polynomial. *BIT Numerical Mathematics 36*, 4 (1996), 664–675. (cite p. 52 et 53)

[28] Carsten, K., Gould, I., and Wathen, A. Constraint preconditioning for indefinite linear systems. 1300–1317. (cite p. 56, 100 et 101)

[29] Chouaki, A. T. Recalage de modèles dynamiques de structures avec amortissement. (cite p. 26)

[30] Cimetière, A., Delvare, F., and Pons, F. Une méthode inverse d'ordre un pour les problèmes de complétion de données. *Comptes rendus mécanique 333*, 2 (2005), 123–126. (cite p. 28)

[31] Coleman, T. F., and Pothen, A. The null space problem i. complexity. *SIAM Journal on Algebraic Discrete Methods 7*, 4 (1986), 527–537. (cite p. 92)

[32] Coleman, T. F., and Pothen, A. The null space problem ii. algorithms. *SIAM Journal on Algebraic Discrete Methods 8*, 4 (1987), 544–563. (cite p. 92)

[33] Cormen, L., and Leiserson, C. Rivest. *Introduction to algorithms 2* (1990). (cite p. 44)

[34] Cot, A. A. Une approche de l'identification en dynamique des structures combinant

l'erreur en relation de comportement et le filtrage de kalman. (cite p. 20, 25, 26, 28, 133 et 159)

[35] Davis, T. Matlab primer. (cite p. 44)

[36] Davis, T. UMFPACK User Guide. *Engineering* (2011), 1–140. (cite p. 70 et 71)

[37] De Niet, A., and Wubs, F. Numerically stable ldl t-factorization of f-type saddle point matrices. *IMA Journal of Numerical Analysis 29*, 1 (2008), 208–234. (cite p. 84)

[38] Deraemaecker, A. Sur la maitrise des modèles en dynamique des structures à partir de résultats d'essais. (cite p. 19)

[39] Deraemaecker, A., Ladevèze, P., and Leconte, P. Reduced bases for model updating in structural dynamics based on constitutive relation error. computer methods in applied mechanics and engineering. *Computer methods in applied mechanics and engineering 191* (2002), 2427–2444. (cite p. 63)

[40] Deraemaeker, A., Ladevèze, P., and Leconte, P. Reduced bases for model updating in structural dynamics based on constitutive relation error. *Computer methods in applied mechanics and engineering 191*, 21 (2002), 2427–2444. (cite p. 26)

[41] Dollar, H. S. Iterative linear algebra for constrained optimization. (cite p. 47 et 97)

[42] Dollar, H. S. Constraint-style preconditioners for regularized saddle point problems. *SIAM Journal on Matrix Analysis and Applications 29*, 2 (2007), 672–684. (cite p. 56, 100, 101, 102 et 160)

[43] Duff, I. The solution of augmented systems. *Pitman research notes in mathematics series* (1994), 40–40. (cite p. 44)

[44] Duff, I. The solution of augmented systems. (cite p. 164)

[45] Duff, I. Ma57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software (TOMS) 30*, 2 (2004), 118–144. (cite p. 85)

[46] Duff, I., Gould, N., Reid, J., Scott, J., and Turner, K. The factorization of sparse symmetric indefinite matrices. *IMA Journal of Numerical Analysis 11*, 2 (1991), 181–204. (cite p. 85 et 160)

[47] Duff, I. S., Erisman, A. M., and Reid, J. Direct methods for sparse matrices. (cite p. 163)

[48] Duff, I. S., and Reid, J. K. The multifrontal solution of indefinite sparse symmetric

linear. *ACM Transactions on Mathematical Software (TOMS) 9*, 3 (1983), 302–325. (cite p. 164)

[49] Duff, I. S., Reid, J. K., Munksgaard, N., and Nielsen, H. B. Direct solution of sets of linear equations whose matrix is sparse, symmetric and indefinite. *IMA Journal of Applied Mathematics 23*, 2 (1979), 235–250. (cite p. 164)

[50] Dutrech, L. Note h-61-2008-01569-fr procédure d'analyse modale applicable aux stators 900 mw bobinés en star. rapport technique,. *EDF R&D*. (cite p. 132)

[51] Dyn, N., and Ferguson, W. The numerical solution of equality constrained quadratic programming problems. *Mathematics of Computation 41*, 163 (1983), 165–170. (cite p. 36)

[52] Elman, H., Silvester, D., and Wathen, A. Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics. (cite p. 36)

[53] Ewins, D. J. Modal testing : Theory and practice. (cite p. 62)

[54] Feissel, P. Vers une stratégie d'identification en dynamique rapide pour des données incertaines. (cite p. 26)

[55] Feissel, P., and Allix, O. Modified constitutive relation error identification strategy for transient dynamics with corrupted data: the elastic case. *Computer methods in applied mechanics and engineering 196*, 13 (2007), 1968–1983. (cite p. 26)

[56] Gansterer, W., SchneidÝ, J., and UeberhuberÝ., C. Mathematical properties of equilibrium systems. (cite p. 44)

[57] Golub, G., and Van Loan, C. Matrix computations. (cite p. 45)

[58] Gould, N., Hribar, M., and Nocedal, J. On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM Journal on Scientific Computing 23*, 4 (2001), 1376–1395. (cite p. 56 et 99)

[59] Gould, N., Orban, D., and Rees, T. Projected krylov methods for saddle-point systems. *SIAM Journal on Matrix Analysis and Applications 35*, 4 (2014), 1329–1343. (cite p. 99)

[60] Greenbaum, A., Pták, V., and Strakoš, Z. Any nonincreasing convergence curve is possible for gmres. *Siam journal on matrix analysis and applications 17*, 3 (1996), 465–469. (cite p. 52)

[61] Gutknecht, M. A brief introduction to krylov space methods for solving linear systems. *Frontiers of Computational Science* (2007), 53–62. (cite p. 52)

[62] Hadj-Sassi, K. Une stratégie d'identification conjointe des paramètres et de l'état de structures à comportements non-linéaires. assimilation de données et erreur en loi de comportement. (cite p. 26)

[63] Hajj, I., Yang, P., and Trick, T. N. Avoiding zero pivots in the modified nodal approach. *Circuits and Systems, IEEE Transactions on 28* (1981), 271–279. (cite p. 45)

[64] Hénon, P., Ramet, P., and Roman, J. PaStiX: A High-Performance Parallel Direct Solver for Sparse Symmetric Definite Systems. *Parallel Computing 28*, 2 (2002), 301–321. (cite p. 113)

[65] Hestenes, M., and Stiefel, E. Methods of conjugate gradients for solving linear systems. (cite p. 50)

[66] Horn, R., and Johnson, C. Matrix analysis. (cite p. 41)

[67] Jimenez, R. Model based structural damage assessment using vibration measurements. (cite p. 26)

[68] Jimenez, R. Model based structural damage assessment using vibration measurements. (cite p. 63)

[69] Joubert, W. On the convergence behavior of the restarted gmres algorithm for solving nonsymmetric linear systems. *Numerical linear algebra with applications 1*, 5 (1994), 427–447. (cite p. 111)

[70] Kaustuv. *IPSOL: An Interior Point Solver for Nonconvex Optimization Problems.* PhD thesis, Stanford University, 2009. (cite p. 92 et 96)

[71] Kuczkowiak, A., Cogan, S., Ouisse, M., Foltete, E., and Corus, M. Robust expansion of experimental mode shapes under epistemic uncertainties. *Proceedings of the 32nd IMAC, A Conference and Exposition on Structural Dynamics 3* (2014), 419–427. (cite p. 19, 26, 62 et 133)

[72] Ladevèze, P. Comparaison de modeles de milieux continus. (cite p. 25 et 26)

[73] Ladevéze, P., and Moës, N. A new a posteriori error estimation for nonlinear time-dependent finite element analysis. *Computer Methods in Applied Mechanics and Engineering 157*, 1-2 (1998), 45–68. (cite p. 26)

[74] Ladeveze, P., Puel, G., Deraemaeker, A., and Romeuf, T. Validation of structural dynamics models containing uncertainties. *Computer methods in applied mechanics and engineering 195*, 4 (2006), 373–393. (cite p. 29)

[75] Ladeveze, P., Reynier, M., and Nedjar, D. Parametric correction of finite element models using modal tests. In *Inverse problems in engineering mechanics*. Springer, 1993, pp. 91–100. (cite p. 26)

[76] Ladevèze, P., and Waeytens, J. Model verification in dynamics through strict upper error bounds. *Computer Methods in Applied Mechanics and Engineering 198*, 21 (2009), 1775–1784. (cite p. 26)

[77] Li, X. S. An overview of superlu: Algorithms, implementation, and user interface. *ACM Trans. Math. Softw. 31*, 3 (Sept. 2005), 302–325. (cite p. 70 et 108)

[78] Liu, J. W. Modification of the minimum-degree algorithm by multiple elimination. *ACM Transactions on Mathematical Software (TOMS) 11*, 2 (1985), 141–153. (cite p. 164)

[79] Luksan, L., and Vlcek, J. Indefinitely preconditioned inexact newton method for large sparse equality constrained nonlinear programming problems. *Numerical linear algebra with applications 5*, 3 (1998), 219–247. (cite p. 56)

[80] Lungten, S., Schilders, W., and Maubach, J. Sparse inverse incidence matrices for schilders' factorization applied to resistor network modeling. (cite p. 48)

[81] Mathieu, C. Elimination des conditions aux limites dualisÃ©es. *Technical Report, EDF R&D, AMA,R3.03.05.* (2015). (cite p. 34)

[82] Meijerink, J., and van der Vorst, H. An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix. *Mathematics of computation 31*, 137 (1977), 148–162. (cite p. 54)

[83] Mercier, S. Fast nonlinear solvers in solid mechanics. (cite p. 32 et 36)

[84] Murphy, M. F., Golub, G. H., and Wathen, A. J. A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing 21*, 6 (2000), 1969–1972. (cite p. 56)

[85] Nguyen, H. M., Allix, O., and Feissel, P. A robust identification strategy for rate-dependent models in dynamics. *Inverse Problems 24*, 6 (2008), 065006. (cite p. 26)

[86] O'Callahan, J., Avitabile, P., and Riemer, R. System equivalent reduction expansion process (serep). In *Proceedings of the 7th international modal analysis conference* (1989), vol. 1, Union College Schnectady, NY, pp. 29–37. (cite p. 63)

[87] Paige, C., and Saunders, M. Solution of sparse indefinite systems of linear equations. *SIAM journal on numerical analysis 12*, 4 (1975), 617–629. (cite p. 51)

[88] Pellet, J. Dualisation des conditions limites. *Technical Report, EDF R&D, AMA,R3.03.01.* (2011). (cite p. 33, 34 et 35)

[89] Perugia, I., Simoncini, V., and Arioli, M. Linear algebra methods in a mixed approximation of magnetostatic problems. *SIAM Journal on Scientific Computing 21*, 3 (1999), 1085–1101. (cite p. 36)

[90] Puel, G., Bourgeteau, B., and Aubry, D. Parameter identification of nonlinear time-dependent rubber bushings models towards their integration in multibody simulations of a vehicle chassis. *Mechanical Systems and Signal Processing 36*, 2 (2013), 354–369. (cite p. 29)

[91] Qu, Z. Model order reduction techniques with applications in finite element analysis. (cite p. 63)

[92] Rees, T., and Scott, J. A comparative study of null-space factorizations for sparse symmetric saddle point systems. *Numerical Linear Algebra with Applications* (2017). (cite p. 48, 49 et 79)

[93] Reid, J. On the method of conjugate gradients for the solution of large sparse systems of linear equations. In *Pro. the Oxford conference of institute of mathmatics and its applications* (1971), pp. 231–254. (cite p. 50)

[94] Reynier, M. Sur le controle de modélisations par éléments finis : recalage à partir d'essais dynamiques. (cite p. 19)

[95] Rozloznik, M., Okulicka-Dłuzewska, F., and Smoktunowicz, A. Indefinite orthogonalization with rounding errors. (cite p. 46)

[96] Rusten, T., and Winther, R. A preconditioned iterative method for saddlepoint problems. *SIAM Journal on Matrix Analysis and Applications 13*, 3 (1992), 887–904. (cite p. 55)

[97] Saad, Y. Iterative methods for sparse linear systems. (cite p. 20, 50, 51 et 54)

[98] Saad, Y., and Schultz, M. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing 7*, 3 (1986), 856–869. (cite p. 51, 52 et 112)

[99] Schöberl, J., and Zulehner, W. Symmetric indefinite preconditioners for saddle

point problems with applications to pde-constrained optimization problems. *SIAM Journal on Matrix Analysis and Applications 29*, 3 (2007), 752–773. (cite p. 99)

[100] Shewchuk, J. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). *University of California at Berkeley 73* (2002). (cite p. 137)

[101] Sifuentes, J. Preconditioning the integral formulation of the helmholtz equation via deflation. (cite p. 52)

[102] Simoncini, V. Block triangular preconditioners for symmetric saddle-point problems. *Applied Numerical Mathematics 49*, 1 (2004), 63–80. (cite p. 56)

[103] Simoncini, V., and Szyld, D. Recent computational developments in krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications 14*, 1 (2007), 1–59. (cite p. 19)

[104] Sontag, E. D. Mathematical control theory, vol. 6 of texts in applied mathematics. (cite p. 36)

[105] Stott, B., and Alsac, O. an overview of sparse matrix techniques for on-line network applications. *IFAC Power Systems and Power Plant Control* (1986). (cite p. 45)

[106] Tarantola, A. Inverse problem theory and methods for model parameter estimation. (cite p. 24 et 28)

[107] Tarantola, A., and Valette, B. Generalized nonlinear inverse problems solved using the least squares criterion. *Reviews of Geophysics 20*, 2 (1982), 219–232. (cite p. 25)

[108] Tikhonov, A., and Arsenin, V. Solutions of ill-posed problems. vh winston & sons, 1977. (cite p. 28)

[109] Van der Vorst, H. Iterative krylov methods for large linear systems. (cite p. 53)

[110] Van der Vorst, H., and Vuik, C. The superlinear convergence behaviour of gmres. *Journal of computational and applied mathematics 48*, 3 (1993), 327–341. (cite p. 111)

[111] Vavasis, S. Stable numerical algorithms for equilibrium systems. *SIAM Journal on Matrix Analysis and Applications 15*, 4 (1994), 1108–1131. (cite p. 44)

ÉCOLE DOCTORALE
Sciences mécaniques et énergétiques, matériaux et géosciences (SMEMAG)

Titre : Solveurs performants pour l'optimisation sous contraintes en identification de paramètres

Mots clefs : Optimisation sous contraintes, Systèmes point-selle, Préconditionneurs par blocs, Problèmes inverses

Résumé : Cette thèse vise à concevoir des solveurs efficaces pour résoudre des systèmes linéaires, résultant des problèmes d'optimisation sous contraintes dans certaines applications de dynamique des structures et vibration (la corrélation calcul-essai, la localisation d'erreur, le modèle hybride, l'évaluation des dommages, etc.). Ces applications reposent sur la résolution de problèmes inverses, exprimés sous la forme de la minimisation d'une fonctionnelle en énergie. Cette fonctionelle implique à la fois, des données issues d'un modèle numérique éléments finis, et des essais expérimentaux. Ceci conduit à des modèles de haute qualité, mais les systèmes linéaires point-selle associés, sont coûteux à résoudre. Nous proposons deux classes différentes de méthodes pour traiter le système. La première classe repose sur une méthode de factorisation directe profitant de la topologie et des propriétés spéciales de la matrice point-selle. Après une première renumérotation pour regrouper les pivots en blocs d'ordre 2. L'élimination de Gauss est conduite à partir de ces pivots et en utilisant un ordre spécial d'élimination réduisant le remplissage. Les résultats numériques confirment des gains significatifs en terme de remplissage, jusqu'à deux fois meilleurs que la littérature pour la topologie étudiée. La seconde classe de solveurs propose une approche à double projection du système étudié sur le noyau des contraintes, en faisant une distinction entre les contraintes cinématiques et celles reliées aux capteurs sur la structure. La première projection est explicite en utilisant une base creuse du noyau. La deuxième est implicite. Elle est basée sur l'emploi d'un préconditionneur contraint avec des méthodes itératives de type Krylov. Différentes approximations des blocs du préconditionneur sont proposées. L'approche est implémentée dans un environnement distribué parallèle utilisant la bibliothèque PETSc. Des gains significatifs en terme de coût de calcul et de mémoire sont illustrés sur plusieurs applications industrielles.

Title : Efficient solvers for constrained optimization in parameter identification problems

Keywords : Constrained optimization, Saddle point systems, Block preconditioners, Inverse problems

Abstract : This thesis aims at designing efficient numerical solution methods to solve linear systems, arising in constrained optimization problems in some structural dynamics and vibration applications (test-analysis correlation, model error localization, hybrid model, damage assessment, etc.). These applications rely on solving inverse problems, by means of minimization of an energy-based functional. This latter involves both data from a numerical finite element model and from experimental tests, which leads to high quality models, but the associated linear systems, that have a saddle-point coefficient matrices, are long and costly to solve. We propose two different classes of methods to deal with these problems. First, a direct factorization method that takes advantage of the special structures and properties of these saddle point matrices. The Gaussian elimination factorization is implemented in order to factorize the saddle point matrices block-wise with small blocks of orders 2 and using a fill-in reducing topological ordering. We obtain significant gains in memory cost (up to 50%) due to enhanced factors sparsity in comparison to literature. The second class is based on a double projection of the generated saddle point system onto the nullspace of the constraints. The first projection onto the kinematic constraints is proposed as an explicit process through the computation of a sparse null basis. Then, we detail the application of a constraint preconditioner within a Krylov subspace solver, as an implicit second projection of the system onto the nullspace of the sensors constraints. We further present and compare different approximations of the constraint preconditioner. The approach is implemented in a parallel distributed environment using the PETSc library. Significant gains in computational cost and memory are illustrated on several industrial applications.

Université Paris-Saclay

Espace Technologique - Immeuble Discovery - Route de l'Orme aux Merisiers RD 128, 91190 Saint-Aubin, France