Statics and inverse dynamics solvers based on strain-mode disassembly

François M. Hemez* — **Emmanuel Pagnacco****

* *Engineering Sciences and Applications (ESA–EA)*
Los Alamos National Laboratory, P.O. Box 1663,
M/S P946, Los Alamos, New Mexico 87545, USA
hemez@lanl.gov

** *Laboratoire de Mécanique de Rouen (LMR)*
UPRES-A CNRS 6104 & INSA de Rouen
BP 8, F-76801 Saint Etienne-du-Rouvray
Emmanuel.Pagnacco@insa-rouen.fr

ABSTRACT. The finite element method is widely used in design engineering for modeling and analyzing structural systems. Two approaches have been developed: the force–based method that exploits the equilibrium of forces and moments at nodal joints of the mesh to formulate the assembly of element–level matrices into master mass and stiffness matrices and its dual counterpart, the flexibility–based method. An alternative formulation of stiffness–based finite element assembly is proposed that decomposes element–level matrices even further into strain mode contributions. This decomposition (referred to as finite element disassembly here) allows the derivation of an efficient numerical solver. It is shown that a single matrix factorization is required for analyzing all models characterized by the same topology. This makes finite element disassembly and the associated inverse solver ideal in cases where multiple design analyzes are performed. In the first part, this publication derives a framework for an alternative finite element assembly of mass and stiffness matrices in the context of linear elasticity. Basically, disassembly consists of representing finite element matrices as a matrix product where topology contributions are isolated from constitutive law or inertia law contributions. Application examples are discussed to illustrate the advantages and limitations of this formulation using various meshes typically encountered in the automotive and aerospace industries. The second area of application discussed in the second part of this publication is the correlation between finite element models and test data. It is shown that numerical models can be updated for improving their correlation with measured frequency response functions with minimum computational cost when the model is disassembled.

RÉSUMÉ. La méthode des éléments finis est l'une des plus populaires en calcul des structures. Deux approches ont été développées. La formulation en force permet l'assemblage des matrices élémentaires en matrices globales grâce aux conditions d'équilibre des forces et moments. La seconde formulation est basée sur la notion de matrices de flexibilité. Nous proposons une représentation alternative. La représentation conventionnelle en force où les matrices élémen-

Revue européenne des éléments finis. Volume 9 – n° 5/2000, pages 511 à 560

taires sont décomposées en modes de déformation. Cette décomposition est appelée désassemblage de modèle et elle permet l'implémentation de solveurs numériques efficaces. Nous démontrons que la factorisation d'une seule matrice est nécessaire afin d'analyser les modèles issus d'une même topologie et métrique. Cette propriété rend les solveurs basés sur le désassemblage idéaux pour tous les problèmes qui nécessitent des analyses multiples comme l'optimisation ou le recalage de modèles. Tout d'abord, un cadre général est proposé pour le désassemblage de modèles arbitraires en élasticité linéaire. Le désassemblage consiste à représenter chaque matrice éléments finis comme un produit de matrices où les contributions de la topologie du maillage sont isolées de celles des lois de comportement. Plusieurs applications sont présentées afin d'illustrer les avantages et inconvénients de cette représentation pour des maillages typiques des industries automobile et aérospatiale. Puis, la formulation de solveurs inverses est abordée dans le contexte de la corrélation calculs-essais. Nous montrons que l'utilisation de modèles désassemblés permet de recaler à moindre coût les modèles numériques pour améliorer leur corrélation avec des fonctions de réponse en fréquences mesurées.

KEYWORDS: disassembly, finite element method, numerical method, fast reanalysis.

MOTS-CLÉS : désassemblage, méthode des éléments finis, solveur numérique, ré-analyse rapide.

Nomenclature

Vectors, Matrices and Tensors

$\mathcal{A}$	Bilinear form for total energy
$\mathbf{B}$	Matrix of shape function derivatives
$\mathbf{b}$	Linear form for energy of applied forces
$\mathcal{D}$	Generic differential operator (in space)
$\mathcal{F}$	Static or dynamic flexibility matrix
$\mathbf{F}$	Applied force vector
$\mathbf{G}_\epsilon$	Condensation matrix
$\mathcal{H}$	Generic Hilbert space
$\mathbf{H}$	Hooke tensor (constitutive law of elasticity)
h	Thickness of a plate or one of its layers
$\mathbf{I}$	Identity matrix
$\mathbf{I}_m$	Tensor of inertia
$\mathbf{J}$	Jacobian of a coordinate transform
$\mathbf{K}$	Stiffness matrix
$\mathbf{L}$	Generic localization matrix
$\mathbf{L}_I$	Interface localization matrix
$\mathcal{M}$	Vector of resultant moments per unit length

$\mathbf{M}$	Mass matrix
$\mathbf{N}$	Matrix of shape functions
$\mathbf{N}_f$	Vector of resultant forces per unit length
$\mathbf{n}$	Outward pointing unit normal vector
$\mathbf{P}$	Matrix of generic disassembly vectors
$\mathbf{p}$	Design parameters of the model
$\mathbf{Q}$	Matrix of generic disassembly vectors
$\mathbf{Q}_c, \overline{\mathbf{Q}_c}$	Stress–strain matrices of the layer of a laminate
$\mathbf{s}$	Frequency
$\mathbf{T}$	Rotation matrix of a coordinate transform
t	Time
$\mathbf{t}_o$	Prescribed traction vector
$\mathbf{u}$	Continuous generalized displacements
$\mathbf{u}_o$	Prescribed displacement vector
$\mathbf{W}$	Diagonal matrix of generic disassembly values
$\mathbf{w}$	Finite element test function
$\mathbf{w}_{ijk}$	Weight for numerical integration
$\mathbf{X}$	Triad of coordinates $(\mathbf{x}; \mathbf{y}; \mathbf{z})$
$\mathbf{x}$	Vector of discrete generalized displacements

Dimensions

N_e	Number of finite elements in the mesh
N_g	Number of Gauss integration points
N_k	Number of disassembly values for the stiffness
N_m	Number of disassembly values for the mass
N_s	Number of subdomains

Greek Symbols

ϵ	Vector of strains
κ	Vector of curvatures
λ	Vector of Lagrange multipliers (interface tractions)
Ω	Generic volume
$\partial\Omega_u$	Boundary of prescribed displacements
$\partial\Omega_t$	Boundary of prescribed tractions
σ	Vector of stresses
θ	Orientation of fibers in a laminate

Subscripts and Upperscripts

$(\bullet)_k$	Quantity pertaining to the stiffness matrix
$(\bullet)_m$	Quantity pertaining to the mass matrix
$(\bullet)_\epsilon$	Strain–mode quantity
$(\bullet)^{(e)}$	Element–level quantity
$(\bullet)^{(k)}$	Quantity pertaining to the kth layer of a laminate
$(\bullet)^+$	Pseudo–inverse quantity
$(\bullet)^{(s)}$	Quantity pertaining to a subdomain
$(\bullet)^T$	Transposed of a matrix or vector

1. Introduction

Design engineering in structural dynamics involves the resolution of linear systems that provide displacement and rotation solutions given loading conditions applied to the system. Typical applications include structural analysis ([PAZ 91]), topology or structural optimization ([SUZ 91], [HAF 93]), and finite element model updating where a distance between measured and computed responses is minimized ([HEM 95]). Such applications are quite different in nature and objectives. However, they all share the necessity of solving inverse systems repeatedly because the optimization associated with these problems is nonlinear.

Classical finite element modeling is based on the concept of assembly. It consists of partitioning the elastic domain into elemental volumes with geometries simple enough that the stress–strain relationship can be derived. Then, equilibrium of forces and moments at each nodal joint of the mesh are exploited to construct the master mass and stiffness matrices by adding their element–level counterparts ([HUG 87]). These matrices represent both the mathematical idealization and the spatial discretization of kinetic and strain energies, respectively.

Hence, the procedure for solving structural dynamics equations consists of, first, assembling master matrices for a mesh with given geometrical and material properties, then, solving the resulting linear system of equations. The solution generally provides displacements and rotations at nodal joints of the mesh and it is associated to a design criterion such as minimizing displacements in given areas of the model, or making sure that stresses do not exceed prescribed levels. In case of finite element updating, the numerical solution is compared with test data to assess the predictive quality of the model. Then, this constraint is enforced by bringing changes to the design and the whole procedure is repeated until convergence. Generally, large numbers of degrees of freedom are involved which results into multiple assemblies and factorizations of large–size, sparse matrices. This motivates the alternative representation of finite element assembly presented in the first part of this work.

The scope of this paper is restricted to linear, undamped elasticity for simplicity. Extension to damped dynamics does not seem to offer any particular difficulty other

than implementation issues which involve complex arithmetic because complex vectors must then be handled. The linear systems are described by equations [1] and [2] below. Equation [1] would typically be solved for the generalized displacements $\{\mathbf{x}\}$ when external static loadings $\{\mathbf{F}\}$ are applied to the structure

$$[\mathbf{K}(\mathbf{p})]\{\mathbf{x}\} = \{\mathbf{F}\} \qquad [1]$$

Equation [2] is the counterpart of static equation [1] when dynamic systems are analyzed in the frequency domain

$$([\mathbf{K}(\mathbf{p})] - \mathbf{s}^2[\mathbf{M}(\mathbf{p})])\{\mathbf{x}(\mathbf{s})\} = \{\mathbf{F}(\mathbf{s})\} \qquad [2]$$

This last equation models the equilibrium between internal, inertia and applied forces. It can be seen that it describes a variety of situations including acoustic problems, eigen–decompositions (when $\{\mathbf{F}\} = 0$) and static problems (when $\mathbf{s} = 0$). Note that, in the case of time–domain resolution, the inversion of a similar matrix is involved, typically with $\mathbf{s} = \frac{2}{\Delta \mathbf{t}}$ where $\Delta \mathbf{t}$ denotes the time step selected for numerical integration. The mass and stiffness matrices depend on design parameters $\{\mathbf{p}\}$ that represent geometrical and material properties for each finite element. These variables are precisely the unknown optimization variables considered in inverse problems such as design optimization or test–analysis reconciliation.

With the stiffness–based method, master mass and stiffness matrices are assembled from contributions of each finite element. In structural dynamics, systems are analyzed by discretizing them into meshes, adopting a mathematical idealization for representing the behavior of each individual finite element, then estimating the global mass and stiffness operators by assembling contributions from each element. The operation of assembly can be represented with the stiffness matrix, for example, as

$$[\mathbf{K}(\mathbf{p})] = \sum_{e=1\cdots N_e} \left[\mathbf{L}^{(e)}\right]^T \left[\mathbf{k}^{(e)}(\mathbf{p}^{(e)})\right] \left[\mathbf{L}^{(e)}\right] \qquad [3]$$

where $\left[\mathbf{k}^{(e)}\right]$ denotes the eth element–level stiffness matrix expressed in the coordinate system local to the element and $\left[\mathbf{L}^{(e)}\right]$ is the corresponding transformation and localization matrix used for adding together equations in the master stiffness $[\mathbf{K}]$. This classical and most widely used formulation of finite element assembly is based on the stiffness method which consists of writing the equilibrium at each nodal joint of a mesh, then exploiting the equality of displacements contributed by all finite elements that share a common node to derive equation [3] ([HUG 87]).

This formulation is very popular because most efficient for assembling finite element matrices. It is however quite inefficient when the final objective is multiple resolutions with changing designs because the master matrices must be re–assembled, at least partly, when the control parameters $\{\mathbf{p}\}$ are modified. These control or design parameters are, typically, constituted of geometrical parameters (thicknesses, cross–sectional areas, etc.) and material properties (moduli of elasticity, shear moduli, densities, etc.). The main motivation for this work is two–fold. First, we revisit the finite

element assembly and provide a very efficient numerical implementation when multiple assemblies are required due to design changes or optimization of parameters $\{\mathbf{p}\}$. Secondly, we develop efficient numerical solvers for multiple design analyses. Basically, the approach proposed here consists of representing the master stiffness matrix as a product of three matrices

$$[\mathbf{K}(\mathbf{p})] = [\mathbf{Q}_k]^T [\mathbf{W}_k(\mathbf{p})] [\mathbf{Q}_k] \quad [4]$$

where the central matrix is diagonal and is the only one that depends on structural parameters $\{\mathbf{p}\}$. Such a dependency must be known algebraically for the efficiency of the procedure and we demonstrate how this property can be achieved for practical applications. The other matrix $[\mathbf{Q}_k]$ exhibits, in the general case, a high degree of sparsity and does not depend on structural parameters. Hence, re–assembling the stiffness matrix based on equation [4] only requires the evaluation of the matrix product aforementioned.

The paper is organized as follows. The next section addresses the motivation of developing efficient numerical solvers for multiple design analyses. Section 3 discusses some of the applications of this work by introducing the resolution of structural dynamics equations using domain decomposition and parallel computing. Finite element theory and matrix assembly are summarized in section 4 to illustrate how disassembly can be achieved in the classical context of the stiffness method. The alternative representation of constitutive law and inertia law leading to finite element disassembly is introduced in section 5. Then, the theory of finite element disassembly is presented in section 6. Two simple examples are derived analytically in section 7 to illustrate how the method would typically be implemented and to present some of its difficulties. Section 8 details the main steps of the numerical solver associated to finite element disassembly in the case of a stiffness–based system. Applications are presented with various finite element meshes to illustrate the numerical performance of the method and to discuss its domain of applicability. Section 10 illustrates the application of finite element disassembly in the case where the solution to a flexibility–based system is sought after. Throughout sections 9 and 10, realistic examples are provided. These involve finite element models typically analyzed in the automotive and aerospace industries. All performance figures reported are obtained using a Matlab 5.2–based library of finite element tools implemented on a Silicon Graphics workstation (equipped with a R10,000-180 processor).

2. Stiffness Versus Flexibility FE Methods

One of the motivations for this work is the development of efficient numerical solvers for structural dynamics applications such as design optimization or finite element model updating. Consider, for example, the following equation of equilibrium in the frequency domain

$$\left([\mathbf{K}(\mathbf{p})] - \mathbf{s}^2 [\mathbf{M}(\mathbf{p})]\right) \{\mathbf{x}(\mathbf{s})\} = \{\mathbf{F}(\mathbf{s})\} \quad [5]$$

The solution procedure consists of solving equation [5] for a known loading case $\{\mathbf{F}\}$ and prescribed boundary conditions. Whether the problem of interest is structural design and analysis, structural optimization or finite element model updating, multiple resolutions of this equation must be obtained. Note that evoking equation [5] does not restrict our discussion since static problems are obtained when $\mathbf{s} = 0$ while time–domain resolutions involve the inversion of a similar matrix where $\mathbf{s} = (\frac{2}{\Delta t})$ as mentioned previously.

Such solutions are given (in the frequency domain) by the following product between the flexibility matrix and the right–hand side vector

$$\{\mathbf{x}(\mathbf{s})\} = [\mathcal{F}(\mathbf{p};\mathbf{s})]\{\mathbf{F}(\mathbf{s})\} \qquad [6]$$

Obviously, the flexibility matrix (or admittance matrix) defined by equation [6] is equal to the inverse of the dynamic stiffness matrix at the frequency s of interest

$$[\mathcal{F}(\mathbf{p};\mathbf{s})] = \left([\mathbf{K}(\mathbf{p})] - \mathbf{s}^2[\mathbf{M}(\mathbf{p})]\right)^{-1} \qquad [7]$$

It can be seen that this solution procedure is quite inefficient in the context of multiple analyses because it accumulates the disadvantage of multiple assemblies (which could become expensive with large dimensional models) to the need for factoring a full–size matrix each time. The alternative is the flexibility–based finite element method for assembling the flexibility matrix $[\mathcal{F}]$ directly, therefore, bypassing the need for a numerical solver ([ARG 60], [FRA 65]). However, flexibility methods have not grown in popularity basically because they do not provide efficient element–level assembly as the stiffness method does.

This issue has been indirectly addressed by Gordis who motivates his work on finite element disassembly as a way, for example, of relating global error estimations to local finite element matrices. It provides a unique identification of which finite elements contribute to an error indicator, which is extremely useful in problems such as mesh refinement, model updating or damage detection [GOR 96]. Gordis' definition of disassembly is however more general than the one defined by equation [4] and used here. His disassembly consists of rendering the master matrix block–diagonal

$$[\mathbf{K}(\mathbf{p})] = [\mathbf{L}]^T \begin{bmatrix} \left[\mathbf{k}_\epsilon^{(1)}(\mathbf{p}^{(1)})\right] & & & 0 \\ & \ddots & & \\ & & \left[\mathbf{k}_\epsilon^{(e)}(\mathbf{p}^{(e)})\right] & \\ & & & \ddots \\ 0 & & & \left[\mathbf{k}_\epsilon^{(N_e)}(\mathbf{p}^{(N_e)})\right] \end{bmatrix} [\mathbf{L}] \qquad [8]$$

where each diagonal block represents a "strain–mode" element–level matrix, that is, a reduced matrix where contributions from rigid body motions have been filtered out. Thus, reduced matrices preserve only the non–rigid dynamics of each finite element,

which is a characteristics "our" disassembled matrices share too. Matrix [L] is obtained as the transformation between global (coupled) generalized coordinates $\{\mathbf{x}\}$ and local (element–level, uncoupled) coordinates $\left\{\mathbf{x}^{(e)}\right\}$

$$\left\{\mathbf{x}^{(e)}\right\} = [\mathbf{L}]\{\mathbf{x}\} \qquad [9]$$

A fundamental difference is that Gordis' work takes the form of a unique decomposition whereas we will show that equation [4] is non–unique. As a result, for the disassembly [8-9] to provide any practical use, the model must represent a structure where the load path between any two points is unique, which is extremely restrictive. Our approach overcomes this major difficulty because, basically, the decomposition is not required to be uniquely determined. Nevertheless, we will discuss the cost to pay: a significant numerical implementation effort and the necessity to filter out the extraneous information.

Ideas for mesh disassembly have also been proposed by Peterson, Doebling and Alvin for structural damage detection using vibration test data ([PET 95]). Their basic idea is the starting point of this work: partitioning element–level matrices according to the contribution of their rigid body and strain modes, then representing global matrices as the superposition of each partition. Relating entries of measured flexibility matrices to element–level stiffnesses of a finite element model enables efficient localization of structural damage. However, results presented in reference [PET 95] involve the disassembly of beam elements only and lack a general framework for generalizing the procedure to arbitrary elements, which is precisely the contribution of the present work.

3. Domain Decomposition and Parallel Computing

Finite element disassembly can also be used to improve techniques for solving the equations of structural mechanics in parallel using a decomposition of the model into subdomains. In this section, we briefly emphasize this issue with two techniques that are similar in formulation but aimed at solving two different problems.

One of the most popular technique for solving structural mechanics problems in parallel is the Finite Element Tearing and Interconnecting (FETI) technique developed by Farhat and Roux ([FAR 91]). Basically, the equation of equilibrium [5] is partitioned according to different subdomains, yielding

$$\left(\left[\mathbf{K}^{(s)}(\mathbf{p})\right] - \mathbf{s}_j^2\left[\mathbf{M}^{(s)}(\mathbf{p})\right]\right)\left\{\mathbf{u}^{(s)}(\mathbf{s})\right\} = \left\{\mathbf{F}^{(s)}(\mathbf{s})\right\} + \left[\mathbf{L}_I^{(s)}\right]\{\lambda\} \qquad [10]$$

for each subdomain ($s = 1 \cdots N_s$). The last term of equation [10] accounts for boundary tractions on the interface between subdomain #s and all the neighboring subdomains. These tractions are required for satisfying the equilibrium of all the subdomains. Equations [10] are solved in parallel on separate processors once the tractions $\{\lambda\}$ have been made available, thus, providing reduced running times compared to a

single–processor architecture. Of course, the compatibility of displacements on the interface must be added

$$\sum_{s=1}^{N_s} \left[\mathbf{L}_I^{(s)}\right]^T \left\{\mathbf{u}^{(s)}(\mathbf{s})\right\} = 0 \qquad [11]$$

where $\left[\mathbf{L}_I^{(s)}\right]$ denotes the transformation that extracts coordinates on the interface for the sth subdomain. Equation [11] basically states that displacements contributed from separate subdomains that share the same interface nodes must be equal. Equations [10] and [11] are combined for obtaining the interface problem which provides a solution for the unknown boundary tractions $\{\lambda\}$

$$\left(\sum_{s=1}^{N_s} \left[\mathbf{L}_I^{(s)}\right]^T \left[\mathcal{F}^{(s)}(\mathbf{p};\mathbf{s})\right] \left[\mathbf{L}_I^{(s)}\right]\right) \{\lambda\} = \left(\sum_{s=1}^{N_s} \left[\mathbf{L}_I^{(s)}\right]^T \left[\mathcal{F}^{(s)}(\mathbf{p};\mathbf{s})\right]\right) \left\{\mathbf{F}^{(s)}(\mathbf{s})\right\} \qquad [12]$$

The FETI approach is based on a dual variational formulation of the partitioned equations of equilibrium that has proven its numerical and computational efficiency for solving a large variety of structural dynamics problems in the time and frequency domains. One illustration is the development at Sandia National Laboratories of SALINAS, a fully object–oriented and parallel finite element program for linear systems constituted of several million degrees of freedom. Another illustration is the development by Farhat and Géradin of a component mode synthesis method based on the same framework ([FAR 92]).

Obviously, equations [10] and [12] feature factorizations of each subdomain dynamic stiffness matrix and the iterative resolution of this system requires multiple inverse resolutions. This justifies our claim that finite element disassembly might be a valuable tool for bypassing these computationally expensive steps.

Building on the FETI approach, Park has exploited an idea similar to disassembly for formulating an algebraic partitioning of the equations of dynamics ([PAG 97], [PAR 97]). Although the overall procedure may not be as efficient as FETI for arbitrary mechanics problems and currently suffers from implementation difficulties, it clearly offers the advantage of enabling the identification of local mass and stiffness properties using the same decomposition as the one illustrated in equation [8].

The main step goes as follows: the transformation from global (coupled) generalized coordinates to local (element–level, uncoupled) coordinates can be applied to the force vector as

$$\{\mathbf{F}\} = [\mathbf{L}]^T \left\{\mathbf{F}^{(e)}\right\} \qquad [13]$$

Combining equations [5], [9] and (13] at zero–frequency leads to the transformation between the full–order static flexibility matrix $[\mathcal{F}]$ and decoupled, element–level flexibilities

$$\begin{bmatrix} [\mathcal{F}^{(1)}(\mathbf{p}^{(1)};0)] & & 0 \\ & \ddots & \\ & & [\mathcal{F}^{(N_e)}(\mathbf{p}^{(N_e)};0)] \end{bmatrix} = [\mathbf{L}]\,[\mathcal{F}(\mathbf{p};0)]\,[\mathbf{L}]^T \qquad [14]$$

Then, minimizing a strain–based version of the system's total energy with the compatibility of displacements and equilibrium of forces at the interface between connected elements provides a Ricatti–like equation. This equation relates local stiffness properties to the global flexibility matrix

$$[\mathcal{F}_\epsilon(\mathbf{p})] - [\mathcal{F}_\epsilon(\mathbf{p})]\,[\mathbf{G}_\epsilon(\mathbf{p})]\,[\mathcal{F}_\epsilon(\mathbf{p})] = ([\mathbf{L}]\,[\mathbf{Q}_k])^T\,[\mathcal{F}(\mathbf{p};0)]\,([\mathbf{L}]\,[\mathbf{Q}_k]) \qquad [15]$$

where matrix $[\mathbf{G}_\epsilon]$ is defined as

$$[\mathbf{G}_\epsilon(\mathbf{p})] = [\mathbf{L}_I]\left([\mathbf{L}_I]^T\,[\mathcal{F}_\epsilon(\mathbf{p})]\,[\mathbf{L}_I]\right)^{-1}[\mathbf{L}_I]^T \qquad [16]$$

In equations [15-16], a block–diagonal partitioning similar to equation [4] is used for the stiffness matrix from which it can be shown that the flexibility matrix is decomposed according to

$$[\mathcal{F}_\epsilon(\mathbf{p})] = [\mathbf{Q}_k]\begin{bmatrix} [\mathcal{F}^{(1)}(\mathbf{p}^{(1)};0)] & & 0 \\ & \ddots & \\ & & [\mathcal{F}^{(N_e)}(\mathbf{p}^{(N_e)};0)] \end{bmatrix}[\mathbf{Q}_k]^T \qquad [17]$$

where, by definition, we set $[\mathcal{F}_\epsilon] = [\mathbf{W}_k]^{-1}$. Note that this approach can theoretically be applied to the decomposition of the mesh into subdomains larger than a single finite element. It should also be emphasized that domain decomposition usually generates free–floating subdomains for which master matrices are singular. In this case, inverses are replaced with pseudo–inverses, for example in equation [16].

The conclusion is that, solving equation [15] for the diagonal matrix $[\mathcal{F}_\epsilon]$ provides an estimation of local inverse stiffness parameters. Hence, measured frequency response functions of a system may replace the flexibility matrix $[\mathcal{F}]$ in equation [15] and the solution procedure may be applied for identifying local stiffness properties given a description of the topology of the structure. Hence, Park's idea of adding an additional constraint of algebraic decomposition (what is referred to as "disassembly" here) provides an elegant framework for solving inverse problems such as health monitoring and damage detection where the main challenge is precisely to relate global error indicators to local mass and stiffness changes over time.

Each of these two examples demonstrates where finite element disassembly may be a useful tool for enabling, or at least enhancing, the resolution of direct and inverse structural dynamics problems. The development of efficient inverse solvers is however complicated by the non–unicity of disassembly and caution must be observed for filtering out extraneous information and obtaining meaningful solutions.

4. Finite Element Discretization and Assembly

After having motivated the need for alternative finite element assembly procedures, a theoretical framework is derived in this section. We start with a brief summary of the classical, stiffness–based finite element procedure: mathematical idealization, discretization and weak formulation are addressed. Our purpose is to remind where design parameters $\{\mathbf{p}\}$ are involved during the matrix assembly because this is critical to the understanding of the disassembly technique. reference [GER 97] provides the complete details for those readers who may not be familiar with the notations indroduced here.

From now on, we will focus on linear elasticity problems that constitute the core of structural mechanics applications. Furthermore, Einstein's summation convention of repeated indices is used for clarity in most equations ($\mathbf{A}_{ij}\mathbf{u}_j = \sum_{j=1\cdots N} \mathbf{A}_{ij}\mathbf{u}_j$). In the general case, the following boundary value problem is considered

$$\begin{cases} \mathbf{I}_{m_{ij}} \frac{\partial^2 \mathbf{u}_j}{\partial t^2}(t) - \mathcal{D}_k\left(\mathbf{H}_{ijkl}\mathcal{D}_l\left(\mathbf{u}_j(t)\right)\right) = \mathbf{F}_i(t) & \text{in domain}\,\Omega \\ \mathbf{u}_i(t) = \mathbf{u}_{0i}(t) & \text{on}\,\partial\Omega_u \\ \sigma_{ij}(\mathbf{u}) = \mathbf{t}_{0i}(t) & \text{on}\,\partial\Omega_t \end{cases} \qquad [18]$$

where the boundary $\partial\Omega$ of elastic domain Ω is partitioned in two disjoint parts $\partial\Omega_u$ and $\partial\Omega_t$ where the displacements and tractions are imposed, respectively. In addition, equation [18] must be completed with the adequate initial conditions (at $t = 0$) that involve the generalized displacements $\mathbf{u}$ or the velocities or accelerations. Equation [18] represents a mathematical idealization of the real behavior of the physical system where inertia and internal forces in the left–hand side balance the applied forces.

Prior to discretization, problem [18] is cast in a variational form given in equations [19-21]. This second form of the same problem is commonly referred to as the "weak" form because its solution $\mathbf{u}$ satisfies equation [18] in average. The weak formulation of equation [18] is given by

$$\mathcal{A}(\mathbf{u};\mathbf{w}) = \mathbf{b}(\mathbf{w}), \forall \mathbf{w} \in \mathcal{H} \qquad [19]$$

where

$$\mathcal{A}(\mathbf{u};\mathbf{w}) = \int_\Omega \left(\mathbf{w}_i \mathbf{I}_{m_{ij}} \frac{\partial^2 \mathbf{u}_j}{\partial \mathbf{t}^2} + \mathbf{H}_{ijkl}\mathcal{D}_l\left(\mathbf{u}_j\right)\mathcal{D}_k\left(\mathbf{w}_i\right)\right) d\Omega \qquad [20]$$

$$\mathbf{b}(\mathbf{w}) = \int_\Omega \mathbf{w}_i \mathbf{F}_i d\Omega + \int_{\partial\Omega_t} \mathbf{w}_i \mathbf{t}_{o_i} d\sigma \qquad [21]$$

It is assumed, as usual, that $\mathbf{w} = 0$ on the boundary $\partial\Omega_u$. In equation [19], $\mathcal{H}$ denotes the adequate Hilbert space. In Mechanical Engineering, Sobolev spaces such as $\mathbf{W}^{1,2}(\Omega)$ and $\mathbf{W}^{2,2}(\Omega)$ are usual choices, depending on the order of partial differential equations analyzed. Applied mathematics tell us that problems [18] and [19-21] are equivalent under fairly general assumptions relevant to the class of problems treated here and which essentially consist of satisfying the continuity and coercivity of the bilinear, symmetric form $\mathcal{A}$ over $\mathcal{H}$ [ODE 76].

For obtaining a computational procedure from the weak formulation, discretization is introduced. The element–level stiffness matrix represents the second–order derivative of the element's strain energy with respect to nodal displacements, that is

$$\left[\mathbf{k}^{(e)}(\mathbf{p}^{(e)})\right] = \frac{\partial^2}{\partial \mathbf{x}^{(e)2}} \int_{\Omega^{(e)}} \left\{\sigma^{(e)}\right\}^T \left\{\epsilon^{(e)}\right\} d\Omega \qquad [22]$$

In the context of linear, homogeneous and isotropic elasticity, this element stiffness matrix can generally be obtained when the element's strain field is expressed in terms of nodal displacements. Without loss of generality, this can be represented as

$$\left[\mathbf{k}^{(e)}(\mathbf{p}^{(e)})\right] = \int_{\Omega^{(e)}} [\mathbf{B}(\mathbf{X})]^T \left[\mathbf{H}(\mathbf{p}^{(e)})\right] [\mathbf{B}(\mathbf{X})]\, d\Omega \qquad [23]$$

where $\mathbf{X} = (\mathbf{x};\mathbf{y};\mathbf{z})$ are spatial variables. In equation [23], matrix $[\mathbf{B}] = \left[\frac{\partial \mathbf{N}}{\partial \mathbf{X}}\right]$ represents the strain–displacement matrix that features derivatives of the element's shape functions

$$\left\{\epsilon^{(e)}\right\} = [\mathbf{B}(\mathbf{X})] \left\{\mathbf{x}^{(e)}\right\} \qquad [24]$$

while matrix $[\mathbf{H}]$ denotes the constitutive matrix that relates the strain and stress fields to each other

$$\left\{\sigma^{(e)}\right\} = \left[\mathbf{H}(\mathbf{p}^{(e)})\right] \left\{\epsilon^{(e)}\right\} \qquad [25]$$

The same formalism applies to the element's mass matrix. Integration can be performed over individual subdomain $\Omega^{(e)}$ (or finite elements), then accumulated in the master quantities. This is the essence of the assembly procedure that is written as

$$[\mathbf{M}(\mathbf{p})] = \sum_{e=1\cdots N_e} \left[\mathbf{L}^{(e)}\right]^T \left[\mathbf{m}^{(e)}(\mathbf{p}^{(e)})\right] \left[\mathbf{L}^{(e)}\right] \qquad [26]$$

$$[\mathbf{K}(\mathbf{p})] = \sum_{e=1\cdots N_e} \left[\mathbf{L}^{(e)}\right]^T \left[\mathbf{k}^{(e)}(\mathbf{p}^{(e)})\right] \left[\mathbf{L}^{(e)}\right] \qquad [27]$$

$$\{\mathbf{F}(\mathbf{t})\} = \sum_{e=1\cdots N_e} \left[\mathbf{L}^{(e)}\right]^T \left\{\mathbf{f}^{(e)}(\mathbf{t})\right\} \qquad [28]$$

where matrix $\left[\mathbf{L}^{(e)}\right]$ denotes the localization of master degrees of freedom to the eth finite element. Practically, these quantities are calculated by numerical integration,

except in the case of simple geometries where analytical solutions are available. For example, element–level stiffness matrices are obtained as

$$\left[\mathbf{k}^{(e)}(\mathbf{p}^{(e)})\right] = \sum_{i,j,k=1\cdots N_g} [\mathbf{B}(i;j;k)]^T \left[\mathbf{H}(\mathbf{p}^{(e)})\right] [\mathbf{B}(i;j;k)]\, \mathbf{w}_{ijk} \det \mathbf{J}(i;j;k) \quad [29]$$

where N_g, the number of Gauss integration points, depends on the degree of the polynomial function being integrated.

Finally, the well–known linear system of equations is obtained by substituting shapes functions (Ritz–Galerkin formulation) into the weak formulation [19-21]

$$[\mathbf{M}(\mathbf{p})] \left\{ \frac{\partial^2 \mathbf{x}}{\partial \mathbf{t}^2}(t) \right\} + [\mathbf{K}(\mathbf{p})] \{\mathbf{x}(t)\} = \{\mathbf{F}(t)\} \quad [30]$$

Note that a transformation in the frequency domain (using the Fourier transform, for example) yields the previous equation [5]. It can be seen that quantities depending on the order of the interpolation performed, that is, the triplet $(i;j;k)$ in equation [29], and quantities depending on the element's design parameters $\{\mathbf{p}^{(e)}\}$ are decoupled. This property is exploited in the next section for a practical formulation of finite element disassembly.

For completeness, the following remarks can be made about these developments:

- Although a Ritz–Galerkin formulation is used for simplicity, nothing prevents us from choosing the test functions $\mathbf{w}$ differently from the shape functions $\{\mathbf{N}_j\}$. Hence, more general approximation methods can be implemented within this framework ([ODE 83]).
- Similarly, mixed or hybrid variational principles are not excluded from this formulation [FEL 89]. Actually, the critical, enabling condition for finite element disassembly is that only inertia and constitutive matrices depend on design parameters $\{\mathbf{p}\}$, which is shared by all approaches.

5. Alternative Representation of Linear Elasticity

Section 4 has established the natural decoupling between topology–dependent and design parameter–dependent quantities in the elemental mass and stiffness matrices. Our goal is now to show that inertia and constitutive laws can be decomposed as

$$\left[\mathbf{I}_m(\mathbf{p}^{(e)})\right] = [\mathbf{P}_m]^T \left[\mathbf{W}_m(\mathbf{p}^{e})\right] [\mathbf{P}_m] \quad [31]$$

$$\left[\mathbf{H}(\mathbf{p}^{(e)})\right] = [\mathbf{P}_k]^T \left[\mathbf{W}_k(\mathbf{p}^{(e)})\right] [\mathbf{P}_k] \quad [32]$$

where matrices $[\mathbf{W}_m]$ and $[\mathbf{W}_k]$ are diagonal and where matrices $[\mathbf{P}_m]$ and $[\mathbf{P}_k]$ do not depend on geometrical and material properties. Because the inertia law can be represented by a diagonal matrix $[\mathbf{I}_m]$, we will focus on the constitutive law of elasticity in this section.

In the following, 1D, 2D, then 3D elasticity is investigated. The main result is to show that, no matter what type of isotropic elasticity is considered, the decomposition [31-32] can always be written algebraically because it involves (at worse) three by three symmetric matrices that can be decomposed "by hand."

5.1. *Monodimensional and Isotropic Elasticity*

The most trivial case is to consider axial deformations only because the stress–strain relationship can be expressed as $\sigma = E\epsilon$, where E represents the material's modulus of elasticity (Young's modulus). Therefore, the decomposition [31] is simply $\mathbf{W}_k = E$ and $\mathbf{P}_k = 1$.

The second interesting case involving pseudo–1D elasticity is that of beam elements. Consider, for example, the 4th–order, 2–node Euler–Bernoulli beam with the degree of freedom ordering given in equation [33]

$$\left\{\mathbf{x}^{(e)}\right\} = \{\mathbf{U}_{x1}, \mathbf{U}_{y1}, \mathbf{U}_{z1}, \theta_{x1}, \theta_{y1}, \theta_{z1}, \mathbf{U}_{x2}, \mathbf{U}_{y2}, \mathbf{U}_{z2}, \theta_{x2}, \theta_{y2}, \theta_{z2}\}^T \qquad [33]$$

Design variables are $\left\{\mathbf{p}^{(e)}\right\} = \{E, A, \nu, J, I_y, I_z\}^T$ and it can be verified that, in this particular case, the disassembly [4] may be achieved by hand because numerical integration is not required. We can write

$$\left[\mathbf{k}^{(e)}(\mathbf{p}^{(e)})\right] = \left[\mathbf{Q}_k^{(e)}\right]^T \left[\mathbf{W}_k^{(e)}(\mathbf{p}^{(e)})\right] \left[\mathbf{Q}_k^{(e)}\right] \qquad [34]$$

where

$$\left[\mathbf{W}_k^{(e)}(\mathbf{p}^{(e)})\right] = \frac{1}{L}\begin{bmatrix} EA & 0 & 0 & 0 & 0 & 0 \\ 0 & GJ & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{EI_y}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{EI_y}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{EI_z}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{EI_z}{2} \end{bmatrix} \qquad [35]$$

with $G = \frac{E}{2(1+\nu)}$ and

$$\left[\mathbf{Q}_k^{(e)}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{2\sqrt{3}}{L} & \frac{2\sqrt{3}}{L} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{2\sqrt{3}}{L} & \frac{2\sqrt{3}}{L} \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1+\sqrt{3} & 1-\sqrt{3} \\ 0 & 0 & -(1+\sqrt{3}) & -(1-\sqrt{3}) & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{2\sqrt{3}}{L} & -\frac{2\sqrt{3}}{L} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{2\sqrt{3}}{L} & -\frac{2\sqrt{3}}{L} \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -(1-\sqrt{3}) & -(1+\sqrt{3}) \\ 0 & 0 & 1-\sqrt{3} & 1+\sqrt{3} & 0 & 0 \end{bmatrix}^T \quad [36]$$

This decomposition is equivalent to the one given in reference [PET 95]. The six–row vectors in matrix $\left[\mathbf{Q}_k^{(e)}\right]$ can be interpreted as the element's static deformation shapes associated with nonzero strain energy, or strain modes. Their number is always equal to the rank of the stiffness matrix. It can be verified that the main two assumptions of disassembly are satisfied: 1) Coefficients of the diagonal matrix $\left[\mathbf{W}_k^{(e)}\right]$ are known explicitely; and 2) Matrix $\left[\mathbf{Q}_k^{(e)}\right]$ does not depend on design parameters $\left\{\mathbf{p}^{(e)}\right\}$. Note also that a similar decomposition can be obtained with the Timoshenko beam where shear coupling is involved.

5.2. *Bidimensional and Isotropic Elasticity*

Bidimensional finite elements such as plates and shells involve membrane and bending deformations. For constant–thickness elements, these constitutive laws are generally equal to

$$\left[\mathbf{H}^{\text{membrane}}(\mathbf{p}^{(e)})\right] = \frac{Eh}{(1-\nu^2)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \quad [37]$$

$$\left[\mathbf{H}^{\text{bending}}(\mathbf{p}^{(e)})\right] = \frac{h^2}{12}\left[\mathbf{H}^{\text{membrane}}(\mathbf{p}^{(e)})\right] \quad [38]$$

where h is the element's thickness. It can be verified easily that the decomposition [31] for membrane deformations is obtained with

$$\left[\mathbf{W}_k^{(e)\text{membrane}}(\mathbf{p}^{(e)})\right] = Eh \begin{bmatrix} \frac{1}{2(1+\nu)} & 0 & 0 \\ 0 & \frac{1}{2(1-\nu)} & 0 \\ 0 & 0 & \frac{1}{2(1+\nu)} \end{bmatrix} \quad [39]$$

$$\left[\mathbf{P}_k^{(e)\text{membrane}}\right] = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad [40]$$

Similarly, the decomposition [31] for bending deformations is obtained with

$$\left[\mathbf{W}_k^{(e)\text{bending}}(\mathbf{p}^{(e)})\right] = \frac{h^2}{12}\left[\mathbf{W}_k^{(e)\text{membrane}}(\mathbf{p}^{(e)})\right] \quad [41]$$

and $\left[\mathbf{P}_k^{(e)\text{bending}}\right] = \left[\mathbf{P}_k^{(e)\text{membrane}}\right]$. It can be seen from equations [37-38] that, in the case of 2D elasticity, the design parameters of interest are restricted to $\{\mathbf{p}^{(e)}\} = \{E, \nu, h\}^T$.

5.3. *Tridimensional and Isotropic Elasticity*

Volume elements rely on the following constitutive matrix in the case of linear, homogeneous and isotropic elasticity

$$\left[\mathbf{H}(\mathbf{p}^{(e)})\right] = \frac{E}{1+\nu}\begin{bmatrix} \frac{1-\nu}{1-2\nu} & \frac{\nu}{1-2\nu} & \frac{\nu}{1-2\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-2\nu} & \frac{1-\nu}{1-2\nu} & \frac{\nu}{1-2\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-2\nu} & \frac{\nu}{1-2\nu} & \frac{1-\nu}{1-2\nu} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix} \quad [42]$$

It can be decomposed into

$$\left[\mathbf{W}_k^{(e)}(\mathbf{p}^{(e)})\right] = \begin{bmatrix} \frac{E}{1-2\nu} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{E}{1+\nu} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{E}{1+\nu} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{E}{2(1+\nu)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{E}{2(1+\nu)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{E}{2(1+\nu)} \end{bmatrix} \quad [43]$$

$$\left[\mathbf{P}_k^{(e)}\right] = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad [44]$$

with design parameters defined as $\{\mathbf{p}^{(e)}\} = \{E, \nu\}^T$.

We emphasize that the decomposition [31-32] is not an eigenvalue factorization that could be performed numerically. Doing so would not provide any computational

advantage over the classical assembly technique. The key point here is to obtain algebraically the expression of matrices $[\mathbf{W}]$ and $[\mathbf{P}]$ so that the decomposition [31-32] can be implemented in any finite element program.

6. Theory of Finite Element Disassembly

In this Section, the procedure for disassembling an arbitrary finite element model is presented, based on decomposition [31-32] of inertia and constitutive laws. Here, we are mostly concerned with decomposing the element–level matrices. Next, the partitioning is collected for each finite element to generate representations of master matrices similar to equation [4].

We assume that the inertia and constitutive laws have been partitioned into N_m and N_k vectors, respectively, according to the procedure detailed in section 5 for the homogeneous and isotropic elasticity. Equations [31-32] can also be written as an accumulation of rank–one matrices

$$\left[\mathbf{I}_m(\mathbf{p}^{(e)})\right] = \sum_{d=1}^{N_m} \mathbf{w}_{m_{dd}}(\mathbf{p}^{(e)}) \{\mathbf{P}_{m_d}\} \{\mathbf{P}_{m_d}\}^T \quad [45]$$

$$\left[\mathbf{H}(\mathbf{p}^{(e)})\right] = \sum_{d=1}^{N_k} \mathbf{w}_{k_{dd}}(\mathbf{p}^{(e)}) \{\mathbf{P}_{k_d}\} \{\mathbf{P}_{k_d}\}^T \quad [46]$$

These equations are simply substituted to matrices $[\mathbf{I}_m]$ and $[\mathbf{H}]$ in the definition of element–level mass and stiffness matrices, respectively. Accounting for numerical integration provides

$$\left[\mathbf{m}^{(e)}(\mathbf{p}^{(e)})\right] = \sum_{d=1}^{N_m} \mathbf{w}_{m_{dd}}(\mathbf{p}^{(e)}) \left(\sum_{i,j,k=1}^{N_g} \left\{\mathbf{Q}_{m_d}^{(e)}(i;j;k)\right\} \left\{\mathbf{Q}_{m_d}^{(e)}(i;j;k)\right\}^T \right) \quad [47]$$

$$\left[\mathbf{k}^{(e)}(\mathbf{p}^{(e)})\right] = \sum_{d=1}^{N_k} \mathbf{w}_{k_{dd}}(\mathbf{p}^{(e)}) \left(\sum_{i,j,k=1}^{N_g} \left\{\mathbf{Q}_{k_d}^{(e)}(i;j;k)\right\} \left\{\mathbf{Q}_{k_d}^{(e)}(i;j;k)\right\}^T \right) \quad [48]$$

where we have defined the following sets of vectors

$$\left\{\mathbf{Q}_{m_d}^{(e)}(i;j;k)\right\} = \sqrt{\mathbf{w}_{ijk} \det \mathbf{J}(i;j;k)} \left[\mathbf{N}(i;j;k)\right] \{\mathbf{P}_{m_d}\} \quad [49]$$

$$\left\{\mathbf{Q}_{k_d}^{(e)}(i;j;k)\right\} = \sqrt{\mathbf{w}_{ijk} \det \mathbf{J}(i;j;k)} \left[\mathbf{B}(i;j;k)\right] \{\mathbf{P}_{k_d}\} \quad [50]$$

Note that the square roots involved in equation [49-50] should not be a concern because integration weights $\mathbf{w}_{ijk}$ are generally positive (see, for example, [BAT 96]) and so are the determinants $\det \mathbf{J}$ of the Jacobian's coordinate transform if the element's outward normals are defined correctly. If these quantities are not positive, a negative sign can always be assigned to the corresponding entry in the diagonal matrix $\mathbf{W}$ without loss of generality.

Finally, the decomposition can be written compactly as

$$\left[\mathbf{m}^{(e)}(\mathbf{p}^{(e)})\right] = \left[\mathbf{Q}_m^{(e)}\right]^T \left[\mathbf{W}_m^{(e)}(\mathbf{p}^{(e)})\right] \left[\mathbf{Q}_m^{(e)}\right] \quad [51]$$

$$\left[\mathbf{k}^{(e)}(\mathbf{p}^{(e)})\right] = \left[\mathbf{Q}_k^{(e)}\right]^T \left[\mathbf{W}_k^{(e)}(\mathbf{p}^{(e)})\right] \left[\mathbf{Q}_k^{(e)}\right] \quad [52]$$

Consider the stiffness matrix: since $\left[\mathbf{W}_k^{(e)}\right]$ is a diagonal matrix, equations [48] and [52] emphasize the mechanism of finite element construction. It is a summation of $(N_k \times N_g)$ rank–one matrices. The same remark applies to the mass and it is easy to understand that similar representations hold for the master matrices because the assembly consists in adding together the element matrices. The disassembled master matrices are denoted by

$$[\mathbf{M}(\mathbf{p})] = [\mathbf{Q}_m]^T [\mathbf{W}_m(\mathbf{p})] [\mathbf{Q}_m] \quad [53]$$

$$[\mathbf{K}(\mathbf{p})] = [\mathbf{Q}_k]^T [\mathbf{W}_k(\mathbf{p})] [\mathbf{Q}_k] \quad [54]$$

where matrices $[\mathbf{W}_m]$ and $[\mathbf{W}_k]$ are diagonal and collect values $\mathbf{w}_{m_{dd}}$ and $\mathbf{w}_{k_{dd}}$, respectively, for each finite element. Similarly, vectors $\left\{\mathbf{Q}_{m_d}^{(e)}\right\}$ and $\left\{\mathbf{Q}_{k_d}^{(e)}\right\}$ are collected in columns of matrices $[\mathbf{Q}_m]$ and $[\mathbf{Q}_k]$, respectively, with the usual ordering of master degrees of freedom, that is

$$\left\{\mathbf{Q}_{m_j}\right\} = \left[\mathbf{L}^{(e)}\right]^T \left\{\mathbf{Q}_{m_d}^{(e)}\right\}, \quad \left\{\mathbf{Q}_{k_j}\right\} = \left[\mathbf{L}^{(e)}\right]^T \left\{\mathbf{Q}_{k_d}^{(e)}\right\} \quad [55]$$

This explains why these matrices exhibit, in general, a very high degree of sparsity. This procedure is illustrated in section 7 using a simple example for which all computations can be performed analytically.

We now have achieved the main result of this work: equations [51-54] show that element–level and master quantities can be decomposed in a summation of rank–one matrices, each multiplied by a scalar value known explicitely and that depends on the model's design parameters. In general, these rank–one matrices require numerical integration and they can not be derived explicitely. However, the key point is that they do not depend on design parameters and they stay constant once the topology and the metric of the mesh has been set. Re–analysis techniques or inverse solvers can take great advantage of this property because only the diagonal matrices $[\mathbf{W}_m]$ and $[\mathbf{W}_k]$ need to be updated.

7. Demonstration Examples

To illustrate how finite element models can be disassembled, two examples are now presented. They both involve simple, planar models because our purpose is to offer a complete overview of the procedure before demonstrating its application to re–analysis, structural optimization and inverse problem solving.

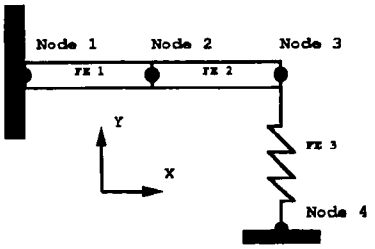

Figure 1. *Finite Element Model Used for Illustration*

7.1. *Example I: Disassembly of a Two-Beam Structure*

Figure 1 depicts the finite element model used for this first illustration. It consists of a uniform, cantilever planar beam connected to a spring. Two Euler–Bernoulli beam elements are used for the discretization. For simplicity, the beam's local axes are aligned with global axes $(\mathbf{x};\mathbf{y})$ and it can be verified that the two element stiffness matrices are equal to

$$\left[\mathbf{k}^{(1)}(\mathbf{p}^{(1)})\right] = \begin{bmatrix} \frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \qquad [56]$$

and

$$\left[\mathbf{k}^{(2)}(\mathbf{p}^{(2)})\right] = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \qquad [57]$$

In equations [56-57], only the active (unrestrained) degrees of freedom are kept. Similarly, the element stiffness matrix of the spring is equal to

$$\left[\mathbf{k}^{(3)}(\mathbf{p}^{(3)})\right] = [k] \qquad [58]$$

where k denotes the sping rigidity. Finally, the boolean table that provides the one–to–one equivalence between local and global degrees of freedom can be defined as

DOF	U_{x1}	U_{y1}	θ_{z1}	U_{x2}	U_{y2}	θ_{z2}	U_{x3}	U_{y3}	θ_{z3}	U_{x4}	U_{y4}	θ_{z4}
1	0	0	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	0	0	1	0	0	0

It can easily be verified that the resulting master stiffness matrix for this simple example is equal to

$$[\mathbf{K}(\mathbf{p})] = \begin{bmatrix} \frac{2EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{24EI}{L^3} & \frac{12EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{12EI}{L^2} & \frac{8EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \left(\frac{12EI}{L^3}+k\right) & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \quad [59]$$

where the unrestrained degrees of freedom consist of the two translations and in–plane rotations at nodes 2 and 3

$$\{\mathbf{x}\} = \{\mathbf{U}_{x2}, \mathbf{U}_{y2}, \theta_{z2}, \mathbf{U}_{x3}, \mathbf{U}_{y3}, \theta_{z3}\} \quad [60]$$

We now use equations [34-36] to generate the disassembly of beam elements. It provides the following decompositions

$$\left[\mathbf{k}^{(1)}(\mathbf{p}^{(1)})\right] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & \frac{2\sqrt{3}}{L} & -\frac{2\sqrt{3}}{L} \\ 0 & 1-\sqrt{3} & 1+\sqrt{3} \end{bmatrix} \begin{bmatrix} \frac{EA}{L} & 0 & 0 \\ 0 & \frac{EI}{2L} & 0 \\ 0 & 0 & \frac{EI}{2L} \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & \frac{2\sqrt{3}}{L} & -\frac{2\sqrt{3}}{L} \\ 0 & 1-\sqrt{3} & 1+\sqrt{3} \end{bmatrix}^T \quad [61]$$

and

$$\left[\mathbf{k}^{(2)}(\mathbf{p}^{(2)})\right] = \left[\mathbf{Q}^{(2)}\right]^T \begin{bmatrix} \frac{EA}{L} & 0 & 0 \\ 0 & \frac{EI}{2L} & 0 \\ 0 & 0 & \frac{EI}{2L} \end{bmatrix} \left[\mathbf{Q}^{(2)}\right]$$

$$\text{with} \quad \left[\mathbf{Q}^{(2)}\right]^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{2\sqrt{3}}{L} & \frac{2\sqrt{3}}{L} \\ 0 & -(1+\sqrt{3}) & -(1-\sqrt{3}) \\ -1 & 0 & 0 \\ 0 & \frac{2\sqrt{3}}{L} & -\frac{2\sqrt{3}}{L} \\ 0 & 1-\sqrt{3} & 1+\sqrt{3} \end{bmatrix} \quad [62]$$

The spring element is simply disassembled as

$$\left[\mathbf{k}^{(3)}(\mathbf{p}^{(3)})\right] = \{1\}(k)\{1\}^T \quad [63]$$

Partitioning the master stiffness matrix simply consists of collectingthe disassembly values and the corresponding vectors in the diagonal matrix $[\mathbf{W}_k]$ and columns of matrix $[\mathbf{Q}_k]$, respectively. Of course, degrees of freedom must be ordered in matrix $[\mathbf{Q}_k]$ the same way they are ordered in the master stiffness matrix [59]. It can be verified that the disassembled master stiffness matrix is given by

$$[\mathbf{W}_k(\mathbf{p})] = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{EI}{2L} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{EI}{2L} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{EI}{2L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{EI}{2L} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & k \end{bmatrix} \quad [64]$$

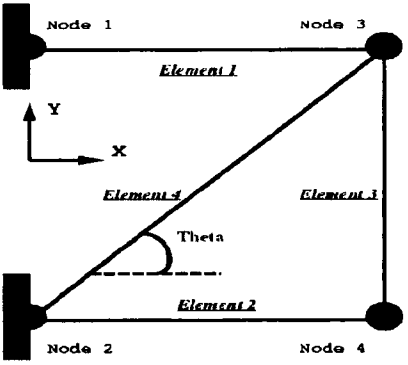

Figure 2. *Illustration Example of a Planar Beam Model*

and

$$[\mathbf{Q}_k] = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{2\sqrt{3}}{L} & -\frac{2\sqrt{3}}{L} & 0 & -\frac{2\sqrt{3}}{L} & \frac{2\sqrt{3}}{L} & 0 \\ 0 & 1-\sqrt{3} & 1+\sqrt{3} & 0 & -(1+\sqrt{3}) & -(1-\sqrt{3}) & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{2\sqrt{3}}{L} & -\frac{2\sqrt{3}}{L} & 0 \\ 0 & 0 & 0 & 0 & 1-\sqrt{3} & 1+\sqrt{3} & 1 \end{bmatrix}^T \quad [65]$$

A total of seven disassembly values are obtained for this particular example, that is, more than the dimension of the model. This is because finite elements are disassembled independently of one another, therefore, redundant information may be introduced such as illustrated here. This is not a concern for the assembly of master matrices. The second part addresses this issue when disassembly is used for algebraically inverting the master matrices.

7.2. *Example II: Disassembly of a Planar Frame Structure*

Before proceeding with the derivation of efficient numerical solvers, we illustrate the disassembly of the simple finite element model depicted in Figure 2. The structure consists of a planar frame where four beams are connected at four nodal joints. The mathematical idealization chosen for this structure is, once again, the Euler–Bernoulli beam model. Nodes 1 and 2 are clamped so that the total number of active degrees of freedom is equal to six.

7.2.1. *Assembly of the Planar Beam Frame Model*

The element stiffness matrix for the Euler–Bernoulli planar beam element is first expressed in each element's local frame system. It yields

$$\left[\mathbf{k}^{(e)}(\mathbf{p}^{(e)})\right] = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \quad [66]$$

where the element's design parameters are defined as $\{\mathbf{p}^{(e)}\} = \{E; A; I\}^T$ and represent the Young modulus, the cross–sectional area and the moment of inertia of the beam. The local–to–global coordinate transform consists of the following 6 by 6 matrix where θ denotes the in–plane angle of the element (measured positive from the horizontal to the element's axial direction)

$$\left[\mathbf{L}^{(e)}\right] = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & 0 & 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad [67]$$

The master stiffness matrix for this model is obtained by substituting definitions [66] and [67] in the assembly equation [27]. The ordering of active degrees of freedom is provided below

$$\{\mathbf{x}\} = \{\mathbf{x}_{x3}; \mathbf{x}_{y3}; \theta_{z3}; \mathbf{x}_{x4}; \mathbf{x}_{y4}; \theta_{z4}\}^T \quad [68]$$

7.2.2. *Disassembly of the Planar Beam Frame Model*

As before, both disassembly values and vectors can be obtained algebraically in the case of beam elements

$$\left[\mathbf{k}^{(e)}(\mathbf{p}^{(e)})\right] = \left[\mathbf{Q}^{(e)}\right]^T \begin{bmatrix} \frac{EA}{L} & 0 & 0 \\ 0 & \frac{EI}{2L} & 0 \\ 0 & 0 & \frac{EI}{2L} \end{bmatrix} \left[\mathbf{Q}^{(e)}\right]$$

$$\text{with} \quad \left[\mathbf{Q}^{(e)}\right]^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{2\sqrt{3}}{L} & \frac{2\sqrt{3}}{L} \\ 0 & -(1+\sqrt{3}) & -(1-\sqrt{3}) \\ -1 & 0 & 0 \\ 0 & \frac{2\sqrt{3}}{L} & -\frac{2\sqrt{3}}{L} \\ 0 & 1-\sqrt{3} & 1+\sqrt{3} \end{bmatrix} \quad [69]$$

Equation [69] shows that each planar beam contributes to three strain–deformation modes. The three vectors of matrix $\left[\mathbf{Q}_k^{(e)}\right]$ represent each one of the element's non–rigid deformation modes. It can be verified that the local matrix [66] is recovered exactly when the product [69] is performed. In addition, the element's three rigid

body modes have been filtered out from the disassembly because all three entries of diagonal matrix $\left[\mathbf{W}_k^{(e)}\right]$ are nonzero. Therefore, the total number of disassembly values and vectors is equal to $N_D = 3 \times 4 = 12$ (three values contributed by each one of the four beams). Then, combining the decomposition of each element stiffness matrix and the local–to–global coordinate transform [67] provides the disassembly of the master stiffness matrix. Each element–level disassembly vector is stored in a row of matrix $[\mathbf{Q}_k]$ below with the same degree of freedom ordering as the one described by equation [68]. The main diagonal of master matrix $[\mathbf{W}_k]$ is composed of twelve entries equal to

$$\left\{\left(\frac{EA}{L}\right)^{(1)}, \left(\frac{EI}{2L}\right)^{(1)}, \left(\frac{EI}{2L}\right)^{(1)}, \ldots, \left(\frac{EA}{L}\right)^{(4)}, \left(\frac{EI}{2L}\right)^{(4)}, \left(\frac{EI}{2L}\right)^{(4)}\right\} \qquad [70]$$

and the corresponding twelve disassembly vectors are

$$[\mathbf{Q}_k] = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{2\sqrt{3}}{L} & 1-\sqrt{3} & 0 & 0 & 0 \\
0 & -\frac{2\sqrt{3}}{L} & 1+\sqrt{3} & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{2\sqrt{3}}{L} & 1-\sqrt{3} \\
0 & 0 & 0 & 0 & -\frac{2\sqrt{3}}{L} & 1+\sqrt{3} \\
0 & -1 & 0 & 0 & 1 & 0 \\
-\frac{2\sqrt{3}}{L} & 0 & -(1+\sqrt{3}) & \frac{2\sqrt{3}}{L} & 0 & 1-\sqrt{3} \\
\frac{2\sqrt{3}}{L} & 0 & -(1-\sqrt{3}) & -\frac{2\sqrt{3}}{L} & 0 & 1+\sqrt{3} \\
\cos(\theta) & -\sin(\theta) & 0 & -\cos(\theta) & \sin(\theta) & 0 \\
-\frac{2\sqrt{3}}{L}\sin(\theta) & -\frac{2\sqrt{3}}{L}\cos(\theta) & -(1+\sqrt{3}) & \frac{2\sqrt{3}}{L}\sin(\theta) & \frac{2\sqrt{3}}{L}\cos(\theta) & 1-\sqrt{3} \\
\frac{2\sqrt{3}}{L}\sin(\theta) & \frac{2\sqrt{3}}{L}\cos(\theta) & -(1-\sqrt{3}) & -\frac{2\sqrt{3}}{L}\sin(\theta) & -\frac{2\sqrt{3}}{L}\cos(\theta) & 1+\sqrt{3}
\end{bmatrix} \qquad [71]$$

Note that the main two assumptions of finite element disassembly are satisfied: matrix $[\mathbf{W}_k]$ is diagonal and components of disassembly vectors $\{\mathbf{Q}_{k_d}\}$ do not depend on design parameters $\{\mathbf{p}\}$ of the model.

This example shows that several designs can be analyzed simply by modifying the diagonal entries $\mathbf{w}_{k_{dd}}^{(e)}$, then performing the matrix product [54]. For large–size meshes, this procedure becomes much cheaper than re–assembling the entire model. Of course, this is valid only to the extent where the topology of the structure does not change because modifications of the load path generally affect disassembly vectors $\{\mathbf{Q}_{k_d}\}$. Using our simple example, we now illustrate the case of topology modification to show that situations may occur where finite element disassembly continues to apply.

7.2.3. *Structural Removal*

First, we are interested in removing a structural member. This case is the simplest one because it consists of deleting contributions from the element that is being removed. For example, we want to remove the transverse beam number 4. This element contributes to disassembly values and vectors 10, 11 and 12. Simply removing them from vector [70] and matrix [71] leads to the disassembled model described by elements 1, 2 and 3 only.

When the removal of one or several finite elements generates a "free–floating" nodal joint (this would arise in situations where all elements contributing to a given degree of freedom are removed), structural integrity may be re–gained by grounding the corresponding nodal joints. Practically, it would simply mean deleting all entries of vectors $\{\mathbf{Q}_{k_d}\}$ that correspond to the free degrees of freedom. This simple example demonstrates that finite element disassembly can be applied to such problems as topology optimization where the load–carrying capability of a system is optimized. The topology optimization algorithms generally attempt to minimize a given cost function by carving the optimum load path out of an elastic domain. The conclusion is that element removal can be handled easily within the general framework of finite element disassembly.

7.2.4. *Addition of Finite Elements*

The second situation considered consists of adding finite elements to an existing mesh. In cases where no nodal joint or degree of freedom is created, adding a new element can be handled by classical assembly as

$$\left[\mathbf{K}^{(\text{updated})}(\mathbf{p})\right] = \left[\mathbf{K}^{(\text{original})}(\mathbf{p})\right] + \left[\mathbf{L}^{(e)}\right]^T \left[\mathbf{k}^{(e)}(\mathbf{p}^{(e)})\right] \left[\mathbf{L}^{(e)}\right] \qquad [72]$$

Similarly, the disassembled model can be updated by adding to matrices $[\mathbf{W}_k]$ and $[\mathbf{Q}_k]$ the diagonal entries and vectors corresponding to the new finite element. For example, introducing a spring element in figure 2 between nodes 1 and 4 simply consists of adding the disassembly value $\left(\frac{EA}{L}\right)^{(5)}$ at the 13th position on the diagonal of matrix $[\mathbf{W}_k]$ and adding the following vector

$$\left\{\mathbf{Q}_k^{(5)}\right\} = \{-\cos\theta, -\sin\theta, 0, \cos\theta, -\sin\theta, 0\}^T \qquad [73]$$

as the 13th column of matrix $[\mathbf{Q}_k]$. The reason that a single strain mode is considered is because the spring element exhibit extensional stiffness only.

When the mesh is modified in such a way that new degrees of freedom are introduced, the procedure remains basically the same except that components of existing vectors $\{\mathbf{Q}_{k_d}\}$ may have to be re–ordered to match the new degree of freedom numbering. Nevertheless, it should be noted that performing this operation on individual vectors is generally much easier than re–ordering rows and columns of an existing master matrix.

8. Numerical Validation

In this section, the technique for disassembling arbitrary finite element models is illustrated. Implementation and numerical aspects are discussed. Our goal is to illustrate the technique using a variety of structures typical of the aerospace and automotive industries. Our first example is a wing structure formed of 1D and 2D finite elements, then, a volume model is disassembled. Finally, a truss structure is used for illustrating the computational advantage of disassembly over classical assembly in the context of repeated analysis.

Figure 3. *Reduced–Scale Model of a Wing Structure*

	Wing Model	Volume Model	Truss Model
Number of Nodes	432	4,805	44
Number of Elements	1,708	2,727	135
Number of Equations	2,556	14,415	120
Sparsity of [**K**]	1.81%	0.39%	5.00%
Storage of [**K**]	1.43 Mbytes	9.8 Mbytes	9,172 bytes

Table 1. *Properties of the Finite Element Models Used*

8.1. *Wing Model*

Our first example features the finite element model of a reduced–scale wing shown in figure 3. The model counts a total of 1,708 elements among which 434 stiffeners (with axial deformation only), 178 Euler–Bernoulli beams and 1,096 3–node shell elements. The wing is cantilevered and it results into a total of 2,556 active degrees of freedom. Table 1 lists the characteristics of the models used. It can be seen that the sparsity of the stiffness matrix is 1.81%, meaning that only 1.81% of its entries (potentially, 2,556 × 2,556 entries) are nonzero. This very low ratio of nonzero terms is typical of aerospace structures. Table 2 lists characteristics of the disassembled model for this system and the other two used. For the stiffness matrix, a total of 17,752 vectors are generated by disassembly. However, the sparsity ratio of matrix $[\mathbf{Q}_k]$ is only 0.62%: as a result, it requires only a little less than three times the memory required to store the master stiffness matrix.

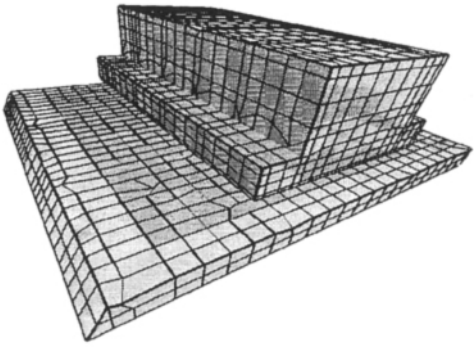

Figure 4. *Simplified Model of a Cylinder–head Block*

As far as computation times are concerned, it can be observed in table 3 that classical assembly and disassembly techniques are very similar for all three examples presented. These examples illustrate that it may even be less expensive to perform a disassembly of the stiffness matrix than to assemble it. Table 3 shows CPU times obtained for various operations with a Matlab–based finite element program. Relatively large assembly times are measured because 1) Programs are interpreted by Matlab rather than compiled; and 2) The program used is a research software in finite element methodologies that is not optimized for computational efficiency as commercial programs would be. However, both assembly and disassembly suffer from the same disadvantages. Hence, the merit of table 3 is to provide a relative comparison between various operations. The CPU times are obtained on the R10,000 processor of a Silicon Graphics workstation. Each time presented is an average of 30 executions of the same operation. In our first example, the computational cost of 2.5 seconds for constructing the master stiffness matrix from its disassembly is approximatively two orders of magnitude less than the cost of classical assembly. Thus, if the design is changed without altering the topology of the structure, master matrices can be reconstructed at a small fraction of the cost of the assembly (about 1%).

8.2. *Cylinder–head Model*

The second example features a simplified, "shoe–box" model of a cylinder–head block from the automotive industry (see figure 4). The model is constructed from 2,727 8–node brick elements with free–floating boundary conditions and it results into a total of 14,415 active degrees of freedom (see table 1). The sparsity ratio of the stiffness matrix is now equal to 0.39%, which basically means that less degrees of freedom are coupled than for the case of the wing model.

Table 2 shows that the stiffness disassembly generates 112,886 vectors stored in columns of matrix $[\mathbf{Q}_k]$. However, the high degree of sparsity of matrix $[\mathbf{Q}_k]$ (with

	Wing Model	Volume Model	Car Model
Length of $[\mathbf{W}_m]$	24,468	65,448	-
Sparsity of $[\mathbf{Q}_m]$	0.39%	0.007%	-
Storage of $[\mathbf{Q}_m]$	0.44 Mbytes	1.11 Mbytes	-
Length of $[\mathbf{W}_k]$	17,752	112,886	113,540
Sparsity of $[\mathbf{Q}_k]$	0.62%	0.14%	0.06%
Storage of $[\mathbf{Q}_k]$	3.99 MBytes	29.20 MBytes	18.7 MBytes

Table 2. *Properties of Disassembled FE Models*

	Wing Model (in seconds)	Volume Model (in seconds)	Car Model (in seconds)
Assembly of $[\mathbf{M}]$	32.2	343.3	-
Disassembly of $[\mathbf{M}]$	53.7	471.8	-
Product of $[\mathbf{Q}_m]^T [\mathbf{W}_m] [\mathbf{Q}_m]$	0.3	2.8	-
Assembly of $[\mathbf{K}]$	280.2	3,023.1	3,700
Disassembly of $[\mathbf{K}]$	237.6	5,294.9	3,000
Product $[\mathbf{Q}_k]^T [\mathbf{W}_k] [\mathbf{Q}_k]$	2.5	78.6	6.7

Table 3. *Typical CPU Times Obtained for the Disassembly*

0.14% of nonzero entries only) yields a memory requirement multiplied by a factor three only compared to storing the master stiffness. The second column in table 3 shows, again, that reconstructing master matrices once their disassemblies are available can be achieved at a fraction of the original cost: 78 seconds with disassembly compared to 3,023 seconds with classical assembly for the stiffness matrix, that is, about 38 times less.

It can be observed that, in general, disassembly tends to generate large numbers of vectors in matrices $[\mathbf{Q}_m]$ and $[\mathbf{Q}_k]$. This is because finite elements are disassembled individually, as mentioned previously. Hence, extraneous (or redundant) information is introduced because the master matrices only span a subspace of dimension equal to their rank. Practically, a filtering technique would be implemented to recover the correct number of independent columns in matrices $[\mathbf{Q}_m]$ and $[\mathbf{Q}_k]$ but the figures listed in table 2 do not take such a filtering into account. Therefore, the CPU times obtained correspond to a worst–case scenario since much more information is generated than actually needed.

8.3. *Car Model*

Finally, we illustrate the disassembly procedure with the body in white of a car shown in figure 5. This model features a total of 3,930 nodal joints and 8,110 triangular shell elements leading to 23,480 degree of freedoms. The DKT18 shell presented in reference [BAT 92] is used for this discretization.

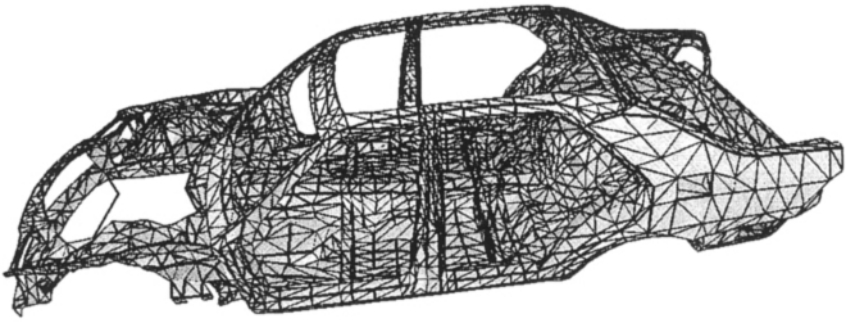

Figure 5. *Finite Element Model of the Car Structure*

Since the DKT18 shell element exhibits a rank order of 11, the number of columns stored in matrix $[\mathbf{Q}_k]$ is equal to 11,3540 after applying boundary conditions. Statistics and CPU times are provided, again, in tables 2 and 3, respectively. They illustrate that the CPU time required for this example is less than the time of assembling the master stiffness matrix. It can also be seen that reconstructing the master matrices once their disassembly is available can be achieved at a marginal fraction of the original cost. Finally the high degree of sparsity of matrix $[\mathbf{Q}_k]$ (with 0.06% of nonzero entries only) yields again a memory requirement multiplied by a factor three compared to storing the symmetrical part of master stiffness.

9. Numerical Solver for Repeated Analysis

In this section, we focus on the problem of solving equations [1-2] with multiple models (that is, when various sets of design variables $\{\mathbf{p}\}$ are tested). Typically, we require multiple resolutions of an equation written as

$$\left([\mathbf{Q}_k]^T\,[\mathbf{W}_k(\mathbf{p})]\,[\mathbf{Q}_k]\right)\{\mathbf{x}\} = \{\mathbf{F}\} \qquad [74]$$

corresponding to the static equilibrium [1]. In this equation, the design $\{\mathbf{p}\}$ is modified at each analysis cycle, and it is assumed that the system matrix is disassembled. For eigenmode extraction equation [74] still hold with the right-hand side defined as

$$\{\mathbf{F}\} = [\mathbf{M}]\,\{\mathbf{x}\}$$

if a subspace iteration solver, for example, is implemented ([GER 97]). We amphasize that the deviation that followsis not restricted to static problems.

Next, in the next section, we introduce the incomplete QR factorization of the disassembly matrix $[\mathbf{Q}_k]$ required for establishing the baseline numerical solver proposed in section 9.2.

9.1. *QR Factorization*

In general, finite element disassembly generates large numbers of vectors. For example, disassembling the model shown in figure 2 generates a total of twelve vectors ($N_D = 12$) whereas the number of active degrees of freedom is equal to six ($N = 6$). Obviously, some of the information generated during disassembly is redundant because the elements are decomposed into strain mode contributions independently from one another.

This section addresses this issue by showing how redundant vectors may be eliminated without loss of vital information about the model. We assume in the remainder that the rank of the matrix decomposed is equal to N_R which is less than or equal to N, the number of equations. Hence, we always work with the following assumption

$$N_D \geq N \geq N_R \qquad [75]$$

The first step to derive a numerical solver for equations [1-2] using disassembly is to "invert" matrix $[\mathbf{Q}_k]$. Since this matrix is generally rectangular and exhibits a high degree of sparsity, a QR factorization is best suited. The QR factorization is denoted by

$$\underset{N_D \times N}{[\mathbf{Q}_k]} = \underset{N_D \times N_D}{[\mathbf{Q}]} \quad \underset{N_D \times N}{[\mathbf{R}]} \qquad [76]$$

where matrix $[\mathbf{Q}]$ is orthogonal, that is, it satisfies

$$[\mathbf{Q}]^T [\mathbf{Q}] = [\mathbf{I}] \qquad [77]$$

and where matrix $[\mathbf{R}]$ would typically feature a $N \times N$ upper triangular part followed by a $(N_D - N) \times N$ zero–block matrix. In equations [76] and [78] below, the convention $N_{row} \times N_{col}$ indicates the number of rows (N_{row}) and the number of columns (N_{col}) of each partition.

It can be seen from equation [76] that the full QR factorization would be prohibitly expensive because it involves the computation of an orthogonal matrix $[\mathbf{Q}]$ of dimension N_D. We could rather implement an incomplete QR factorization. This algorithm is widely available for sparse matrix algebra based either on Householder transformations ([BUS 65]) or Givens rotations ([GEO 80]), the latter being less computationally efficient than the former but offering the advantage of introducing less fill–in. An incomplete QR factorization is denoted in the following by

$$\underset{N_D \times N}{[\mathbf{Q}_k]} = \underset{N_D \times N_R}{[\mathbf{Q}_1} \quad \underset{N_D \times (N_D - N_R)}{\mathbf{Q}_2]} \quad \begin{matrix} \left[\begin{matrix} \mathbf{R}_{11} & \mathbf{R}12 \\ 0 & 0 \end{matrix} \right] \\ \underset{N_R}{} \quad \underset{(N - N_R)}{} \end{matrix} \begin{matrix} N_R \\ (N_D - N_R) \end{matrix} \qquad [78]$$

where the orthogonality condition [77] becomes

$$[\mathbf{Q}_1]^T [\mathbf{Q}_1] = [\mathbf{I}], [\mathbf{Q}_2]^T [\mathbf{Q}_2] = [\mathbf{I}], [\mathbf{Q}_1]^T [\mathbf{Q}_2] = [0] \qquad [79]$$

Decomposition [78] is called incomplete because the partition $[\mathbf{Q}_2]$ is not actually computed. However, the incomplete QR factorization provides the same information regarding the subspace spanned by the original matrix as a complete decomposition, that is

$$\text{Range}\left([\mathbf{Q}_k]\right) = \text{Range}\left([\mathbf{Q}_1]\right) \qquad [80]$$

Equation [80] proves that redundant information generated during finite element disassembly may be filtered out because a minimum number of column–vectors can be collected in matrix $[\mathbf{Q}_1]$ that span the same subspace as the master stiffness matrix. In equation [78], matrix $[\mathbf{R}_{11}]$ is a square, upper triangular and non–singular matrix of dimension N_R. The second non-zero block $[\mathbf{R}_{12}]$ gathers all singular columns of the triangular matrix. For a mechanical system with no mechanism, a maximum of six rigid body modes are extracted from the stiffness matrix. Therefore, the rank would typically be obtained in the range $(N - 6) \leq N_R \leq N$ and partition $[\mathbf{R}_{12}]$ would collect no more than six columns. In the case where the stiffness matrix is full–rank (that is, $N_R = N$), it can be shown that the upper triangular matrix $[\mathbf{R}_{11}]$ is equal to the Choleski factorization of matrix $\left([\mathbf{Q}_k]^T [\mathbf{Q}_k]\right)$ ([GOL 90]).

The advantage of an incomplete QR factorization is that partition $[\mathbf{Q}_2]$ need not to be calculated. This results in significant computational savings. Reference [GOL 90] provides several numerical procedures for computing the incomplete QR factorization, the cost of which can be estimated to $(2N_D N^2)$ floating point operations (Flops). In comparison, it requires approximatively $(\frac{2}{3}Nb^2)$ Flops for factoring the master stiffness matrix where b denotes the sparse storage's bandwidth. Practically, partitions $[\mathbf{R}_{11}]$ and $[\mathbf{R}_{12}]$ are extracted from the upper triangular matrix $[\mathbf{R}]$ by identifying which columns possess nonzero diagonal entries and which columns possess zero diagonal entries, respectively. The latter characterizes the singular columns that are stored in partition $[\mathbf{R}_{12}]$. Then, we are left with the non–singular part $[\mathbf{R}_{11}]$. The inverse of this matrix is denoted by $[\mathbf{R}_{11}]^{-1}$ in the remainder; However, it should be kept in mind that this operation is simply a backward substitution since the "inverted" matrix is upper triangular.

We now investigate the derivation of a pseudo–inverse matrix using the incomplete QR factorization obtained previously. For solving a rectangular system of equations $[\mathbf{A}]\{\mathbf{x}\} = \{\mathbf{b}\}$, a pseudo–inverse matrix $[\mathbf{A}^+]$ is generally defined as the matrix that minimizes the Frobenius (or Euclidean) norm of the following residue

$$\min_{[\mathbf{A}^+]} \left\| [\mathbf{A}]\left[\mathbf{A}^+\right] - [\mathbf{I}] \right\|_2 \qquad [81]$$

Going over the derivation of the minimization problem [81] where $[\mathbf{A}] = [\mathbf{Q}_k]$ provides the following solution

$$\left[\mathbf{Q}_k^+\right] = [\mathbf{R}_{11}]^{-1} [\mathbf{Q}_1]^T \qquad [82]$$

An alternative way of establishing this result is to verify that the pseudo–inverse matrix satisfies the four necessary and sufficient Moore–Penrose conditions listed in equation [83-86] below. The pseudo–inverse is the unique matrix that satisfies these conditions

$$[\mathbf{Q}_k]\,[\mathbf{Q}_k^+]\,[\mathbf{Q}_k] = [\mathbf{Q}_k] \tag{83}$$

$$[\mathbf{Q}_k^+]\,[\mathbf{Q}_k]\,[\mathbf{Q}_k^+] = [\mathbf{Q}_k^+] \tag{84}$$

$$([\mathbf{Q}_k]\,[\mathbf{Q}_k^+]) \text{ is symmetric and equal to } \left([\mathbf{Q}_1]\,[\mathbf{Q}_1]^T\right) \tag{85}$$

$$([\mathbf{Q}_k^+]\,[\mathbf{Q}_k]) \text{ is symmetric and equal to } [\mathbf{I}] \tag{86}$$

Again, we emphasize the fact that partition $[\mathbf{Q}_2]$ is not required in the solution procedure [82]. Similarly, the notation $[\mathbf{Q}_k^+]$ is employed in the following for the sake of clarity. However, this pseudo–inverse needs not to be calculated explicitely which also saves a great deal of memory and CPU requirements. Therefore, implementing an incomplete QR factorization provides a computationally efficient procedure for "inverting" matrix $[\mathbf{Q}_k]$ while featuring at the same time a practical filtering of redundant information introduced by finite element disassembly. Of course, this procedure is significantly more expensive than a simple Choleski factorization of the master matrix. Nevertheless, we will show that this initial higher cost is rapidly compensated by the very efficient disassembly–based solution procedure for repeated analysis.

9.2. *Baseline Numerical Solver*

The solution to equation [74] can now be expressed as

$$\{\mathbf{x}\} = \left([\mathbf{Q}_k^+]\,[\mathbf{W}_k(\mathbf{p})]^{-1}\,[\mathbf{Q}_k^+]^T\right)\{\mathbf{F}\} \tag{87}$$

where the pseudo–inverse matrix $[\mathbf{Q}_k^+]$ is provided symbolically by equation [82]. Of course this solution is obtained in a least square sense since the number of disassembly vectors generated is usually greater than the dimension of the problem ($N_D \geq N$), as illustrated by previous numerical examples. However, the obtained solution could be seen as an excellent starting guess for an iterative solving procedure, which necessitate only few step refinement for leading to an acceptable solution. But in practical applications shown in sections 9.3 to 9.6, no refinement steps are involved since solutions obtained with [87] are readily acceptable.

Practically, the solution procedure only involves forward–backward substitutions and it takes advantage of orthogonality conditions [79] to avoid constructing and storing the pseudo–inverse matrix. The numerical implementation of this solver is provided and discussed briefly below:

$$1.\ \text{Solve } \left\{\mathbf{x}^{(1)}\right\} = [\mathbf{R}_{11}]^{-T}\{\mathbf{F}\} \tag{88}$$

$$2.\ \text{Multiply } \left\{\mathbf{x}^{(2)}\right\} = [\mathbf{Q}_1]\left\{\mathbf{x}^{(1)}\right\} \tag{89}$$

$$3.\ \text{Solve}\ \left\{\mathbf{x}^{(3)}\right\} = [\mathbf{W}_k(\mathbf{p})]^{-1}\left\{\mathbf{x}^{(2)}\right\} \qquad [90]$$

$$4.\ \text{Multiply}\ \left\{\mathbf{x}^{(4)}\right\} = [\mathbf{Q}_1]^T\left\{\mathbf{x}^{(3)}\right\} \qquad [91]$$

$$5.\ \text{Solve}\ \{\mathbf{x}\} = [\mathbf{R}_{11}]^{-1}\left\{\mathbf{x}^{(4)}\right\} \qquad [92]$$

Step 1 of solver [88-92] consists of a forward substitution with a system of N_R linearly independent equations. It requires approximately $(\frac{1}{2}Nb)$ Flops where b denotes the sparse storage's bandwidth. Note that the load vector possesses a total of N entries. However, if the system is rank–deficient (that is, $N_R \leq N$), only N_R components need to be solved for. Typically, these components are identified during the QR factorization by storing indices of columns where a nonzero entry is found on the diagonal of triangular matrix $[\mathbf{R}]$. Step 2 of solver [88-92] is a matrix–vector multiply that results into a maximum of $(N_D N_R)$ Flops. Step 3 involves the inversion of a diagonal matrix and Step 4 is another matrix–vector multiply: these totalize a maximum of $N_R\,(N_D + 1)$ Flops. Finally, Step 5 consists of a backward substitution with N_R equations. The computational burden of solver [88-92] is therefore equal to $N_R\,(2N_D + b + 1)$, at most, if it is assumed that the orthogonal matrix $[\mathbf{Q}_1]$ is full and that the triangular matrix $[\mathbf{R}_{11}]$ is stored using a skyline profile of bandwidth b. Obviously, this requirement is marginal compared to the $(2N_D N^2)$ Flops required for QR factorization.

Also, we see that implementing the solution procedure [88-92] using a second set of design parameters $\{\mathbf{p}\}$ only requires to modify Step 3 because only matrix $[\mathbf{W}_k]$ depends on the model's design variables. If the load vector $\{\mathbf{F}\}$ is unchanged, repeating Steps 3 to 5 involves a maximum of $N_R\,(N_D + \frac{b}{2} + 1)$ Flops in addition to the cost of re–assembling the diagonal of matrix $[\mathbf{W}_k]$, which can be kept to a minimum given the adequate implementation effort. In the remainder of this section, the procedure is illustrated using two different finite element models.

9.3. *Disassembly of a Truss Structure*

Our first example is the truss structure shown in figure 6 and presented in tables 1-3. It features a total of 44 nodal joints and it is modeled using 135 bar elements (with extensional stiffness only). The ten–bay truss is cantilevered, its number of active degrees of freedom is equal to 120 and each finite element contributes to essentially one strain mode. Therefore, disassembly yields a matrix $[\mathbf{Q}_k]$ formed of 120 rows and 135 columns. Its sparsity is 2.12%, meaning that 97.88% of its entries are not stored because they are equal to zero. The CPU time required for total finite element disassembly is equal to 0.84 seconds and QR factorization is achieved in 0.03 seconds. In comparison, the CPU time required for assembly of the master stiffness matrix is equal to 1.15 seconds and a Choleski factorization is obtained in 0.003 seconds. The sparsity ratio of the master stiffness matrix is equal to 5.00%.

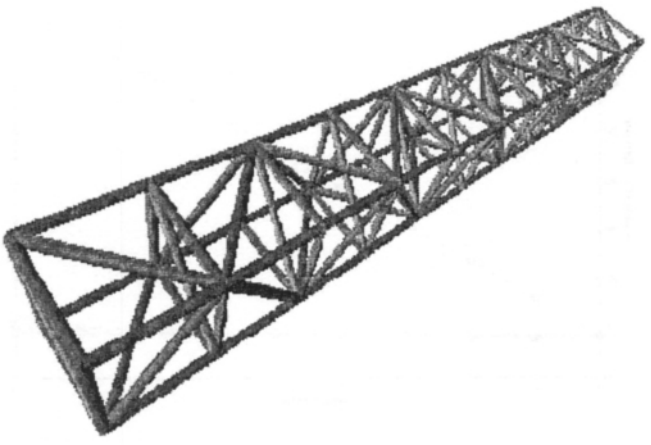

Figure 6. *Finite Element Model of NASA Langley's DSMT Truss Structure*

9.4. *Re–analysis of a Truss Model With Various Solvers*

Figure 7 compares CPU times obtained with three implementations as the design is changed and multiple solutions to equation [74] are sought after: 1) The stiffness matrix is re–assembled, re–factored and multiple load cases are analyzed; 2) No new assembly or factorization is necessary and only forward–backward resolutions are carried out; and 3) The disassembly technique is implemented together with solver [87]. Note that the second strategy is not realistic as soon as the design is changed because the stiffness matrix must be re–factored. However, since the forward–backward resolution is the most efficient solver for sparse matrices, a comparison with case 2 provides us with insight regarding the computational efficiency of disassembly. In figure 7, five loading cases are analyzed for each design and 10 design cycles are considered. As expected, figure 7 shows that case 1 is by far the most expensive solution procedure. Cases 2 and 3 are very similar in terms of computational efficiency for this example. The reader should however keep in mind that these results depend on the problem's size, its degree of sparsity and the type of finite elements involved.

Next, we apply a loading in the vertical direction at each of the four nodes at the free end of the truss and we are interested in computing the structural response for various combinations of design parameters $\{\mathbf{p}\}$. Three solvers are compared in the following. The first one consists of re–assembling the master stiffness matrix, performing a Choleski factorization and forward–backward resolutions for each model considered. The second solver implements the Woodbury method: in the case where modifications brought to the stiffness matrix can be represented as a superposition of rank–one updates, the solution of a modified system of equations can be obtained from the previous solution without having to perform a new factorization. Equations [93]

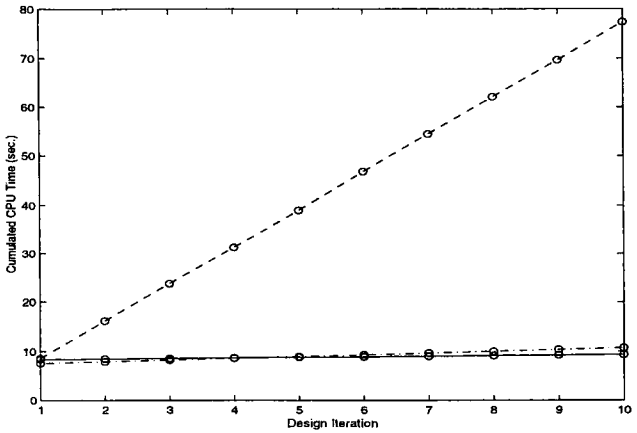

Figure 7. *Comparison of Three Solvers for Structural Re-analysis. Case 1 is shown with the dashed line: the master matrix is re-assembled and re-factored for each new analysis. Case 2 is shown with the solid line: only forward-backward resolutions are implemented for each new analysis. Case 3 is shown with the dashed/dotted line: model disassembly and solver [87] are implemented*

to [95] below summarize the formulation of this second solver. We assume that the "new" stiffness matrix is obtained from the previous one by

$$\left[\mathbf{K}(\mathbf{p}^{(new)})\right] = [\mathbf{K}(\mathbf{p})] + [\mathbf{E}_w]\left[\mathbf{D}_w(\mathbf{p}^{(new)})\right][\mathbf{E}_w]^T \qquad [93]$$

where $[\mathbf{D}_w]$ is a diagonal matrix that collects changes brought to the model and $[\mathbf{E}_w]$ is a localization matrix used for propagating these changes to entries of the master stiffness matrix. Design changes would, for example, represent rank–one perturbations assigned to individual finite elements between two successive design cycles. Then, it can be verified that the "new" solution can be expressed as a function of the previous solution by

$$\left\{\mathbf{x}^{(new)}\right\} = \left([\mathbf{I}] - [\mathbf{K}(\mathbf{p})]^{-1}[\mathbf{E}_w][\mathbf{W}_w]^{-1}[\mathbf{E}_w]^T\right)\{\mathbf{x}\} \qquad [94]$$

where

$$[\mathbf{W}_w] = \left(\left[\mathbf{D}_w(\mathbf{p}^{(new)})\right]^{-1} + [\mathbf{E}_w]^T[\mathbf{K}(\mathbf{p})]^{-1}[\mathbf{E}_w]\right) \qquad [95]$$

The solution procedure of the Woodbury–based solver goes as follows: the first analysis consists of solving the linear system $[\mathbf{K}]\{\mathbf{x}\} = \{\mathbf{F}\}$. Then, the model is perturbed according to equation [93] and solution [94] is implemented for any subsequent analysis. Since a Choleski factorization of the original stiffness matrix is available, equation [94] consists only of forward–backward resolutions and matrix–vector multiplications, which is cheaper than having to re–factor the new stiffness matrix.

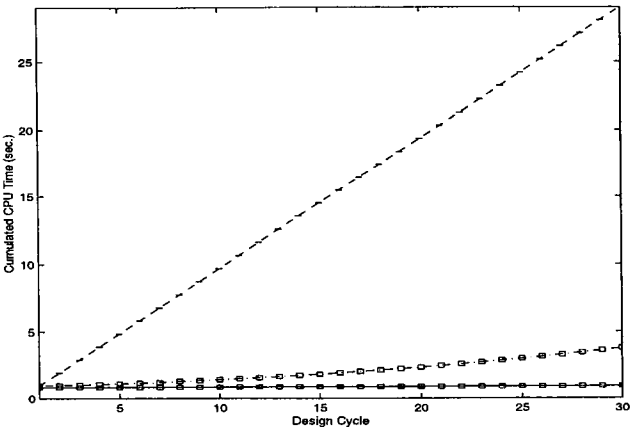

Figure 8. *Comparison of Three Solvers for Structural Re-analysis of a Truss Model. Case 1 is shown with the dashed line: the master matrix is re-assembled and factored for each new analysis. Case 2 is shown with the dashed/dotted line: the Woodbury-based solver [93-95] is implemented with updates of rank (n-1) at the n-th design cycle. Case 3 is shown with the solid line: the disassembly-based solver [88-92] is implemented*

It can be seen that the numerical efficiency of this second solver depends to a great extent on the size of matrix $[\mathbf{W}_w]$ that is the only one to require factorization. Equation [95] shows that this matrix is full and that its size is equal to the number of column vectors in matrix $[\mathbf{E}_w]$. Typically, an update of rank N_R involves exactly N_R linearly independent vectors and, therefore, matrix $[\mathbf{W}_w]$ in equation [95] is a square matrix of size N_R. To maximize the numerical efficiency of this solver in our numerical example, we restrict ourselves to rank–one updates: the cross–sectional area of a single bar element is modified at each new design cycle. Hence, the size of matrix $[\mathbf{W}_w]$ is equal to $N_R = (n-1)$ at the nth design cycle. Note that, for a rank–one update, equations [93-95] degenerate into the well–known Sherman–Morisson update.

The third solver is the disassembly–based algorithm described in equation [88-92]. The CPU times obtained are equal to 0.95 seconds, 0.13 seconds and 0.03 seconds for the Choleski–based, Woodbury–based and disassembly–based solvers, respectively. We emphasize that these figures represent the average CPU time required for a single static resolution.

Figure 8 illustrates the cumulated CPU requirements for each one of the three algorithms. It can be observed that the cost of the first solver increases linearly with the number of design cycles: this makes sense since similar operations (assembly, factorization and forward–backward resolutions) are repeated each time. With the Woodbury–based solver, the first analysis is the same solution procedure as before. Then, figure 8 shows that additional iterations are much cheaper because a matrix of

small size $N_R = (n-1)$ is factored at iteration n. It also shows that the Woodbury–based solver loses its numerical efficiency as N_R, the rank of the update, increases. Clearly, our disassembly–based solver provides the best result. In a worst–case scenario, the computational cost associated to finite element disassembly and QR factorization would be greater than that of solver one, although this is not observed in our numerical simulation. In any case, these expensive manipulations are performed only once and the computational requirement of each subsequent iteration represents a marginal fraction of that cost. Moreover, this requirement does not depend, as in the case of a Woodbury–based solver, on the rank of the update. The conclusion is that our disassembly–based solver is best suited to situations where enough designs are analyzed to absorb the potentially higher initial cost.

9.5. *Re–analysis of a Wing Model With Various Solvers*

We will now discuss the re–analysis of the wing model presented in section 8.1 (see figure 3) using the same three solvers as before. Disassembly yields a matrix $[\mathbf{Q}_k]$ formed of 2,556 rows and 17,752 columns. Since we are essentially interested in constructing an inverse stiffness matrix, the largest disassembly values $\mathbf{w}_{k_{dd}}$ are filtered out because their inverse would be very small compared to other components, therefore, the contribution from corresponding disassembly vectors $\{\mathbf{Q}_{k_d}\}$ can be neglected. Hence, the number of disassembly vectors is reduced down from 17,752 to 3,039 which greatly lowers the computational burden, yet yields a solution accurate up to the second decimal (the maximum error is 1%).

Next, static forces and moments are applied at the free end of the wing to simulate a structural response similar to the first torsional mode. Again, a total of $n = 30$ resolutions are repeated in the same condition as before, that is, a single perturbation (or rank–one modification) is introduced between any two consecutive design cycles. CPU times obtained for a single static resolution are equal to 279.03 seconds, 39.75 seconds and 133.04 seconds for the Choleski–based, Woodbury–based and disassembly–based solvers, respectively. Figure 9 illustrates the cumulated CPU requirements for the three algorithms. The Woodbury–based solution procedure now appears to be cheaper than the other two: this is because any two successive designs only differ by a rank–one update, which is the best possible situation for this solver, yet not the most realistic one. Nevertheless, the cost of the second solver grows quadratically each time as the rank of the update is increased because matrix [95] is full and it must be factored. The cost of the disassembly–based solver, to the contrary, is independent of the amount of change brought to the model.

9.6. *Application to Fully Stressed Design*

One last illustration of the applicability of finite element disassembly is the problem of minimum mass design given known operating conditions. The baseline truss model shown in figure 6 is optimized to decrease its total mass as much as possible

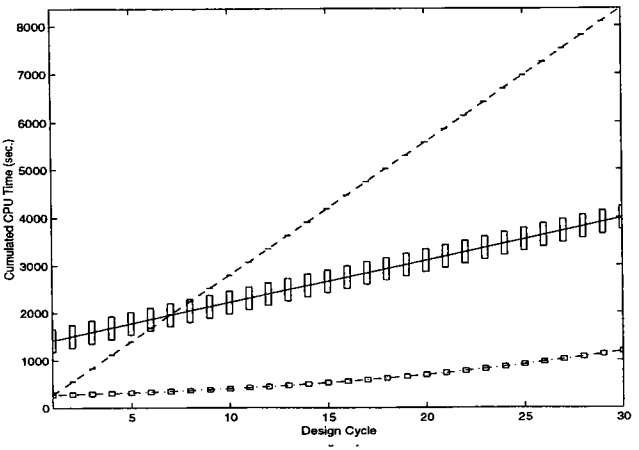

Figure 9. *Comparison of Three Solvers for Structural Re-analysis of a Wing Model. Case 1 is shown with the dashed line: the master matrix is re-assembled and factored for each new analysis. Case 2 is shown with the dashed/dotted line: the Woodbury-based solver [93-95] is implemented with updates of rank (n-1) at the n-th design cycle. Case 3 is shown with the solid line: the disassembly-based solver [88-92] is implemented*

given the same static loading as before (vertical forces are applied at each of the four nodes at the free end of the truss). The optimization algorithm described briefly below is implemented and performance is compared using two static solvers: the first solver involves classical assembly followed with a Choleski factorization and the second solver features finite element disassembly and implementation [88-92].

Many formulations and solution procedures have been proposed for this problem ([RAZ 65], [ROZ 89]), among which we choose to implement the fully stressed design algorithm developed by Souza de Cursi and Pagnacco because of its simplicity ([SOU 95]). The problem is formulated as the minimization of the total mass of the structure

$$\min_{\{\mathbf{p}\}} (mass) \qquad [96]$$

given a constraint on the stress distribution generated in the model by the applied loading. This limitation can be expressed as

$$\Psi(\mathbf{x};\sigma) \leq \Psi_{max} \qquad [97]$$

where the design criterion Ψ typically represents a Von Mises–like criterion of plastic failure and Ψ_{max} represents the material's limit of elasticity. Using mathematical considerations, it is shown in reference [SOU 95] that the optimum design $\{\mathbf{p}^{(opt)}\}$ can be obtained by seeking the solution to the equation $\{\mathbf{p}^{(opt)}\} = \mathbf{f}(\Psi(\mathbf{p}^{(opt)}))\{\mathbf{p}^{(opt)}\}$ where $\mathbf{f}$ is a somewhat arbitrary, user–defined function. This problem is solved by con-

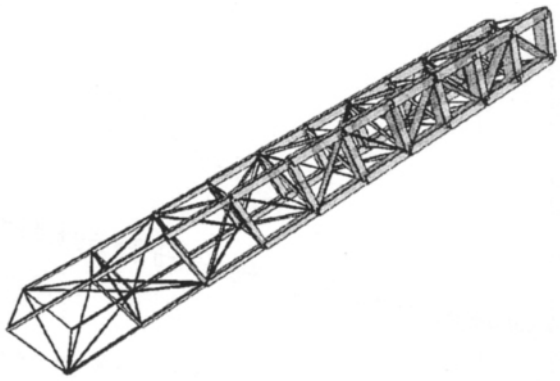

Figure 10. *Finite Element Model of NASA Langley's DSMT Truss Structure After Fully Stressed Design Optimization*

structing a sequence of models $\{\mathbf{p}^{(n)}\}$ that converge to the optimum design $\{\mathbf{p}^{(opt)}\}$ using the following numerical procedure:

1. Assembly of Matrix $\left[\mathbf{K}(\mathbf{p}^{(n)})\right]$ [98]

2. Resolution of $\left[\mathbf{K}(\mathbf{p}^{(n)})\right]\left\{\mathbf{x}^{(n)}\right\} = \{\mathbf{F}\}$ [99]

3. Determine the Stress Distribution $\left\{\sigma^{(n)}\right\} = \left\{\sigma(\mathbf{x}^{(n)})\right\}$ [100]

4. Determine the Design Objective $\Psi^{(n)} = \Psi(\mathbf{x}^{(n)}; \sigma^{(n)})$ [101]

5. Optimization $\left\{\mathbf{p}^{(n+1)}\right\} = (1-\omega)\left\{\mathbf{p}^{(n)}\right\} + \omega\mathbf{f}\left(\frac{\Psi^{(n)}}{\Psi_{max}}\right)\left\{\mathbf{p}^{(n)}\right\}$ [102]

6. Apply Constraints $\mathbf{p}_k^{(n+1)} = \min\left(\mathbf{p}_{\max}; \max\left(\mathbf{p}_{\min}; \mathbf{p}_k^{(n+1)}\right)\right)$ [103]

In the application example below, $\mathbf{f}(\mathbf{x}) = \mathbf{x}$ is chosen for simplicity. Step 6 of the algorithm is implemented to ensure that each design parameter $\mathbf{p}_k$ is optimized within user–defined bounds $[\mathbf{p}_{\min}; \mathbf{p}_{\max}]$. The relaxation parameter ω can be optimized for improving the convergence rate of the algorithm. In our example, it is kept constant at $\omega = \frac{1}{2}$. Convergence is assessed by estimating the total amount of change brought to the model between any two iterations.

The total mass of the truss illustrated in figure 6 is optimized using the same static loading as in section 9.4. The fully stressed design algorithm [98-103] requires multiple inverse resolutions during step 2 and two implementations are tested. The first one involves classical assembly and a Choleski–based solver. The second one involves finite element disassembly (performed at iteration $n = 1$ only) and the solver [88-92]. Note that the Woodbury–based solver [93-95] would typically be inadequate in this situation because all of the model's design variables are modified simultaneously at each iteration. Figure 10 illustrates the optimum design reached after 15 iterations

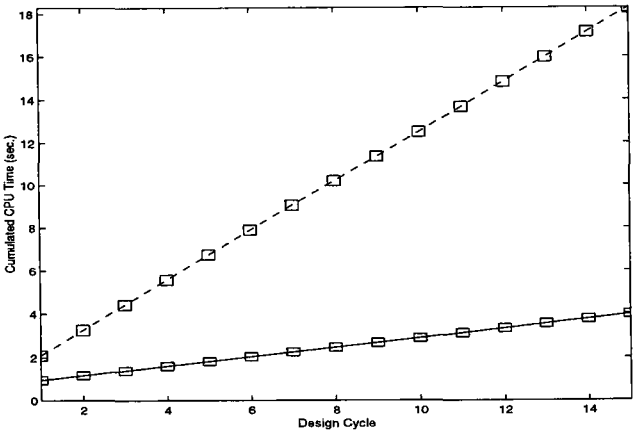

Figure 11. *CPU Time Requirements for Fully Stressed Design Optimization. Case 1 is shown with the dashed line: the master matrix is re-assembled and factored for each new analysis. Case 2 is shown with the solid line: the disassembly-based solver [88-92] is implemented*

only. Optimization variables consist of the cross–sectional areas of the 135 truss members. The solution makes perfect engineering sense since stiffer members (indicated by thicker cross–sectional areas in figure 10) are placed near the cantilever boundary condition where strain energy is the highest. At the same time, cross–sectional areas are decreased to minimize the mass as much as possible in areas that do not carry significant loading. This optimization results into a minimization of the total mass by 83%: the structure's weight is decreased from 0.0574 lbf (or 10.05 kg) before optimization to 0.0097 lbf (or 1.71 kg) after. Both solvers provide the same design because assembly–based and disassembly–based solutions at step 2 are identical. However, computational requirements are quite different, as illustrated in figure 11. Although a small number of iterations are necessary to achieve convergence, finite element disassembly and its associated solver provide a dramatic reduction of the total computational requirement.

10. Inverse Problem Solving Using Flexibility Data

An application to test–analysis reconciliation is now discussed. First, we formulate the inverse problem of finite element model updating. This discussion is aimed at showing that most updating techniques result into similar resolutions ; the only difference between any two methods being the definition of a residue vector. It is shown here that the particular choice of flexibility data offers significant advantages in the context of finite element disassembly, not to mention that flexibility matrices are easily measured during modal tests.

10.1. *Formulation of Finite Element Model Updating*

Updating methods can generally be formulated as the minimization of residue vectors. Residues measure the (lack of) correlation between test data and responses simulated using the finite element model. Typical examples include the difference between identified and simulated frequencies or vectors. Indirect measures of the correlation between the test article and the model can also be defined, such as out–of–balance forces in the model. Reference [HEM 98] provides a description of these popular residues.

No matter what definition is used, design parameters are sought after that minimize residues at the $(n+1)$th iteration given a distribution of residues at the n-th iteration. The best–case scenariowould consist of reaching zero–residues, that is

$$\left\{\mathcal{R}(\mathbf{p}^{(n+1)};\mathbf{s})\right\} = 0 \qquad [104]$$

where the updated design parameters are obtained from current ones as

$$\left\{\mathbf{p}^{(n+1)}\right\} = \left\{\mathbf{p}^{(n)}\right\} + \left\{\delta\mathbf{p}^{(n+1)}\right\} \qquad [105]$$

Usually, substituting definition [105] into equation [104] leads to a nonlinear system of equations. First–order linearizations are attractive alternatives to the implementation of computationally intensive nonlinear solvers. It yields

$$\left\{\mathcal{R}(\mathbf{p}^{(n+1)};\mathbf{s})\right\} \approx \left\{\mathcal{R}(\mathbf{p}^{(n)};\mathbf{s})\right\} + \sum_{k=1\cdots N_p} \left\{\frac{\partial\mathcal{R}}{\partial\mathbf{p}_k}(\mathbf{p}^{(n)};\mathbf{s})\right\}\delta\mathbf{p}_k^{(n+1)} \qquad [106]$$

At any updating iteration, the system [106] is solved for the unknown correction parameters $\{\delta\mathbf{p}\}$ by inverting in a least–squares sense a correction matrix $[\mathcal{G}]$ that collects the residue's gradient vectors in its columns. The compact form of system [106] is provided below

$$\left[\mathcal{G}(\mathbf{p}^{(n)};\mathbf{s})\right]\left\{\delta\mathbf{p}^{(n+1)}\right\} = -\left\{\mathcal{R}(\mathbf{p}^{(n)};\mathbf{s})\right\} \qquad [107]$$

with

$$\left[\mathcal{G}(\mathbf{p}^{(n)};\mathbf{s})\right] = \left[\left\{\frac{\partial\mathcal{R}}{\partial\mathbf{p}_1}(\mathbf{p}^{(n)};\mathbf{s})\right\}\left|\left\{\frac{\partial\mathcal{R}}{\partial\mathbf{p}_2}(\mathbf{p}^{(n)};\mathbf{s})\right\}\right|\cdots\left|\left\{\frac{\partial\mathcal{R}}{\partial\mathbf{p}_{N_p}}(\mathbf{p}^{(n)};\mathbf{s})\right\}\right] \qquad [108]$$

Regularization can be added to improve the condition # of matrices to invert. This would typically modify the definition of matrix [108] but it does not change the conclusion reached below. For clarity, regularization is not assumed in the following.

It can be seen from equation [107] that the solution procedure consists of forming matrix [108], then inverting it to get the solution and adjust the model. All updating algorithms using gradient–based optimization techniques involve a similar procedure. In the following, we show that the computational burden associated to this procedure can be greatly reduced when the correlation involves flexibility data.

10.2. *Using Flexibility Data and Reduced FE Models*

Using flexibility data consists of defining the residue vectors as the difference between measured and analytical Frequency Response Functions (FRFs). Since FRFs are obtained by inverting the dynamic stiffness matrix at a given frequency s, a flexibility–based residue matrix can be defined as

$$[\mathcal{R}(\mathbf{p};\mathbf{s})] = [\mathbf{Z}_R(\mathbf{p};\mathbf{s})]^{-1} - [\mathcal{F}_{id}(\mathbf{s})] \qquad [109]$$

The advantage of this definition is that the measurements $[\mathcal{F}_{id}]$ can be obtained directly from test data by estimating the transfer functions between input excitations and output measurements. Measuring the flexibility matrix is a pre–requisite to any identification algorithm and it is generally much cheaper and less troublesome than attempting to best–fit mode shapes and frequencies, especially when modally complex structures are tested and identified.

We now emphasize the fact that only a small number of degrees of freedom are measured during a modal test, typically no more than 10% in aerospace applications, even less in automotive applications. If otherwise, the updating problem offers no challenge because it can be solved for in a single iteration. Therefore, any comparison between test data and finite element quantities involves a procedure for resolving this spatial incompleteness. For example, the dynamic stiffness matrix is reduced to match the size of the experimental model. Accordingly, our disassembled, analytic model can be reduced down to the measurement points by

$$[\mathbf{Z}_R(\mathbf{p};\mathbf{s})] = [\mathbf{T}]^T \left([\mathbf{K}(\mathbf{p})] - \mathbf{s}^2 [\mathbf{M}(\mathbf{p})]\right) [\mathbf{T}] \qquad [110]$$

$$= \left([\mathbf{Q}_{k,m}] [\mathbf{T}]\right)^T [\mathbf{W}_{k,m}(\mathbf{p};\mathbf{s})] \left([\mathbf{Q}_{k,m}] [\mathbf{T}]\right) \qquad [111]$$

where matrix [T] denotes the model condensation operator. Finally, using the results of section 9, the disassembled equation [111] can be inverted to provide the model–based flexibility matrix as

$$[\mathbf{Z}_R(\mathbf{p};\mathbf{s})]^{-1} = \left[\mathbf{Q}^+_{k,m}\right] [\mathbf{W}_{k,m}(\mathbf{p};\mathbf{s})]^{-1} \left[\mathbf{Q}^+_{k,m}\right]^T \qquad [112]$$

In equation [112], the original disassembly matrix $[\mathbf{Q}_{k,m}]$ is kept for simplifying the notations although it should actually be replaced with the condensed matrix $([\mathbf{Q}_{k,m}][\mathbf{T}])$. From now on, the former will denote the condensed disassembly matrix. Again, a pseudo–inverse matrix is used for clarity in equation [112] but the reader should keep in mind that it is never actually required, as discussed in section 9.

10.3. *Two-Step Updating Algorithm*

We have seen that a residue matrix [109] can be defined based on measured flexibility data and condensed finite element models. The solution procedure for model updating as outlined in section 10.1 is now detailed using this particular choice. It

is shown that disassembly can be exploited to greatly reduce the computational burden. Basically, no matrix inversion nor QR factorization needs to be performed for updating the model.

We recall that the computational model is adjusted by solving the parameter correction equation below with our particular choice of flexibility–based residues

$$\sum_{k=1}^{N_p} \frac{\partial}{\partial \mathbf{p}_k} \left(\left[\mathbf{Z}_R(\mathbf{p}^{(n)}; \mathbf{s}) \right]^{-1} \right) \delta \mathbf{p}_k^{(n+1)} = - \left[\mathcal{R}(\mathbf{p}^{(n)}; \mathbf{s}) \right] \qquad [113]$$

Using the disassembly representation [112] for the reduced finite element flexibility matrix, the previous system of equations becomes

$$\left[\mathbf{Q}_{k,m}^{+} \right] \left(\sum_{k=1}^{N_p} \frac{\partial}{\partial \mathbf{p}_k} \left(\left[\mathbf{W}_{k,m}(\mathbf{p}^{(n)}; \mathbf{s}) \right]^{-1} \right) \delta \mathbf{p}_k^{(n+1)} \right) \left[\mathbf{Q}_{k,m}^{+} \right]^T = - \left[\mathcal{R}(\mathbf{p}^{(n)}; \mathbf{s}) \right] \qquad [114]$$

Equation [114] holds because only the disassembly values stored in diagonal matrix $[\mathbf{W}_{k,m}]$ depend on design variables. This is however not exact strictly speaking since $[\mathbf{Q}_{k,m}]$ depends on a condensation matrix. The latter would typically be derived from the model, therefore, introducing an implicit dependency with respect to the current design $\{\mathbf{p}^{(n)}\}$. However, it is common practice to neglect this relationship in first order approximation because the condensation operator $[\mathbf{T}]$ is generally computed using the original finite element model and used for several refinement iterations without being updated. When it is assessed that the current model is far away from the starting point, test–analysis correlation can be established and the condensation matrix can be updated based on the current model.

We can re–write this correction system using a generic matrix $[\mathcal{G}]$ equal to the summation of diagonal sensitivity matrices in the left–hand side of equation [114]. It gives

$$\left[\mathbf{Q}_{k,m}^{+} \right] \left[\mathcal{G}(\mathbf{p}^{(n)}; \mathbf{s}) \right] \left[\mathbf{Q}_{k,m}^{+} \right]^T = - \left[\mathcal{R}(\mathbf{p}^{(n)}; \mathbf{s}) \right] \qquad [115]$$

We may now consider without loss of generality that the problem consists of solving equation [115] for the unknown quantity $[\mathcal{G}]$. However, the solution matrix is not arbitrary: it is diagonal and exhibits the specific pattern described by equation [114]. This leads to a two–step updating algorithm where equation [115] is solved first, then the particular form of matrix $[\mathcal{G}]$ is best–fitted to the solution. The first resolution is referred to as the "global" updating problem because all equations are coupled. It can be verified easily that the solution is given simply by

$$\left[\mathcal{G}(\mathbf{p}^{(n)}; \mathbf{s}) \right] = - \left[\mathbf{Q}_{k,m} \right] \left[\mathcal{R}(\mathbf{p}^{(n)}; \mathbf{s}) \right] \left[\mathbf{Q}_{k,m} \right]^T \qquad [116]$$

It is remarquable that no matrix inversion nor QR factorization is required to solve the global updating problem once the disagreement [109] between test and analysis flexibility matrices has been calculated. Note also that solution [116] should yield a

diagonal matrix if the connectivity of the model is an exact representation of the test structure's load path. This is however rarely the case due to finite element approximations. Moreover, measurement noise and numerical roundoff prevents the solution matrix from being exactly diagonal in most realistic applications. Therefore, the finite element parameterization is best–fitted by solving the following system that involves the diagonal of matrix $[\mathcal{G}]$ only

$$\sum_{k=1\cdots N_p} \frac{\partial}{\partial \mathbf{p}_k} \left(\left[\mathbf{W}_{k,m}(\mathbf{p}^{(n)}; \mathbf{s}) \right]^{-1} \right) \delta \mathbf{p}_k^{(n+1)} = \text{Diag} \left(\left[\mathcal{G}(\mathbf{p}^{(n)}; \mathbf{s}) \right] \right) \quad [117]$$

This second resolution is referred to as the "local" updating problem because equations are actually decoupled for each finite element. In the best–case scenario, solving equation [117] consists of inverting a diagonal matrix when a single design variable is updated per finite element. In the worst–case scenario, solving equation [117] consists of inverting as many small–size systems as there are finite elements adjusted. The size of an individual system for a specific element is dictated by the number of design parameters corrected within this element. Therefore, the local updating problem introduces hardly no computational requirement at all since resolutions can be performed in parallel and independently from one another.

This solution procedure yields the following iterative solver:

1. Disassembly of Matrices $\left[\mathbf{K}(\mathbf{p}^{(n)}) \right], \left[\mathbf{M}(\mathbf{p}^{(n)}) \right]$ [118]

2. Computation of the Reduction Matrix $\left[\mathbf{T}^{(n)} \right] = \left[\mathbf{T}(\mathbf{p}^{(n)}; \mathbf{s}) \right]$ [119]

3. Estimation of Residues :

$$\left[\mathcal{R}(\mathbf{p}^{(n)}; \mathbf{s}) \right] = \left[\mathbf{Z}_R(\mathbf{p}^{(n)}; \mathbf{s}) \right]^{-1} - \left[\mathcal{F}_{id}(\mathbf{s}) \right] \quad [120]$$

4. Resolution the of Global Problem :

$$\left[\mathcal{G}(\mathbf{p}^{(n)}; \mathbf{s}) \right] = - \left[\mathbf{Q}_{k,m} \right] \left[\mathcal{R}(\mathbf{p}^{(n)}; \mathbf{s}) \right] \left[\mathbf{Q}_{k,m} \right]^T \quad [121]$$

5. Resolution of the Element – level, Local Problems :

$$\sum_{k=1}^{N_p^{(e)}} \frac{\partial}{\partial \mathbf{p}_k^{(e)}} \left(\left[\mathbf{W}_{k,m}^{(e)}(\mathbf{p}^{(n)}; \mathbf{s}) \right]^{-1} \right) \delta \mathbf{p}_k^{(n+1)} = \text{Diag} \left(\left[\mathcal{G}(\mathbf{p}^{(n)}; \mathbf{s}) \right] \right) \quad [122]$$

6. Update the Finite Element Model $\left\{ \mathbf{p}^{(n+1)} \right\} = \left\{ \mathbf{p}^{(n)} \right\} + \left\{ \delta \mathbf{p}^{(n+1)} \right\}$ [123]

7. Apply Constraints $\mathbf{p}_k^{(n+1)} = \min \left(\mathbf{p}_{\max}; \max \left(\mathbf{p}_{\min}; \mathbf{p}_k^{(n+1)} \right) \right)$ [124]

It is generally not necessary to implement steps 1-2 above at each updating iteration: it suffices to update the condensation matrix only when the finite element model has been significantly modified. Many choices are available from the literature that yields reduction matrices [**T**] that preserve the dynamics of the system within a given

Figure 12. *Finite Element Model and Sensing Configuration of the Ladder Structure*

frequency range and/or that extrapolate the response accuratly for structural modification. Another reason for avoidin steps 1-2 is that model condensation costs are very significant. In comparison, the computational cost of the global problem [116] increases linearly with the number of measurement points while the cost of the local problem [117] is linear with the number of finite elements updated. Section 10.4 discusses an application example where the finite element model of an automotive frame structure is adjusted for improving its joint stiffness characteristics.

10.4. *Validation of Flexibility–based Model Updating*

The flexibility–based reconciliation algorithm is now illustrated using the finite element model depicted in figure 12. The structure is referred to as the "ladder" structure and it represents a simplified model of engine cradle used in the automotive industry. The Euler–Bernoulli beam model is chosen for mathematical idealization and the system is discretized into 120 nodal joints and 120 finite elements. Boundary conditions are free–free. Figure 12 shows the sensing configuration adopted (three translations at 16 nodal joints are measured) as well as rigid elements introduced in the finite element model to account for measurement offsets. A total of 48 out of the 672 active degrees of freedom are measured during the simulation.

Although test data are available, FRFs used for this validation are simulated numerically. This is to provide us with an assessment of the method in a situation where the finite element model is a perfect idealization of the structure's load path and where no measurement noise is introduced. Sparsity of the master stiffness matrix is equal to 0.72% and the corresponding storage requirement is 41.6 KBytes. In comparison, the sparsity of disassembly matrix $[\mathbf{Q}_k]$ is equal to 0.39% and the corresponding storage requirement is 24.9 KBytes. Assembly and disassembly require roughly the same amount of computation, with a slight advantage to disassembly.

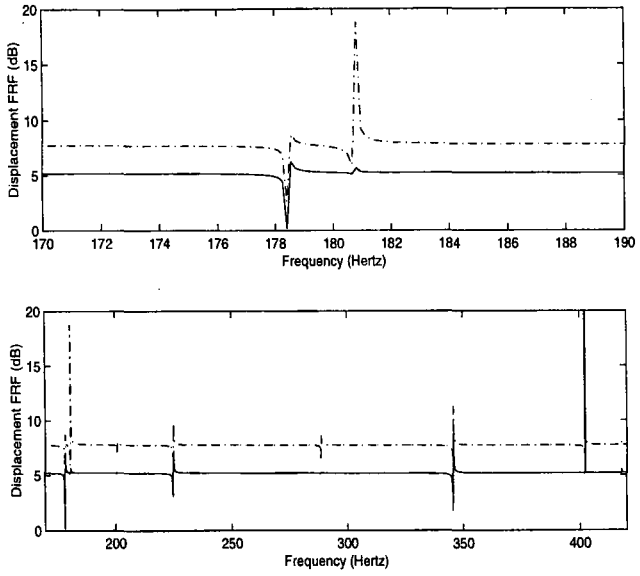

Figure 13. *Comparison of FRF Data Before and After Updating. The nominal finite element model (before updating) is shown with the dashed/dotted line. The adjusted finite element model (after disassembly-based updating) is shown with the dashed line. The reference, "truth" structure is shown with the solid line. No difference between the test data and response of the optimized model is visible*

Then, a translational spring element is added to the model at each one of the four nodal joints where longitudinal and transverse beams are connected. In this example, the modeling error consists of underestimating these spring constants by a factor of six. In other words, "test" data are simulated using spring stiffnesses six times superior to their values in the nominal finite element model. The data set consists of the full, 48 by 48, FRF matrix at 180 Hertz. This sampling frequency is chosen for no particular reason other than for being close to the model's first bending mode (at 178.48 Hertz).

Figure 13 compares FRF data before and after updating for a particular input–output pair but over the entire frequency range of interest. The improvement brought to the model after three updating iterations is clearly visible since the updated model (dashed line) matches the "test" data (solid line) over the entire frequency range, even though only FRFs at 180 Hertz were provided to the updating algorithm. Figure 14 compares the final stiffnesses to their exact value. The solution has undoubtedly been identified with acceptable accuracy after three iterations only. This example illustrates the performance of updating algorithm [118-124] when the finite element model is disassembled. The advantage of this procedure over modal–based techniques should be emphasized: here, the input data consist of small–size FRF blocks that can be obtained more easily than resonant frequencies and mode shapes.

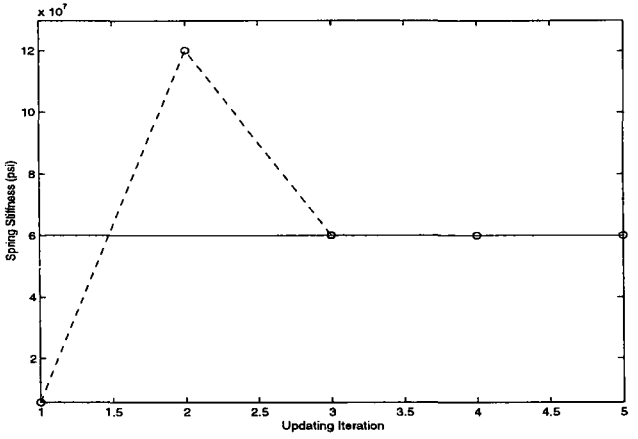

Figure 14. *Adjustment Brought to the Disassembled Ladder Model. Updated stiffness values (shown with the dashed line) are compared to their reference, "truth" value (shown with the solid line) after the disassembled finite element model has been updated using a single FRF block*

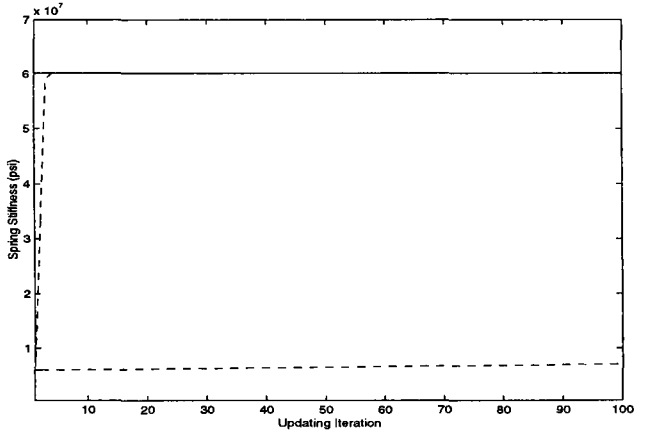

Figure 15. *Adjustment Brought to the Assembled Ladder Model. Updated stiffness values (shown with the dashed line) are compared to their reference, "truth" value (shown with the solid line) after 100 iterations of updating the assembled finite element model using a single FRF block*

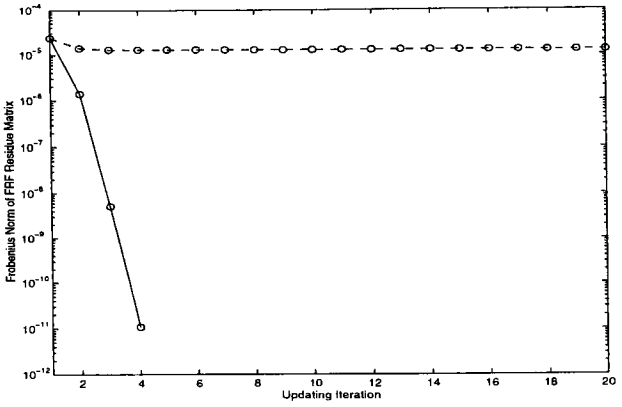

Figure 16. *Convergence During Model Updating of the Ladder Structure. Case 1 is shown with the solid line: the disassembly–based solver [118-124] is implemented. Case 2 is shown with the dashed line: finite element matrices are not disassembled and equation [113] is inverted numerically*

In terms of computational requirement, the solver [118-124] is compared with a similar version where the finite element model is not disassembled. When no disassembly is available, global and local inverse problems (steps 4 and 5, respectively) must be replaced with equation [113] that is inverted numerically. This second version therefore proceeds with finite element assembly and model reduction at each updating iteration, next, the correction system [113] is solved for in a least–squares sense. In both versions, the cost of constructing the condensation matrix [T] and reducing the finite element model from 672 degrees of freedom down to the 48 measurement points remain similar. Figure 15 compares the stiffness parameters obtained after 100 iterations to their exact value. Two parameters converge rapidly; Obtaining the other two requires over 1,000 updating iterations. After 100 iterations, these two parameters are increased by 15.3% only. This example clearly illustrates that the disassembly–based solution procedure [118-124] is more efficient than its numerical counterpart because, in the first case (with disassembly), algebraic solutions are implemented while, in the second case (with assembly), matrices must be inverted numerically which yields approximate, least–squares solutions.

Figure 16 depicts the convergence of both algorithms. It can be observed that convergence of the second solver is slowed down significantly after the third iteration when two of the updating parameters have reached their solution. Small incremental changes are then brought to the remaining two springs, which explains why the residue [109] decreases very slowly. Finally, figure 17 illustrates the cumulated CPU requirements as a function of the updating iteration. It demonstrates that significant computational savings are provided when finite element disassembly is exploited: CPU times required to complete the first four optimizations are reduced by a factor of three with finite element disassembly (6.23 seconds as opposed to 17.62 seconds).

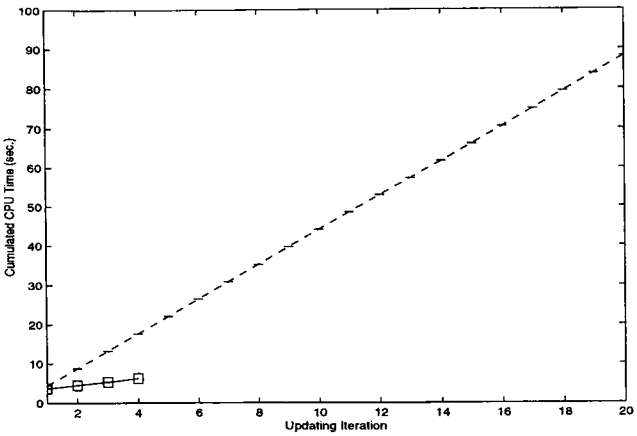

Figure 17. *CPU Time Requirements for Model Updating of the Ladder Structure. Case 1 is shown with the solid line: the disassembly–based solver [118-124] is implemented. Case 2 is shown with the dashed line: finite element matrices are not disassembled and equation [113] is inverted numerically*

11. Conclusion

This publication deals with the disassembly of finite element models for structural dynamics applications and inverse problem solving. Disassembly consists of partitioning element–level matrices into inertia modes (for the mass operator) and strain modes (for the stiffness operator). These contributions can be collected in global storages to yield the disassembly of master mass and stiffness matrices. Although implementing the algebraic disassembly of finite element matrices may involve significant programming efforts as well as important computational resources, it is shown that this particular representation can be exploited for deriving efficient inverse solvers.

Since disassembly decouples information regarding the design from information regarding the topology and the metric of the structure, updating a disassembled model can be performed at a fraction of the cost required for re–assembling the system matrices when the design is modified. Basically, only a diagonal matrix is updated.

Within this context, numerically efficient solvers for inverse structural dynamics problems are derived. Problems of interest include multiple analyses, structural optimization and finite element model updating. These share the need for solving linear systems repeatedly with multiple designs and, most often, with a known topology and metric. In this situation, finite element disassembly provides a very efficient numerical procedure that involves a single, incomplete QR factorization, no matter how many analyses are performed. Then, solving a linear system only requires two matrix multiplications. Similarly, solving a linear system with a modified design only adds to this cost the updating of a diagonal matrix.

Application examples are presented that feature models typically encountered in the automotive and aerospace communities. In all cases considered (multiple analysis, fully stressed design and flexibility–based model updating), the disassembly–based solver is shown to provide significant computational speed–up ratios compared to traditional solvers. Future work includes the investigation of numerical disassembly that reduces the implementation effort and that may be interfaced with any commercially–available finite element package.

12. References

[ARG 60] ARGYRIS, J.H., and KELSEY, S., *Energy Theorems and Structural Analysis*, Butterworths, London, U.K., 1960.

[BAT 92] BATOZ J.L., and DHATT G., *Modélisation de Structures par Eléments finis*, Vol. 3, Hermès Editeur, 1992.

[BAT 96] BATHE K.J., *Finite Element Procedures*, Prentice Hall, M.I.T. Press, 1996.

[BUS 65] BUSINGER, P.A., and GOLUB, G.H., "Linear Least Squares Solutions by Householder Transformations," *Journal of Numerical Mathematics*, Vol. 7, 1965, pp. 269–276.

[CAL 83] CALLADINE, C.R., *Theory of Shell Structures*, Cambridge University Press, New York, NY, 1983.

[FAR 91] FARHAT, C., and ROUX, F.–X., "A Method of Finite Element Tearing and Interconnecting and its Parallel Solution Algorithm," *International Journal of Numerical Methods in Engineering*, Vol. 32, 1991, pp. 1205–1227.

[FAR 92] FARHAT, C., and GÉRADIN, M., "A Hybrid Formulation of a Component Mode Synthesis Method," *33rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, # AIAA–92–2383–CP, Dallas, TX, Apr. 13–15, 1992, pp. 1783–1796.

[FEL 89] FELIPPA, C.A., and MILITELLO, C., "Developments in Variational Methods For High Performance Plate and Shell Elements," *Analytical and Computational Models of Shells*, Edited by Noor, A.K., Belytschko, T., and Simo, J.C., ASME Winter Annual Meeting, San Francisco, CA, Dec. 10–15, 1989, pp. 191–215.

[FRA 65] FRAEIJS DE VEUBEKE, B.M., "Displacement and Equilibrium Models in the Finite Element Method," *Stress Analysis*, Edited by Zienkiewicz, O.C, and Hollister, G., Wiley, London, U.K., 1965, pp. 145–197.

[GEO 80] GEORGE, J.A., and HEATH, M.T., "Solution of Sparse Linear Least Squares Problems Using Givens Rotations," *Journal of Linear Algebra and Its Applications*, Vol. 34, 1980, pp. 69–83.

[GER 97] GÉRADIN, M., and RIXEN, D., *Mechanical Vibration, Theory and Application to Structural Dynamics*, J. Wiley & Sons, New York, N. Y., Second Edition, 1997.

[GOL 90] GOLUB, G.H., and VAN LOAN, C.F., *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Maryland, Second Edition, 1990.

[GOR 96] GORDIS, J.H., "On the Analytic Disassembly of Structural Matrices," *Modal Analysis: the International Journal of Analytic and Experimental Modal Analysis*, Vol. 11, No. 1 & 2, July 1996, pp. 39–48.

[HAF 93] HAFTKA, R.T., AND GÜRDAL, Z., *Elements of Structural Optimization*, Solid Mechanics and Its Applications, Kluwer Academic Publishers, Third Edition, 1993.

[HEM 95] HEMEZ, F.M., and Farhat, C., "Structural Damage Detection Via a Finite Element Model Updating Methodology," *Journal of Modal Analysis*, Vol. 10, No. 3, July 1995, pp. 152–166.

[HEM 98] HEMEZ, F.M., "Can Model Updating Tell the Truth?," *16th SEM International Modal Analysis Conference*, Santa Barbara, CA, Feb. 2–5, 1998, pp. 1–7.

[HUG 87] HUGHES, T.J.R., *The Finite Element Method: Linear, Static and Dynamic Finite Element Analysis*, Prentice–Hall, Englewood Cliffs, NJ, 1987.

[JON 75] JONES, R.M., *Mechanics of Composite Materials*, Taylor & Francis, 1975.

[McG 90] McGOWAN, P.E., SMITH, S.W., and JAVEED, M., "Experiments For Locating Damaged Members in a Truss Structure," *2nd USAF/NASA Workshop on System Identification and Health Monitoring of Precision Space Structures*, Pasadena, CA, Mar. 27–29, 1990, pp. 571–615.

[ODE 76] ODEN, J.T., *An Introduction to the Mathematical Theory of Finite Elements*, Wiley, New York, NY, 1976.

[ODE 83] ODEN, J.T., and REDDY, J.N., *Variational Methods in Theoretical Mechanics*, Springer–Verlag, Berlin, Germany, 2nd Edition, 1983.

[PAG 97] PAGNACCO, E., and PARK, K.C., "Damage Location Using a Localized Flexibility–Based Method," Center for Aerospace Structures, University of Colorado at Boulder, Boulder, CO 80309–0429, Nov. 1997.

[PAR 97] PARK, K.C., REICH, G.W., and ALVIN, K.F., "Structural Damage Detection Using Localized Flexibilities," Center for Aerospace Structures, University of Colorado at Boulder, Boulder, CO 80309–0429, # CU–CAS–97–14, July 1997.

[PAZ 91] PAZ, M., *Structural Dynamics, Theory and Computation*, Van Nostrand Reinhold, New York, NY, Third Edition, 1991.

[PET 95] PETERSON, L.D., DOEBLING, S.W., and ALVIN, K.F., "Experimental Determination of Local Structural Stiffness by Disassembly of Measured Flexibility Matrices," *36th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, # AIAA–95–1090–CP, New Orleans, LA, Apr. 10–13, 1995, pp. 2756–2766.

[RAZ 65] RAZANI R., "Behavior of Fully Stressed Design of Structures and Its Relationship to Minimum-Weight Design," *AIAA Journal*, Vol. 3, No. 12, 1965.

[ROZ 89] ROZVANY G.I.N., *Structural Design Via Optimilaty Criteria*, Kluwer, Dordrecht, The Netherlands, 1989.

[SOU 95] SOUZA DE CURSI, J.E., and PAGNACCO, E., "Minimum Mass Spare Parts in 2D Elasticity," *Structural and Multidisciplinary Optimization*, Edited by Olhoff, N., and Rozvany, G.I.N., Pergamon Press, 1995, pp. 231–236.

[SUZ 91] SUZUKI, K., and KIKUCHI, N., "A Homogenization Method for Shape and Topology Optimization," *Computer Methods in Applied Mechanics and Engineering*, Vol. 93, 1991, pp. 291–318.