Three-dimensional Finite Element Formulation and Scalable Domain Decomposition for High Fidelity Rotor Dynamic Analysis

Anubhav Datta
ELORET Corporation
AFDD at Ames Research Center
Moffett Field, CA 94035

Wayne Johnson
Aeromechanics Branch
NASA Ames Research Center
Moffett Field, CA 94035

ABSTRACT

This paper implements and analyzes a dual-primal iterative substructuring method, that is parallel and scalable, for the solution of a three-dimensional finite element dynamic analysis of helicopter rotor blades. The finite element analysis is developed using isoparametric hexahedral brick elements. Particular emphasis is placed on the formulation of the inertial terms that are unique to rotary wing structures. The scalability of the solution procedure is studied using two prototype problems – one for steady hover (symmetric) and one for transient forward flight (non-symmetric) – both carried out on up to 48 processors. In both hover and forward flight, a linear speed-up is observed with number of processors, up to the point of substructure optimality. Substructure optimality and the range of linear speed-up are shown to depend both on the problem size as well as a corner based global coarse problem selection. An increase in problem size extends the linear speed-up range up to the new substructure optimality. A superior coarse problem selection extends the optimality to a higher number of processors. The method also scales with respect to problem size. The key conclusion is that a three-dimensional finite element analysis of a rotor can be carried out in a fully parallel and scalable manner. The careful selection of substructure corner nodes, that are used to construct the global coarse problem, is the key to extending linear speed-up to as high a processor number as possible, thus minimizing the solution time for a given problem size.

Nomenclature

Δ	= denotes incremental quantities
δ	= denotes variational quantities
ϵ_{ij}	= Green-Lagrange strains
η^s	= subdomain interface fluxes
κ	= condition number
κ_{ij}	= non-linear strains
λ	= interface dual variables
Ω	= rotor rotational speed
ψ	= blade azimuth angle
ψ^{BI}	= rotation of frame B relative to I
σ_{ij}	= second Piola-Kirchhoff stresses
θ	= instantaneous control angle
$\tilde{\omega}, \dot{\tilde{\omega}}$	= skew symmetric angular velocity and accelerations
ε_{ij}	= linear strains
$\vec{r}$	= displacement of a point relative to inertial frame
$\vec{x}^{BI}, \dot{\vec{x}}^{BI}$	= displacement and velocity of frame B relative to frame I
ξ, η, ζ	= finite element curvilinear, natural, coordinate axes
a	= subscript denoting finite element nodal quantities
$B^s_{R/E/C}$	= subdomain Boolean restrictions
BI/B	= quantity in B relative to I measured in B
C	= Cauchy-Green deformation tensor
C^{BI}	= orientation of frame I relative to B

Presented at the American Helicopter Society Annual Forum 65, Grapevine, Texas, May 27–29, 2009

D, D'	= linear constitutive relations
f^s	= subdomain force vector
f^*_C	= coarse problem force vector
H_a	= finite element shape function for node a
I, B	= inertial and non-inertial body fixed frames
I_3	= unit matrix 3×3
J	= Jacobian of transformation from physical to finite element natural coordinate axes
K^s	= subdomain stiffness matrices
K^*_{CC}	= coarse problem stiffness matrix
L	= composite ply angle transformation
M	= interface preconditioner
q	= finite element degrees of freedom
S	= discretized substructure interface
s	= superscript denoting substructure quantities
$T_{m/c/k/f}$	= inertial mass, damping, stiffness, and force
u^g_C	= global solution vector for corner nodes
u^s	= subdomain solution vector = Δq^s
u_i	= physical deflections, $i = 1, 2, 3$ in frame B
x^0_i	= undeformed coordinates of a point in frame B
x_i	= deformed coordinates of a point in frame B

INTRODUCTION

One hundred years ago what is now called the *Poincaré-Steklov* operator was introduced. This operator, which governs the interface of a problem generated by decomposing a larger problem into many smaller subproblems, has spectral properties that are superior to the original problem. In particular, its condition number (ratio of maximum to minimum eigenvalues, which determines the number of iterations needed for an iterative solution) grows at a rate that is an order lower than that of the original problem. Every modern method of iterative substructuring is based on one or more variational forms of this operator.

The word *substructuring* and the method was introduced fifty years ago by Przemieniecki [1, 2]. Together with Denke [3], Argyris and Kelsey [4], and Turner et al. [5], they laid the foundations of displacement and force finite element analyses of partitioned structures. These partitioned methods were the only avenues to obtain results for practical structures for which the original problem far exceeded the capacity of computers at the time. The method of substructures has remained the fastest (time), most efficient (memory), most reliable (accurate), and most flexible (heterogeneous physics and properties) method of partitioned analysis of large scale structures. The modern methods of primal and dual iterative substructuring have their origin and genesis in these original contributions – established long before the advent of parallel computers.

The advent of parallel computers opened opportunity for solving each partitioned substructure using a separate processor. A straight forward implementation, however, leads to a dead end. First, the high condition number and lack of sparsity of the interface equation by itself poses a significant challenge for convergence. Second, the condition number grows rapidly both with problem size and with number of partitions, producing a dramatic increase in the required number of iterations. The recognition, that a finite element representation of the substructure interface is precisely a discrete equivalent of the original Poincaré-Steklov operator, allows the mathematical theories of domain decomposition to be brought to bear directly towards the resolution of this key problem. Today, methods are available that precondition the interface and solve it iteratively, in a parallel and scalable manner, requiring only local substructure calculations. These methods are called *iterative substructuring* methods. Their objective is to provide optimal numerical scalability, i.e. to ensure that the condition number of the pre-conditioned interface does not grow with the number of substructures and grows at most poly-logarithmically with mesh refinement within each substructure.

The mathematical theory of domain decomposition provides scalable algorithms for two broad classes of partitioning: overlapping and non-overlapping [6]. Overlapping partitioning leads to *additive Schwarz* methods, which are variants of block Jacobi preconditioners. These are widely used in fluid mechanics, but are of little importance in structural mechanics – due to the very high condition numbers ($10^8 - 10^{10}$) and high bandwidth of practical structures. Structural mechanics prefer non-overlapping partitioning. They lead to iterative substructuring methods, a name borrowed from, and indicative of the deep connections to the long and successful tradition of substructuring. Henceforth, the words ‘subdomain’ and ‘substructure’ are used synonymously.

The growth of the mathematical theory of iterative substructuring can be traced to the seminal work of Agoshkov et al. [7, 8] and Bramble et al. [9] in the mid-1980s. The former provided a detailed analysis of the Poincaré-Steklov operator. The latter provided one of the earliest scalable interface preconditioners for problems governed by 2nd order elliptic partial differential equations (henceforth, mentioned as 2nd order problems) with homogeneous coefficients. Subsequent algorithmic developments in this area have continued through the 1990s and 2000s (the interested reader is referred to monographs by Quarteroni and Valli [10] and Toselli and Widlund [11]) culminating in the increasing application of these methods for High Performance Computing (HPC) based large scale problems of computational mechanics (see special eds. [12, 13]). Today, Neumann-

Neumann type primal methods known as Balancing Domain Decomposition with Constraints (BDDC) (see [14] and references therein) and the Dirichlet-Dirichlet type dual methods known as Finite Element Tearing and Interconnecting (FETI) methods (see [15] and references therein) provide scalable preconditioners that are optimal up to 4th order elliptic problems that include highly discontinuous and heterogeneous subdomains.

Both the Neumann-Neumann and FETI methods act on the discrete equivalents of the Poincaré-Steklov interface operator. The former acts on its primal form. The latter acts on its dual form. A primal form consists of variables that are a direct subset of the original unknowns, e.g., the interface displacements. A dual form consists of a different set of variables that are not a subset of the original unknowns, but whose equality must still be guaranteed across the interface, e.g., the reaction forces. Both methods are based on simultaneous Dirichlet and Neumann solves within each substructure — one for preconditioning and one for residual calculation. Note that the Neumann solves are non-invertible for floating substructures that arise naturally from multiple partitioning of a structure. In Neumann-Neumann, this singularity occurs in the preconditioner, in FETI this singularity occurs in the residual calculation.

The first objective of this paper is to apply an advanced multi-level FETI method, the FETI-Dual Primal (DP) method, pioneered by Farhat et al. [15, 16], for the parallel solution of a large scale rotary wing structural dynamics problem. The FETI-DP method acts both on the primal and dual form of the interface - each form for a separate subset of the interface. We apply this method in the present work because there is a substantial volume of published literature demonstrating its high level of performance for large scale engineering problems (see references above). Note, however, that both the FETI-DP and the BDDC methods are closely connected, and we refer the interested reader to the recent work of Mandel et al. [17] for an excellent exposition of this connection.

The state-of-the-art in rotary wing structural modeling involves a variational-asymptotic reduction of the 3-dimensional (3-D) nonlinear elasticity problem into a 2-D linear cross-section analysis and a 1-D geometrically exact beam analysis – based on Berdichevsky [18] and pioneered by Hodges and his co-workers [19]. Aeroelastic computations are performed on the beam, followed by a recovery of the 3-D stress field. The method is efficient and accurate – except near end-edges and discontinuities for which a 3-D analysis is still needed – as long as the cross-sections are small compared to the wavelength of deformations along the beam. Modern hingeless and bearingless rotors contain 3-D flexible load bearing components near the hub that have short aspect ratios and cannot be treated as beams. Moreover, treatment of blades, depending on their advanced geometry and material anisotropy, also requires continuous improvements based on refinements to the asymptotic cross-section analysis [20, 21].

A second objective of this paper, therefore, is to develop a 3-D Finite Element Model (FEM) for rotary wing structures that can be used to analyze generic 3-D non-beam like hubs as well as advanced geometry blade shapes. With the emergence of rotorcraft Computational Fluid Dynamics (CFD), physics-based models containing millions of grid points can carry out Reynolds Averaged Navier-Stokes (RANS) computations on 100s of cores, routinely, in a research environment for the rotor, and even for the entire helicopter. Applications are focused today on coupling CFD with relatively simple engineering-level structural models. The structural analyis is carried out on a single processor while the remaining processors lie idle. Therefore, the purpose of the second objective is to explore the possibility of integrating 3-D FEM as the physics-based Computational Structural Dynamics (CSD) model in the structures domain.

There is no question that such a capability will be powerful. First, it will provide enabling technology for modeling hingeless and bearingless rotors with advanced geometries. Second, it will enable the direct calculation of stresses on critical load bearing components near the hub, a capability that is not available today. Third, it will provide a true representation of the 3-D structure, consistent with the high level of fidelity sought in large scale CFD. And finally, even though this research is targeted towards large scale HPC based analysis, as a by-product, it will always provide a means for extracting sectional properties with which efficient lower order design analyses can still be carried out. The key question for such a capability is that of an efficient solution procedure.

The primary objective of this paper is to answer this key question directly. As in CFD, the tremendous capabilities of HPC is also envisioned here to be the key technology driver and enabler.

Scope of Present Work

A parallel Newton-Krylov solver is developed in this study for the solution of 3-D FEM analysis of rotors in hover and forward flight. The solver is based on the FETI-DP method of iterative substructuring. The primary emphasis in this paper is on the scalability of this solver.

The formulation of the 3-D FEM analysis pays particular attention on the structural non-linearities and inertial terms that are unique to rotary wings. The Krylov solver is equipped with a Generalized Minimum Residual (GMRES) update, in addition to its traditional Conjugate Gradient (CG) update, to accommodate the non-symmetric nature of the rotary wing inertial terms.

Advanced finite element capabilities like locking-free elements, hierarchical elements, nonlinear constitutive models, composite ply modeling are beyond the scope of this initial work. Grid generation is not part of this endeavor. Simple grids are constructed that are adequate for research purposes. Domain partitioning, on the other hand, is a part of this work. Standard graph partitioners, which are readily available, will not suffice for reasons described herein. Most key elements of a comprehensive rotorcraft analysis are not considered at present: airloads model, trim model, extraction of periodic dynamics, and multibody dynamics, are all part of future work.

The paper is organized as follows. The next, i.e., the second section, describes the formulation of the 3-D FEM analysis, followed by preliminary verification using thin plate and rotating beam results. The third section presents a brief description of the iterative substructuring algorithm and its parallel implementation. The numerical scalability of the algorithm is established in this section. The fourth section introduces the key components of the 3-D rotor analysis: geometry and grids, partitioning and corner selection, the hover prototype, and the transient forward flight prototype. The fifth section is focused on scalability. Calculations performed on up to 48 processors – the maximum available to the authors at present – are reported here. The paper ends with the key conclusions of this work and a summary of the future research directions that are critical to the success of this endeavor.

3-D FINITE ELEMENTS FOR ROTORS

The Finite Element formulation is based on well established, standard procedures [22, 23]. The non-linear, geometrically exact implementation, follows an incremental approach, using Green-Lagrange strain and second Piola-Kirchhoff stress measures, within a total Lagrangian formulation. The main contribution here is the formulation of the inertial terms that are unique to rotary wings.

Strain Energy

Let the deformed coordinates of a material point in the blade at any instant be given by

$$\begin{aligned} x_1 &= x_1^0 + u_1(x_1^0, x_2^0, x_3^0) \\ x_2 &= x_2^0 + u_2(x_1^0, x_2^0, x_3^0) \\ x_3 &= x_3^0 + u_3(x_1^0, x_2^0, x_3^0) \end{aligned} \tag{1}$$

where x_i^0 and u_i, $i = 1, 2, 3$ denote the undeformed coordinates and the 3-D deformation field respectively. The deformation gradient of the point with respect to its undeformed configuration is then ∇

$$\nabla = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{bmatrix} \quad \text{where} \quad x_{i,j} = \frac{\partial x_i}{\partial x_j^0} \tag{2}$$

The Green-Lagrange strain relates the deformed length of a line element, dl, to its original length on the undeformed blade, dl^0, in the following manner

$$\epsilon_{ij}\, d x_i\, d x_j = (1/2)[(dl)^2 - (dl^0)^2] \tag{3}$$

where $(dl)^2 = d x_i\, d x_i$ and $(dl^0)^2 = d\, x_i^0 d x_i^0$. The Green-Lagrange strain tensor follows

$$\epsilon = (1/2)(C - I_3) \tag{4}$$

Here I_3 is a 3×3 unit matrix and C is the left Cauchy-Green deformation tensor given by

$$C = \nabla^T\ \nabla \tag{5}$$

The elements of the Green-Lagrange strain tensor have the well-known form

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j^0} + \frac{\partial u_j}{\partial x_i^0} + \frac{\partial u_k}{\partial x_i^0}\ \frac{\partial u_k}{\partial x_j^0}\right) \qquad k = 1, 2, 3 \tag{6}$$

The total Lagrangian formulation is based on virtual work per unit *original* volume. The stress measure that is energetically conjugate to Green-Lagrange strain is the second Piola-Kirchhoff stress tensor, σ. That is, the strain energy of the deformed structure can now be calculated using the above strain and stress measures with integration over original volume. The variation in strain energy is then simply

$$\delta U = \int_V \sigma\ \ \delta\epsilon_{ij}\ \ dV \tag{7}$$

For non-linear analysis, an incremental procedure is followed. The strain at a current state $t + \Delta t$ is expressed in terms of incremental strains measured from a previous state t

$$\epsilon_{ij}(t + \Delta t) = \epsilon_{ij}(t) + \Delta\epsilon_{ij} \tag{8}$$

where $\Delta\epsilon_{ij}$ is the incremental strain. It must be understood that the variation in strain at the current state is simply the variation in incremental strains. That is,

$$\delta\epsilon_{ij}(t + \Delta t) = \delta\Delta\epsilon_{ij} \tag{9}$$

The incremental strain is related to the incremental deformations Δu_i.

$$\begin{aligned} \Delta u_i &= x_i(t + \Delta t) - x_i(t) \\ &= u_i(t + \Delta t) - u_i(t) \end{aligned} \tag{10}$$

Substitution of the strain expression Eq. 6 in Eq. 8 and use of $u_i(t + \Delta t) = u_i(t) + \Delta u_i$ gives

$$\begin{aligned} \Delta\epsilon_{ij} =& \frac{1}{2}\left(\frac{\partial \Delta u_i}{\partial x_j^0} + \frac{\partial \Delta u_j}{\partial x_i^0} + \frac{\partial u_k}{\partial x_i^0}\ \frac{\partial \Delta u_k}{\partial x_j^0} + \frac{\partial \Delta u_k}{\partial x_i^0}\ \frac{\partial u_k}{\partial x_j^0}\right) \\ &+ \frac{1}{2}\ \frac{\partial \Delta u_k}{\partial x_i^0}\ \frac{\partial \Delta u_k}{\partial x_j^0} = \Delta\varepsilon_{ij} + \Delta\kappa_{ij} \qquad k = 1, 2, 3 \end{aligned} \tag{11}$$

where the linear and non-linear strains are separately denoted as ε_{ij} and κ_{ij}. The variations follow

$$\delta\Delta\epsilon_{ij} = \delta\Delta\varepsilon_{ij} + \delta\Delta\kappa_{ij} \tag{12}$$

where

$$\begin{aligned} &\delta\Delta\varepsilon_{ij} = \\ &\frac{1}{2}\left(\frac{\partial \delta\Delta u_i}{\partial x_j^0} + \frac{\partial \delta\Delta u_j}{\partial x_i^0} + \frac{\partial u_k}{\partial x_i^0}\ \frac{\partial \delta\Delta u_k}{\partial x_j^0} + \frac{\partial \delta\Delta u_k}{\partial x_i^0}\ \frac{\partial u_k}{\partial x_j^0}\right) \\ &\delta\Delta\kappa_{ij} = \\ &\frac{1}{2}\left(\frac{\partial \Delta u_k}{\partial x_i^0}\ \frac{\partial \delta\Delta u_k}{\partial x_j^0} + \frac{\partial \delta\Delta u_k}{\partial x_i^0}\ \frac{\partial \Delta u_k}{\partial x_j^0}\right) \end{aligned} \tag{13}$$

Similarly decompose the stress

$$\sigma_{ij}(t+\Delta t) = \sigma_{ij}(t) + \Delta\sigma_{ij} \tag{14}$$

Use Eqs. 12, 13 and 14 in Eq. 7 to obtain

$$\begin{aligned}\delta U = &\int_V \Delta\sigma_{ij}\ \delta\Delta\varepsilon_{ij}\ dV + \int_V \sigma_{ij}(t)\ \delta\Delta\varepsilon_{ij}\ dV + \\ &\int_V \underline{\sigma_{ij}(t)\ \delta\Delta\kappa_{ij}\ dV} + \int_V \Delta\sigma_{ij}\ \delta\Delta\kappa_{ij}\ dV\end{aligned} \tag{15}$$

The integration, as before, is over the original volume. This expression is exact for large deformations, large strains, and material non-linearities. For linear elastic materials, the incremental stress is related to the incremental *linear* strain by a constitutive relation of the form

$$\Delta\sigma_{ij} = D_{ijmn}\Delta\varepsilon_{mn}$$

D_{ijmn} has the general form $L^T\,D'\,L$ with D' containing the material constitution and L the composite ply angle transformation. The first term in Eq. 15 then becomes the standard linear finite element term. The second term is an incremental term involving the existing state of stress and linear strains. The third term, underlined, is the key term for rotorcraft. It is an incremental term involving the existing state of stress and the non-linear strains. This term produces the structural couplings between axial and transverse deformations (torsion is not a separate state of motion in 3-D but a function of transverse deformations) in response to rotational effects. It carries within it the classical extension-bending and flap-lag structural couplings. The fourth term is dropped as part of linearization, with the assumption $D_{ijmn}\varepsilon_{mn}\delta\kappa_{ij} \approx 0$. The final expression then becomes

$$\begin{aligned}\delta U = &\int_V D_{ijmn}\ \Delta\varepsilon_{mn}\ \delta\Delta\varepsilon_{ij}\ dV + \\ &\int_V \sigma_{ij}(t)\ \delta\Delta\varepsilon_{ij}\ dV + \int_V \sigma_{ij}(t)\ \delta\Delta\kappa_{ij}\ dV\end{aligned} \tag{16}$$

Iterations are required, of course, primarily for $\sigma_{ij}(t)$ but also for the linearization. The latter is usually insignificant. For example, for a static solution, given a prescribed, deformation-independent, non-inertial external forcing, the equation of motion takes the form

$$\delta U = \delta W_E$$

where δW_E is the external virtual work. The iterative procedure is then

$$\begin{aligned}&\int_V D_{ijmn}\ \Delta\varepsilon_{mn}\ \delta\Delta\varepsilon_{ij}\ dV + \int_V \sigma_{ij}(t)\ \delta\Delta\kappa_{ij}\ dV \\ &\qquad = \delta W_E - \int_V \sigma_{ij}(t)\ \delta\Delta\varepsilon_{ij}\ dV\end{aligned} \tag{17}$$

readily recognized as a Newton-Raphson iteration. If $\sigma_{ij}(t)$ is updated only on the right hand side, the procedure is a modified Newton iteration. For a rotor, it must be updated on both sides initially to obtain the correct non-linear stiffness. Thereafter, modified Newton iteration is enough for the purposes of airload non-linearities.

Kinetic Energy

The variation in kinetic energy or the virtual work by inertial forces is given by

$$\delta T = -\delta W_I = \int_V \rho\,\ddot{\vec{r}}\cdot\delta\vec{r}\ dV$$

where $\ddot{\vec{r}}$ and $\delta\vec{r}$ are the acceleration and virtual displacement of a material point P on the blade relative to an inertial frame I. Let P lie in a non-inertial frame B. The frames I and B are associated with corresponding coordinate axes or basis. At any instant, frame B has a displacement $\vec{x}^{BI}$ relative to the frame I and an orientation defined by a rotation matrix C^{IB}. C^{IB} defines the orientation of I relative to B, i.e. it rotates the axis from B to I. If the components of $\vec{x}^{BI}$ expressed in B and I basis are denoted by $x^{BI/B}$ and $x^{BI/I}$, then

$$x^{BI/I} = C^{IB}x^{BI/B}$$

Recall, that the time derivative of the rotation matrix is related to the skew symmetric angular velocities by

$$\dot{C}^{IB} = C^{IB}\tilde{\omega}^{BI/B} = -\tilde{\omega}^{IB/I}C^{IB}$$

where $\tilde{\omega}^{BI/B}$ is the angular velocity of B relative to I and measured in B, and $\tilde{\omega}^{IB/I}$ is the angular velocity of I relative to B and measured in I. The components of the motion of the point P relative to I and B and expressed in I and B frames satisfy

$$\begin{aligned}r^{PI/I} &= C^{IB}(x^{BI/B} + r^{PB/B}) \\ \dot{r}^{PI/I} &= C^{IB}(v^{BI/B} + \dot{r}^{PB/B} + \tilde{\omega}^{BI/B}r^{PB/B}) \\ \ddot{r}^{PI/I} &= C^{IB}(\dot{v}^{BI/B} + \tilde{\omega}^{BI/B}v^{BI/B} + \ddot{r}^{PB/B} + \\ &\quad 2\,\tilde{\omega}^{BI/B}\dot{r}^{PB/B} + \tilde{\omega}^{BI/B}\tilde{\omega}^{BI/B}r^{PB/B} + \\ &\quad \dot{\tilde{\omega}}^{BI/B}r^{PB/B})\end{aligned} \tag{18}$$

where the frame motions have been expressed in body-fitted coordinates as $\dot{x}^{BI/I} = v^{BI/I} = C^{IB}v^{BI/B}$ and $\ddot{x}^{BI/I} = C^{IB}\dot{v}^{BI/B} + C^{IB}\tilde{\omega}^{IB/B}v^{BI/B}$. The components of virtual displacement are

$$\begin{aligned}\delta r^{PI/I} &= C^{IB}(\delta x^{BI/B} + \delta\tilde{\psi}^{BI/B}r^{PB/B} + \delta r^{PB/B}) \\ &= C^{IB}(\delta x^{BI/B} - \tilde{r}^{PB/B}\delta\psi^{BI/B} + \delta r^{PB/B}) \\ &= C^{IB}R^T\delta q\end{aligned} \tag{19}$$

where

$$R = \begin{pmatrix} I \\ -\tilde{r}^{PB/B} \\ R_b^T \end{pmatrix} \qquad \delta q = \begin{pmatrix} \delta x^{BI/B} \\ \delta\psi^{BI/B} \\ \delta q_b \end{pmatrix}$$

$x^{BI/B}$ and $\psi^{BI/B}$ are the rigid body translational and rotational states of frame B. The virtual displacement

$\delta r^{PB/B}$ in B frame is written in terms of the variations in finite element degrees of freedom $\delta r^{PB/B} = R_b^T \delta q_b$. The kinetic energy is then

$$\delta T = \int_V \rho \, (\delta r^{PI/I})^T \, \ddot{r}^{PI/I} \, dV \tag{20}$$

In general, frame B may contain a flexible component. For a simple illustration, consider B to be the undeformed blade rotating frame, containing the entire blade, with origin at the rotor hub. It undergoes control motions $\theta, \dot{\theta}, \ddot{\theta}$ with respect to another rotating frame with origin at the hub. This frame undergoes rotational motions $\Omega, \dot{\Omega}, \ddot{\Omega}$ with respect to a non-rotating inertial frame I at the hub. B has no rigid body states with respect to I. Thus $\delta x^{BI/B} = 0$, $\delta\psi^{BI/B} = 0$, and $v^{BI/B} = \dot{v}^{BI/B} = 0$. It follows

$$C^{IB} = \begin{bmatrix} c_\psi & -s_\psi c_\theta & s_\psi s_\theta \\ s_\psi & c_\psi c_\theta & -c_\psi s_\theta \\ 0 & s_\theta & c_\theta \end{bmatrix} \tag{21}$$

where $\psi = \Omega t$ is the blade azimuth angle, θ is the instantaneous control angle, and $c_\theta = \cos\theta, s_\theta = \sin\theta$ etc.

$$\omega^{BI/B} = \begin{pmatrix} \dot{\theta} \\ 0 \\ 0 \end{pmatrix}^T e^B + \begin{pmatrix} 0 \\ 0 \\ \Omega \end{pmatrix}^T e^I = \begin{pmatrix} \dot{\theta} \\ \Omega s_\theta \\ \Omega c_\theta \end{pmatrix}^T e^B \tag{22}$$

e^B and e^I are the unit basis vectors in the B and I frame respectively. The non-zero components of $\ddot{r}^{PI/I}$ in the kinetic energy expression Eq. 20 then become

$$\ddot{r}^{PB/B} = \begin{pmatrix} \ddot{u}_1 \\ \ddot{u}_2 \\ \ddot{u}_3 \end{pmatrix} \tag{23}$$

$$2\tilde{\omega}^{BI/B}\dot{r}^{PB/B} = 2 \begin{bmatrix} 0 & -\Omega c_\theta & \Omega s_\theta \\ \Omega c_\theta & 0 & -\dot{\theta} \\ -\Omega s_\theta & \dot{\theta} & 0 \end{bmatrix} \begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \end{pmatrix} \tag{24}$$

$$\tilde{\omega}^{BI/B} \, \tilde{\omega}^{BI/B} \, r^{PB/B} = \begin{bmatrix} -\Omega^2 & \dot{\theta}\Omega s_\theta & \dot{\theta}\Omega c_\theta \\ \dot{\theta}\Omega s_\theta & -\Omega^2 c_\theta^2 - \dot{\theta}^2 & \Omega^2 c_\theta s_\theta \\ \dot{\theta}\Omega c_\theta & \Omega^2 c_\theta s_\theta & -\Omega^2 s_\theta^2 - \dot{\theta}^2 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{25}$$

$$\dot{\tilde{\omega}}^{BI/B} r^{PB/B} = \begin{bmatrix} 0 & -\dot{\Omega} c_\theta + \Omega\dot{\theta} s_\theta & \dot{\Omega} s_\theta + \Omega\dot{\theta} c_\theta \\ \dot{\Omega} c_\theta - \Omega\dot{\theta} s_\theta & 0 & \ddot{\theta} \\ -\dot{\Omega} s_\theta - \Omega\dot{\theta} c_\theta & \ddot{\theta} & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{26}$$

The virtual displacement is simply the variation of the incremental displacements

$$\delta r^{PB/B} = \delta u; \qquad \delta r^{PI/I} = C^{IB} \delta u \tag{27}$$

Thus, the kinetic energy Eq. 20 takes the following form

$$\delta T = \int_V \delta u^T \, \rho \, (T_m \, \ddot{u} + T_c \, \dot{u} + T_k \, u + T_f) dV \tag{28}$$

With the simplest assumption of only a collective, and steady rotation, we have from Eq. 23–Eq. 26 the following mass, damping, stiffness, and force contributions from inertia.

$$\begin{aligned} T_m &= I_3 \\ T_c &= -2\Omega \begin{bmatrix} 0 & c_\theta & -s_\theta \\ -c_\theta & 0 & 0 \\ s_\theta & 0 & 0 \end{bmatrix} \\ T_k &= -\Omega^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\theta^2 & -s_\theta c_\theta \\ 0 & -s_\theta c_\theta & s_\theta^2 \end{bmatrix} \\ T_f &= T_k \, [\, x_1^0 \, x_2^0 \, x_3^0 \,]^T \end{aligned} \tag{29}$$

Virtual Work by External Forces

A surface lies, always, on one or more of the element faces, defined by its natural coordinates $\xi \pm 1$, $\eta \pm 1$, or $\zeta \pm 1$ (see following section). A differential change $d\xi$ in the natural coordinates (ξ, η, ζ) creates the following changes in the geometric coordinates (x_1, x_2, x_3).

$$dx_i = x_{i,\xi} d\xi = \sum_{k=1}^{N} H_{k,\xi} x_i^k \qquad i = 1, 2, 3 \tag{30}$$

Denote the vector $[x_{1,\xi}, x_{2,\xi}, x_{3,\xi}]$ as $x_{,\xi}$. Similarly, differential changes $d\eta$ and $d\zeta$ create the vectors $x_{i,\eta} d\eta$ and $x_{i,\zeta} d\zeta$, $i = 1, 2, 3$, in the geometric coordinates. The area $d\xi d\eta$, for example, then corresponds to the area of a parallelogram between the two vectors $x_{,\xi} d\xi$ and $x_{,\eta} d\eta$ in the geometric domain, defined by their cross product or cross product matrices: $x_{,\xi} \times x_{,\eta} = -\tilde{x}_{,\xi} x_{,\eta} = x_{,\xi} \tilde{x}_{,\eta}$

$$x_{,\xi} \times x_{,\eta} = \begin{bmatrix} x_{2,\xi} x_{3,\eta} - x_{2,\eta} x_{3,\xi} \\ x_{3,\xi} x_{1,\eta} - x_{3,\eta} x_{1,\xi} \\ x_{1,\xi} x_{2,\eta} - x_{1,\eta} x_{2,\xi} \end{bmatrix}$$

For example, if normal pressure p on an elemental face defined by ζ =constant produces a virtual work $p \, d\vec{A} \cdot \delta\vec{u}$, the components of $d\vec{A}$ are simply $x_{,\xi} \tilde{x}_{,\eta} \, d\xi \, d\eta$. The components of $\delta\vec{u}$ are the virtual deformations $\delta[u_1 \, u_2 \, u_3]^T = \delta q^T H^T$, where H are structural shape functions (see next section). The virtual work expression is then

$$\delta W_E = \delta q^T \int_A p H^T \; x_{,\xi} \tilde{x}_{,\eta} \; d\xi \, d\eta = \delta q^T \, Q \tag{31}$$

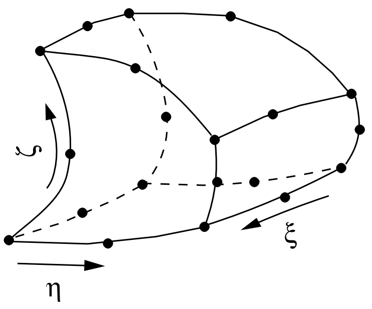
A general surface stress distribution σ_S is incorporated in exactly the same manner using $(\sigma_S \cdot d\vec{A}) \cdot \delta\vec{u}$ as virtual work.

$$\delta W_E = \delta q^T \int_A H^T \, \sigma_S \; x_{,\xi} \tilde{x}_{,\eta} \; d\xi \, d\eta = \delta q^T \, Q \tag{32}$$

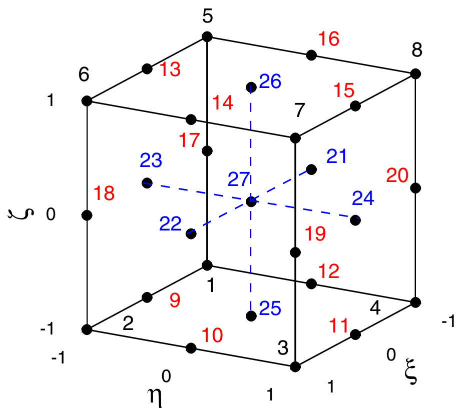
Because fluid stress are deformation dependent, the area must be evaluated at the deformed configuration. The deformed configuration is not known a priori, therefore, iterations are required. This is discussed further in the subsection 'Steady Hover Prototype' under '3-D FEM Rotor Analysis'.

Brick Finite Elements

The analysis of bending dominated problems involving thin structures using 3-D elements suffer from severe stiffening known as *element locking* as the element thickness tends to zero. A simple but effective way to prevent locking is to use higher-order elements – as in this study – containing sufficient number of internal nodes. Devising more efficient lower-order locking-free brick elements, based on reduced-integration or Enhanced Assumed Strain methods are beyond the scope of this initial development. The main concern at present is accuracy.



(a) Element in physical coordinates (only edge nodes shown)



(b) Element in natural coordinates (all nodes shown)

Figure 1: **27-node isoparametric, hexahedral brick element in curvilinear natural coordinates; $4 \times 4 \times 4$ Gauss integration points.**

Figure 1 shows an isoparametric, hexahedral, quadratic brick element developed in this study. It consists of 8 vertex nodes and 19 internal nodes – 12 edge nodes, 6 face nodes, and 1 volume node. Within isoparametric elements, geometry and displacement solution are both interpolated using the same shape functions. Thus

$$\begin{aligned} x_i^0 &= \sum_{a=1}^{N} H_a \; x_{i\,a}^0; \qquad \Delta u_i = \sum_{a=1}^{N} H_a \; \Delta u_{i\,a} \\ x_i &= \sum_{a=1}^{N} H_a \; x_{i\,a}; \qquad u_i = \sum_{a=1}^{N} H_a \; u_{i\,a} \end{aligned} \tag{33}$$

where a is the elemental node point index. $N = 27$. The shape functions are expressed in element natural axes ξ, η, and ζ, where $-1 \le \xi, \eta, \zeta \le 1$. We consider Lagrange polynomials in each direction.

$$H_a = H_a(\xi, \eta, \zeta) = L_I^n(\xi)\; L_J^m(\eta)\; L_K^p(\zeta) \tag{34}$$

with $n = m = p = 2$; and each of I, J, K vary as $1, 2, 3$ respectively. The second-order polynomials in, say η, are $\eta(\eta-1)/2$, $1-\eta^2$, and $\eta(\eta+1)/2$. Based on the local node ordering shown in Fig. 1(b), we have for example the shape function corresponding to node 11

$$H_{11} = L_2^2(\xi)\; L_3^2(\eta)\; L_1^2(\zeta) = \frac{1}{4}\,\eta\,\zeta(1-\xi^2)\,(\eta+1)\,(\zeta-1)$$

The strains require the derivatives of the shape functions with respect to geometric coordinates

$$\frac{\partial u_i}{\partial x_j^0} = \sum_{a=1}^{N} \left(\frac{\partial H_a}{\partial x_j^0} \right) u_{i\,a}; \qquad \frac{\partial \Delta u_i}{\partial x_j^0} = \sum_{a=1}^{N} \left(\frac{\partial H_a}{\partial x_j^0} \right) \Delta u_{i\,a} \tag{35}$$

These are calculated from the derivatives with respect to natural coordinates as follows

$$\begin{pmatrix} \frac{\partial H_a}{\partial \xi} \\ \frac{\partial H_a}{\partial \eta} \\ \frac{\partial H_a}{\partial \zeta} \end{pmatrix} = \underbrace{\begin{bmatrix} \frac{\partial x_1^0}{\partial \xi} & \frac{\partial x_2^0}{\partial \xi} & \frac{\partial x_3^0}{\partial \xi} \\ \frac{\partial x_1^0}{\partial \eta} & \frac{\partial x_2^0}{\partial \eta} & \frac{\partial x_3^0}{\partial \eta} \\ \frac{\partial x_1^0}{\partial \zeta} & \frac{\partial x_2^0}{\partial \zeta} & \frac{\partial x_3^0}{\partial \zeta} \end{bmatrix}}_{J} \begin{pmatrix} \frac{\partial H_a}{\partial x_1^0} \\ \frac{\partial H_a}{\partial x_2^0} \\ \frac{\partial H_a}{\partial x_3^0} \end{pmatrix} \tag{36}$$

The evaluation of the Jacobian, J, is straight forward using the derivatives of the shape functions with respect to element natural axes and the location of the nodal points (i.e. the grid points), i.e., the first of Eq. 33

$$J = \begin{bmatrix} H_{1,\xi} & H_{2,\xi} & \dots & H_{N,\xi} \\ H_{1,\eta} & H_{2,\eta} & \dots & H_{N,\eta} \\ H_{1,\zeta} & H_{2,\zeta} & \dots & H_{N,\zeta} \end{bmatrix} \begin{bmatrix} x_{11}^0 & x_{21}^0 & x_{31}^0 \\ x_{12}^0 & x_{22}^0 & x_{32}^0 \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ x_{1N}^0 & x_{2N}^0 & x_{3N}^0 \end{bmatrix} \tag{37}$$

The required derivatives are then

$$\begin{pmatrix} \frac{\partial H_a}{\partial x_1^0} \\ \frac{\partial H_a}{\partial x_2^0} \\ \frac{\partial H_a}{\partial x_3^0} \end{pmatrix} = \frac{1}{J} \begin{pmatrix} \frac{\partial H_a}{\partial \xi} \\ \frac{\partial H_a}{\partial \eta} \\ \frac{\partial H_a}{\partial \zeta} \end{pmatrix} \tag{38}$$

$$B_{L0} = \begin{bmatrix} H_{1,1} & 0 & 0 & H_{2,1} & \dots & 0 \\ 0 & H_{1,2} & 0 & 0 & \dots & 0 \\ 0 & 0 & H_{1,3} & 0 & \dots & H_{N,3} \\ H_{1,2} & H_{1,1} & 0 & H_{2,2} & \dots & 0 \\ 0 & H_{1,3} & H_{1,2} & 0 & \dots & H_{N,2} \\ H_{1,3} & 0 & H_{1,1} & H_{2,3} & \dots & H_{N,1} \end{bmatrix}$$

$$B_{L1} = \begin{bmatrix} l_{11}H_{1,1} & l_{21}H_{1,1} & l_{31}H_{1,1} & l_{11}H_{2,1} & \dots & l_{31}H_{N,1} \\ l_{12}H_{1,2} & l_{22}H_{1,2} & l_{32}H_{1,2} & l_{12}H_{2,2} & \dots & l_{32}H_{N,2} \\ l_{13}H_{1,3} & l_{23}H_{1,3} & l_{33}H_{1,3} & l_{13}H_{2,3} & \dots & l_{33}H_{N,3} \\ l_{11}H_{1,2}+l_{12}H_{1,1} & l_{21}H_{1,2}+l_{22}H_{1,1} & l_{31}H_{1,2}+l_{32}H_{1,1} & l_{11}H_{2,2}+l_{12}H_{2,1} & \dots & l_{31}H_{N,2}+l_{32}H_{N,1} \\ l_{12}H_{1,3}+l_{13}H_{1,2} & l_{22}H_{1,3}+l_{23}H_{1,2} & l_{32}H_{1,3}+l_{33}H_{1,2} & l_{12}H_{2,3}+l_{13}H_{2,2} & \dots & l_{32}H_{N,3}+l_{33}H_{N,2} \\ l_{11}H_{1,3}+l_{13}H_{1,1} & l_{21}H_{1,3}+l_{23}H_{1,1} & l_{31}H_{1,3}+l_{33}H_{1,1} & l_{11}H_{2,3}+l_{13}H_{2,1} & \dots & l_{31}H_{N,3}+l_{33}H_{N,1} \end{bmatrix}$$

Henceforth the above derivatives are denoted as $H_{a,1}, H_{a,2}$, and $H_{a,3}$. In addition, the first quantity in Eq. 35 is denoted by l_{ij}, i.e.,

$$l_{ij} = \sum_{a=1}^{N} H_{a,j}\ u_{i\,a}$$

From the linear strain $\Delta\varepsilon_{ij}$ as defined in Eq. 11, the strain-displacement relation now takes the following form

$$\Delta\hat{\varepsilon} = (B_{L0} + B_{L1})\Delta q \tag{39}$$

where

$$\begin{aligned} \Delta\hat{\varepsilon}^T &= \Delta[\varepsilon_{11}\ \ \varepsilon_{22}\ \ \varepsilon_{33}\ \ 2\varepsilon_{12}\ \ 2\varepsilon_{23}\ \ 2\varepsilon_{13}] \\ \Delta q^T &= [u_{11}\, u_{21}\, u_{31}\ \ u_{12}\, u_{22}\, u_{32}\ \ \dots\ \ u_{1N}\, u_{2N}\, u_{3N}] \end{aligned} \tag{40}$$

and the expressions for B_{L0} and B_{L1} are given above. The first term in the strain energy Eq. 16 then becomes

$$\delta\Delta q^T \left(\int_V (B_{L0} + B_{L1})^T\ D\ (B_{L0} + B_{L1})\ dV \right) \Delta q \tag{41}$$

The second term is treated is the same manner to obtain

$$\delta\Delta q^T \left(\int_V (B_{L0} + B_{L1})^T\ \hat{\sigma}(t)\ dV \right) \Delta q \tag{42}$$

where $\hat{\sigma}(t) = [\sigma_{11}\ \ \sigma_{22}\ \ \sigma_{33}\ \ \sigma_{12}\ \ \sigma_{23}\ \ \sigma_{13}]^T$. Consider the non-linear incremental strain $\Delta\kappa_{ij}$ as defined in Eq. 11. It is directly in a quadratic form, and hence the third term can be re-arranged as

$$\delta\Delta q^T \left(\int_V \bar{B}_{NL}^T\ \bar{\sigma}(t)\ \bar{B}_{NL}\ dV \right) \Delta q \tag{43}$$

with the matrices having special forms

$$\bar{\sigma}(t) = \begin{bmatrix} \sigma_{ij}(t) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_{ij}(t) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sigma_{ij}(t) \end{bmatrix};\quad \mathbf{0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\bar{B}_{NL} = \begin{bmatrix} B_{NL} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & B_{NL} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & B_{NL} \end{bmatrix};\quad \mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$B_{NL} = \begin{bmatrix} H_{1,1} & 0 & 0 & H_{2,1} & 0 & 0 & \dots & H_{N,1} \\ H_{1,2} & 0 & 0 & H_{2,2} & 0 & 0 & \dots & H_{N,2} \\ H_{1,3} & 0 & 0 & H_{2,3} & 0 & 0 & \dots & H_{N,3} \end{bmatrix}$$

The volume integrations are performed using 4 Gauss points along each natural coordinate axes, a total of 64 integration points. Note that

$$dV = \det(J)\ d\xi\ d\eta\ d\zeta$$

VERIFICATION OF 3-D FEM

A preliminary verification of the 3-D FEM analysis is carried out by reproducing non-rotating plate and rotating beam frequencies. The former verifies the locking-free behavior. The later verifies the non-linear implementation.

Here, and henceforth, a 3-D grid is denoted as $n_1 \times n_2 \times n_3$ or (n_1, n_2, n_3). They denote the number of elements along span, chord, and thickness, respectively.

Thin Plate Frequencies

The locking-free behavior of the brick elements, in shear, is verified by re-producing classical Kirchhoff thin plate frequencies for a square cantilevered plate.

The plate is modeled using a $4 \times 4 \times 4$ brick grid (Fig. 2), i.e., there are 4 layers of bricks across thickness. Starting from a solid cube, the variation in natural frequencies with a gradual reduction in thickness is shown in Fig. 3. It is clear that the frequencies approach those of a thin plate. The plate frequencies are obtained from converged rectangular plate finite elements and are validated easily with classical solutions [24]. The discrepancy at the higher modes are mostly resolved with a finer mesh converged solution (Table 1). The remaining differences are due to shear, not present in the Kirchhoff solution, but present in the brick solution.

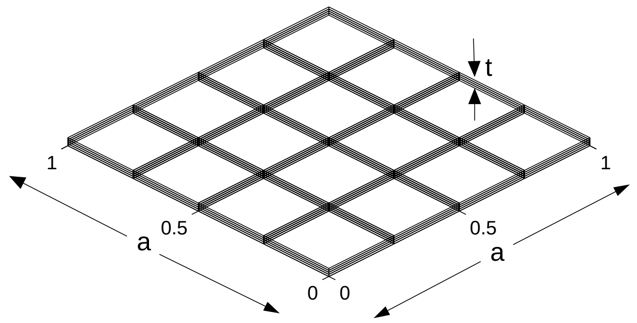


Figure 2: **A cantilevered square plate with $4 \times 4 \times 4$ brick element grid; thickness $t = 1$ in; dimensions $a = 39.4$ in**

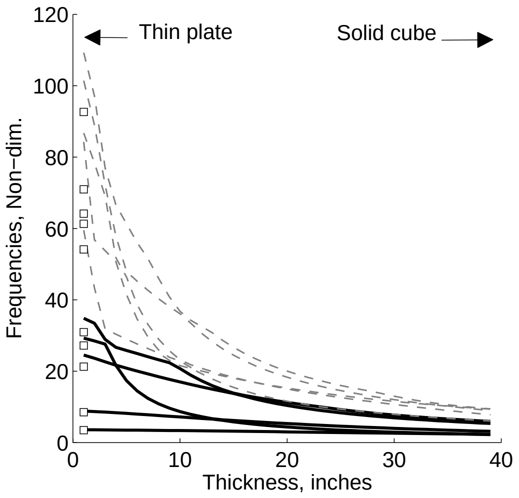


Figure 3: **Natural frequencies of a solid cube approaching Kirchhoff thin plate frequencies (symbols) with reduced thickness**

Mode number	$4 \times 4 \times 4$ bricks	$8 \times 8 \times 4$ bricks	Kirchhoff plate
1	3.55	3.50	3.47
2	8.68	8.53	8.51
3	22.93	21.58	21.29
4	28.16	27.29	27.19
5	32.84	31.19	30.96
6	56.78	54.31	54.13
7	71.19	63.09	61.29
8	74.75	64.96	64.16
9	83.68	72.50	70.98

Table 1: **Plate frequencies using 3-D FEM: nondimensionalized w.r.t. $\sqrt{D/\rho t a^4}$, a: square plate dimension, t: thickness, ρ: density, $D = Et^3/12(1-\nu^2)$, E: Young's Modulus and ν: Poisson's ratio**

section grid	Torsion 1	Torsion 2
1×1	1.710	5.132
2×2	1.586	4.758
3×3	1.577	4.733
4×4	1.576	4.728
5×5	1.575	4.726

Table 2: **Non-rotating beam torsion frequencies vs. cross-section grid refinement; 8 span-wise elements; nondimensionalized w.r.t. $\sqrt{GJ/IL^2}$; EB values are 1.571 and 4.712; beam dimension:** $100c \times c \times c$

Thickness t	$n_1 = 8$	$n_1 = 16$	$n_1 = 20$
c	4.733	4.726	4.725
c/2	4.757	4.746	4.742
c/4	4.824	4.794	4.784
c/8	4.886	4.839	4.824

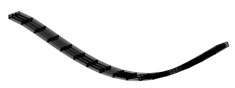
Table 3: **Non-rotating second torsion frequency vs. span-wise grid refinement; grids are $n_1 \times 3 \times 3$; nondimensionalized w.r.t. $\sqrt{GJ/IL^2}$; EB value is** 4.712**; beam dimension:** $100c \times c \times t$

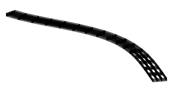
Slender Beam Frequencies

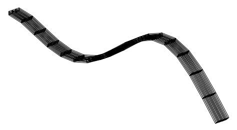
Next, the brick element is verified using a slender geometry that behaves as a beam over a large variation of thickness and aspect ratios. The bending frequencies are easy to re-produce, we focus on torsion. Consider a uniform beam of aspect ratio 100 and a square cross-section, i.e., dimensions of $100c \times c \times c$, in length, width, and thickness. The torsion frequencies converge towards Euler-Bernoulli (EB) values with 4 to 9 cross-section elements (Table 2) for this simple geometry. The remaining difference stems from span-wise resolution.

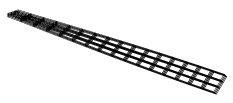
The effect of span-wise resolution is shown in the first row of Table 3, starting from $8 \times 3 \times 3$ grid as the baseline. The higher (second) torsion frequency is shown. Span-wise resolution becomes more important as the beam thickness is reduced. This is shown in the subsequent rows and columns of Table 3. The rows show the variation of torsion frequency with a progressively thinner beam. The columns show the effect of span-wise refinement for each thickness. There is increased deviation from Euler-Bernoulli values as the thickness reduces. With thickness fixed at $c/4$ and grid at $16 \times 3 \times 3$, the aspect ratio of the beam is now progressively reduced. Table 4 shows that from 100 to 20 the frequencies remain nominally constant. At aspect ratio 5 there is still only an error of $5-6\%$. This deviation stems clearly from the short aspect ratio of the problem – even though its exact nature and source is not studied here in detail.

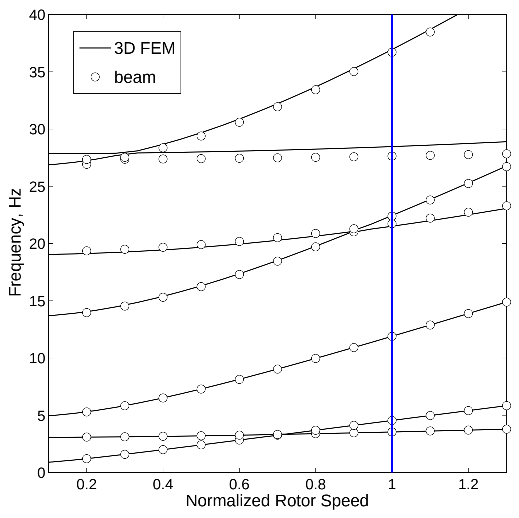
The configuration with aspect ratio 20, i.e. dimensions $20c \times c \times c/4$, is now used to compare the rotating frequencies. The material modulii are $E = 8.2700 \times 10^7$


(a) Mode 3: Flap 2


(b) Mode 4: Lag 1


(c) Mode 5: Flap 3


(d) Mode 6: Torsion 1



(e) Fan plot showing rotating frequencies

Figure 4: **Rotating frequencies and mode shapes of an uniform soft in-plane hingeless rotor of aspect ratio 20, thickness 25% chord; 3-D bricks vs 1-D beam;** $16 \times 3 \times 3$ **grid**

Aspect Ratio	Torsion 1	Torsion 2
100	1.598	4.794
40	1.599	4.799
20	1.603	4.813
15	1.606	4.826
10	1.615	4.860
8	1.622	4.890
6	1.635	4.919
5	1.645	4.991
4	1.662	5.064
3	1.794	5.478

Table 4: **Non-rotating torsion frequencies vs. aspect ratio;** $16 \times 3 \times 3$ **grid; nondimensionalized w.r.t.** $\sqrt{GJ/IL^2}$; EB values remain 1.571 and 4.712 for all aspect ratios.

Pa and $G = 3.4458 \times 10^7$ Pa (Poisson's ratio $\nu = 0.2$), density is $\rho = 192.2208$ kg/m^3, and the dimension $c = 0.0864$ m. The rotation axis is at mid-chord.

A few of the key rotating modes are shown in Figs. 4(a)– 4(d). The frequency plot, Fig. 4(e), shows the the beam frequencies are almost exactly reproduced by 3-D FEM – and the small grid size of $16 \times 3 \times 3$ is adequate for this simple problem. Note that this serves as a verification of the non-linear formulation. The torsion frequency shows an error of 5% consistent with the deviation in the non-rotating case at this level of grid refinement. The torsion frequency is relatively high for this structure, and occurs only as the sixth mode. The rotating frequencies for $\Omega = 27$ rad/s are tabulated in Table 5.

Mode no.	Beam freqs. (/rev)	3-D freqs. (/rev)	Mode type
1	0.826	0.824	Lag 1
2	1.059	1.058	Flap 1
3	2.768	2.769	Flap 2
4	5.058	5.006	Lag 2
5	5.211	5.223	Flap 3
6	6.431	6.625	Torsion 1
7	8.542	8.597	Flap 4

Table 5: **Rotating frequencies for a soft in-plane hingeless rotor;** $16 \times 3 \times 3$ **grid in 3-D**

ITERATIVE SUBSTRUCTURING USING PARALLEL KRYLOV SOLVER

The parallel Krylov solver developed in this study to provide an efficient and scalable 3-D FEM solution is described in this section.

The effectiveness of the method of substructures in solving large scale problems is mentioned already in the introduction. Each substructure (or subdomain) can use a solver tailored to its local condition number. Most often, for reasons cited in the introduction, the substructures are required to be solved using direct factorization. This is the approach taken in the present paper. Iterative substructuring then provides a preconditioned iterative solver for the substructure interface problem.

The interface problem is more amenable to an iterative solver, unlike the substructures themselves, as its condition number (ratio of maximum to minimum eigenvalues) grows with substructure mesh resolution at a rate that is one order lower compared to the original problem

— i.e. by $O(h^{-1})$ for 2nd order and by $O(h^{-3})$ for 4th order PDEs where h is a typical mesh resolution within each substructure. However, it also grows necessarily at $O(H^{-1})$ where H is an average subdomain size. Indeed, for a 2nd order, elliptic, positive definite, and coercive operator, and an uniform finite element meshing, the precise condition number κ of the interface S is proven to be [25]

$$\kappa(S) \leq c\frac{H}{h\,H_m^2}$$

where H is the maximum and H_m is the minimum subdomain size. c is a constant independant of h, H, and H_m. The objective of iterative substructuring is to provide parallel preconditioning methods such that the preconditioned interface problem has a condition number independent of both h and H. Such a preconditioner is called an *optimal preconditioner*.

The dependence on $0(H_m^{-2})$ cannot be removed without a higher level coarse problem – a mechanism to propagate local substructure information globally. Communication only between neighboring subdomains will always show this dependence. In FETI-DP, the coarse problem is constructed out of a selected set of substructure corner nodes.

The building blocks of a preconditioned Krylov solver (to solve $M^{-1}Ax = M^{-1}b$, where M is the preconditioner) are: (1) residual calculation $r = b - Ax$, (2) preconditioning $M^{-1}r$ and, (3) a general matrix-vector multiplication procedure $A\,v$. An iterative substructuring algorithm computes these building blocks in a parallel manner. Once the building blocks are provided, constructing an iterative update (Krylov update) using these blocks is straight forward. The blocks and the update then form the Krylov solver. Unlike a simple Conjugate Gradient (CG) update, a Generalized Minimum Residual (GMRES) update, however, poses a few parallelization issues of its own.

The iterative substructuring algorithm, its numerical scalability, and the parallelization issues of the CG and GMRES updates are documented below.

The FETI-DP Algorithm

For a 3-D substructure, each interface node can be a face, edge, or vertex node. Of these, the edge and vertex nodes — that are common to more than two substructures — are designated as corner nodes. The degrees of freedom (DOFs) associated with the corner nodes are formulated as a primal interface. The rest are formulated as a dual interface. The corner nodes form the coarse problem, and, are key to ensuring optimal scalability.

The number of DOFs associated with a corner depend on the order of the problem, e.g., 3 for 2nd order brick FEM or 6 for 4th order plate or shell elements. Thus, it renders the coarse mesh automatically denser with increase in order. The FETI-DP method and its implementation follows entirely the work of Refs. [15, 16]. A detailed exposition of our implementation is not provided, only a brief description of the key components are summarized below.

If the nodes of a subdomain are re-ordered with internals first (I^s), followed by the interface edges and faces (Γ_E^s), and then the corners (Γ_C^s), where the subscript s denotes subdomain quantities, then a subdomain matrix, say the stiffness matrix, takes the following form

$$K^s = \begin{bmatrix} K_{II}^s & K_{IE}^s & K_{IC}^s \\ K_{EI}^s & K_{EE}^s & K_{EC}^s \\ K_{CI}^s & K_{CE}^s & K_{CC}^s \end{bmatrix} = \begin{bmatrix} K_{RR}^s & K_{RC}^s \\ K_{CR}^s & K_{CC}^s \end{bmatrix} \tag{44}$$

where the internal and edge nodes are denoted together as R^s nodes. The subdomain forcing, f^s, and unknowns, u^s, are correspondingly

$$f^s = \begin{pmatrix} f_R^s \\ f_C^s \end{pmatrix} \qquad u^s = \begin{pmatrix} u_R^s \\ u_C^s \end{pmatrix} \tag{45}$$

with

$$u_R^s = \begin{pmatrix} u_I^s \\ u_E^s \end{pmatrix} \qquad f_R^s = \begin{pmatrix} f_I^s \\ f_E^s \end{pmatrix} \tag{46}$$

Two Boolean restrictions are defined for each subdomain. The first Boolean restriction, B_R^s, restricts u_R^s to u_E^s, and assigns a $+1$ or -1 sign such that equality of the interface degrees of freedom are guaranteed upon convergence.

$$\sum_s B_R^s u_R^s = 0$$

The summation sign denotes assembly over sub domains. The second Boolean restriction, B_C^s, restricts the global corner nodes to subdomain corners. Note that, for a re-ordered subdomain, the first restriction, B_R^s has the form and size

$$\begin{array}{ccc} & \longleftarrow I^s \longrightarrow \quad \longleftarrow \Gamma_E^s \longrightarrow & \\ B_R^s = & \left[\begin{array}{ccc|c} 0 & & & \\ & \dots & & B_E^s \\ & & 0 & \end{array}\right] & \begin{array}{c}\uparrow \\ \Gamma_E^s \\ \downarrow\end{array} \end{array} \tag{47}$$

where B_E^s is a diagonal matrix with entries $+1$ or -1. The dual-primal procedure computes a set of dual variables associated with the interface which on convergence allows the recovery of the subdomain internal and edge DOFs u_R^s as

$$K_{RR}^s u_R^s = f_R^s - {B_R^s}^T \lambda^s - K_{RC}^s B_C^s u_C^g \tag{48}$$

and all the global corner DOFs u_C^g as

$$K_{CC}^* \; u_C^g = F_{CR}\lambda + f_C^* \tag{49}$$

where λ and λ^s are the dual variables and their subdomain restrictions. The corner problem, i.e. the coarse grid problem, is also constructed subdomain by subdomain. Formally,

$$\begin{aligned} K_{CC}^* &= \sum_s {B_C^s}^T \left[K_{CC}^s - K_{CR}^s {K_{RR}^s}^{-1} K_{RC}^s \right] B_C^s \\ F_{CR}\lambda &= \sum_s {B_C^s}^T K_{CR}^s {K_{RR}^s}^{-1} {B_R^s}^T \lambda^s \\ f_C^* &= \sum_s {B_C^s}^T \left[f_C^s - K_{CR}^s {K_{RR}^s}^{-1} f_R^s \right] \end{aligned} \tag{50}$$

The solve, however, is carried out in every subdomain. Thus, before the interface iterations begin, the subdomain contributions to K^*_{CC} are constructed and factorized in every subdomain, and communicated globally. Thereafter, during the Krylov iterations, the coarse problem is only a repeated right hand side solve.

The building blocks of the Krylov solver are briefly stated below.

Residual calculation

The residual is calculated in two parts from Eq. 48.

$$r = \sum_s B^s_R u^s_R = r_1 + r_2 \tag{51}$$

The first part r_1 is calculated as

$$r_1 = \sum_s B^s_R {K^s_{RR}}^{-1} f^s_R - \sum_s B^s_R {K^s_{RR}}^{-1} {B^s_R}^T \lambda^s \tag{52}$$

The second part r_2 is calculated after the coarse solve

$$r_2 = -\sum_s B^s_R {K^s_{RR}}^{-1} K^s_{RC} B^s_C u^g_C \tag{53}$$

Using Eq. 49, the total residual then takes the form $d - F\lambda$. Note that the residual calculation requires ${K^s_{RR}}^{-1}$. Therefore, during subdomain partitioning, the corner node selection must ensure null kernels.

Preconditioner

The subdomain residuals, r^s, are used to construct subdomain fluxes

$$\eta^s = K^s_{EI} w^s + K^s_{EE}\, B^s_E r^s \tag{54}$$

with w^s obtained using subdomain Dirichlet solves

$$w^s = -{K^s_{II}}^{-1} K^s_{IE}\, B^s_E r^s \tag{55}$$

from which the preconditioned residual is generated

$$M^{-1} r = \sum_s B^s_E \eta^s \tag{56}$$

Expanding the above expressions, we have, formally

$$M^{-1} = \sum_s B^s_R \begin{bmatrix} 0 & 0 \\ 0 & S^s_{EE} \end{bmatrix} {B^s_R}^T \tag{57}$$

where S^s_{EE} are the subdomain Schur complement matrices. A more efficient preconditioner (but not optimal) is obtained by skipping the Dirichlet solve above and calculating the fluxes directly as

$$\eta_s = K^s_{EE}\, B^s_E r^s$$

This leads formally to

$$M^{-1} = \sum_s B^s_R \begin{bmatrix} 0 & 0 \\ 0 & K^s_{EE} \end{bmatrix} {B^s_R}^T \tag{58}$$

where the Schur complement matrices have been approximated by their leading terms. The two preconditioners above are called the Dirichlet and Lumped preconditioners. All results shown in this paper use the Dirichlet preconditioner, even though for 3-D brick problems the Lumped preconditioner is obviously more efficient.

Matrix-vector multiplication

This is identical to the residue calculation, except u^s_R is now calculated in Eq. 48 using ${B^s_R}^T v^s$ on the right hand side, instead of $f^s_R - {B^s_R}^T \lambda^s$. v^s is the subdomain restriction of a global vector v that is to be multiplied.

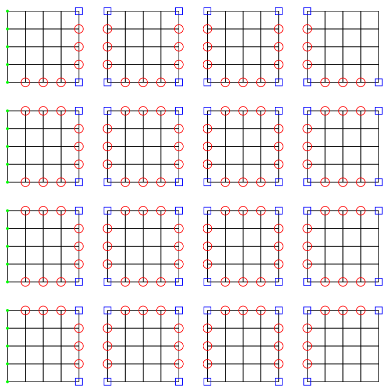


Figure 5: **A 4×4 plate partitioning; 16 elements in each partition; $H = 1/4, h = 1/16$; Interface and corner nodes shown in circles and squares respectively.**

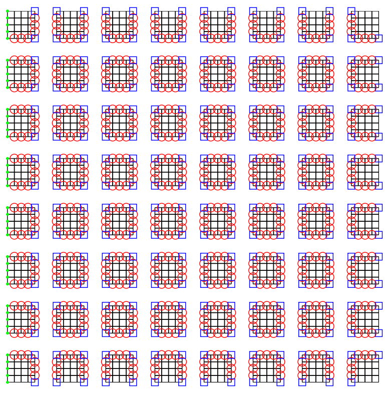


Figure 6: **A 8×8 plate partitioning; 16 elements in each partition; $H = 1/8, h = 1/32$; Interface and corner nodes shown in circles and squares respectively.**

Numerical scalability

For a symmetric and coercive elliptic operator the condition number of the preconditioned FETI-DP interface problem can be shown to grow as

$$\kappa(S) = O\left(1 + \log\frac{H}{h}\right)^m; \qquad \text{where} \quad m \le 3$$

If the subdomains have size H, and the finite element mesh has size h, then the condition number of the interface does not grow with the number of subdomains as long as the mesh within each subdomain is refined to keep H/h constant. This is the definition of optimal numerical scalability. Thus, a bigger problem with additional subdomains require the same iteration count as a smaller problem.

The optimality of the algorithm is verified on a cantilevered square plate of unit dimensions. Plate bending, like beam bending, is governed by 4th order partial differential equations and is considered a challenge for iterative solvers because their condition numbers grow at a rate $O(h^{-4})$. Tables 6 and 7 show that the iteration count increases with increase in H/h (Table 6), and decreases with decrease in H/h (Table 7). However, if H/h is held fixed, Table 8 shows, as desired, the iteration count remains relatively constant. Thus, the two plate problems shown in Figs. 5 and 6 converge at the same rate (see rows marked with $\Leftarrow$ in Table 8)— even though the latter is four times larger.

h	n_s	n_{DOF}	FETI-DP CG	FETI-DP GMRES
1/8	16	216	18	21
1/12	16	468	23	26
1/16	16	816	26	31
1/20	16	1260	29	36
1/24	16	1800	32	39
1/32	16	3168	37	46

Table 6: **Iteration count vs. increase in mesh refinement. Total number of subdomains fixed n_s=16, i.e., H fixed. H/h increased.**

H	n_s	n_{DOF}	FETI-DP CG	FETI-DP GMRES
1/3	9	1260	31	46
1/4	16	1260	32	41
1/6	36	1260	29	33
1/8	64	1260	25	29
1/12	144	1260	21	23

Table 7: **Iteration count vs. increase in number of subdomains. Total problem size or number of DOFs fixed $h = 1/24$, i.e., h fixed. H/h decreased.**

h	n_s	n_{DOF}	FETI-DP CG	FETI-DP GMRES
1/12	9	468	23	29
1/16	16	816	26	31$\Leftarrow$
1/20	25	1260	28	32
1/24	36	1800	29	33
1/28	49	2436	30	34
1/32	64	3168	30	34$\Leftarrow$

Table 8: **Iteration count vs. increase in number of subdomains for an increasing total problem size. Total number of DOFs increases, i.e., h decreases; Total number of subdomains increases, i.e., H decreases; but H/h fixed.**

Parallel Implementation of CG

A standard Conjugate Gradient (CG) update is as shown below. The main building blocks that are constructed using the parallel FETI-DP procedure are highlighted in bold.

```
$\lambda_0 = 0; \quad \mathbf{r_0 = d - F\lambda_0}$
for $k = 1, 2, \ldots$
    $z_{k-1} = \mathbf{M^{-1} r_{k-1}}$
    $\xi_k = \left(\underline{z_{k-1}^T r_{k-1}}\right) / \left(z_{k-2}^T r_{k-2}\right)$    with $\xi_1 = 0$
    $p_k = z_{k-1} + \xi_k p_{k-1}$    with $p_1 = z_0$
    $\gamma_k = \left(z_{k-1}^T r_{k-1}\right) / \left(\underline{p_k^T \mathbf{F p_k}}\right)$
    $\lambda_k = \lambda_{k-1} + \gamma_k p_k$
    $r_k = r_{k-1} - \gamma_k F p_k$
end
```

In addition to the communication requirements for the FETI-DP, the CG update requires processor synchronization points of its own. These are points beyond which calculations cannot proceed unless all processors reach that point. All vector inner products are synchronization points. The two synchronization points are underlined above. An additional synchronization point is required to calculate the norm of the preconditioned residual $||z_{k-1}||_2$, to determine the stopping criteria. In the case of CG, the total number of points can be reduced to one, using advanced norm estimation techniques [26, 27]. This refinement has not been included at present, but is needed eventually when thousands of distributed memory nodes are used.

Parallel Implementation of GMRES

A standard Generalized Minimum Residual (GMRES) update requires an Arnoldi algorithm and a solution of a least-square problem. The Arnoldi algorithm implemented in this study uses a Reorthogonalized Classical Gram-Schmidt procedure, based on Ref. [28]. This procedure produces levels of orthogonalization that is superior to Modified Gram-Schmidt [29], while preventing the unacceptable communication costs of the latter (explained below). A Classical Gram-Schmidt (without Reorthogonalization) is numerically unstable and is not used in practice.

Recall, given an initial estimate x_0 and residual $r_0 = b - Ax_0$, every m-th GMRES update for the solution of $Ax = b$ is given by $x_m = x_0 + K$ where K lies in the Krylov subspace of dimension m associated with A and r_0, $K_m(A, r_0) = \text{span}(r_0, Ar_0, \ldots, A^{m-1}r_0)$, and minimizes the norm $||b - Ax||_2$.

The Arnoldi algorithm in dimension m constructs an orthonormal basis $V_m = [v_1, v_2, \ldots, v_m]$ of the Krylov subspace $K_m(A, r_0)$. The procedure also generates a matrix $\bar{H}_m$ of size $(m+1) \times m$ the top $m \times m$ block of which is an upper Hessenberg matrix H_m. The m-th update is computed as $x_m = x_0 + V_m y_m$ where y_m is calculated such x_m minimizes $||b - Ax_m||_2$. This amounts to calculating a y_m which minimizes $||\beta e_1 - \bar{H}_m y_m||_2$ where $\beta = ||r_0||_2$ and e_1 is the first canonical vector of $\Re^{m+1}$. A QR factorization — employing Givens rotations — is used here to solve this least squares problem.

The classical GMRES method expands the Krylov subspace dimension to n and terminates in at most n iterations, where n is the size of A. However, each iteration requires every one of the previous basis vectors, and hence the method is expensive in terms of memory. A restarted version of GMRES restricts the expansion to, say, m dimensions and restarts the Arnoldi algorithm using x_m as it new initial guess. These restarts are called the outer iterations. We use a restarted GMRES(m) method. It is as follows.

```
$\lambda_0 = 0; \quad \mathbf{r_0} = \mathbf{d} - \mathbf{F}\lambda_\mathbf{0}$
$\mathbf{z_0} = \mathbf{M^{-1}r_0}$
$\beta = ||z_0||_2$
for $k = 1, 2, \ldots$ till convergence
    $v_1 = z_{k-1}/\beta$
    Arnoldi algorithm
    Least-square solve of order $m$:
        Calculate $y_m$ to minimize
        min $_{y \in \Re^m} ||\beta e_1 - \bar{H}_m y||_2$.
        Use $QR$ factorization of $\bar{H}_m$.
    $\lambda_k = \lambda_{k-1} + V_m y_m$
    $\mathbf{r_k} = \mathbf{d} - \mathbf{F}\lambda_\mathbf{k}$
    $\mathbf{z_k} = \mathbf{M^{-1}r_k}$
    $\beta = ||z_k||_2$
end
```

The Arnoldi algorithm consists of three steps. Given A and an initial vector v_1, the m orthonormal basis vectors are constructed as follows.

```
for $j = 1, 2, \ldots, m$
    1. Basis expansion: $w_{j+1} = Av_j$
    2. Orthogonalization: orthogonalize $w_{j+1}$
       with respect to all previous Arnoldi
       vectors $(v_1, v_2, \ldots, v_j)$
    3. Normalization: $h_{j+1,j} = ||w_{j+1}||_2$
       and $v_{j+1} = w_{j+1}/h_{j+1,j}$.
```

The orthogonalization is the main step. Traditionally, a Modified Gram-Schmidt procedure is preferred in this step because of its numerical stability over Classical Gram-Schmidt. It is as follows — with the synchronization points underlined.

```
for $j = 1, \ldots, m$
    $w = \mathbf{Fv_j}$
    $t = \mathbf{M^{-1}w}$
    for $i = 1, \ldots, j$
        $h_{i,j} = \underline{v_i^T t}$
        $t = t - h_{i,j} v_i$
    end
    $h_{j+1,j} = \underline{||t||_2}$
    $v_{j+1} = t/h_{j+1,j}$
end
```

A complication is immediately apparent – the first set of points, i.e. the $v_i^T t$ calculations, presents a high communication requirement. Within each step j, the vector t, once generated, is immediately projected to, and subtracted from, each and every one of the previous Arnoldi vectors v_i. Each projection, a vector inner product, requires a global synchronization. In a Classical Gram-Schmidt, the projection and the subtraction steps could be carried out separately, with a single synchronization step in-between. However, because Classical Gram-Schmidt is unstable (though mathematically equivalent to Modified Gram-Schmidt), a second orthogonalization step is needed. Thus, the final Reorthogonalized Classical Gram-Schmidt algorithm is as follows.

```
for $j = 1, \ldots, m$
    $w = \mathbf{Fv_j}$
    $t = \mathbf{M^{-1}w}$

    for $i = 1, \ldots, j$
        $h_{i,j} = v_i^T t$
    end
    Global synchronization 1
    for $i = 1, \ldots, j$
        $t = t - h_{i,j} v_j$
    end

    for $i = 1, \ldots, j$
        $h'_{i,j} = v_i^T t$
    end
    Global synchronization 2
    for $i = 1, \ldots, j$
        $t = t - h'_{i,j} v_j$
    end

    $h = h + h'$
    $h_{j+1,j} = \underline{||t||_2}$
    $v_{j+1} = t/h_{j+1,j}$
end
```

3-D FEM ROTOR ANALYSIS COMPONENTS

The main components of the 3-D rotor FEM analysis are described in this section. They are: geometry and grids, partition and corner selection, steady hover prototype, and the transient forward flight prototype.

These are mere prototypes because we do not use real airloads, do not have a trim mechanism, and do not have at present a true representation of a blade structure. In addition, for research purposes, we will consider only small size problems, with which a large number of cases could be constructed using the 48 processors that were at our disposal.

On the other hand, every fundamental aspect of the physics of the structural dynamics of an isolated rotor blade is incorporated. And, the parallel solution procedure is generic, i.e. it is independent of the type of air-

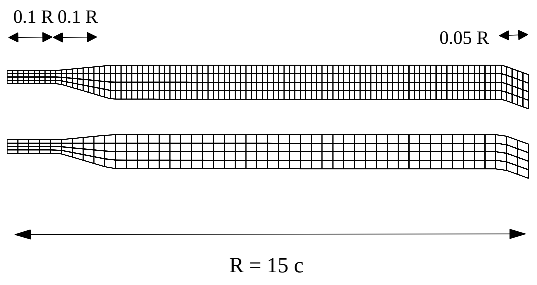


Figure 7: **Planform of a hingeless rotor blade showing two different span-wise grid resolutions used in this study; $c = 0.53$ m**

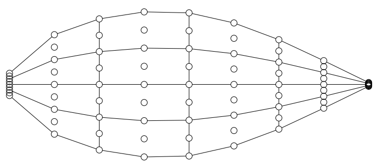


Figure 8: **Cross-section of prototype rotor blade; 4×4 bricks with internal nodes; 5% t/c, exaggerated scale.**

loads, control angle variation, grid, and material constitution. The objective is to study the parallel scalability of the Newton-Krylov solver.

Geometry and Grid

We consider a hingeless rotor blade, discretized as shown in Figs. 7 and 8. The simple grid generator for this study requires that the cross-sectional discretization remains the same along span and that all sections be solid. With these assumptions, it is straight forward to accommodate an arbitrary variation of airfoil shape, twist, planform, and advanced tips. A key limitation at present is that only one continuous structure can be gridded. Grid generation, however, is not the focus of this work. It is assumed that a suitable grid will be available to the solver from other sources.

The surface geometry, required for external forcing, is defined by the sectional airfoil coordinates. We use a generic, symmetric airfoil with 5% thickness (Fig. 8).

We consider a set of four finite element discretizations (Table 9). As before, $n_1 \times n_2 \times n_3$ refers to numbers of elements along span, chord, and thickness. Note that each element contains 81 degrees of freedom and 64 integration points.

Grid	$n_1 \times n_2 \times n_3$	Total DOFs
1	$48 \times 4 \times 2$	12,960
2	$48 \times 4 \times 4$	25,920
3	$96 \times 4 \times 2$	25,920
4	$96 \times 4 \times 4$	46,656

Table 9: **3-D FEM rotor grids**

Each finite element, naturally, can accommodate its own constitutive material model and ply direction – we use simple isotropic properties: $E = 73$ GPa; $\nu = 0.3$; and $\rho = 2700$ kg/m^3 (corresponding to Aluminum).

Along with the dimension $c = 0.53$ m, these generate similar order of magnitude non-dimensional values of stiffness and inertia as soft in-plane hingeless rotors. No attempt is made to place the sectional offsets with respect to quarter-chord. Thus, the blade may not be dynamically stable. However, the values will generate typical deflections with typical airloads. The airloads are an uniform 4000 N/m^2 baseline (around 375 lb/ft along span), and two and four times that magnitude, to generate moderately large deformations. These are referred to as airloads 1, 2, and 3. The pressure airloads act only on the top surface and have the non-linear characteristics of a follower force – i.e., they act normal to the deformed surface which is not know a priori. The rotational speed is $\Omega = 27$ rad/s (steady).

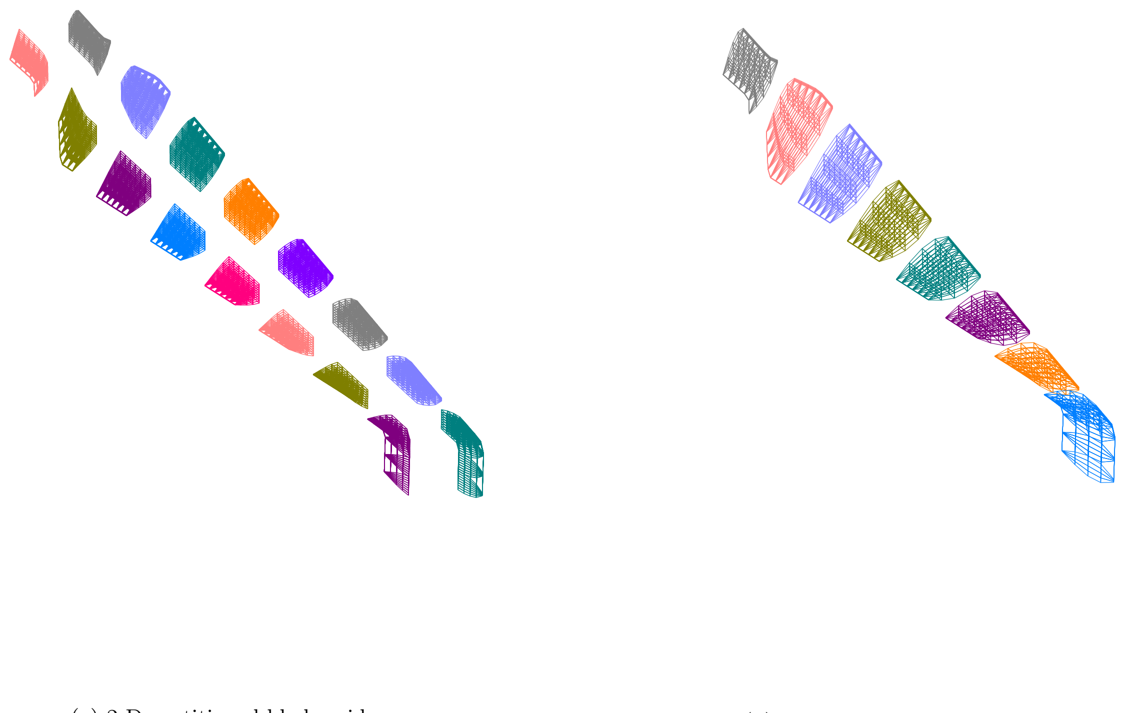
Grid Partitioning and Corner Selection

The partitioning requirements are unique. The generation of subdomain grids from a global grid via a recalculation of the finite element connectivity is straightforward. Partitioners are widely available in public domain, that carry out this task ensuring an optimal balancing of processor loads. However, for structures, this is not the most important requirement. The most important requirement is that the the coarse problem be picked to ensure a null kernel in every substructure, i.e. K^s_{RR} be invertible.

The partitioner we develop as part of this research is simple – in that it handles only the brick elements we developed, and it makes the same assumptions on grid type as our simple grid generator. Namely, the cross-sectional grids must remain the same throughout the span, regardless of variations in geometry.

Figure 9(a) shows a generic 2-D partition that is used in the present study. The blade can be divided into any number of substructures in the span-wise and chord-wise directions. Figure 9(b) shows an alternative 1-D partition. It will be shown that the 1-D partition, though naturally load balanced, is a poor partition and should not be used.

The partitioner performs the following tasks:



(a) 2-D partitioned blade grid

(b) 1-D partitioned blade grid

Figure 9: **2-D and 1-D partitioning of blade grid into substructures. Each substructure is solved in a separate processor.**

1. Designates the corner nodes.
2. Re-orders the subdomain nodes into interior, face, edge, vertex, and boundary nodes. Re-calculates the subdomain finite element connectivity.
3. Sets up domain connectivity maps for substructure to substructure communication.

The first and second are the key tasks. The third is merely a matter of book-keeping.

Corner Selection

Consider the 2-D partition of Fig. 9(a). The nodes on the subdomain edges — that are common to more than two subdomains — are immediately designated as corner nodes. It is clear, however, that this definition makes the two substructures at the tip end (or extremities of the tip end in case of more than two chord-wise strips) indefinite. Each substructure then carries a rotational rigid body mode making K^s_{RR} non-invertible. Thus, the definition of corner nodes must include, in addition those edge nodes, which may be common only to two subdomains, but which occur at the boundaries of the structure. With this definition, the corner nodes of a typical substructure are now as shown in Fig. 10. This definition also enables the selection of corner nodes for a typical substructure of the 1-D partition (Fig. 11), otherwise, there would be no corner nodes. For a large scale 3-D problem, a large number of corner nodes is generated by this procedure. A superior choice is simply the subdomain vertices, and like before, additionally those that occur at the boundaries (Fig. 12). There is then always a maximum of only 8 corner nodes per subdomain – regardless of the grid. In this paper, this selection is not implemented. We use the previous selection as shown in Fig. 10.

Node Reorder

The re-ordering brings the interior nodes first, followed by interface nodes, then corner nodes, and lastly the boundary nodes. The procedure depends on the grid, partition (1-D or 2-D), and the selection of corner nodes. The N elemental nodes in each brick is associated with a natural order within each substructure. The natural order is then associated with a reordered order, and a reverse association back to natural. In addition, the natural order is associated with the global order because the geometry and material constitution is defined in the latter.

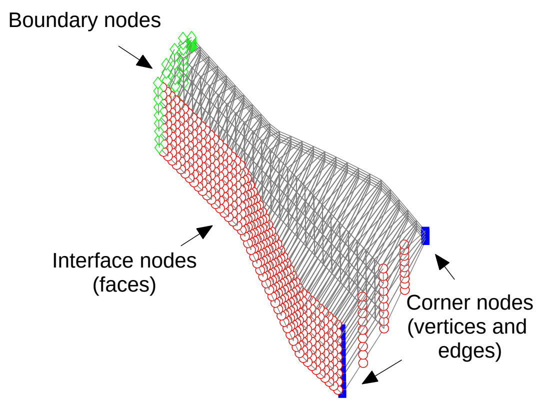


Figure 10: **Node designation for a typical subdomain from a 2-D partition. Corner nodes are edge nodes connecting more than two substructures and those occurring at the boundaries.**

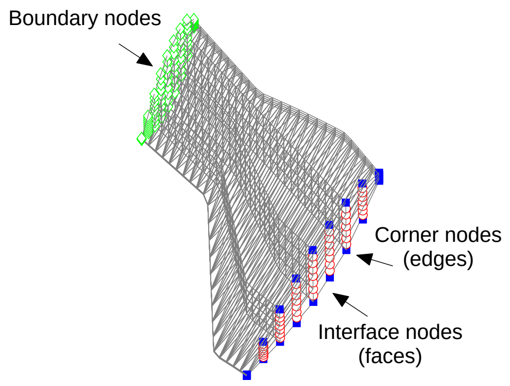


Figure 11: **Node designation for a typical subdomain from a 1-D partition.**

Domain Connectivity

For a Lagrangian formulation, the connectivity remains static, and needs to be calculated only once for a grid. The non-floating and non-overlapping nature of the partitions, and the conforming nature of the finite elements lead to no search, interpolation, or projection requirements. Consequently, there are no errors introduced during substructure to substructure communication. Each substructure carries a destination map and a reception map. The destination map contains the substructures to which quantities are to be dispatched, and the node numbers to which they correspond. The reception maps contains the substructures from which quantities are to be received, and the internal node numbers to which they will correspond.

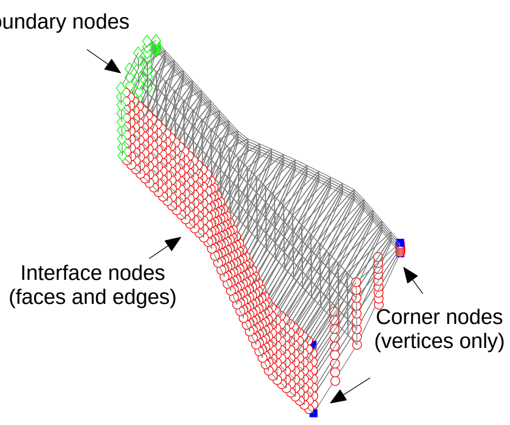


Figure 12: **Optimal Node designation for a typical subdomain from a 2-D partition. Corner nodes are vertex nodes only including those occurring at the boundaries.**

Steady Hover Prototype

The steady hover prototype simply solves for blade response using a prescribed pressure airload at a constant collective pitch angle. The non-linear solution procedure uses Newton-Raphson outer iterations. The FETI-DP inner solver uses CG update in hover. This is adequate as the stiffness matrix is symmetric.

Several initial updates of the stiffness matrix are necessary to include the non-linear structural stiffness – which provides the key extension-bending non-linearity associated with rotation. Figure 13 shows the convergence of this non-linearity in the left hand side of the figure. Around 8 to 10 iterations are required. Subsequently, the stiffness matrices can be updated only at certain intervals if desired, while using modified Newton iterations in between. Once the structural non-linearities are converged, the pressure airloads are imposed on the blade. The direction of the pressure airloads is deformation-dependent and unknown a priori. Thus, equilibrium iterations are required. The convergence of the airload iterations are shown in the same plot on the right hand side (corresponding to airloads 1, 2, and 3). The two parts are shown separately only because the latter are equilibrium iterations where the geometric stiffness need not be updated. The results shown, however, are with fully updated stiffness in every iteration.

In each iteration, the virtual work is calculated based on the previous iteration deformation state. An alternative and more rigorous approach is to linearized the forcing using incremental displacements. This leads to a non-symmetric stiffness contribution, which is however easily handled by replacing the geometric stiffness K with $1/2(K+K^T)$, since the role of the stiffness is only to converge the Newton iterations. However, iterations are still necessary and therefore we believe the previous configuration approach is more efficient. The relatively large de-

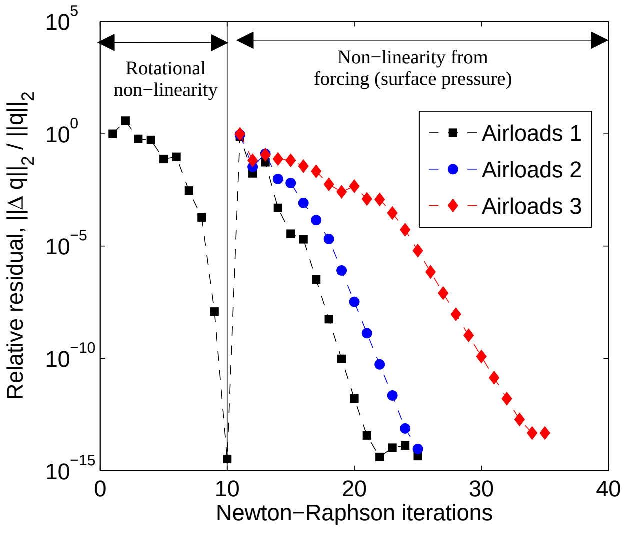


Figure 13: **Convergence of Newton-Raphson outer iterations for rotational and forcing non-linearities in hover**

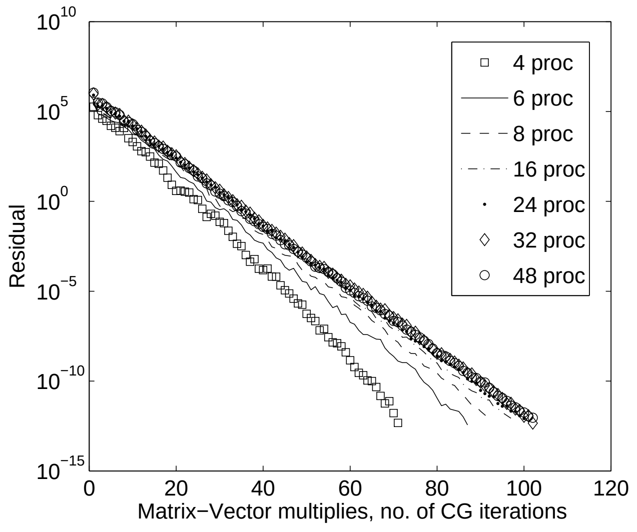


Figure 15: **Convergence pattern of FETI-DP (CG) updates in steady hover calculations on** 4 **to** 48 **processors.**

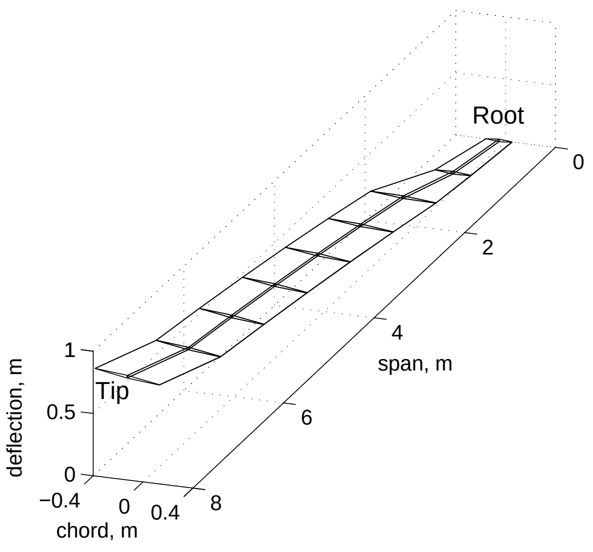


Figure 14: **Blade steady deflection in hover using prescribed pressure airloads 3 (only grid outlines are shown)**

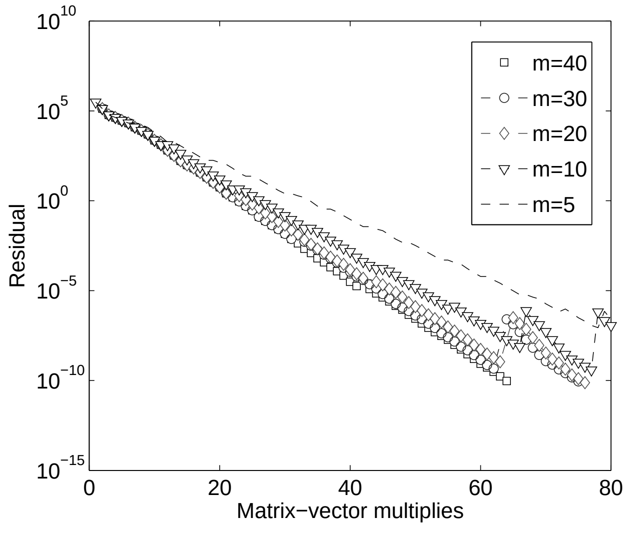


Figure 16: **Convergence of FETI-DP (GMRES) updates for various restart parameters in forward flight; all calculations are with** 32 **processors.**

formation corresponding to airloads 3 is shown in Fig. 14.

The convergence criteria of the inner Krylov solver is set tightly to 10^{-12} for all cases in this study. Fig. 15 shows the convergence of the solver when run on 4 to 48 processors. The convergence corresponding to the first Newton iteration is shown, as it begins with zero guess and takes the most number of iterations. The parallel scalability of these calculations are examined in more detail in the next section. Here, we note that the iteration count shows an initial increase with processors. This is contrary to Table 7 but is recognized for 3-D brick problems in Ref. [16]. The remedy is an edge-based augmentation procedure that is not incorporated in this work.

Transient Forward Flight Prototype

The transient forward flight prototype solves for blade response using the same set of prescribed pressure airloads as in hover, but now the stiffness and forcing values are those that arise out of a single time step of a time-marching procedure. We consider a Newmark scheme with a 5° azimuth step. The control angle variation is taken as $\theta(\psi) = 20^\circ + 5^\circ \cos\psi - 5^\circ \sin\psi$. The dynamic stiffness now contains the complete non-symmetric inertial terms. Therefore, the FETI-DP inner solver now uses a GMRES update.

The matrix structure will be identical in every time step, therefore, for purposes of scalability it is enough to study only one time step. Unlike CG where each update requires only two previous solutions, in GMRES, each update requires all of the previous updates. The restarted version, as explained earlier, uses at the most m updates at a time, starting afresh with a superior initial guess each time.

Figure 16 shows, that for the small grid size used here (i.e. a small bandwidth), the convergence pattern remains similar over a wide range of restart parameters – even $m = 5$ is adequate. For purposes of a realistic scalability study, however, we will consider $m = 30$, 40, and 50. These are deemed more suitable for large scale problem sizes.

SCALABILITY OF 3-D ROTOR ANALYSIS

The parallel scalability of the 3-D FEM solver is documented here in detail. The first section deals with the steady hover prototype. The Krylov solver uses a CG update here. The idea of substructure optimality and the definition of scalability are introduced. The second section deals with the transient forward flight prototype. The Krylov solver is equipped with a GMRES update here.

Steady Hover

Consider the grid of size $96 \times 4 \times 2$. It is partitioned into 6 to 48 substructures using a 2-D partitioning, i.e., having an arrangement as Fig. 9(a) with 3×2 to 24×2 subdomains.

n_s	FE	Subdomain LU	Coarse problem	FETI-DP CG	Solver total
6	201	757	130	775	1668
8	200	463	91	622	1180
12	199	246	63	458	770
16	194	165	52	361	580
24	191	96	51	267	416
32	190	66	71	213	350
48	190	39	168	187	395

Table 10: **Solver time (secs) vs. number of substructures n_s on a single processor**

n_p	FE	Subdomain LU	Coarse problem	FETI-DP CG	Solver total
6	32.6	121.80	21.56	94.07	237.87
8	24.2	57.44	12.33	57.23	127.28
12	16.2	20.95	5.80	26.85	53.73
16	12.6	10.54	3.64	14.93	29.19
24	8.18	4.16	2.71	7.96	14.89
32	6.01	2.20	2.92	5.70	10.85
48	4.24	0.88	5.62	5.15	11.70

Table 11: **Solver time (secs) vs. number of processors n_p; each processor contains one substructure**

The dramatic reduction in solution time is clear from Tables 10 and 11. The details are described later. Here, we note that a direct solution of this problem takes more than 50,000s. The iterative substructuring method, with 32 substructures, but still on a single processor, brings it down to only 350s (Table 10). In addition, the method is now fully parallel. Therefore, a scalable implementation, on 32 processors, finally brings it down drastically to 10.85s (Table 11).

Consider the solver times for this problem on a single processor (Fig. 17). The importance of substructuring is immediately apparent. There is a steep drop in solution time with increasing number of substructures. For a problem of fixed size, a condition of diminishing return must eventually be reached, with an optimal number of substructures producing the minimum solution time. We shall call this the substructure optimality number. For this problem it is 32. Note, however, that the rise in solution time beyond the optimality point is not nearly as steep as its decline prior to it, and there is a large region over which it remains flat. For this problem, this region is roughly from 16 to 48 substructures. This flat region is a gift of iterative substructuring. It is shown later that this region is sensitive to the partitioning and corner selection scheme. A good partitioner and corner selector will keep this region flat, a poor one will produce a steep rise.

The parallel implementation solves each substructure on a separate processor. To calculate parallel speedup, it is important to use the single processor time with the same number of substructures as the baseline. That is, the solver time with, say 32 processors, must be com-

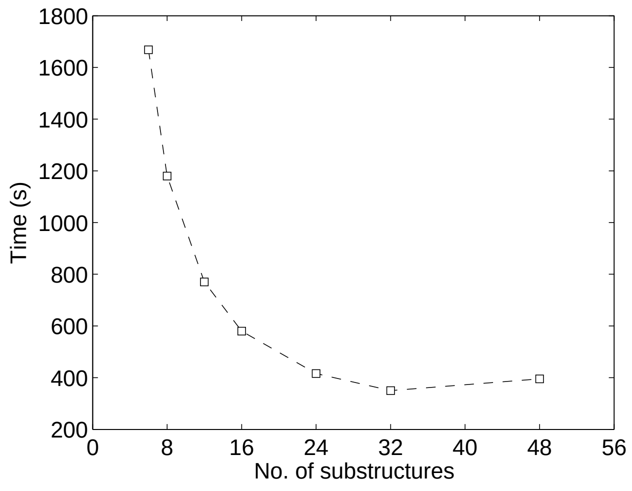


Figure 17: **Solver time vs. number of substructures for calculations on a single processor.**

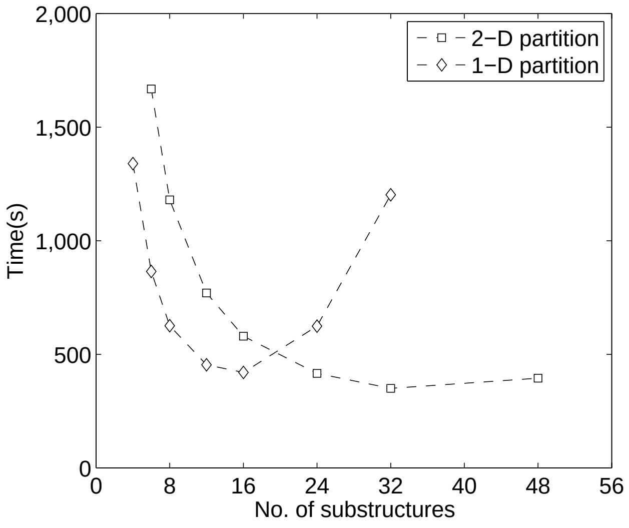


Figure 19: **Effect of grid partitioning and corner selection on solver time as a function of number of substructures; calculations on a single processor.**

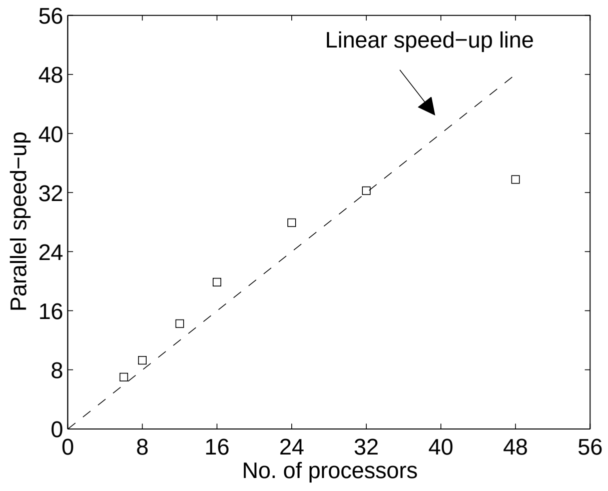


Figure 18: **Parallel speed-up for calculations on multiple processors; each substructure solved in a separate processor.**

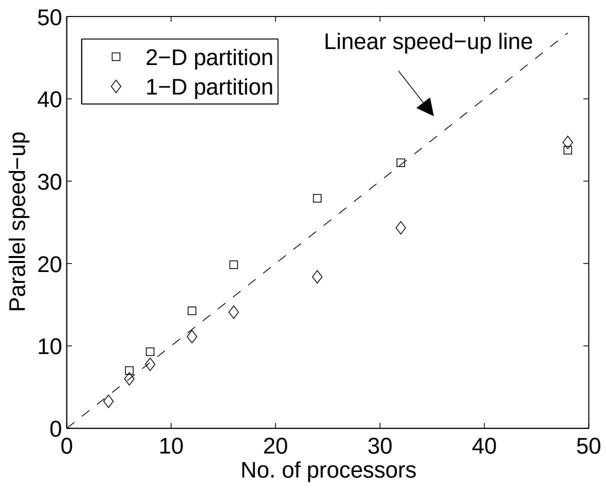


Figure 20: **Effect of grid partitioning and corner selection on parallel speed-up for calculations on multiple processors.**

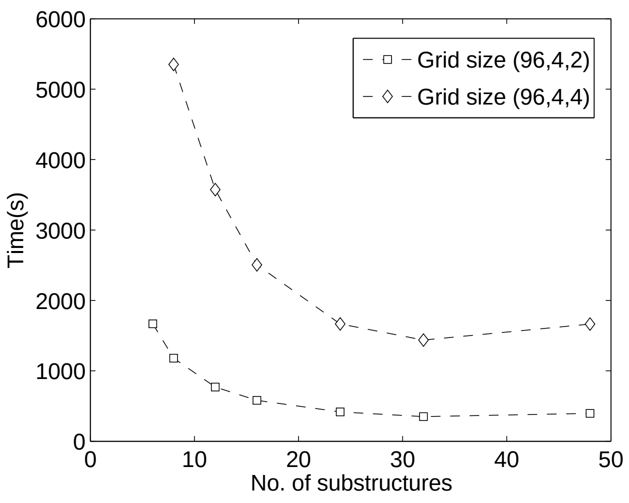


Figure 21: **Two different problem sizes with the same substructure optimality; solver time vs. number of substructures on a single processor.**

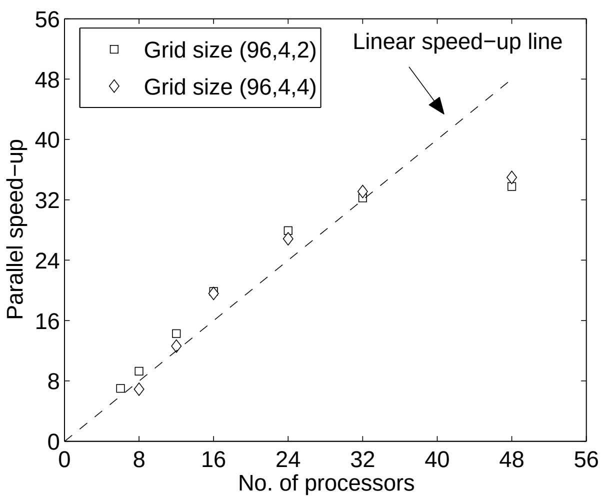


Figure 22: **Parallel speed-up for problem sizes with the same substructure optimality; solver time vs. number of processors.**

pared with the corresponding solver time on a single processor that uses 32 substructures. This is to ensure that computations of the same complexity are compared, otherwise, the speed-up is contaminated with the benefits of substructuring and a super-linear number is always obtained. This is because using 32 substructures on a single processor by itself reduces the solver time by more than 32 – a fact that has nothing to do with parallelization but substructuring itself. The parallel speed-up is shown in Fig. 18. Even for this fixed problem size, it has a perfectly linear trend up to the point of substructure optimality. Note that the point corresponding to 32 processors has nothing to do with the point corresponding to 16 processors. Indeed, the 32 processor run takes only 10.9s whereas the 16 processor run takes 29.2s, i.e. less than half the time, due to the dramatic reduction in subdomain LU time (see Table 10). What the speed-up plot shows is that 32 processors run the problem exactly 32 times faster compared to a single processor using 32 substructures.

The drop off in scalability beyond 32 processors is studied using the detailed timings for the different parts of the computation. The timings for the single processor and parallel calculations are given in Tables 10 and 11. In the tables, 'FE' refers to the time taken to construct the structural matrices. 'Solver total' refers to the total solver time. The two together constitute the total simulation time. 'Solver total' consists of three parts: (1) 'Subdomain LU' time, which refers to the subdomain factorization, (2) 'Coarse problem' time, which refers to the coarse problem factorization and communication, and (3) the 'FETI-DP' time, refers to the Krylov solver time. Note that the later includes the computation and communication costs of the residual, preconditioner, and matrix-vector multiplies, and the additional communications required for the updates. The communications costs are, of course, incurred only during the parallel calculations.

From Table 10, which shows the single processor timings, the reason behind the flat region in Fig. 17 is clear. The growth in the coarse problem is offset by the reduction in the Krylov solver time. This is expected – as the purpose of the coarse problem is precisely that – but the key point is that the coarse solver should be just enough to serve this purpose and no larger, so that the substructure optimality is pushed to as high a processor number as possible. Beyond the optimality point, any growth in the coarse problem is an indicator of increased communication cost for the parallel implementation. Note that the coarse problem is solved in every processor and as such, requires a global communication. The drop off in Fig. 18 is a direct consequence of this communication cost. To summarize, a key objective of the coarse problem selection should be to suppress the growth beyond substructure optimality to as gradual as possible. This has little bearing upon scalability with problem size, but serves to extend linear speed-up for a fixed problem size, to as high a processor number as possible.

We illustrate the importance of the coarse problem with a worse partitioning. The same problem, when treated with a 1-D partitioning as illustrated earlier in Figs. 9(b) and 11, generates a timing and scalability plot as shown in Figs. 19 and 20. The 2-D partitioning results are also plotted for comparison. Clearly, from Fig. 19, the same problem now has a substructure optimality of 16, as opposed to 32. A good parallel implementation should guarantee a linear speed-up to at least 16 processors. Scalability beyond this number is expected to be affected adversely by communication costs of the coarse problem. This is exactly what is observed in Fig. 20.

Thus, the linear speed-up range is not a function

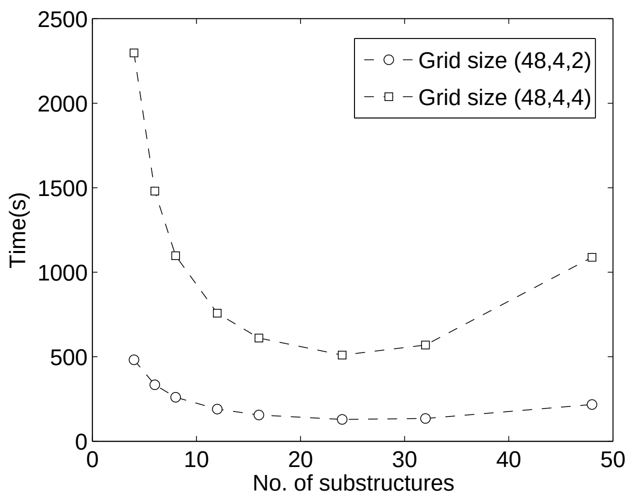


Figure 23: **Two different problem sizes with the same substructure optimality; solver time vs. number of substructures on a single processor.**

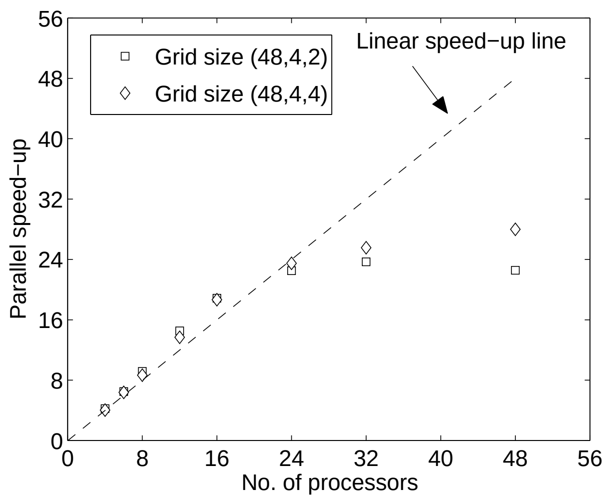


Figure 24: **Parallel speed-up for problem sizes of same substructure optimality.**

of problem size alone but also of substructure optimality. For example, Figs. 21 and 22 compare the single processor solution time and parallel speed-up of a bigger problem of size $96 \times 4 \times 4$. From the single processor solution time, it is clear that the substructure optimality is still 32. As a result, the linear speed-up range still extends only up to 32. The same conclusions hold for very small problem sizes, as shown in Figs. 23 and 24. The smallest grid of $48 \times 4 \times 2$ shows a linear speed-up up to 24 processors – it's substructure optimality – in exactly the same manner as a grid of $48 \times 4 \times 4$ twice its size.

In summary, given a problem size, the solver shows a linear speed-up – for at least as many processors as its substructure optimality. To extend this linear speed-up range, a smaller coarse problem is required. An example of such a selection was shown earlier in Fig. 12.

The scalability with increasing problem size is illustrated in Table 12. Problem 2 has twice the size of Problem 1 (from Table 9). Problem 3 has the same size as Problem 2, only different grid characteristics. Problem 2 and 3, therefore, have similar solver times, up to substructure optimality which sets in at 24 processors for Problem 3. The timings of Problem 1 and Problem 2 provide a simple illustration of scalability with increasing size. Because Problem 2 is twice the size, twice the number of processors provide approximately the same solve time. For example, Problem 2 on 12 processors take similar time as Problem 1 on 6. Problem 2 on 16 take similar time as Problem 1 on 8.

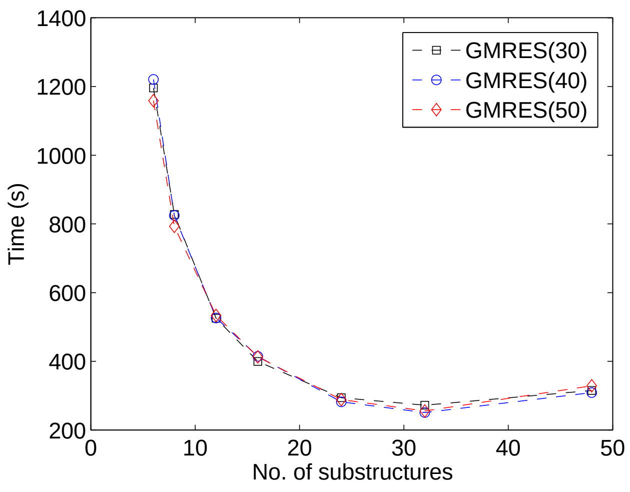


Figure 25: **Solver time vs. number of substructures on a single processor; FETI-DP (GMRES).**

n_p	Problem 1 $48 \times 4 \times 2$	Problem 2 $96 \times 4 \times 2$	Problem 3 $48 \times 4 \times 4$
6	51.52	237.87	232.00
8	28.39	127.28	126.70
12	13.07	53.73	55.43
16	8.22	29.19	32.72
24	5.73	14.89	21.69
32	5.70	10.85	22.28
48	9.62	11.70	38.89

Table 12: **Parallel solver times (secs) showing scalability with respect to problem size up to limits of substructure optimality.**

Transient Forward Flight

The conclusions drawn on effective partitioning and substructure optimality in the previous section are carried over to this section. These results, which are similar to those shown in hover, are not repeated here. The scalability results for a single grid size $96 \times 4 \times 2$ (with substructure optimality of 32) is presented. The results for the other grids compare similarly to hover.

The scalability of the parallel FETI-DP (GMRES) solver involving the non-symmetric matrices in forward

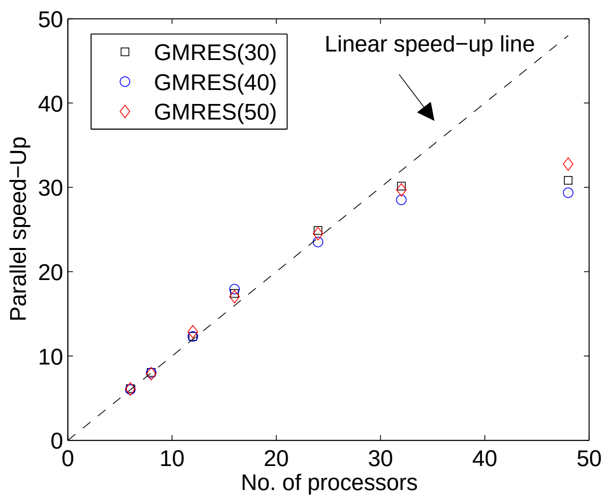


Figure 26: **Parallel speed-up of FETI-DP (GMRES) solver using Classical Gram-Schmidt with Reorthogonalization based Arnoldi procedure.**

n_p	Modified GS	Classical GS with Reorth.
6	194.08	190.17
8	109.47	100.47
12	41.69	41.43
16	23.92	24.24
24	12.33	11.77
32	8.88	8.61
48	10.67	10.04

Table 13: **Parallel FETI-DP (GMRES) solver times (secs) with Modified Gram-Schmidt and Classical Gram-Schmidt with Reorthogonalization based Arnoldi procedures.**

flight shows a similar linear trend as those of the FETI-DP (CG) solver in hover. The single processor timings are shown in Fig. 25. All calculations use the Reorthogonalized Classical Gram-Schmidt Arnoldi procedure. The increasing restart parameters all show the same timings on a single processor – as expected, because their affect is only on memory – but incur increasing communication costs for a parallel calculation. However, for the small sized problems considered here, there are no discernible differences between the three versions even in parallel. As a result, the scalability plot shown in Fig. 26 is identical for all three cases. Indeed, even the Modified Gram-Schmidt procedure show the same scalability behavior (Fig. 27). The latter is expected to be drastically inferior for large subdomain problems. Note however, regardless of scalability, the actual solution times for Reorthogonalized Classical Gram-Schmidt are by themselves lower compared to the Modified Gram-Schmidt. This difference is expected to be drastic for large scale problems. This trend is clear in Table 13.

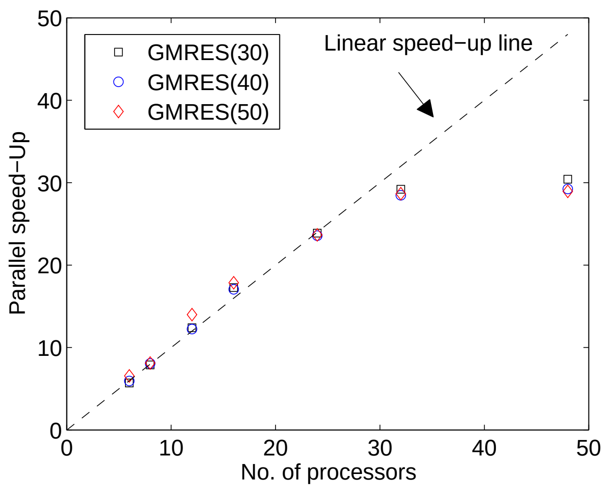


Figure 27: **Parallel speed-up of FETI-DP (GMRES) solver using Modified Gram-Schmidt based Arnoldi procedure.**

CONCLUDING OBSERVATIONS

The main objective of this paper was to demonstrate a parallel and scalable solution procedure for a 3-D FEM based dynamic analysis of helicopter rotor blades. The 3-D FEM analysis was formulated with an emphasis on the non-linear structural, and inertial terms that are unique to rotorcraft. Second order, isoparametric, hexahedral brick elements, that are well suited to discretize a rotor blade structure, were developed to carry out this study. A dual-primal iterative substructuring based Krylov solver was developed for a fully parallel solution procedure. The iterative substructuring method was built upon the FETI-DP domain decomposition algorithm. The Krylov solver was equipped with GMRES updates, in addition to its traditional CG update, due to the the non-symmetric nature of the inertial terms. A detailed study was carried out on the scalability of the solution procedure for both hover and transient forward flight conditions. Based on this study, the following key conclusions are drawn.

Key conclusions

1. A 3-D FEM based rotor dynamic analysis can be carried out using a fully parallel and scalable solution procedure.

2. Given a fixed problem size, there is always an optimal number of substructures into which it can be decomposed – termed substructure optimality – so as to require the minimum solution time.

3. The 3-D FEM analysis presented in this paper is scalable, i.e. shows a linear speed-up up to the point of substructure optimality. A p-processor calculation with a separate substructure in each processor takes $1/p$ the time compared to a single processor with p substructures. It also scales with problem

size. That is, a n-times larger problem takes similar time with $n \times p$ processors.

4. For a fixed problem size, a drop off in scalability eventually occurs – but not before the subdomain optimality number is reached. At that point, there is no reason to use any more processors – unless a larger problem is attacked – in which case, linear speed-up is restored again up to the new optimal. However, even if more processors than optimal is used, the scalability only reaches a plateau, and does not show a dramatic fall. The plateau stems from iterative substructuring and is due to the flat region at the bottom of the time vs. substructure curve.

5. The drop in scalability beyond substructure optimality is traced to two factors: the increasing substructure to substructure communication cost, and, the global coarse problem communication cost. The first penalty is relatively minor, and can be minimized with a minimum number of synchronization points during the Krylov update. This is more relevant to the GMRES. The second penalty, which stems from the coarse problem, is an important factor.

6. The size of the coarse problem is the key driver for both parallel scalability as well as solution time. The global communication required by the coarse problem drives scalability. The size of the coarse problem drives solution time. In order to ensure scalability while decreasing solution time, a minimal coarse problem should be selected.

7. The coarse problem is selected by the domain partitioning algorithm. Partitioning has special requirements in structures – not just any will do. The key idea is that the coarse problem must ensure null kernels in each substructure while satisfying the criteria of minimal selection (conclusion 6).

In summary, the parallel solver developed in this study can solve both hover and transient forward flight response in a scalable manner. For example, each Newton iteration of a $25,920$ DOFs problem, could be solved in approximately $8 - 10$ seconds on 32 processors with a residual reduction level of 10^{-12}. A direct solve on a single processor, in comparison, is not feasible – requiring more than $50,000$s. Real blade structures will contain millions of DOFs. The scalability of the solver must then be tested on 100s and 1000s of processors. Actual solver timings would be as important as scalability.

The next fundamental challenge towards the application of this solver is the problem of periodic response. Transient response is of little use in rotary wing dynamics due to the requirement of obtaining periodic solutions in presence of undamped modes. The requirement for a finite element in time or non-linear harmonic balance type method will be realized more acutely during high fidelity analysis. A time based domain decomposition mechanism must be innovated. This, and other key challenges are summarized below in way of conclusion.

Future Research Areas

A suggested list for future directions of this research is given below subdivided into three categories.

Fundamental Research

1. Periodic dynamics – scalable domain decomposition in time for temporal boundary value problems.

2. 3-D FEM and multibody dynamics coupling.

Applied Research

1. Nodeless elements for multibody dynamics coupling. Efficient, locking-free, hierarchical elements.

2. 3-D fluid-structure interfaces. Innovative delta coupling for trim solution in level and turning flight.

Application Development

1. Interfacing with 3-D solid geometry and grid tools.

2. Smart substructuring – corner node selection, interface localization, and nodal reordering.

ACKNOWLEDGMENTS

We acknowledge Dr. Roger Strawn's support for this research at the U. S. Army AFDD as part of the HPC Institute of Advanced Rotorcraft Modeling and Simulation (HI-ARMS) program supported by the U. S. DoD HPC Modernization Program Office. We thank Prof. Charbel Farhat, Stanford University, for his kind advice and insights. We also thank Dr. Guru Guruswamy, NASA Ames Research Center, for his useful comments and suggestions.

REFERENCES

[1] Przemieniecki, J. S., "Matrix Structural Analysis of Substructures," *AIAA Journal*, Vol. 1, (1), January 1963, pp. 138–147.

[2] Przemieniecki, J. S. and Denke, P. H., "Joining of Complex Substructures by the Matrix Force Method," *Journal of Aircraft*, Vol. 3, (3), May–June 1966, pp. 236–243.

[3] Denke, P. H., "A General Digital Computer Analysis of Statically Indeterminate Structures," NASA TN D-1666, December, 1962.

[4] Argyris, J. H. and Kelsey, S., *Modern Fuselage Analysis and the Elastic Aircraft*, Butterworth's Scientific Publications Ltd., London, 1963.

[5] Turner, M. J., Martin, H. C., and Weikel, R. C., "Further Development and Applications of the Stiffness Method," *Matrix Methods of Structural Analysis*, AGARDograph 72, Pergamon Press, New York, 1964, pp. 203–266.

[6] Smith, B., Bjørstad, P., and Gropp, W., *Domain Decomposition – Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, UK, 1996.

[7] Agoshkov, V. I. and Lebedev, V. I., "Poincaré–Steklov Operators and the Methods of Partition of the Domain in Variational Problems," *Computational Processes and Systems*, Vol. 2, ed., Marchuk, G. I., Nauka, Moscow, pp. 173–227. In Russian.

[8] Agoshkov, V. I., "Poincaré–Steklov Operators and Domain Decomposition Methods in Finite Dimensional Spaces," First International Symposium on Domain Decomposition Methods for Partial Differential Equations, eds., Glowinski et al., SIAM, Philadelphia, pp. 73–112.

[9] Bramble, J. H., Pasciak, J. E., and Schatz, A. H., "The Construction of Preconditioners for Elliptic Problems by Substructuring," I, II, III, and IV, *Mathematics of Computation*, Vol. 47, (175), 1986, pp. 103–134, Vol. 49, (179), pp. 1–16, 1987, Vol. 51, (184), pp. 415–430, 1988, and Vol. 53, (187), pp. 1–24, 1989.

[10] Quarteroni, A. and Valli, A., *Domain Decomposition Methods for Partial Differential Equations*, Oxford University Press, Oxford, UK, 1999.

[11] Toselli, A. and Widlund, O., *Domain Decomposition Methods – Algorithms and Theory*, Springer Series in Computational Mathematics, Springer-Verlag Berlin Heidelberg, 2005.

[12] *Computers and Structures*, Vol. 85, (9), special issue on "High Performance Computing for Computational Mechanics," eds., Magouls, F. and Topping, B. H. V., May 2007, pp.487-562.

[13] *Computer Methods in Applied Mechanics and Engineering*, Vol. 196, (8), special issue on "Domain-Decomposition Methods: Recent Advances and New Challenges in Engineering," eds., Magouls, F. and Rixen, D., January 2007, pp. 1345-1622.

[14] Mandel, J., and Dohrmann, C. R., "Convergence of a Balancing Domain Decomposition by Constraints and Energy Minimization," *Numerical Linear Algebra with Applications*, Vol. 10, (7), 2003, pp. 639–659.

[15] Farhat, C., Lesoinne, M., and Pierson, K., "A Scalable Dual-Primal Domain Decomposition Method," *Numerical Linear Algebra with Applications*, Vol. 7, (7–8), 2000, pp. 687–714.

[16] Farhat, C., Lesoinne, M., LeTallec, P., Pierson, K., and Rixen, D., "FETI-DP: A Dual-Primal Unified FETI Method - Part I: A Faster Alternative to the Two-level FETI Method," *International Journal of Numerical Methods in Engineering*, Vol. 50, pp. 1523–1544, 2001.

[17] Mandel, J., Dohrmann, C. R., and Radek, T. "An Algebraic Theory for Primal and Dual Substructuring Methods by Constraints," *Applied Numerical Mathematics*, Vol. 54, (2), July 2005, pp. 167–193.

[18] Berdichevsky, V. L., "Variational-Asymptotic Method of Constructing a Theory of Shells," *Journal of Applied Mathematics and Mechanics*, translated from *Prikladnaya Matematika i Mekhanika*, Vol. 43, (4), 1979, pp. 664–687.

[19] Hodges, D. H., *Nonlinear Composite Beam Theory*, AIAA, Reston, VA, 2006.

[20] Volovoi, V. V., and Hodges, D. H., "Theory of Anisotropic Thin-Walled Beams," *Journal of Applied Mechanics*, Vol. 67, (3), September 2000, pp. 453–459.

[21] Yu, W., Hodges, D. H., Volovoi, V. V., and Cesnik, C. E. S., "On Timoshenko-Like Modeling of Initially Curved and Twisted Composite Beams," *International Journal of Solids and Structures*, Vol. 39, (19), September 2002, pp. 5101-5121.

[22] Bathe, K., *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, Inc., NJ, 1982.

[23] Zienkiewicz, O. C., and Taylor, R. L., *The Finite Element Method for Solid and Structural Mechanics*, Elsevier Butterworth-Heinemann, Oxford, UK, Sixth edition, 2006.

[24] Zienkiewicz, O. C., *The Finite Element Method in Structural and Continuum Mechanics*, McGraw-Hill, London, U.K., 1967.

[25] Le Tallec, P., "Domain Decomposition Methods in Computational Mechanics," *Computational Mechanics Advances*, Vol. 1, (2), 1994, pp. 121–220.

[26] Meurant, G., "Multitasking the Conjugate Gradient method on the CRAY X-MP/48", *Parallel Computing*, Vol. 5, (3), July 1987, pp. 267–280.

[27] Saad, Y., "Krylov Subspace Methods on Supercomputers," *SIAM Journal on Scientific and Statistical Computing*, Vol. 10, (6), November 1989, pp. 1200–1232.

[28] Daniel, J. W, Gragg, W. B., Kaufman, L. and Stewart, G. W., "Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization," *Mathematics of Computation*, Vol. 30, (136), October 1976, pp. 772-795.

[29] Giraud, L., Langou, J. and Rozioznik, M., "The Loss of Orthogonality in the Gram-Schmidt Orthogonalization Process," *Computers and Mathematics with Applications*, Vol. 50, (7), October 2005, pp. 1069–1075.