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

American Helicopter Society 65th Annual Forum
Grapevine, TX, May 27–29, 2009

ABSTRACT

This paper has two objectives. The first objective is to formulate a 3-dimensional Finite Element Model for the dynamic analysis of helicopter rotor blades. The second objective is to implement and analyze a dual-primal iterative substructuring based Krylov solver, that is parallel and scalable, for the solution of the 3-D FEM analysis. The numerical and parallel scalability of the solver is studied using two prototype problems – one for ideal hover (symmetric) and one for a transient forward flight (non-symmetric) – both carried out on up to 48 processors. In both hover and forward flight conditions, a perfect linear speed-up is observed, for a given problem size, up to the point of substructure optimality. Substructure optimality and the linear parallel speed-up range are both shown to depend on the problem size as well as on the selection of the coarse problem. With a larger problem size, linear speed-up is restored up to the new subtructure optimality. The solver also scales with problem size – even though this conclusion is premature given the small prototype grids considered in this study.

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, have spectral properties that are superior to the original problem. In particular, its condition number 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 many 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 analysis of partitioned structures. These partitioned methods were the only avenues to obtain results for practical structures for which the original problem far exeeded the capacity of computers at the time. The method of substructures have remained the fastest (time), most efficient (memory), most reliable (accurate), and most flexible (heterogenous 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 opportu-

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

nity for solving each partitioned substructure in 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 total interface size grows both with problem size and with 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, procedures are available which preconditions the interface and solves it iteratively, in a parallel and scalable manner, requiring only local substructure calculations. These procedures 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 polylogarithmically 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]. The 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 or no importance in structural mechanics – due to the very high condition numbers ($10^8 - 10^{10}$) and high bandwidth of practical structures. Structural mechanics use non-overlapping partitioning. They lead to iterative substructuring methods, a name borrowed from, and as explained earlier, indicative of the deep connections to the long and successful tradition of substructuring.

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 a 2-nd order elliptic problem with homogenous 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 [12, 13]. Today, Neumann-Neumann type balancing 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 Integration (FETI) methods (see [15] and references therein) provide scalable preconditioners that are optimal for up to 4-th order problems that include highly discontinuous and heterogenous 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 (explained later). Both are based on simultaneous Dirichlet and Neumann solves within each substructure – one for preconditioning and one for residual calculation – only in reverse order to one another. Note that the Neumann solves – which lie at the heart of these methods – 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 primary 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 on a separate portion 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 3D nonlinear elasticity problem into a 2D linear cross-section analysis and a 1D geometrically exact beam analysis – based on Berdichevsky [18] and pioneered by Hodges and his coworkers [19]. Aeroelastic computations are performed on the beam, followed by a recovery of the 3D 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 configurations contain 3-D flexible load bearing components near the root 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, 22].

A second objective of this paper, therefore, is to develop a 3-D FEM analysis 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 CFD, physics-based models containing millions of grid points can carry out RANS computations using 100's of cores, routinely, in a research environment for the rotor, and even for the entire helicopter. Current research is focused today on coupling between CFD and relatively simple engineering-level rotor structural dynamics. The purpose of the second objective of this paper, therefore, is to explore the possibility of integrating 3D FEM as the physics-based model in the CSD domain.

A 3-D FEM analysis will provide enabling technol-

ogy for modeling advanced hingeless and bearingless rotors. It will provide enabling technology for calculating the stresses and strains directly on the critical load bearing components near the hub. It will provide a model that is a true representation of the 3-D structure, consistent with the high fidelity sought in large scale CFD computations. Thus, this research is primarily targetted towards large-scale, high-fidelity, HPC based comprehensive rotorcraft simulations. However, as a by-product, it will still provide a means for extracting 2-D sectional properties for generic structures, with which lower fidelity analysis can always be carried out. There is no question that such a capability will be powerful. The question is that of an efficient solution procedure. As in CFD, the tremendous capabilities of HPC is also envisioned here to be the key technology driver and enabler. The primary objective of this paper is therefore to address this key challenge directly.

Scope of Present Work

A 3-D FEM analysis for rotary wing dynamics is developed with emphasis on identifying the inertial terms that are unique to rotors. The 3-D FEM solver is then used to implement and analyze a parallel Newton-Krylov solver developed using the FETI-DP method of iterative substructuring. This solver is equipped with a Generalized Minimum Residual (GMRES) update, in addition to its traditional CG based update, to accomodate the nonsymmetric nature of rotary wing dynamics. However no formal proof of convergence or condition estimates are derived for this nonsymmetric operator.

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 that 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 multi-body dynamics, are all part of future work.

The paper is organized as follows. The second section describes the formulation of 3-D Finite Element Modeling (FEM) for rotors, 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 3D FEM analysis: geometry and grids, partitioning and corner selection, the hover prototype, and the transient forward flight prototype. The fifth section is focussed on scalability. Analyses were performed on up to 48 processors – the maximum available to the authors at present. 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 endeavour.

3D FINITE ELEMENTS FOR ROTORS

The Finite Element formulation is based on well established, standard procedures [23, 24]. The non-linear, geometrically exact implementation, follows incremental approach, using Green-Lagrange strain and Second Piola-Kirchoff stress measures, within a total Lagrangian formulation. The main contribution here is on identifying the inertial terms that are unique to rotorcraft.

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 X_0

$$X_0 = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{bmatrix} \quad \text{where} \quad x_{i,j} = \frac{\partial x_i}{\partial x_j^0} \tag{2}$$

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

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

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

$$\epsilon = (1/2)(C - I)$$

where C is the left Cauchy-Green deformation tensor given by

$$C = X_0^T\ X_0$$

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

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

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

$$\delta U = \int_V \sigma\ \ \delta\epsilon_{ij}\ \ dV \tag{4}$$

For non-linear analysis, an incremental procedure is followed. The variation in strain energy in the current state is expressed in terms of incremental deformations measured from a previous known state. It must be understood that the variation in strain in the current state is simply the variation in incremental strains. That is,

$$\delta\epsilon_{ij}(t+\Delta t) = \delta\Delta\epsilon_{ij} \tag{5}$$

where $\Delta\epsilon_{ij}$ is the incremental strain

$$\epsilon_{ij}(t+\Delta t) = \epsilon_{ij}(t) + \Delta\epsilon_{ij} \tag{6}$$

The incremental strain is related to the incremental deformations Δu_i. They are defined as follows

$$x_i(t) = x_i^0 + u_i(t)$$

$$x_i(t+\Delta t) = x_i^0 + u_i(t+\Delta t)$$

$$\Delta u_i = x_i(t+\Delta t) - x_i(t) = u_i(t+\Delta t) - u_i(t)$$

Substitution of the strain expression 3 in 6 and use of $u_i(t+\Delta t) = u_i(t) + \Delta u_i$ gives

$$\begin{aligned}\Delta\epsilon_{ij} =& \frac{1}{2}\left(\frac{\partial\Delta u_i}{\partial x_j^0} + \frac{\partial\Delta u_j}{\partial x_i^0} + \frac{\partial u_k}{\partial x_i^0}\,\frac{\partial\Delta u_k}{\partial x_j^0} + \frac{\partial\Delta u_k}{\partial x_i^0}\,\frac{\partial u_k}{\partial x_j^0}\right)\\ &+ \frac{1}{2}\,\frac{\partial\Delta u_k}{\partial x_i^0}\,\frac{\partial\Delta u_k}{\partial x_j^0} = \Delta\varepsilon_{ij} + \Delta\kappa_{ij} \qquad k = 1,2,3\end{aligned} \tag{7}$$

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

$$\delta\Delta\epsilon_{ij} = \delta\Delta\varepsilon_{ij} + \delta\Delta\kappa_{ij} \tag{8}$$

where

$$\begin{aligned}&\delta\Delta\varepsilon_{ij} =\\ &\frac{1}{2}\left(\frac{\partial\delta\Delta u_i}{\partial x_j^0} + \frac{\partial\delta\Delta u_j}{\partial x_i^0} + \frac{\partial u_k}{\partial x_i^0}\,\frac{\partial\delta\Delta u_k}{\partial x_j^0} + \frac{\partial\delta\Delta u_k}{\partial x_i^0}\,\frac{\partial u_k}{\partial x_j^0}\right)\\ &\delta\Delta\kappa_{ij} =\\ &\frac{1}{2}\left(\frac{\partial\Delta u_k}{\partial x_i^0}\,\frac{\partial\delta\Delta u_k}{\partial x_j^0} + \frac{\partial\delta\Delta u_k}{\partial x_i^0}\,\frac{\partial\Delta u_k}{\partial x_j^0}\right)\end{aligned} \tag{9}$$

Similarly decompose the stress

$$\sigma_{ij}(t+\Delta t) = \sigma_{ij}(t) + \Delta\sigma_{ij} \tag{10}$$

Use 8, 9 and 10 in 4 to obtain

$$\begin{aligned}\delta U = &\int_V \Delta\sigma_{ij}\ \delta\Delta\varepsilon_{ij}\ dV + \int_V \sigma_{ij}(t)\ \delta\Delta\varepsilon_{ij}\ dV +\\ &\underline{\int_V \sigma_{ij}(t)\ \delta\Delta\kappa_{ij}\ dV} + \int_V \Delta\sigma_{ij}\ \delta\Delta\kappa_{ij}\ dV\end{aligned} \tag{11}$$

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 11 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 inertial 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

$$\begin{aligned}\delta U = &\int_V D_{ijmn}\ \Delta\varepsilon_{mn}\ \delta\Delta\varepsilon_{ij}\ dV +\\ &\int_V \sigma_{ij}(t)\ \delta\Delta\varepsilon_{ij}\ dV + \int_V \sigma_{ij}(t)\ \delta\Delta\kappa_{ij}\ dV\end{aligned} \tag{12}$$

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-independant, 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

$$\begin{aligned}\int_V D_{ijmn}\ \Delta\varepsilon_{mn}\ \delta\Delta\varepsilon_{ij}\ dV + &\int_V \sigma_{ij}(t)\ \delta\Delta\kappa_{ij}\ dV\\ &= \delta W_E - \int_V \sigma_{ij}(t)\ \delta\Delta\varepsilon_{ij}\ dV\end{aligned} \tag{13}$$

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 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 $\tilde{\omega}^{IB/B}$ is the angular velocity of B relative to I and measured in B, and $\tilde{\omega}^{BI/I}$ 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

$$\begin{aligned}
r^{PI/I} &= C^{IB}(x^{BI/B} + r^{PB/B}) \\
\dot{r}^{PI/I} &= C^{IB}(v^{BI/B} + \dot{r}^{PB/B} + \tilde{\omega}^{BI/B} r^{PB/B}) \\
\ddot{r}^{PI/I} &= C^{IB}(\dot{v}^{BI/B} + \tilde{\omega}^{IB/B} v^{BI/B} + \ddot{r}^{PB/B} + \\
&\quad \tilde{\omega}^{BI/B}\dot{r}^{PB/B} + \tilde{\omega}^{BI/B}\tilde{\omega}^{BI/B} r^{PB/B} + \dot{\tilde{\omega}}^{BI/B} r^{PB/B})
\end{aligned} \tag{14}$$

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

$$\begin{aligned}
\delta r^{PI/I} &= C^{IB}(\delta x^{BI/B} + \delta\tilde{\psi}^{BI/B} r^{PB/B} + \delta r^{PB/B}) \\
&= C^{IB}(\delta x^{BI/B} - \tilde{r}^{PB/B}\delta\psi^{BI/B} + \delta r^{PB/B}) \\
&= C^{IB} R^T \delta q
\end{aligned} \tag{15}$$

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 its finite element degrees of freedom $\delta r^{PB/B} = R_b^T \delta q_b$. The kinetic energy is then

$$\delta T = \int_V (\delta r^{PI/I})^T \ \ddot{r}^{PI/I} \ dV \tag{16}$$

In general, frame B may contain a general flexible component. Consider a simple case for illustration. Let B be the undeformed blade rotating frame, containing the entire blade, with origin at the rotor hub. It undergoes control angle motions $\theta, \dot{\theta}, \ddot{\theta}$ with respect to another rotating frame, R, 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 $v^{BI/B} = \dot{v}^{BI/B} = 0$.

$$C^{IB} = \begin{bmatrix} c_\psi & -s_\psi c_\theta & s_\psi s_\theta \\ s_\psi & c_\psi c_\theta & -c_\psi s_\theta \\ 0 & s_\theta & c_\theta \end{bmatrix} \tag{17}$$

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

$$\omega^{BI/B} = \begin{pmatrix} \dot{\theta} \\ 0 \\ 0 \end{pmatrix} e^B + \begin{pmatrix} 0 \\ 0 \\ \Omega \end{pmatrix} e^I = \begin{pmatrix} \dot{\theta} \\ \Omega s_\theta \\ \Omega c_\theta \end{pmatrix} e^B \tag{18}$$

e^B and e^I are the basis vectors in the B and I frame repectively. The non-zero components of $\ddot{r}^{PI/I}$ in the kinetic energy expression 16 become

$$\ddot{r}^{PB/B} = \begin{pmatrix} \ddot{u}_1 \\ \ddot{u}_2 \\ \ddot{u}_3 \end{pmatrix} \tag{19}$$

$$2\tilde{\omega}^{BI/B}\dot{r}^{PB/B} = 2\begin{bmatrix} 0 & -\Omega c_\theta & \Omega s_\theta \\ \Omega c_\theta & 0 & -\dot{\theta} \\ -\Omega s_\theta & \dot{\theta} & 0 \end{bmatrix} \begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \end{pmatrix} \tag{20}$$

$$\tilde{\omega}^{BI/B}\,\tilde{\omega}^{BI/B}\,r^{PB/B} = \begin{bmatrix} -\Omega^2 & \dot{\theta}\Omega s_\theta & \dot{\theta}\Omega c_\theta \\ \dot{\theta}\Omega s_\theta & -\Omega^2 c_\theta^2 - \dot{\theta}^2 & \Omega^2 c_\theta s_\theta \\ \dot{\theta}\Omega c_\theta & \Omega^2 c_\theta s_\theta & -\Omega^2 s_\theta^2 - \dot{\theta}^2 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{21}$$

$$\dot{\tilde{\omega}}^{BI/B} r^{PB/B} = \begin{bmatrix} 0 & -\dot{\Omega}c_\theta + \Omega\dot{\theta}s_\theta & \dot{\Omega}s_\theta + \Omega\dot{\theta}c_\theta \\ \dot{\Omega}c_\theta - \Omega\dot{\theta}s_\theta & 0 & \ddot{\theta} \\ -\dot{\Omega}s_\theta - \Omega\dot{\theta}c_\theta & \ddot{\theta} & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{22}$$

The virtual displacement is simply the variation of the incremental displacements

$$\delta r^{PB/B} = C^{IB}\delta u \tag{23}$$

Thus, the kinetic energy 16 takes the following form

$$\delta T = \int_V \delta u^T \ \rho\,(T_m\,\ddot{u} + T_c\,\dot{u} + T_k\,u + T_f) dV \tag{24}$$

For example, with the simplest assumption of only a collective, $\dot{\theta} = 0$, and steady rotation, $\dot{\Omega} = 0$, we have from 19– 22 the following mass, damping, stiffness, and force

contributions from inertia.

$$
\begin{aligned}
T_m &= I_3 \\
T_c &= -2\Omega \begin{bmatrix} 0 & c_\theta & -s_\theta \\ -c_\theta & 0 & 0 \\ s_\theta & 0 & 0 \end{bmatrix} \\
T_k &= -\Omega^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\theta^2 & -s_\theta c_\theta \\ 0 & -s_\theta c_\theta & s_\theta^2 \end{bmatrix} \\
T_f &= T_k \left[x_1^0 \, x_2^0 \, x_3^0 \right]^T
\end{aligned}
\tag{25}
$$

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).

$$
dx_i = x_{i,\xi} d\xi = \sum_{k=1}^{N} H_{k,\xi} x_i^k \qquad i = 1, 2, 3 \tag{26}
$$

Similarly, differential changes $d\eta$ and $d\zeta$ create the vectors $x_{,\eta} d\eta$ and $x_{,\zeta} \zeta$ 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 $x_{,\xi} d\eta$ 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 incremental 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

$$
\delta W_E = \delta q^T \int_A p H^T \; x_{,\xi} \tilde{x}_{,\eta} \; d\xi \, d\eta = \delta q^T Q \tag{27}
$$

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

$$
\delta W_E = \delta q^T \int_A H^T \sigma_S \; x_{,\xi} \tilde{x}_{,\eta} \; d\xi \, d\eta = \delta q^T Q \tag{28}
$$

Because fluid stress are deformation dependant, 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 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 primary focus at present is on accuracy.

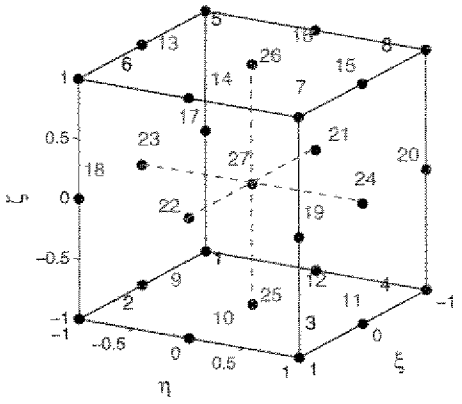


Figure 1: **27-node isoparametric brick element in curvilinear natural coordinates; 64 gauss integration points.**

Figure 1 shows a quadratic, Lagrangian, isoparametric 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

$$
x_i^0 = \sum_{a=1}^{N} H_a \; x_{ia}^0 \qquad \Delta u_i = \sum_{a=1}^{N} H_a \; \Delta u_i^a \tag{29}
$$

and therefore

$$
x_i = \sum_{a=1}^{N} H_a \; x_{ia} \qquad u_i = \sum_{a=1}^{N} H_a \; u_i^a \tag{30}
$$

where a is the elemental node point index, $N = 27$ here. The shape functions are expressed in element natural coordinates ξ, η, and ζ. We consider Lagrange polynomials in each direction.

$$
H_a = H_a(\xi, \eta, \zeta) = L_I^n(\xi) \; L_J^m(\eta) \; L_K^p(\zeta) \tag{31}
$$

In the present formulation $n = m = p = 2$; and $I, J, K = 1, 2, 3$. The second order Lagrange polynomials in, say x, are $x(x-1)/2$, $1 - x^2$, and $x(x+1)/2$. For example, based on the local node ordering shown in the figure, we have the shape function corresponding to node 11 as

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

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

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

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

$$\frac{\partial u_i}{\partial x_j^0} = \sum_{a=1}^{N} \left(\frac{\partial H_a}{\partial x_j^0} \right) u_{i\,a} \qquad \frac{\partial \Delta u_i}{\partial x_j^0} = \sum_{a=1}^{N} \left(\frac{\partial H_a}{\partial x_j^0} \right) \Delta u_{i\,a} \tag{32}$$

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

$$\begin{pmatrix} \frac{\partial H_a}{\partial \xi} \\ \frac{\partial H_a}{\partial \eta} \\ \frac{\partial H_a}{\partial \zeta} \end{pmatrix} = \underbrace{\begin{bmatrix} \frac{\partial x_1^0}{\partial \xi} & \frac{\partial x_2^0}{\partial \xi} & \frac{\partial x_3^0}{\partial \xi} \\ \frac{\partial x_1^0}{\partial \eta} & \frac{\partial x_2^0}{\partial \eta} & \frac{\partial x_3^0}{\partial \eta} \\ \frac{\partial x_1^0}{\partial \zeta} & \frac{\partial x_2^0}{\partial \zeta} & \frac{\partial x_3^0}{\partial \zeta} \end{bmatrix}}_{J} \begin{pmatrix} \frac{\partial H_a}{\partial x_1^0} \\ \frac{\partial H_a}{\partial x_2^0} \\ \frac{\partial H_a}{\partial x_3^0} \end{pmatrix} \tag{33}$$

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

$$J = \begin{bmatrix} H_{1,\xi} & H_{2,\xi} & \dots & H_{N,\xi} \\ H_{1,\eta} & H_{2,\eta} & \dots & H_{N,\eta} \\ H_{1,\zeta} & H_{2,\zeta} & \dots & H_{N,\zeta} \end{bmatrix} \begin{bmatrix} x_{11}^0 & x_{21}^0 & x_{31}^0 \\ x_{12}^0 & x_{22}^0 & x_{32}^0 \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ x_{1N}^0 & x_{2N}^0 & x_{3N}^0 \end{bmatrix} \tag{34}$$

The required derivatives are then

$$\begin{pmatrix} \frac{\partial H_a}{\partial x_1^0} \\ \frac{\partial H_a}{\partial x_2^0} \\ \frac{\partial H_a}{\partial x_3^0} \end{pmatrix} = \frac{1}{J} \begin{pmatrix} \frac{\partial H_a}{\partial \xi} \\ \frac{\partial H_a}{\partial \eta} \\ \frac{\partial H_a}{\partial \zeta} \end{pmatrix} \tag{35}$$

Henceforth the above derivatives are denoted as $H_{a,1}, H_{a,2}, H_{a,3}$ and $H_{a,\xi}, H_{a,\eta}, H_{a,\zeta}$. In addition, the first quantity in 32 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 7, the strain-displacement relation now takes the following form

$$\Delta\hat{\varepsilon} = (B_{L0} + B_{L1})\Delta q \tag{36}$$

where

$$\begin{aligned} \Delta\hat{\varepsilon}^T &= \Delta[\varepsilon_{11}\;\; \varepsilon_{22}\;\; \varepsilon_{33}\;\; 2\varepsilon_{12}\;\; 2\varepsilon_{23}\;\; 2\varepsilon_{13}] \\ \Delta q^T &= [u_{11}\, u_{21}\, u_{31}\;\; u_{12}\, u_{22}\, u_{32}\;\; \dots\;\; u_{1N}\, u_{2N}\, u_{3N}] \end{aligned} \tag{37}$$

and the expressions for B_{L0} and B_{L1} are given above. The first term in the strain energy 12 now becomes

$$\delta\Delta q^T \left(\int_V (B_{L0} + B_{L1})^T\; D\; (B_{L0} + B_{L1})\; dV \right) \Delta q \tag{38}$$

The second term is treated is the same manner to obtain

$$\delta\Delta q^T \left(\int_V (B_{L0} + B_{L1})^T\; \hat{\sigma}(t)\; dV \right) \Delta q \tag{39}$$

where $\hat{\sigma}(t) = [\sigma_{11}\;\; \sigma_{22}\;\; \sigma_{33}\;\; \sigma_{12}\;\; \sigma_{23}\;\; \sigma_{13}]^T$. Now consider the non-linear incremental strain $\Delta\kappa_{ij}$ as defined in 7. Clearly, the strain is non-linear and a strain-displacement relationship cannot be found. However it has a quadratic form, and hence can be re-arranged as

$$\delta\Delta q^T \left(\int_V \bar{B}_{NL}^T\; \bar{\sigma}_{i,j}(t)\; \bar{B}_{NL}\; dV \right) \Delta q \tag{40}$$

with the matrices having special forms

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

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

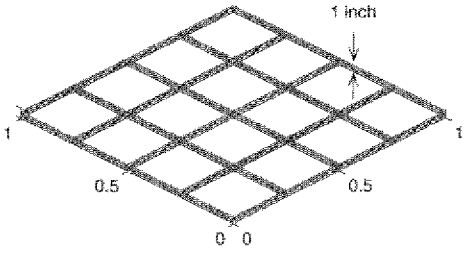


Figure 2: **A cantilevered plate using (4,4,4) brick element grid**

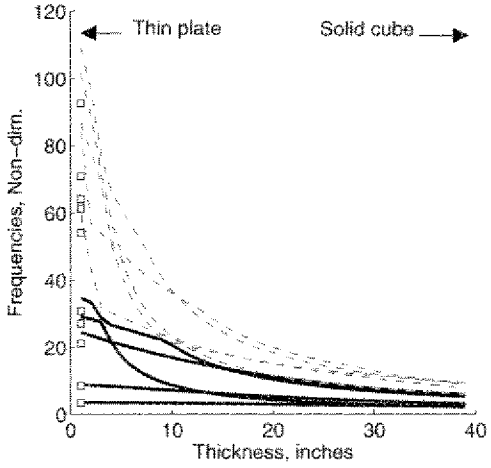


Figure 3: **Verification of brick elements; Natural frequencies of a (4,4,4) solid block approaching Kirchoff plate frequencies (symbols) with reduced thickness**

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

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

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

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

VERIFICATION of 3-D FEM

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

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

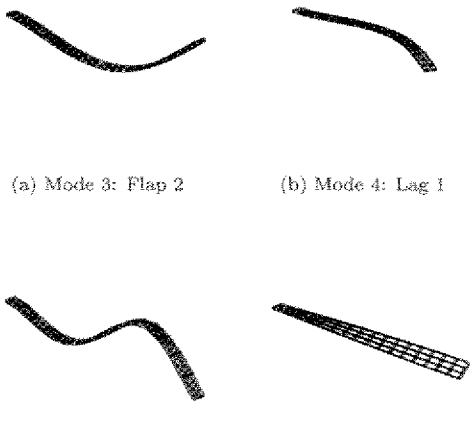
Table 2: **Beam torsion frequencies vs. cross-section grid refinement; 8 spanwise elements; nondimensionalized w.r.t.** $\sqrt{GJ/IL^2}$**; EB values are** 1.571 **and** 4.712**; Beam dimensions are** $L \times c \times c, L = 100c$

Thickness	8 3×3	16 3×3	20 3×3
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: **Second torsion frequency vs. spanwise grid refinement;** 3×3 **cross-section; nondimensionalized w.r.t.** $\sqrt{GJ/IL^2}$**; EB values are** 1.571 **and** 4.712

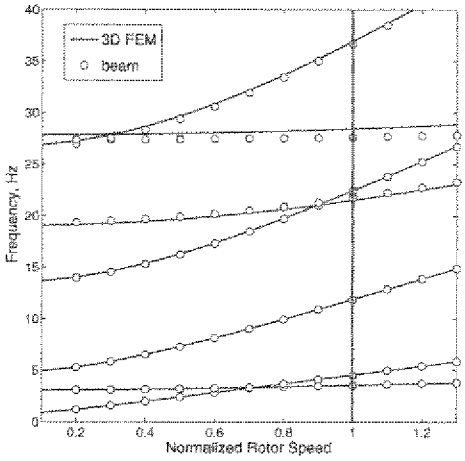
Thin Plate Frequencies

The locking-free behavior of the brick elements, in shear, is verified by re-producing Kirchhoff plate frequencies for a thin plate, using a $(4,4,4)$ brick mesh Fig. 3. The plate frequencies are obtained from converged 2D rectangular plate finite elements $(20,20)$ and are validated easily with documented classical solutions. The discrepancy at the higher modes are resolved using a finer mesh converged solution, see Table 1. The residual differences are due to shear, not present in the Kirchhoff solution, but present in the brick solution.



(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 hingeless blade of aspect ratio 15, thickness 25% chord, and rectangular cross-section; 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: **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.

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-inplane hingeless rotor;** $16 \times 3 \times 3$ **grid in 3-D; L: Lag, F: Flap, T: Torsion**

Slender Beam Frequencies

Next, the 3D element is verified using a slender of uniform geometry that behaves as a beam over a reasonably large variation in thickness and aspect ratio. The bending frequencies are easy to re-produce, torsion in general requires greater resolution. An uniform slender beam of aspect ratio 100 and square cross-section, i.e., the dimensions are $100c$, c, and c, in length, width, and thickness, is considered. The torsion frequencies converge towards Euler-Bernoulli numbers with two to three cross-section elements 2. The remaining difference stems from spanwise resolution.

The effect of spanwise resolution is shown in the first row of Table 3, starting from $8 \times 3 \times 3$ grid as the baseline. Spanwise 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 spanwise refinement for each thickness There is increased deviation from Euler-Bernoulli values as the thickness reduces. With the thickness fixed at $c/4$ and the 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 relatively constant. At aspect ratio 5 there is still only an error of $5 - 6\%$. This deveation is expected from the physics of the problem and is not an error in the FEM formulation.

Consider the configuration with length $20c$, width c, and thickness $c/4$. The rotating modes of this simple beam structure are shown in Fig. 4. The frequency plot, Fig. 4(e), shows the the beam frequencies are almost exactly reproduced by 3D 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 from Euler-Bernoulli frequencies in the non-rotating case for this level of grid refinement. For this structure, the torsion frequency is relatively high, and occurs only as the sixth mode. The rotating frequencies at $\Omega = 27$ rad/s are given in Table 5, both for a beam and the 3-D analysis.

ITERATIVE SUBSTRUCTURING USING PARALLEL KRYLOV SOLVER

A parallel Newton-Krylov solver is developed to provide an efficient and scalable 3-D FEM solution.

Large-scale structural dynamics problems are solved most efficiently using the method of substructures. Substructuring involves partitioning a structure into non-overlapping subdomains. It is the most accurate method, because each subdomain can have its own internal solver depending on its local condition number. Almost always, a direct factorization is preferred. A real structure contains significant heterogeneities – thin and slender geometries, plate and shells by biharmonic PDEs, 3-D bricks with high bandwidths (> 1000), non-linear materials, and constraint forces – factors that routinely give rise to condition numbers of 10^8–10^{10}.

Modern methods of iterative substructuring provides a domain-decomposition based preconditioned iterative solver for the interface problem. The interface problem need not be of primal type, but can also be of dual type. A primal problem consists of variables that are a subset of the original unknowns, e.g. the displacements at the interface. A dual problem consists of variables that are not a subset of the original unknowns but whose equality must still be gauranteed, e.g. the reaction forces at the interface. Depending on the problem dual variables differ – bending problems will involve transverse shears and moments, whereas a plane stress or strain problem will onvolve only in-plane stresses. Regardless of type, whether primal or dual, all finite element interface descriptions are precisely the discrete equivalents of the Poincarè-Steklov operator.

The interface has attractive spectral properties as a result of which it is more amenable to iterative solve. In particular, unlike the substructures themselves, the condition number of the interface problem grows at a rate that is an order slower compared to the original problem, e.g., by $0(h^{-1})$ for second order and by $0(h^{-3})$ for fourth order PDEs. However, it also grows, necessarily, at $0(H^{-2})$ where H is the subdomain size. For a second-order, elliptic, positive definite, and coercive operators, the precise number for uniform finite element meshing given by [25]

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

where H is the maximum and H_m is the minimum subdomain diameter. The main objective of iterative substructuring is to provide preconditioners such that the preconditioned interface problem has a condition number independant of both h and H. Such a preconditioner is an optimal preconditioner. At the same time, it must be constructed in parallel, subdomain by subdomain, requiring only communication between subdomains but no other serial operation – otherwise the primary purpose of iterative substructuring is defeated.

The dependance on $0(H_m^{-2})$ cannot be prevented without a coarse grid solver — communication only between neighboring subdomains will always show this dependance. Thus, a coarse problem i.e. a general mechanisms to propagate local information globally, is a central requirement of any scalable solver.

The general building blocks of a preconditioned Krylov solver (for solving $M^{-1}Ax = M^{-1}b$ are: (1) residual calculation $r = b - Ax$, (2) preconditioning $M^{-1}r$ and, (3) a matrix-vector multiplication Ax. An iterative substructuring algorithm provides these building blocks in a subdomain independant, parallel manner. Once the building blocks are provided, constructing a Krylov solver is trivial. Unlike the CG update, the GMRES update, however, poses its own parallelization issues due to the Arnoldi procedure.

The FETI-DP Algorithm

In iterative substructuring, the subdomain interface nodes are first separated into vertex, edge and face nodes. The vertex nodes and a subset of edge nodes are then designated as corner nodes. In FETI-DP, the degrees of freedom (DOFs) associated with the corner nodes are formulated as a primal interface. The DOFs associated with the rest are formulated as a dual interface. The corner nodes form a coarse problem that propagates local subdomain information globally. Because the number of DOFs associated with a corner depend on the order of the problem, i.e. 3 DOFs for 2nd order brick FEM or 6 DOFs for 4th order plate or shell elements, it automatically renders the coarse mesh appropriately denser with increase in order. The FETI-DP method and its implementation in this study is entirely based on the work of Refs. [15, 16]. A detailed exposition of our implementation is not provided here. A brief description is provided below summarizing its key aspects.

For a given subdomain, if its nodes are re-ordered with internal nodes I^s first, followed by interface nodes Γ_E^s, and then corner nodes Γ_C^s (for a selection of corner nodes in 3-D, see next section), then a subdomain matrix, say the stiffness matrix, takes the following form

$$K^s = \begin{bmatrix} K_{II}^s & K_{IE}^s & K_{IV}^s \\ K_{EI}^s & K_{EE}^s & K_{EV}^s \\ K_{VI}^s & K_{VE}^s & K_{VV}^s \end{bmatrix} = \begin{bmatrix} K_{RR}^s & K_{RV}^s \\ K_{VR}^s & K_{VV}^s \end{bmatrix} \tag{41}$$

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

$$f^s = \begin{pmatrix} f_R^s \\ f_V^s \end{pmatrix} \qquad u^s = \begin{pmatrix} u_R^s \\ u_V^s \end{pmatrix} \tag{42}$$

where

$$u_R^s = \begin{pmatrix} u_I^s \\ u_E^s \end{pmatrix} \qquad f_R^s = \begin{pmatrix} f_I^s \\ f_E^s \end{pmatrix} \tag{43}$$

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 gauranteed upon convergence.

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

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

$$B_R^s = \begin{array}{c} \longleftarrow I^s \longrightarrow \quad \longleftarrow \Gamma_E^s \longrightarrow \\ \left[\begin{array}{ccc|c} 0 & & & \\ & \ldots & & B_E^s \\ & & 0 & \end{array}\right] \begin{array}{c} \uparrow \\ \Gamma_E^s \\ \downarrow \end{array} \end{array} \tag{44}$$

where B_E^s is a diagonal matrix with entries $+1$ or -1. The dual-primal procedure computes a set of dual variables (auxilliary variables that are not part of the original problem) which on convergence allows the recovery of the subdomain internal and edge nodes as

$$K_R^s u_R^s = f_R^s - {B_R^s}^T \lambda^s \tag{45}$$

and the global corner nodes as

$$K_{VV}^* \; u_V^g = F_{RV}^T \lambda + f_V^* \tag{46}$$

The corner problem, which propagates error globally, is a coarse grid problem that is also constructed subdomain by subdomain. Formally,

$$\begin{aligned} K_{VV}^s &= \sum_s {B_V^s}^T \left[K_{VV}^s - {K_{RV}^s}^T {K_{RR}^s}^{-1} K_{RV}^s \right] B_V^s \\ F_{RV}^T \lambda &= \sum_s {B_V^s}^T \left[-{K_{RV}^s}^T {K_{RR}^s}^{-1} {B_R^s}^T \right] \lambda^s \\ f_V^* &= \sum_s {B_V^s}^T \left[{K_{RV}^s}^T {K_{RR}^s}^{-1} f_R^s - f_V^s \right] \end{aligned} \tag{47}$$

The solve, however, is carried out in every subdomain. Thus, before the interface iterations begin, the subdomain contributions to the left hand side of the corner problem are constructed and factorized in every subdomain, and globally communicated. Thereafter, during the interface Krylov iterations, the coarse problem solve is only a repeated right hand side solve.

The building blocks of the Krylov iteration: residual calculation, preconditioning, and matrix-vector multiplication procedure are briefly stated below.

Residue calculation

The first part of the residual r_1 is obviously

$$\begin{aligned} r_1 &= \sum_s B_R^s u_R^s \\ &= \sum_s B_R^s {K_{RR}^s}^{-1} f_R^s - \sum_s B_R^s {K_{RR}^s}^{-1} {B_R^s}^T \lambda^s \end{aligned} \tag{48}$$

The second part of the residual r_2 is obtained after the coarse solve

$$r_2 = -F_{RV} \; u_V^g = \sum_s B_R^s {K_{RR}^s}^{-1} K_{RV}^s B_V^s u_V^s \tag{49}$$

The residual is then $r = r_1 + r_2$. Note that the residual calculation is based on subdomain Neumann solves. Therefore, the subdomain partitioning, and corner node selection must ensure null kernels.

Preconditioner

The residuals are used to construct subdomain fluxes

$$\eta^s = K_{EI}^s w^s + K_{EE}^s \, B_E^s r_E^s \tag{50}$$

with w^s obtained using subdomain Dirichlet solves

$$K_{II}^s w^s = {K_{II}^s}^{-1} K_{IE}^s \, B_E^s r_E^s \tag{51}$$

from which the preconditioned residual follows

$$M^{-1} r = \sum_s B_E^s \eta^s \tag{52}$$

Expanding the above expressions, we have, formally

$$M^{-1} = \sum_s B_R^s \begin{bmatrix} 0 & 0 \\ 0 & S_{EE}^s \end{bmatrix} {B_R^s}^T \tag{53}$$

where S_{EE}^s 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

$$\eta_s = K_{EE}^s \, B_E^s r_E^s$$

This leads formally to

$$M^{-1} = \sum_s B_R^s \begin{bmatrix} 0 & 0 \\ 0 & K_{EE}^s \end{bmatrix} {B_R^s}^T \tag{54}$$

The subdomain Schur complement matrices have been approximated here by their leading terms. The two preconditioners above are termed the *Dirichlet* and *Lumped* preconditioners. All results shown later in this paper use the Dirichlet preconditioner, even though for 3-D brick problems the Lumped preconditioners are known to be faster.

Matrix-vector multiplication

This is identical to the residue calculation, except $f_R^s - {B_R^s}^T \lambda^s$ is now replaced with ${B_R^s}^T p^s$. Here p^s is the subdomain restriction of the vector to the multiplied.

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

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

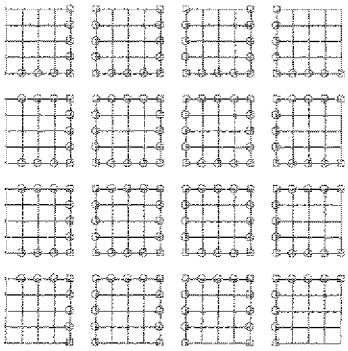


Figure 5: **A 4×4 plate partitioning; 16 elements in each partition. Interface, corner, and boundary nodes shown in red, blue, and green respectively.**

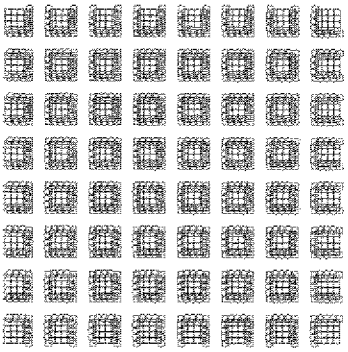


Figure 6: **A 16×16 plate partitioning; 16 elements in each partition.**

. That is, if the subdomains have size H, and the finite element mesh has size h, then the condition number of the interface grows only as H/h. The condition number determines the iteration count required for convergence. For optimal scalability algorithm, the iteration count does not grow with the number of subdomains as long as the mesh within each subdomain is refined to keep H/h constant. Thus, a bigger problem with additional subdomains require the same iteration count as a smaller problem as long as both contain the same mesh resolution.

The optimality of the algorithm is verified on a plate bending problem. Plate bending, like beam bending, is governed by 4-th order partial differential equations and is considered a challenge for iterative solvers because their condition numbers grow at a rate $0(h^{-4})$. Tables 6 and 7 show the change in iteration count with decreasing values of h and H respectively. The iteration count increases with increase in H/h and decreases with decrease in H/h. 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 – even though the latter is four times as big as the former.

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: **Number of substructures fixed n_s=16 (4×4 mesh partition). Iteration count vs. increase in problem size.**

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: **Problem size fixed DOFs=1260 (h=1/24 mesh). Iteration count vs. incrase in number of substructures.**

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

Table 8: **Problem size per substructure fixed H/h=4 (16 elements per substructure). Iteration count vs. increase in problem size.**

Parallel Implementation of CG

A standard Conjugate Gradient (PCG) 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 $k = 1, 2, \ldots$
$z_{k-1} = \mathbf{M^{-1} r_{k-1}}$
$\xi_k = \left(z_{k-1}^T r_{k-1}\right) / \left(z_{k-2}^T r_{k-2}\right)$ with $\xi_1 = 0$
$p_k = z_{k-1} + \xi_k p_{k-1}$ with $p_1 = z_0$
$\gamma_k = \left(z_{k-1}^T r_{k-1}\right) / \left(p_k^T \mathbf{F p_k}\right)$
$\lambda_k = \lambda_{k-1} + \gamma_k p_k$
$r_k = r_{k-1} - \gamma_k F p_k$

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 predonditioned 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 it is desired when thousands of distributed memory nodes are eventually used.

Parallel Implementation of GMRES

A GMRES update require an Arnoldi procedure and a solution of a least-square problem. A Reorthogonalized Classical Gram-Schimdt Arnoldi procedure is implemented in this study, based on the seminal work of Daniel et al. [28]. On the one hand, this algorithm produces high levels of orthogonalization (down to machine precision) and is superior to Modified Gram-Schmidt [29]. On the other hand, it remedies the unacceptable communication costs of the latter (explained below). Note that, a Classical Gram-Schimdt (i.e., 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 iterate 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 procedure 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 iterate 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, iteration after iteration, to n and terminates in at most n iterations. Each iteration requires every previous basis vector. A restarted version of GMRES, on the other hand, restricts the expansion to, say, m dimensions and restarts the Arnoldi procedure using x_m as it new initial guess. These restarts are called the outer iterations. Given below is the restarted GMRES(m) algorithm.

$\lambda_0 = 0; \quad \mathbf{r_0} = \mathbf{d} - \mathbf{F}\lambda_0$

$z_0 = M^{-1} r_0$
$\beta = ||z_0||_2$
for $k = 1, 2, \ldots$ till convergence
  $v_1 = z_{k-1}/\beta$
  Arnoldi procedure
  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 $\bar{H}_m$.
  $\lambda_k = \lambda_{k-1} + V_m y_m$
  $\mathbf{r_k} = \mathbf{d} - \mathbf{F}\lambda_\mathbf{k}$
  $\mathbf{z_k} = \mathbf{M^{-1}} \mathbf{r_k}$
  $\beta = ||z_k||_2$
end

The Arnoldi procedure is the heart of the algorithm. It requires three steps. For a given A and an initial vector v_1, the m basis vectors are constructed as:

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 $v_{j+1} = w_{j+1}/h_{j+1,j}$.

Orthogonalization is the main step. Traditionally a Modified Gram-Schmidt algorithm is always preferred at this step (over Classical Gram-Schmidt) because of its numerical stability. It is as follows. The synchronization points are underlined.

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

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-Schimdt, the projection and the substraction steps could be carried out separately, with a single synchronization step in-between. However, because Classical Gram-Schimdt is unstable (though mathematically equivalent to Modified Gram-Schimdt), a second orthogonalization step is needed. Thus, the final Reorthogonalized Classical Gram-Schmidt algorithm is as follows.

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

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

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

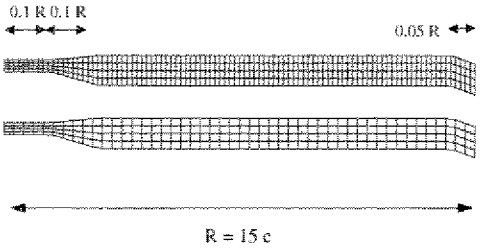


Figure 7: **Planform of prototype rotor blade; $c = 0.53$ m; two different spanwise grid resolutions used in this study are shown.**

3-D FEM ROTOR ANALYSIS COMPONENTS

In this section, the main components of the 3-D rotor FEM analysis are described. They are the 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.

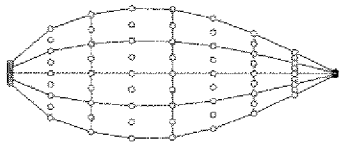


Figure 8: **Cross-section of prototype rotor blade; 5% t/c, exaggerated scale; 4×4 bricks with internal nodes.**

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 independant of the type of airloads, control angle variation, grid, and material constitution. The objective is to study the numerical scalability of the Newton-Krylov solver and the practical scalability of its present implemention.

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 accomodate 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 at the 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. $n_1 \times n_2 \times n_3$ 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: **3D FEM Rotor Grids**

Each finite element, naturally, can accomodate its own constitutive material model and ply direction – we

use simple isotropic properties: $E = 73$ GPa; $\mu = 0.3$; and $\rho = 2700$ kg/m^3 (corresponding to Aluminum).

Along with the dimension $c = 0.53m$, these generate similar order of magnitude non-dimensional values of stiffness and inertia as soft inplane 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) baseline, and two and four times that amount, to generate large deformation cases. The airloads imposed on the blade are prescribed – they act only on the top surface and they are normal pressure airloads. Thus, they 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 – not just any partitioner will do. The generation of subdomain grids from a global grid via a re-calculation of the finite element connectivities is straight-forward. Partitioners are widely available in public domain, that carry out this task intelligently 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.

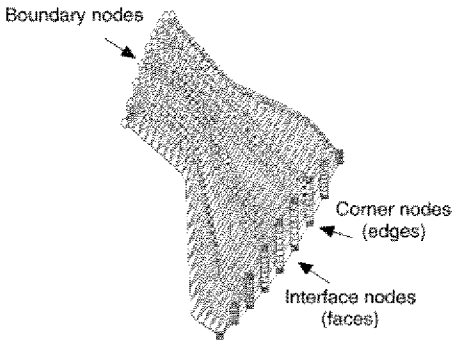


Figure 10: **Substructure node designation for one-way partition.**

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 the grid type as our simple grid generator. Namely, that the cross-sectional grids must remain same throughout the span, regardless of variations in geometry.

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

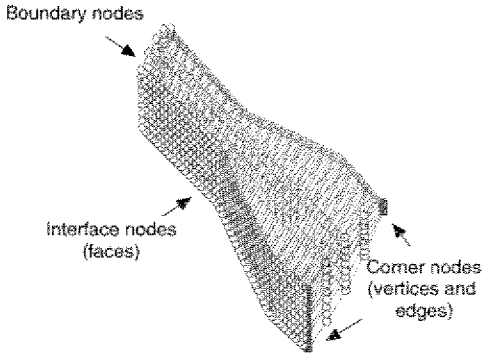


Figure 11: **Substructure node designation for two-way partition. Corner nodes are edge nodes connecting more than two substructures and those occuring at the boundaries.**

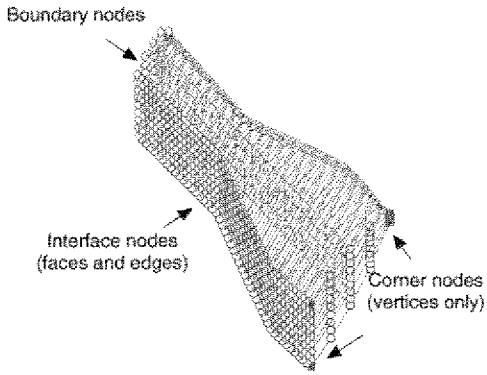
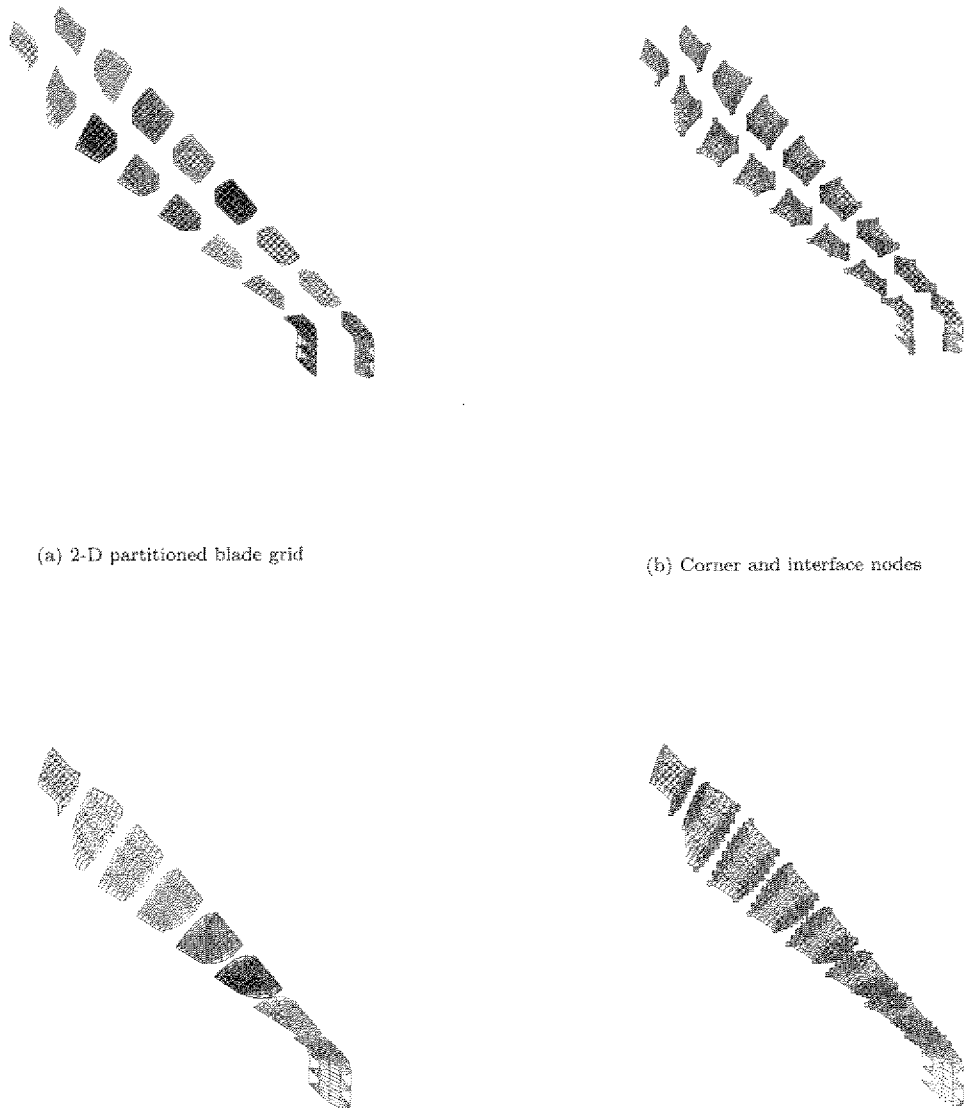


Figure 12: **Optimal substructure node designation for two-way partition. Corner nodes are vertex nodes including those occuring at the boundaries.**

The partitioner performs the following tasks:

1. Designates the corner nodes.
2. Reorders the subdomain nodes into interior, face, edge, vertex, and boundary nodes. Recalculates the subdomain finite element connectivities.
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.



(a) 2-D partitioned blade grid

(b) Corner and interface nodes

(c) 1-D partitioned blade grid

(d) Corner and interface nodes

Figure 9: **Partitioning of blade grid into substructures or subdomains. The corner nodes must ensure a null kernel for each substructure. Each substructure will be solved in a separate processor.**

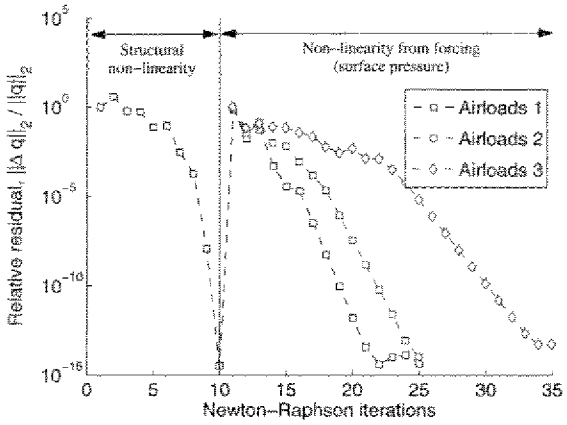


Figure 13: **Convergence of Newton-Raphson outer iterations for structural and forcing non-linearities in hover**

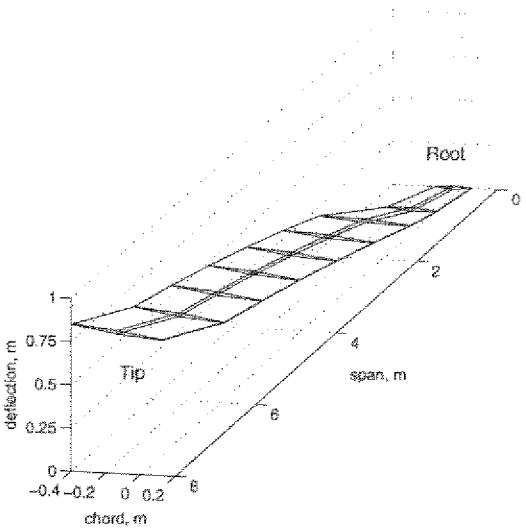


Figure 14: **Blade steady deflection in hover using prescribed pressure airloads (only grid outlines are shown)**

Corner Selection

For a generic 3D 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. Now consider the 2-way partition of Fig. 9(a). The nodes on the subdomain edges are immediately designated as corner nodes. It is clear, however, that this definition makes the two substructures at the tip end (or extremeties of the tip end in case of more than two chordwise strips) indefinite. Each substructure then carries a rotational rigid body mode. Thus, the definition of corner nodes most include in addition, those edge nodes, that are common to only two subdomains, but which occur at the boundaries of the structure. With this definition, the corner nodes are now as shown in Fig. 9(b). This definition also enables the selection of corner nodes for the 1-D partition in Fig. 9(c), otherwise, there would be no corner nodes. For a very large-scale 3D problem, a large number of corner nodes is generated by this procedure, leading to a moderately large coarse problem. A superior choice of corner nodes for a 3-D problem is simply the subdomain vertices, and like before, additionally those that occur at the boundaries. 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. 9(b). Note, the corner nodes define super elements that constitute the coarse problem, and, for a 2-D partition the optimal selection can leave the super elements without internal nodes. The concern for element locking under these circumstances require a closer examination that has not been carried out yet.

Node Reorder

The reordering 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* within each substructure. The natural order is then associated with a *reordered* order, and its reverse, with an association back to natural. In addition, The natural order is associated with the global order as the geometry and material constitution is defined in the latter.

Domain Connectivity

Domain Connectivity is merely a matter of bookkeeping. For a Lagrangian problem, the connectivity remains static, and needs to be calculated only once for a grid. Because of the non-floating, non-overlapping, and conforming nature of the partitions and elements, there are 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 internal 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.

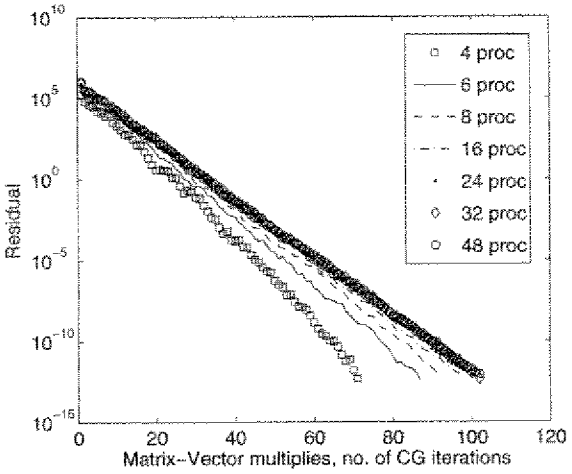


Figure 15: **Convergence pattern of FETI-DP/Preconditioned Conjugate Gradient updates for simulations on 4 to 48 processors.**

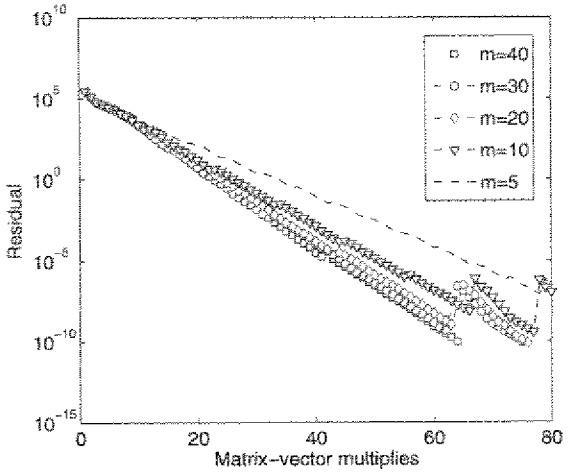


Figure 16: **Convergence of GMRES updates for various restart numbers; all calculations use 32 processors.**

Steady Hover Prototype

The steady hover (ideal) prototype simply solves for blade response using a prescribed pressure airload at a constant collective angle. The non-linear solution procedure uses Newton-Raphson outer iterations around FETI-DP inner solves. The FETI-DP inner solves use a Conjugate Gradient (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 initial part of the plot. Around 8 to 10 iterations are required for a tight convergence. Subsequently, the rotor stiffness matrices can be updated only at certain intervals if desired, while using modified Newton in between. Once the structural non-linearities are converged, the pressure airloads are imposed on the blade. The convergence of the airload iterations are shown in the same plot in the right hand side. The two parts are separated as conceptually these are fluid-structure iterations where the geometric stiffness need not be updated. The results shown, however, are with fully updated stiffnesses in every iteration. The direction of the pressure airloads is deformation-dependant and unknown a-priori. Thus, equilibrium iterations are required.

In each iteration, the virtual work is calculated based on the previous deformation state. An alternative, and more rigorous, approach is to linearize the forcing using incremental displacements. This leads to a non-symmetric stiffness contribution, which is however easily handled by replacing the geometric stiffness J with $1/2(J+J^T)$ as 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 deformation corresponding to airloads 3 is shown in Fig. 14.

Within each Newton iteration is the Krylov solver, FETI-DP with CG updates. The convergence criteria 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. We will always show the convergence corresponding to the first Newton iteration as it begins with a zero guess and takes the most number of iterations. The scalability (fixed problem size) of these calculations are examined in more detail in the next section. Here, we note that the number of iterations increase with an increase in the number of processors, but only gradually – the feature of an algorithm where the preconditioned interface problem grows slowly, and only at a polylogarithmic rate with increase in the number of subdomains. Physically, it means that the increasing coarse problem allows a greater transfer of substructure information across the global structure.

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 in a time-marching procedure. We consider a Newmark scheme with a 5^o azimuth step. The control angle variation is taken as $\theta(\psi) = 20^o + 5^o \cos\psi - 5^o \sin\psi$. The dynamic stiffness now contains the complete inertial terms and non-symetric due the gyroscopic damping. The FETI-DP inner solves now use a Generalized Minimum Residual (GMRES) update.

The matrix structure will be identical in every time step, therefore, for purposes of scalability study it is enough to consider only one. Issues related to paralleliza-

tion arise immediately due to the presence of the Arnoldi algorithm within GMRES. This issue, its resolution, and its impact on scalability if any, are described and studied in detail in the next section. Unlike CG where each update require only two previous updates for construction, in GMRES, each update require information from every previous updates. For example, as shown earlier in Fig. 15 if 100 iterations are required, 100 such vectors have to be stored. A standard procedure is to use a restarted version where only m previous updates are used at a time. Once a set of m updates are obtained an estimate of the solution is constructed. The m updates are then thrown away and the construction of a new set begins with the current estimate of the solution.

Figure 16 shows that for the small grid size used here, the convergence pattern is similar over a broad range of restart parameter – even $m = 5$ is adequate to obtain convergence. For purposes of a realistic scalability study, however, we will consider $m = 30$, 40, and 50. These are deemed adequate for large-scale structural configurations with dense interface matrices.

SCALABILITY OF 3-D ROTOR ANALYSIS

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

Steady Hover

n_s	FE	Subdom LU	Coarse	FETI-DP /PCG	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 vs. number of substructures on a single processor**

Consider the grid of size $96 \times 4 \times 2$. It is partitioned into 6 to 48 substructures using a 2-way partitioning, i.e. substructure arrangement similar to Fig. 9(a) having 3×2 to 24×2 blocks. The solver time for each of these problems are shown in Fig. 17. The importance of substructuring is immediately apparant. There is a steep drop in solution time with increasing number of substructures. For any problem of a fixed size, a condition of diminishing return must eventually be reached, with an optimal number of substructure producing the minimum solution time for that problem size. 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 between 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 procedure. A good partitioner and corner selector will keep this region flat, a poor one will produce a steep rise.

n_s	FE	Subdom LU	Coarse	FETI-DP /PCG	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 vs. number of processors; each processor contains one substructure**

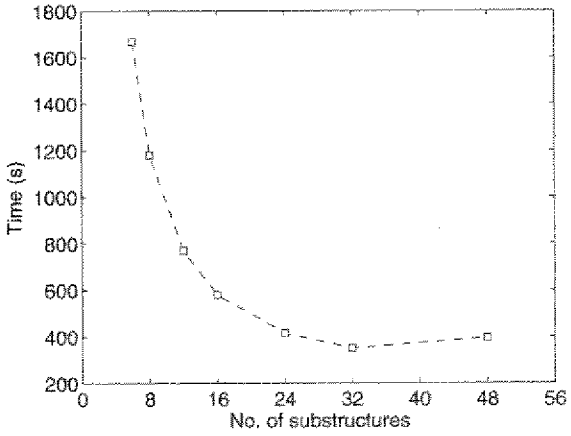


Figure 17: **Solver time vs. number of substructures for calculations on a single processor.**

A parallel implementation solves each substructure on a separate processor. Note that, it is important to calculate the speed-up obtained, using the single processor time with the same number of substructures. That is, the solver time with, say 32 processors, must be compared 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 paralleliza-

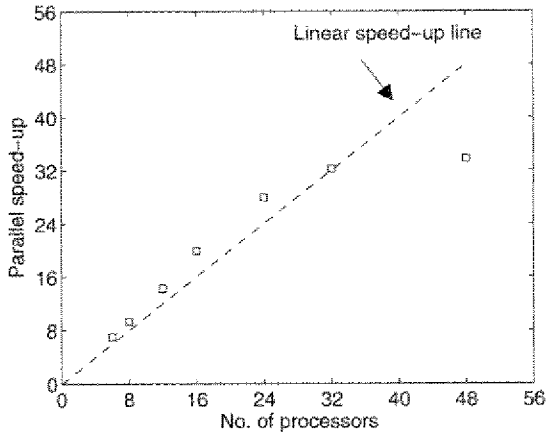


Figure 18: **Parallel speed-up for calculations on multiple processors; each substructure solved in a separate processor.**

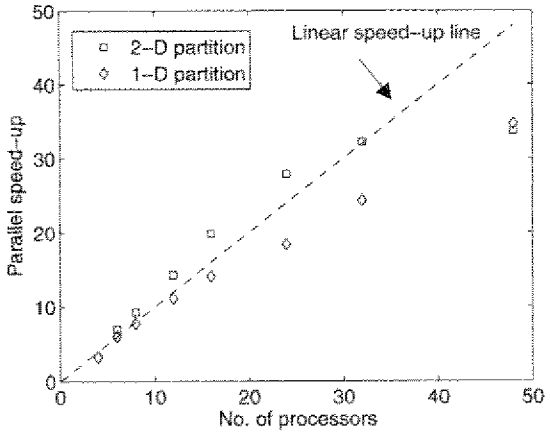


Figure 20: **Effect of grid partitioning and corner selection on parallel speed-up for calculations on multiple processors.**

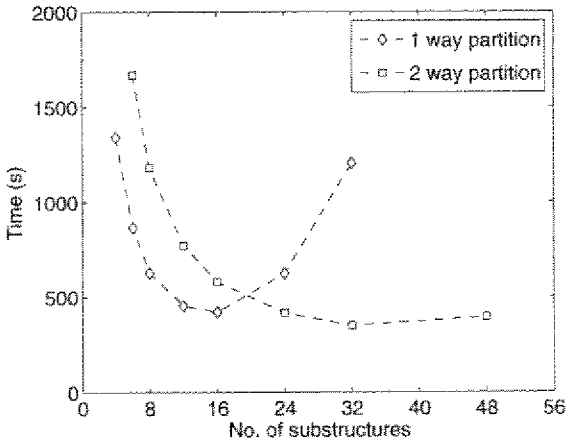


Figure 19: **Effect of grid partitioning and corner selection on solver time as a function of number of substructures; calculations on a single processor.**

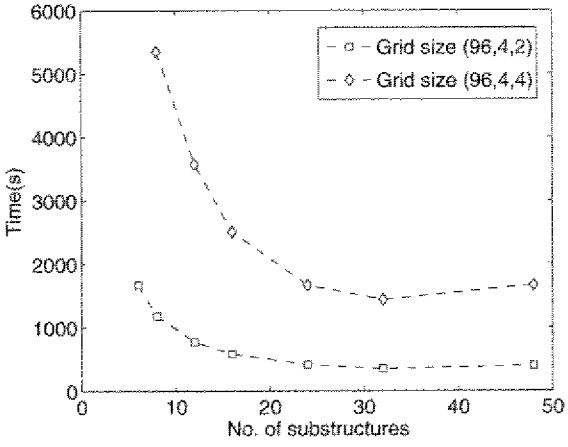


Figure 21: **Two different problem sizes with the same substructure optimality; solver time vs. number of substructures on a single processor.**

tion 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 10.8s whereas the 16 processor run takes 29.2s, i.e. less than half the time. What the speed-up plot shows is that 32 processor run the problem exactly 32 times faster compared to a single processor using 32 substructures.

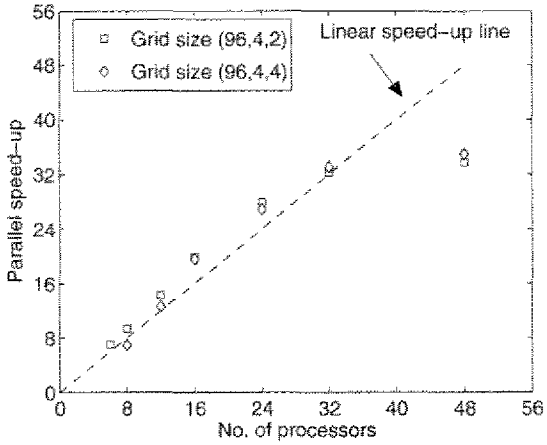


Figure 22: **Parallel speed-up for problem sizes with the same substructure optimality; solver time vs. number of substructures.**

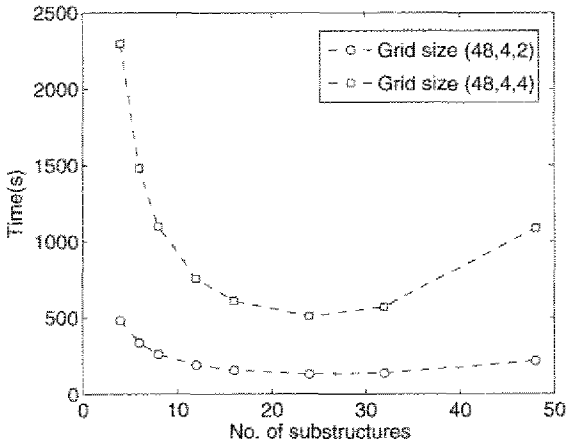


Figure 23: **Two different problem sizes with the same substructure optimality; solver time vs. number of substructures on a single processor.**

The drop off in scalability beyond 32 processors is studied using the exact timings for the different parts of the computation. The timings for the single processor and parallel calculations are given in Tables 10 and

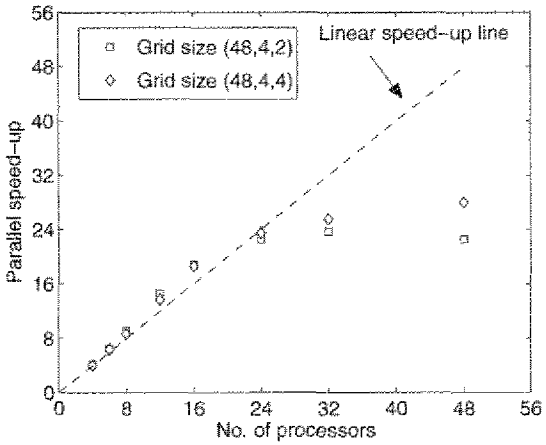


Figure 24: **Parallel speed-up for problem sizes of same substructure optimality.**

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. The 'Solver total' time consists of three parts: (1) 'Subdom LU' time, which refers to the subdomain factorization, (2) 'Coarse' time, which refers to the coarse problem factorization and communication, and (3) the 'FETI-DP/PCG' 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 as expected – as the purpose of the coarse problem is precisely that – but the main 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 should be to keep the growth beyond substructure optimality to as gradual as increase as possible. This has no bearing upon scalability with respect to 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-way partitioning as illustrated in

Figs. 9(c), 9(d), and 10, generates a timing and scalability plot as shown in Figs. 19 and 20. The 2-way 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 gaurantee a linear speed-up at least upto 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.

The linear speed-up range is not a function of problem size but of substructure optimality. For example, Figs. 21 and 22, compares 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 $48 \times 4 \times 2$ still shows a linear speed-up up to 24 processors equal to its value of substructure optimality. In the same manner as a grid of $48 \times 4 \times 4$ which is twice its size.

In summary, for a given problem size, the present solver shows a perfectly 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. Problem 3 has 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 an approximately 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.

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 showing scalability with respect to problem size up to limits of substructure optimality.**

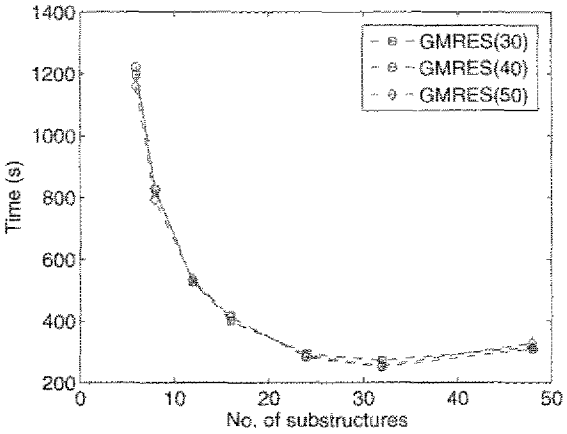


Figure 25: **Solver time vs. number of substructures on a single processor; FETI-DP/PGMRES.**

Transient Forward Flight

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

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/PGMRES solver times (secs.) with Modified Gram-Schimdt and Classical Gram-Schimdt with Reorthogonalization based Arnoldi procedures.**

The scalability of the parallel FETI-DP/PGMRES solver involving the non-symmetric matrices in forward flight shows as linear a trend as those of the FETI-DP/PCG solver in hover. The single processor timings are shown in Fig. 25. All calculations use the Reorthogonalized Classical Gram-Schimdt 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 incurr increasing communcation costs for a parallel calculation. However, for the small size problem considered here, there is no discernable 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-Schimdt

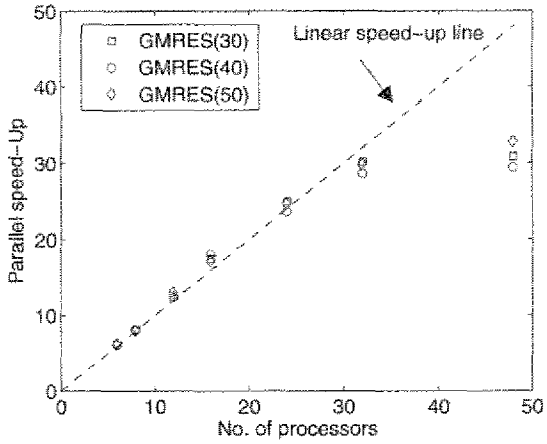


Figure 26: **Parallel speed-up of FETI-DP/PGMRES solver using Classical Gram-Smidt with Reorthogonalization based Arnoldi procedure.**

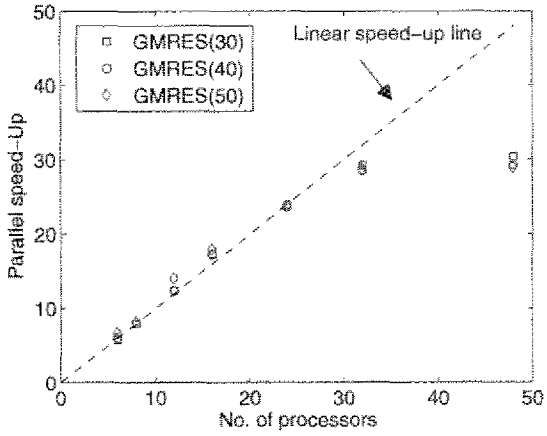


Figure 27: **Parallel speed-up of FETI-DP/PGMRES solver using Modified Gram-Smidt based Arnoldi procedure.**

procedure show the same scalability behavior. 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-Schimdt are by themselves lower compare to the Modified Gram-Schimdt. This difference is expected to be drastic for large scale problems. This trend is clear in Table 13, where the former provides a marginally faster timing trend. Given the small size of the problem, a concrete conclusion is still premature.

CONCLUDING OBSERVATIONS

A 3-D FEM analysis for rotary wing dynamics was formulated with emphasis on the inertial terms unique to rotorcraft. A dual primal iterative substructuring based Krylov solver was developed for a fully parallel solution produre. The FETI-DP domain decomposition algorithm was used for this purpose. The algorithm was equipped with a GMRES update, in addition to its traditional CG based implementation, due to the unique non-symmetric nature of the rotary wing inertial terms. The scalability of this solver was studied in detail both for hover and transient forward flight prototypes. The key components of rotorcraft analysis: multibody dynamics, periodic response solution, blade airloads, and hence trim were not part of this study. The focus was purely on the scalability of a 3-D FEM based large scale structural dynamic analysis. That is the key contribution of this paper. The following are the key conclusions of this study.

Key conclusions

1. Given a fixed problem size, there is always an optimal number of substructures or subdomains into which the problem can be sub-divided so as to require the minimum solution time. This subdomain optimality grows with problem size. Beyond this optimality, the benefits of smaller sized subdomains are offset by the increasing interface. This has less bearing on scalability but is critical for the actual solution time.

2. The present implementation of the 3D FEM rotor code is scalable and shows a linear speed-up. That is, an 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. 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 optimum. However, even if more processors than optimum is used, the scalability only reaches a plateau, and does not show a dramatic drop off.

3. This plateau stems from the nature of iterative substructuring. Unlike classical substructuring, here, there is a flat region in time vs. substructure curve, such that the penalty incurred by using more than the optimum number of processors is very gradual. On the other hand, the difference between the plateau and the linear speed-up line stems from two practical considerations. First, the subdomain to subdomain communication cost, and second, the global coarse problem communication cost.

4. The first penalty is a minor issue that cannot be avoided. It can be minimized by by minimizing the number of syncronization points during the Krylov updates. This is pertinent more to GMRES where

the modified Gram-Scmidt algorithm – the most stable procedure for generating the Krylov basis - incurrs a significant communication cost. In this paper we have compared the modified Gram-Scmidt with a cheaper classical Gram-Scmidt. The later requires an additional re-orthogonalizing step that is optional and is carried out only when necessary.

5. The second penalty is a major issue. The size of the coarse problem is a key driver of scalability. The rule is to select corners which are common to more than two subdomains. In 3D this generates a very large number. In addition, depending on the partitioning, there may not be any corner node at all. The key idea is that the number of corner nodes can vary depending on partitioning but they must be selected to ensure nullity of the subdomain kernels. Thus, a special, smart partitioner is needed, that minimizes the selection of corner nodes while ensuring the above requirement. Not just any partitioner will do. In the present paper this task has been easily accomplished manually, due to the simple nature of the geometry and grid.

6. Finally, we note, that the theoretical condition number estimates and proof of optimality in domain decomposition is based on the assumption of symmetric and coercive operators with CG updates. They are less developed for non-symmetric systems — and are usually based on the assumption of the dominance of the symmetric operator. One approach is to build upon algorithms that are provably optimal for the former to extend them to the later. This is the procedure followed in this paper. We first verify the performance of the algorithms in ideal hover. We then extend them to real forward flight conditions. We equip FETI-DP with GMRES updates for this purpose and demonstrated optimal convergence scalability patterns computationally. We do not attempt to provide formal theoretical estimates in this paper.

Thrust Areas for Future Research

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

Fundamental Research

1. Scalable solution for periodic dynamics – temporal domain decomposition for boundary value problems in time.

2. 3-D FEM / multibody dynamics coupled analysis – mechanisms and methodology.

Applied Research

1. Locking-free elements, hierarchical elements, nodeless elements to facilitate FEM / multibody dynamics architechture.

2. Exact and generic 3-D Fluid-Structure interfaces.

3. Exact delta coupling procedures for trim solution in level and turning flight.

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

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

ACKNOWLEDGMENTS

The authors acknowledge Dr. Roger Strawn's support for this research at the U. S. Army Aeroflightdynamics Directorate under the High Performance Computing Institute for Advanced Rotorcraft Modeling and Simulation (HI-ARMS). We wish to thank Prof. Charbel Farhat, Stanford University, for his insightful comments on dual primal substructuring. We also thank Dr. Guru Guruswamy, NASA Ames Research Center, for his good advice on FEM and substructuring methods.

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] Hodges, D. H., and Yu, W., "A Rigorous, Engineer-Friendly Approach for Modeling Realistic, Composite Rotor Blades,", *Wind Energy*, Vol. 10, (2), 2007, pp. 179–193.

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

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

[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.