Parallel Fast-Floquet Analysis of Trim and Stability for Large Helicopter Models*

S. Subramanian — Research Associate
G. H. Gaonkar — Professor
Department of Mechanical Engineering
Florida Atlantic University
Boca Raton, FL 33431

Abstract

Classical Floquet theory is widely applied to predict trim and stability. However, the sequential run time for trim grows between quadratically and cubically with the number of states or order because many integrations through one complete period are required. By comparison, the fast Floquet theory requires integration through T/Q, where T is the period and Q the number of blades in a rotor with identical blades; the mode identification becomes simpler as well. Accordingly, a parallel shooting method based on the fast Floquet theory and damped Newton iteration is developed to predict trim and the equivalent Floquet transition matrix (EFTM). A parallel QR library is used to predict stability from the eigenanalysis of the EFTM. The parallel fast Floquet analysis comprises these shooting and QR methods. The computational reliability is measured by the condition numbers of the Jacobian matrix in Newton iteration as well as by the eigenvalue condition numbers and residual errors of the eigenpairs. Similarly, the parallel-performance measures include the dominance of the parallelizable part, parallel and sequential run times and their rates of growth with the order, and efficiency. Basically, efficiency shows how effectively the processors are used, allowing analysts to guard against processor underutilization. The parallel fast-Floquet analysis is compared with three other analyses: parallel analysis based on classical Floquet theory and sequential analyses based on classical and fast Floquet theories. Nearly 500 states are used to treat isolated-rotor trim and stability with dynamic stall and wake. The data on computational reliability and parallel-performance measures demonstrate the feasibility of the parallel fast-Floquet analysis with thousands of states.

Nomenclature

Unless otherwise stated, the symbols below are nondimensional:

a	lift curve slope, rad^{-1}
c	number of control inputs
$\mathbf{c}$	control-input vector
C_{d_0}	constant profile drag coefficient
C_d	resultant profile drag force in the plane of the rotor disk opposite to the flight direction
C_l	rolling moment coefficient
C_m	pitching moment coefficient
C_T	thrust coefficient
C_w	weight coefficient of the helicopter
$\mathbf{E}^k$	error vector for k-th iteration
E_p	efficiency using p processors
$\overline{f}$	equivalent flat plate area of parasite drag
f	sequential fraction
$\mathbf{I}$	identity matrix
M	number of states or state variables, $M = N + c$
N	number of structural and aerodynamic states or state variables
N_b	number of blade states
N_w	number of dynamic wake states
p	number of processors
P_β	flap natural frequency, rotating
$\mathbf{P}$	permutation matrix
Q	number of blades
S_p	Speedup
$\mathbf{s}$	state vector $\mathbf{x}$ augmented with control-input vector $\mathbf{c}$
t	time unit such that $T = 2\pi$
t_p	parallel run time on p processors, *sec*
t_1	uniprocessor or sequential run time, *sec*
t_{1_s}	sequential part of t_1, *sec*
t_{1_p}	parallel part of t_1, *sec*
T	period, *sec*
$\mathbf{x}$	state vector
$\mathbf{X}$	matrix of augmented initial-condition vectors, Eq. (30)
$\mathbf{y}$	solution vector with components $\mathbf{x}(2\pi)$ and δ corresponding to initial response vector $\mathbf{s}$, Eq. (25)
$\mathbf{y}^i$	solution vectors with components $\mathbf{x}(2\pi)$ and δ corresponding to perturbed vectors, Eq. (25)
$\mathbf{Y}$	matrix of augmented solution vectors having components $\mathbf{y}^i$, Eq. (31)
z_k	k-th eigenvalue of the FTM
$\overline{z}_k$	k-th eigenvalue of $[\mathbf{P}\phi(\Delta T)]$
α_j^r, β_j^r	wake states
ξ_k	k-th mode nonunique frequency of z_k
$\overline{\xi}_k$	k-th mode nonunique frequency of $\overline{z}_k$
χ	Newton damping parameter
δ	trim-error vector for control inputs

* *Paper presented at the 22nd European Rotorcraft Forum and 13th European Helicopter Association Symposium, Brighton, UK, Sep 16–19, 1996.*

ΔT	T/Q
ϵ	perturbation quantity
γ	Lock number (blade inertia parameter)
σ	rotor solidity
σ_k	k-th mode damping of z_k
$\overline{\sigma}_k$	k-th mode damping of $\overline{z}_k$
μ	advance ratio
ϕ	state transition matrix
Φ	Jacobian or Partial Derivative Matrix
ω_ζ	lag natural frequency, rotating
$[\,]^T$	transpose of $[\,]$
$\|\,\|$	Euclidean norm of a vector or matrix
$(\dot{\mathbf{x}})$	time derivative of $\mathbf{x}$

Introduction

Classical Floquet theory provides a rigorous basis to *predict trim and stability; trim means control* settings and the corresponding periodic responses, and stability means frequencies and damping levels (Ref. 1). Broadly stated, the shooting method is used to predict trim, which gives the Floquet transition matrix (FTM) as a byproduct, and the QR method is used for the eigenvalues and eigenvectors of the FTM, from which stability is predicted (Refs. 1–4). Despite the basis and other attractive computational features, the sequential run time for trim becomes prohibitive; in fact it grows between quadratically and cubically with the number of states or order (Ref. 5). Thus, on sequential computers, classical Floquet theory can be routinely applied to relatively small-order models, say order $N < 100$. More important, its utility is limited in cases of practical interest such as comprehensive and design analyses that require models with thousands of states. Recently, the fast Floquet theory with sequential computing and classical Floquet theory with parallel computing have been pursued to remove or alleviate this run-time constraint (Refs. 5–7). In this paper, we present a new approach that exploits both the fast Floquet theory and parallel computing and thereby demonstrate the feasibility of treating models with thousands of states routinely; as it turns out, the potential practical utility is dramatically borne out by the numerical results. We begin with a mention of why the run time for trim becomes prohibitive. This facilitates a better appreciation of why the combination of the fast Floquet theory and parallel computing provides nearly a tailor-made solution to the run-time constraint. Moreover, we primarily address trim with only a passing reference to stability and use the run times for trim and stability and for trim almost interchangeably. This is because for large systems ($N > 100$) the run time for stability is hardly 1% of the run time for trim.

Prediction of trim is a demanding and computer-heavy exercize that couples nonlinear differential equations of motion with algebraic-transcendental equations of trim (*e.g.* Ref. 2). We have to predict the control settings that satisfy the flight conditions and then (given these controls) find the initial conditions that guarantee periodic response. Furthermore, the control settings appear not only in the damping and stiffness matrices but in the forcing-function matrix as well, and they are specified indirectly to satisfy flight conditions of prescribed thrust level and force-moment equilibrium. Therefore, we predict trim iteratively, starting with assumed values, say for N initial conditions for an N-order model and for c number of controls. Then we integrate the equations of motion through one complete period, find the error in satisfying the trim and periodicity conditions, improve the starting values according to Newton iteration and perturb the starting values one at a time. The cycle of perturbing, integrating and improving continues till convergence. Thus, if k cycles are required for convergence, the procedure requires $k(N+c)$ perturbations, integrations through one complete period and improvements. Here, two points need to be stressed since they have considerable bearing on the fast Floquet theory and parallel computing. First, *a large number of integrations through one complete period* is required; $k(N+c+1)$ integrations to be precise. Second, a fairly large number of integrations in each cycle are *independent*; that is, $N+c+1$ independent integrations. The first point leads to the fast Floquet theory since integrations through T/Q are required, where T is the period and Q the number of blades; the only restriction is that the rotors have identical blades. Similarly, the second point leads to parallel computing since these independent integrations could be executed in parallel or concurrently. Given the fact that the integrations take the bulk of the sequential run time, the fast Floquet theory and parallel computing, in principle, reduce the run time by a factor of nearly $Q(N+c+1)$. This is the motivation for bringing together the fast Floquet theory and parallel computing. Parallel fast Floquet analysis does just that. To put it in perspective, the state of the art of classical and fast Floquet theories is presented next.

Relatively few studies are available that apply classical and fast Floquet theories to predict trim and stability of large systems with sequential or parallel computing (Refs. 1, 5–7). In fact, predictions of trim and stability are based almost exclusively on classical Floquet theory, even for rotors with identical blades. In Ref. 1, Gaonkar and Peters review these applications of classical Floquet theory with sequential computing up to 1986. As reviewed therein (Ref. 1), the applications are limited to relatively small systems, the order N hardly exceeding 35, owing to run-time constraint. Developments since 1986 are covered by Chunduru (Ref. 5); he applied both classical and fast Floquet theories to large systems ($N \approx 500$) with sequential computing and shows that the fast Floquet theory indeed brings in nearly Q-fold reduction in run time for a Q-bladed rotor. Similarly, Subramanian *et al.* address large systems ($N \approx 500$) by exploiting parallel computing (Refs. 6 and 7); classical Floquet theory

is used in Ref. 6 and the fast Floquet theory in Ref. 7. A case in point is two earlier studies due to Peters (Ref. 8) and to McVicar and Bradley (Ref. 9). Peters presents the theoretical aspects of the fast Floquet theory with respect to single- and multi-rotor models and two of these aspects merit special mention: the fast-Floquet theory makes the mode identification simpler (vis-a-vis the classical Floquet theory) and lends itself equally well to multirotor aircraft. In Ref. 9, McVicar and Bradley apply the fast Floquet theory and parallel computing to simulate tilt-rotor trim in real time. This paper is a condensed and amplified version of Ref. 7.

Parallel fast Floquet analysis comprises a shooting method based on the fast Floquet theory and on parallel computing, and a parallel QR library. It is compared with three other analyses: parallel analysis based on classical Floquet theory and sequential analyses based on classical and fast Floquet theories. To make this comparison comprehensive and realistic we treat models of order as high as 430 and present data on the following items: 1) computational reliability, 2) sequential and parallel run times and their rates of growth with the problem size or order and 3) parallelizable versus nonparallelizable parts. These comparative data also include efficiency of parallel computing, which is a measure of how busy the processors are kept; for example, an efficiency of unity means the processors are used effectively without underutilization. Both efficiency and run time together provide a systematic means of scaling the processors with the order as a compromise between how effectively the processors can be used and how fast the problem needs to be solved as the order increases.

To sum up: We present parallel fast Floquet analysis, apply it to models with as many as 430 states and then present a comprehensive set of comparative data to demonstrate its practical utility. The computations are done on a MasPar MP-1, architecturally a Single-Instruction-Multiple-Data or SIMD massively parallel computer with 8192 processors. Unlike sequential computing, parallel computing is architecture dependent, as such the preceding development would need adaptations to other architectures. Nevertheless, the potential practical utility of this development should motivate further research on these adaptations and other applications of parallel computing. All in all, there is broad agreement on "the inevitability of the eventual success and widespread use of massively parallel processing technology" (Ref. 10); also see Ref. 11. And only the barest beginnings have been made in developing and applying parallel computing concepts and methods to rotorcraft aeroelasticity (Ref. 6). Therefore the present work should provide a useful reference.

Classical and Fast Floquet Theories

Parallel fast Floquet analysis, based on the fast Floquet theory, predicts trim and stability of single-rotor as well as multirotored models of rotorcraft. It is presented here primarily with reference to a single-rotor model with Q blades. The extension to multirotor models is straightforward and not explicitly elaborated. The only restriction is that each rotor has identical blades. Compared to parallel Floquet analysis (Refs. 6 and 7), which is based on classical Floquet theory, there are changes such as interval of integration and formulation of forces and moments in predicting trim, and generation of the eigenvalues and eigenvectors of the FTM and mode identification in predicting stability. These changes bring in considerable simplicity to the algorithm with significant benefits: simpler mode identification and dramatically reduced run time. To help explain these features we begin with a mention of predicting periodic-response and stability of a linear system by applying classical Floquet theory, then take up briefly parallel Floquet analysis and finally come to parallel fast Floquet analysis.

Classical Floquet Theory

We consider a linear periodic-coefficient system with $N \times 1$ state vector $\mathbf{x}(t)$, whose equations of motion can be written as

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\,\mathbf{x}(t) + \mathbf{G}(t) \tag{1}$$

where the state matrix $\mathbf{A}(t)$ and the forcing function or input vector $\mathbf{G}(t)$ are 2π-periodic, and are of dimension $N \times N$ and $N \times 1$, respectively. The $N \times N$ state transition matrix $\phi(t)$ is the fundamental solution matrix of the matrix differential equation

$$\dot{\phi} = \mathbf{A}(t)\phi, \quad 0 \le t \le 2\pi \tag{2}$$

with initial condition $\phi(0) = \mathbf{I}$. The FTM is given by $\phi(2\pi)$ and its eigenvalues, z_k, determine system stability (Ref. 1). The modal damping and frequencies are given by

$$\sigma_k = \frac{1}{2\pi} \ln |z_k| \tag{3a}$$

$$\xi_k = \frac{1}{2\pi} \arg(z_k) = \frac{1}{2\pi} \tan^{-1}\left(\frac{\mathrm{Im}(z_k)}{\mathrm{Re}(z_k)}\right) \tag{3b}$$

For the complete or nonhomogeneous Eq. (1), the initial conditions that yield periodic forced response, that is, $\mathbf{x}(0) = \mathbf{x}(2\pi)$, are given by (Ref. 1)

$$[\mathbf{I} - \phi(2\pi)]\,(\mathbf{x}(0) - \mathbf{x}_E(0)) = (\mathbf{x}_E(2\pi) - \mathbf{x}_E(0)) \tag{4}$$

where $\mathbf{x}_E(2\pi)$ is the non-periodic solution at $t = 2\pi$ for any arbitrary initial state $\mathbf{x}_E(0)$.

In general, rotorcraft systems are governed by nonlinear periodic-coefficient equations and can be represented as

$$\dot{\mathbf{x}} = \mathbf{G}(\mathbf{x}, t) \tag{5}$$

The initial conditions to generate periodic response are obtained by applying a Newton-type iteration. Specifically, for the k-th iteration, Eq. (4) reads:

$$\mathbf{x}(0)_{k+1} = \mathbf{x}_E(0)_k + \chi\left[\mathbf{I} - \boldsymbol{\phi}(2\pi)\right]_k^{-1} \left(\mathbf{x}_E(2\pi) - \mathbf{x}_E(0)\right)_k \quad (6)$$

where $\boldsymbol{\phi}(2\pi)$ converges to FTM, and $[\mathbf{I} - \boldsymbol{\phi}(2\pi)]^{-1}$ is the Jacobian or partial derivative matrix $\boldsymbol{\Phi}$ and χ is the Newton damping parameter. Here after, the Jacobian matrix (Jacobian, for short) and the partial derivative matrix (PDM) are used interchangeably.

With the role of unknown control-input vector $\mathbf{c}$ explicitly shown, the equations governing the rotorcraft motions can be written as

$$\dot{\mathbf{x}} = \mathbf{G}(\mathbf{x}, \mathbf{c}, t) \quad (7)$$

Then the initial condition $\mathbf{x}(0)$ that yields periodic response will satisfy

$$\mathbf{x}\left(2\pi, \mathbf{x}(0)\right) - \mathbf{x}(0) = \mathbf{0} \quad (8)$$

where $\mathbf{x}(2\pi, \mathbf{x}(0))$ represents the state at $t = 2\pi$ with initial state $\mathbf{x}(0)$. Beside the initial conditions for periodic response, we also compute the control-input vector $\mathbf{c}$ such that the periodic response satisfies the trim equations of force and moment balance. Symbolically, the trim equations are represented as

$$\mathbf{f}(\mathbf{x}, \mathbf{c}) = \mathbf{0} \quad (9)$$

Equations (8) and (9), which represent the response periodicity and desired flight conditions, respectively, are nonlinear algebraic transcendental equations. Combined, they can be expressed as

$$\mathbf{f}(\mathbf{s}) = \mathbf{0} \quad (10)$$

where $\mathbf{s} = [\mathbf{x},\ \mathbf{c}]^T$ is the augmented-state vector of state variables and control inputs.

Therefore, the trim analysis boils down to the generation and solution of Eq. (10). This is done by formulating a shooting strategy with damped Newton iteration (Ref. 2). Thus, the initial conditions that yield the periodic response and the unknown control inputs that satisfy the required flight conditions are improved according to

$$\begin{Bmatrix} \mathbf{x}(0) \\ \mathbf{c} \end{Bmatrix}_{k+1} = \begin{Bmatrix} \mathbf{x}_E(0) \\ \mathbf{c} \end{Bmatrix}_k - \chi \begin{bmatrix} \boldsymbol{\Phi}_{11} - \mathbf{I} & \boldsymbol{\Phi}_{12} \\ \boldsymbol{\Phi}_{21} & \boldsymbol{\Phi}_{22} \end{bmatrix}^{-1} \begin{Bmatrix} \mathbf{x}_E(2\pi) - \mathbf{x}_E(0) \\ \boldsymbol{\delta} \end{Bmatrix} \quad (11)$$

where $\boldsymbol{\delta}$ is the error in satisfying Eq. (9); that is, $\mathbf{f}(\mathbf{x}, \mathbf{c}) = \boldsymbol{\delta}$, and $\boldsymbol{\Phi}$ is the Jacobian matrix. Furthermore, the submatrix $\boldsymbol{\Phi}_{11}$ converges to the FTM.

Trim with Fast Floquet Theory

In classical Floquet theory, the Jacobian is generated by integrating the equations of motion through one complete period 2π. However, for a Q-bladed rotor with identical blades, the FTM can be generated by integrating the equations of motion through $\Delta T = 2\pi/Q$. The basis is that at $\psi = \Delta T$, the first blade is in exactly the same position that was occupied by the second blade at $\psi = 0$. Therefore, the state transition matrix for the time interval from $\psi = \Delta T$ to $\psi = 2\Delta T$ is identical to the one in the time interval from $\psi = 0$ to $\psi = \Delta T$ except that the blade indices need to be permuted. This shows the feasibility of finding the transition matrix between any two time instants that differ by ΔT from the transition matrix from $\psi = 0$ to $\psi = \Delta T$ and the permutation matrix $\mathbf{P}$ (details to follow). Therefore, the general relation between any two time instants $\psi = n\Delta T$ to $\psi = (n+1)\Delta T$ can be expressed as (Refs. 7–9)

$$\mathbf{P}^n \mathbf{x}\left[(n+1)\Delta T\right] = \boldsymbol{\phi}(\Delta T)\mathbf{P}^n \mathbf{x}\left[n\Delta T\right] \quad n = 0, 1, \ldots, Q-1 \quad (12)$$

From Eq.(12), one can write

$$\mathbf{P}\mathbf{x}(\Delta T) = \mathbf{P}\boldsymbol{\phi}(\Delta T)\mathbf{x}(0) \quad (13a)$$

$$\mathbf{P}^2\mathbf{x}(2\Delta T) = \left[\mathbf{P}\boldsymbol{\phi}(\Delta T)\right]^2 \mathbf{x}(0) \quad (13b)$$

$$\mathbf{P}^Q\mathbf{x}(Q\Delta T) = \mathbf{x}(2\pi) = \left[\mathbf{P}\boldsymbol{\phi}(\Delta T)\right]^Q \mathbf{x}(0) \quad (13c)$$

which leads to

$$\boldsymbol{\phi}(2\pi) = \left[\mathbf{P}\boldsymbol{\phi}(\Delta T)\right]^Q \quad (14)$$

This is a crucially important equation that connects the FTM $\boldsymbol{\phi}(2\pi)$ with $\boldsymbol{\phi}(\Delta T)$. Therefore, we need to compute $\boldsymbol{\phi}(\Delta T)$ instead of $\boldsymbol{\phi}(2\pi)$ since the Q-th power of $[\mathbf{P}\boldsymbol{\phi}(\Delta T)]$ gives the FTM. The practical utility of Eq. (14) is keyed to the fact that the eigenvalues z_k of $\boldsymbol{\phi}(2\pi)$ are related to the eigenvalues $\overline{z}_k$ of $[\mathbf{P}\boldsymbol{\phi}(\Delta T)]$ by the relation $z_k = \overline{z}_k^Q$. Since the modal damping levels and frequencies are found by taking the logarithm of z_k, it is not really necessary to raise the eigenvalues $\overline{z}_k$ to the Q-th power. Instead we take the logarithm of $\overline{z}_k$ and simply multiply it by Q. Therefore, the modal damping and frequencies are found from the eigenvalues $\overline{z}_k$ of $[\mathbf{P}\boldsymbol{\phi}(\Delta T)]$:

$$\overline{\sigma}_k = \frac{Q}{2\pi} \ln |\overline{z}_k| \quad (15a)$$

$$\overline{\xi}_k = \frac{Q}{2\pi} \arg(\overline{z}_k) = \frac{Q}{2\pi} \tan^{-1}\left(\frac{\mathrm{Im}(\overline{z}_k)}{\mathrm{Re}(\overline{z}_k)}\right) \quad (15b)$$

As seen from Eqs. (14) and (15), $[\mathbf{P}\boldsymbol{\phi}(\Delta T)]$ has one-to-one equivalence to the FTM; hence it is referred to as the equivalent FTM (EFTM).

In trimmed flight, the blades experience periodic variations in the sectional angle of attack as well as in the air velocity components, which in turn produce periodic variations in rotor forces and moments. For a rotor with identical blades, the contribution of each blade to the rotor forces and moments will be the same at a given azimuthal position since each azimuthal position has its own pitch and air velocity. Therefore,

the period of oscillations of these forces and moments depends on the number of blades in the rotor. Thus, for a Q-bladed rotor, it is required to rotate through $2\pi/Q$ radians so that it has a blade in all azimuthal positions instantaneously. Thus, the variation of trim forces and moments in one period (2π radians) can be described completely in $2\pi/Q$ radians of rotor rotation. Therefore,

$$\left\{ \begin{array}{c} C_T \\ C_d \\ C_l \\ C_m \end{array} \right\}_{\psi=0} = \left\{ \begin{array}{c} C_T \\ C_d \\ C_l \\ C_m \end{array} \right\}_{\psi=2\pi/Q} \tag{16}$$

Similarly, the inflow forcing functions are influenced by the blade sectional lift, which also can be described completely over $2\pi/Q$ radians. Therefore, for inflow or wake states, the periodicity condition is satisfied when

$$\left\{ \begin{array}{c} \alpha_j^r \\ \beta_j^r \end{array} \right\}_{\psi=0} = \left\{ \begin{array}{c} \alpha_j^r \\ \beta_j^r \end{array} \right\}_{\psi=2\pi/Q} \tag{17}$$

However, in the case of blade states, the rotor has to rotate through 2π *rad* for each blade to pass through all azimuthal positions. Thus, one complete rotor revolution is required to describe the trimmed blade states. In other words, one complete revolution is required to check for periodicity of the blade states. Nevertheless, it is possible to ascertain periodicity of blade states in $2\pi/Q$ *rad* of revolution since the trajectory followed by all the blades is the same as they go through one revolution with a phase shift of $2\pi/Q$ *rad* between the paths of each blade. Therefore, the states of an arbitrary q-th blade at an azimuthal position $\psi = 2\pi/Q$ will map onto the initial states of an identical $(q+1)$-th blade at $\psi = 0$. (The blade states can include displacements, velocities, angular displacements, angular velocities and blade-fixed aerodynamic states.) Thus, for the periodicity of blade states, we have

$$\mathbf{x}_{b_{\psi=2\pi/Q}} = \mathbf{P}_b \mathbf{x}_{b_{\psi=0}} \tag{18}$$

where $\mathbf{x}_b$ is the $N_b \times 1$ vector of blade states defined as

$$\mathbf{x}_b = \left[\mathbf{x}_{\text{blade 1}},\ \mathbf{x}_{\text{blade 2}},\ \cdots,\ \mathbf{x}_{\text{blade }Q}\right]^T \tag{19}$$

In Eq. (18), $\mathbf{P}_b$ is the $QN_b \times QN_b$ permutation matrix and is given by

$$\mathbf{P}_b = \begin{bmatrix} 0 & \mathbf{I}_b & 0 & \cdots & 0 & 0 \\ 0 & 0 & \mathbf{I}_b & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \mathbf{I}_b \\ \mathbf{I}_b & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{20}$$

where $\mathbf{I}_b$ is the $N_b \times N_b$ unit matrix.

Now, combining the blade states and the inflow states, we write

$$\{\mathbf{x}\}_{\psi_q=2\pi/Q} = \mathbf{P}\,\{\mathbf{x}\}_{\psi_q=0} \tag{21}$$

where the $N \times N$ permutation matrix $\mathbf{P}$ is given by

$$\mathbf{P} = \begin{bmatrix} 0 & \mathbf{I}_b & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \mathbf{I}_b & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \mathbf{I}_b & 0 \\ \mathbf{I}_b & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \mathbf{I}_w \end{bmatrix} \tag{22}$$

and $\mathbf{I}_w$ is the unit matrix of size $N_w \times N_w$. In summary, Eqs. (16–21) show that the rotor trim forces, moments and periodic responses can be described in the interval $2\pi/Q$. Therefore, Eq. (11), which improves the initial conditions that guarantee the periodic response as well as the unknown control inputs that satisfy the required flight conditions is modifed as (Refs. 7 and 9)

$$\begin{aligned} \left\{ \begin{array}{c} \mathbf{x}(0) \\ \mathbf{c} \end{array} \right\}_{k+1} &= \left\{ \begin{array}{c} \mathbf{x}_E(0) \\ \mathbf{c} \end{array} \right\}_k \\ &- \chi \begin{bmatrix} \mathbf{\Phi}_{11} - \mathbf{P} & \mathbf{\Phi}_{12} \\ \mathbf{\Phi}_{21} & \mathbf{\Phi}_{22} \end{bmatrix}^{-1} \\ &\left\{ \begin{array}{c} \mathbf{x}_E(\Delta T) - \mathbf{P}\mathbf{x}_E(0) \\ \boldsymbol{\delta} \end{array} \right\} \end{aligned} \tag{23}$$

Moreover, the matrix $\mathbf{P}^{\mathrm{T}}\mathbf{\Phi}_{11}$ converges to $[\mathbf{P}\ \phi(\Delta T)]$.

Parallel Fast-Floquet Analysis

Trim

Now we present a parallel fast-Floquet algorithm of helicopter trim — parallel shooting based on the fast Floquet theory — and consider a system with N structural and aerodynamic states and with c number of control settings or inputs. A noteworthy feature is that in each iteration cycle, while the sequential algorithm generates $(N+c)^2$ elements of the Jacobian one element at a time, the parallel algorithm generates all $(N+c)^2$ elements once. Conceptually, we solve $(N+c)^2$ times as fast on $(N+c)^2$ processors as on a single processor. To facilitate a better appreciation of this significant feature of the developed parallel algorithm, it is expedient to begin with the algorithmic details of sequential shooting based on the fast Floquet theory.

Sequential Fast Shooting

The sequential shooting algorithm centers on Eq. (11) and has the following seven instructions:

1. Assume $N + c = M$ arbitrary starting or initial values of the augmented vector $\mathbf{s}$; that is, $N \times 1$ initial state $\mathbf{x}(0)$ and $c \times 1$ control-input vector $\mathbf{c}$.

2. Form the permutation matrix $\mathbf{P}$ of size $N \times N$ according to Eq. (22).

3. Perturb the M initial values *one initial value at a time* by a small amount ϵ_i, $i = 1, 2, \ldots, M$ and

form $M+1$ vectors of starting values:

$$\mathbf{s}+\begin{Bmatrix}\epsilon_1\\0\\\vdots\\0\end{Bmatrix},\mathbf{s}+\begin{Bmatrix}0\\\epsilon_2\\\vdots\\0\end{Bmatrix},\cdots,\mathbf{s}+\begin{Bmatrix}0\\0\\\vdots\\\epsilon_M\end{Bmatrix},\mathbf{s}(24)$$

4. Integrate nonlinear Eq. (7) $M+1$ times for the $M+1$ vectors of starting values through a time interval $\Delta T = 2\pi/Q$ and generate the solution vectors:

$$\mathbf{y}^i = \begin{Bmatrix}\mathbf{x}(\Delta T)\\ \boldsymbol{\delta}\end{Bmatrix}_{s+\epsilon_i} \text{ where } i=1,2,\ldots,M$$
$$\text{and } \mathbf{y} = \begin{Bmatrix}\mathbf{x}(\Delta T)\\ \boldsymbol{\delta}\end{Bmatrix}_{s} \qquad (25)$$

The vector $\boldsymbol{\delta}$ represents the trim error in satisfying Eq. (9). Moreover, the subscripts s and $s+\epsilon_i$, respectively, indicate the differences in the starting values; that is, one solution vector with starting-value vector $\mathbf{s}$ and M solution vectors with M vectors of perturbed starting values.

5. Form M columns of the Jacobian matrix Φ using

$$\begin{Bmatrix}\dfrac{\mathbf{y}^i-\mathbf{y}}{\epsilon_i}\end{Bmatrix}, \quad i=1,2,\ldots,M \text{ or equivalently}$$
$$\boldsymbol{\Phi} = \begin{bmatrix}\boldsymbol{\Phi}_{11}-\mathbf{P} & \boldsymbol{\Phi}_{12}\\ \boldsymbol{\Phi}_{21} & \boldsymbol{\Phi}_{22}\end{bmatrix} \qquad (26)$$

where $\mathbf{P}^{\mathrm{T}}\boldsymbol{\Phi}_{11}$ converges to $\mathbf{P}\boldsymbol{\phi}(\Delta T)$.

6. Generate the error vector $\mathbf{E}^k$. Specifically, at the k-th iteration counter

$$\mathbf{E}^k = \begin{Bmatrix}\mathbf{x}(\Delta T)-\mathbf{P}\mathbf{x}(0)\\ \boldsymbol{\delta}\end{Bmatrix}^k \qquad (27)$$

where $\mathbf{x}(\Delta T)$ is the solution vector at the end of $2\pi/Q$ period and $\boldsymbol{\delta}$ is the trim-error vector corresponding to the initial-condition vector $\mathbf{s}$.

7. Improve the solution with Newton damping parameter χ:

$$\mathbf{s}^{k+1} = \mathbf{s}^k - \chi\boldsymbol{\Phi}^{-1}\mathbf{E}^k \qquad (28)$$

The instructions 3–7 are repeated till the convergence of control inputs and periodic responses.

Parallel Fast Shooting

According to Eqs. (24–28), each instruction is applied to a large number of input data and the operations on these data are independent. The parallel fast shooting algorithm exploits this independency so that all the processors execute the same instruction but on different data. Given this background, we present the parallel algorithm with seven instructions in the format that is implemented.

1. Assume an arbitrary vector of starting values for trim results of periodic responses and control inputs $\mathbf{s} = [s_1\ s_2\cdots s_M]^T$. Then form a matrix of size $M\times(M+1)$ by replicating $(M+1)$ times the vectors $\mathbf{s}$, that is,

$$\begin{bmatrix}s_1 & s_1 & \cdots & s_1 & s_1\\ s_2 & s_2 & \cdots & s_2 & s_2\\ \vdots & \vdots & \cdots & \vdots & \vdots\\ s_M & s_M & \cdots & s_M & s_M\end{bmatrix} \qquad (29)$$

2. Following Eq. (22), form the $N\times N$ permutation matrix $\mathbf{P}$ in such a way that each element of $\mathbf{P}$ is stored in one processor.

3. For the $M\times M$ submatrix (excluding the last column of the matrix in Eq. (29)), perturb along the leading diagonal by a small amount ϵ_i, $i=1,2,\ldots,M$, and generate the $M\times(M+1)$ matrix of initial conditions:

$$\mathbf{X} = \begin{bmatrix}s_1+\epsilon_1 & s_1 & \cdots & s_1 & s_1\\ s_2 & s_2+\epsilon_2 & \cdots & s_2 & s_2\\ \vdots & \vdots & \cdots & \vdots & \vdots\\ s_M & s_M & \cdots & s_M+\epsilon_M & s_M\end{bmatrix}$$
$$= \begin{bmatrix}s_1^1 & s_1^2 & \cdots & s_1^M & s_1^{M+1}\\ s_2^1 & s_2^2 & \cdots & s_2^M & s_2^{M+1}\\ \vdots & \vdots & \cdots & \vdots & \vdots\\ s_M^1 & s_M^2 & \cdots & s_M^M & s_M^{M+1}\end{bmatrix} \qquad (30)$$

This instruction is sketched schematically in Fig. 1. As shown therein each *data element* of an array (*e.g.*, s_1) is associated with one processor (depicted by a box) and all the processors execute the same instruction on their own data element.

4. Using the above $M\times(M+1)$ matrix of initial conditions, integrate Eq. (7) in parallel and generate the solution matrix $\mathbf{Y}$ at the end of period $\psi = 2\pi/Q$:

$$\mathbf{Y} = \begin{bmatrix}y_1^1 & y_1^2 & \cdots & y_1^M & y_1^{M+1}\\ y_2^1 & y_2^2 & \cdots & y_2^M & y_2^{M+1}\\ \vdots & \vdots & \cdots & \vdots & \vdots\\ y_M^1 & y_M^2 & \cdots & y_M^M & y_M^{M+1}\end{bmatrix}$$
$$= \left[\mathbf{y}^1\ \mathbf{y}^2\cdots\mathbf{y}^M\ \mathbf{y}^{M+1}\right] \qquad (31)$$

According to instruction 3, the integration of equations of motion begins at $\psi=0$ and proceeds through an interval $\psi=\Delta T$ with a finite number of azimuthal positions as integration steps. As shown in Fig. 1, this operation is represented by a series of parallelepipeds, one below the other with each one representing an azimuthal position; $\psi_0=0 \rightarrow \psi_1=\psi_0+\Delta\psi \rightarrow \cdots \rightarrow \psi_n = 2\pi/Q$.

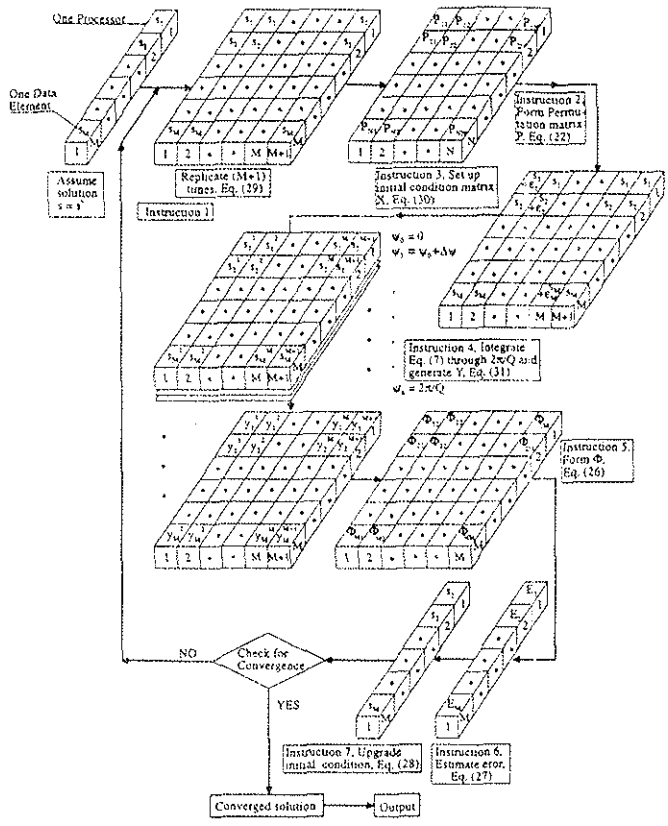

Figure 1: Schematic of Parallel Fast Shooting.

Moreover, each box of the parallelepiped represents one processor associated with one data element. The details of operation at each azimuthal position involve parallel computations associated with the equations of motion and trim. Therefore, we emphasize that computations of the error in satisfying Eq. (9) and the $(M+1)$ sets of integrations for the $M \times (M+1)$ matrix of initial conditions are carried out in parallel and that each element of the solution matrix $\mathbf{Y}$ is computed by one processor. In other words, $M(M+1)$ processors are used to generate $M(M+1)$ elements of the matrix $\mathbf{Y}$.

5. Form the Jacobian matrix $\mathbf{\Phi}$ according to Eq. (26).

6. Estimate the error using Eq. (27).

7. Improve the solution following Eq. (28).

The instructions 3–7 are repeated till convergence of trim results.

Stability

For large systems, the eigenanalysis for stability results is not the dominant issue in the run time of fast-Floquet analysis. The QR method for eigenanalysis takes much less run time. In fact, compared with the run time for trim analysis, the eigenanalysis hardly takes 1% of the total run time. Given this background, we generate the stability results such as the modal damping levels and frequencies from the eigenvalues of the matrix $[\mathbf{P}\phi(\Delta T)]$; see Eqs. (15a) and (15b). As seen from Eq. (15b), the frequency is determined from the inverse arctangent function, which results in multiple values. To help predict the frequencies and thereby identify all modes, we follow the method of Ref. 4 with some modifications that result from replacing the FTM by $[\mathbf{P}\phi(\Delta T)]$; see Eq (23). Specifically, we first compute the eigenvectors of the matrix $[\mathbf{P}\phi(\Delta T)]$. Then for each eigenvector, say $\mathbf{x}_i$, the complex ratio of the derivative $\dot{x}_i$ and the state x_i corresponding to the most dominant component is obtained; for details see Ref. 4. The imaginary part of this ratio with a suitable correction, which is an integer multiple of Q, closely approximates the frequency of the mode. In this study, an LAPACK eigenvalue subroutine DGEEV for real unsymmetric matrices is used to compute the eigenvalues and eigenvectors of the matrix $[\mathbf{P}\phi(\Delta T)]$ (Ref. 12).

Computational Reliability

Predicting trim and stability is demanding, involving nonlinear differential equations of motion coupled with algebraic-transcendental equations of trim. Even for relatively small-order systems, the computations are intensive in that repeated operations like integrations, linearizations for Newton improvement and iterations with improved starting values are required. Some of these operations grow quadratically with the order; therefore, for large models, the computations become extensive as well. Given the complexity and extent of the computations, a means of quantifying their reliability is required.

As seen from Eqs. (26) and (28), the Jacobian determines the improved values of trim in the corresponding iteration cycle, and $[\mathbf{P}\phi(\Delta T)]$ is its submatrix in the final converged cycle. Moreover, the frequencies, mode identification and damping levels are derived from the eigenvalues and eigenvectors of the matrix $[\mathbf{P}\phi(\Delta T)]$. Thus, generating the Jacobian and eigenanalysis of the matrix $[\mathbf{P}\phi(\Delta T)]$ is crucially important to predicting trim and stability. Accordingly, we use three reliability parameters; the first one concerning the Jacobian in Newton iteration, and the other two concerning the eigenvalues and eigenpairs of $[\mathbf{P}\phi(\Delta T)]$ that correspond to the modes of interest. For completeness, a brief account of these reliability parameters is given in the sequel; this also helps present the numerical results subsequently.

The first reliability parameter is the condition number of the Jacobian $\mathbf{\Phi}$ and is given by

$$\text{Cond.}(\mathbf{\Phi}) = \frac{\left[\text{max. eigenvalue of } \mathbf{\Phi}^T\mathbf{\Phi}\right]^{\frac{1}{2}}}{\left[\text{min. eigenvalue of } \mathbf{\Phi}^T\mathbf{\Phi}\right]^{\frac{1}{2}}} \qquad (32)$$

Similarly, let $\mathbf{x}$ and $\mathbf{y}$ represent the right and left eigenvectors of the matrix $[\mathbf{P}\phi(\Delta T)]$ corresponding to its eigenvalue λ; that is,

$$[\mathbf{P}\phi(\Delta T)]\mathbf{x} = \lambda\mathbf{x} \text{ and } [\mathbf{P}\phi(\Delta T)]^{\mathrm{T}}\mathbf{y} = \lambda\mathbf{y} \tag{33}$$

By definition, the condition number of λ and the residual error for the eigenpair $(\lambda, \mathbf{x})$ are, respectively, given by

$$\text{Cond.}(\lambda) = |\mathbf{y}^T\mathbf{x}|^{-1} \tag{34}$$

$$\varepsilon = \frac{\|[\mathbf{P}\phi(\Delta T)]\mathbf{x} - \lambda\mathbf{x}\|}{\|\lambda\mathbf{x}\|} \tag{35}$$

For additional details see Ref. 13.

Parallel Performance

Under ideal conditions in the problem and algorithm, parallelism with p processors reduces the uniprocessor run time by a factor of p, and the 'best' these processors could do has been extracted. The parallel performance metrics collectively provide a measure of how well the actual conditions in completing the job approximate the ideal. The lack of standards on the number, use and interpretations of these metrics is well recognized and has been receiving increasing attention (*e.g.* Refs. 10 and 14). Following the literature (e.g., Ref. 15), we use four such metrics. The first one is a directly measured quantity and the easiest to interpret: the growth of run time with the order and number of processors. The parallel run time is the elapsed time from the start of the parallel computation to the end till all the computations are completed. The other three are speedup, sequential fraction and efficiency, which are derived from the measured run time, number of processors used and predicted uniprocessor run time. We emphasize that memory bottleneck prevents direct measurement of uniprocessor run time for large models since massively parallel computers, particularly SIMD systems, represent an assemblage of thousands of relatively small computers (Ref. 10).

Speedup S_p measures how a parallel algorithm running on p processors compares with itself running on one processor. Thus, if t_j represents the parallel run time on j processors, speedup is defined as

$$S_p = \frac{t_p}{t_1} \leq p \tag{36}$$

Similarly, efficiency E_p measures how effectively the processors are used; in other words, how busy they are kept relative to each other:

$$E_p = \frac{S_p}{p} \leq 1 \tag{37}$$

For example, an efficiency of unity means we are "getting to the best" that the processors can do (Ref. 14). It is important that the results of S_p and E_p be interpreted in a relative sense. These two sets of results simply provide a means of compromising between how fast the job needs to be completed and how the processors are kept busy.

Under the widely used assumptions that the problem can be divided into completely sequential and parallel parts and that the overheads such as those due to interprocessor communication are negligible, the uniprocessor time t_1 can be expressed as (Refs. 10, 14 and 15)

$$t_1 = t_{1_s} + t_{1_p} \tag{38}$$

Since only the parallel part can be speeded up, the run time with p processors is

$$t_p = t_{1_s} + \frac{t_{1_p}}{p} \tag{39}$$

The above equation is used to predict the uniprocessor run time $t_1 (= t_{1_s} + t_{1_p})$. Basically, the same job is run for different values of p, the corresponding parallel run times t_p are measured, and then t_{1_s} and t_{1_p} are computed by a least-square approach. It is convenient to express t_{1_s} and t_{1_p} in terms of dimensionless serial fraction f, which by definition is:

$$t_{1_s} = ft_1 \text{ and } t_{1_p} = (1-f)t_1 \tag{40}$$

Now, the parallel run time t_p in Eq. (39) can be expressed as

$$t_p = ft_1 + \frac{(1-f)t_1}{p} \tag{41}$$

and the speedup in Eq. (36) takes the form

$$S_p = \frac{1}{f + \frac{1-f}{p}} \tag{42}$$

Or, equivalently, serial fraction f can be expressed in terms of speedup S_p and number of processors p:

$$f = 1 - \frac{1 - 1/S_p}{1 - 1/p} \tag{43}$$

After normalizing the uniprocessor run time to unity and differentiating speedup S_p with respect to parallel fraction $(1-f)$, we get

$$\frac{dS_p}{d(1-f)} = p^2 - p \tag{44}$$

which shows the dominant influence of the parallel fraction on speedup. Although p takes on only integer values and the serial fraction is not strictly independent of the number of processors, an instructive result follows from differentiating $1/E_p$ with respect to p:

$$\frac{d}{dp}\left(\frac{1}{E_p}\right) = f \tag{45}$$

The above equation clearly shows that the serial fraction is an indirect measure of efficiency. As elaborated subsequently on the basis of numerical results, the serial fraction f, which is basically problem dependent, is a very small number in Floquet analysis and generally decreases with increasing order. Thus, Eq. (45) shows that any appreciable deviation from linearity in the $1/E_p$-versus-p curve is an indication of the loss of parallelism (Ref. 15).

Modeling and Formulation

Structural Dynamics

We use both rigid flap-lag and elastic flap-lag-torsion models of multibladed rotors resting on rigid or stationary supports. The rigid blade representation describes a rotor with rigid blades executing flap and lag degrees of freedom. The elastic flap-lag-torsion representation includes flap, lag and torsion degrees of freedom. The equations of flap bending, lag bending and torsion are based on Hamilton's principle with a second-order ordering scheme. The spatial dependence is treated by a Galerkin scheme using the uncoupled nonrotating mode shapes corresponding to a uniform cantilever beam. The Galerkin-type integrals are calculated numerically. For additional details on these equations of motion, see Ref. 7.

Aerodynamics

The airfoil aerodynamics is based on the ONERA dynamic stall models of lift, drag and pitching moment (Refs. 16 and 17), and the downwash dynamics is based on a finite-state three-dimensional wake model (Ref. 18). Consistently upgraded wake modeling is used, in which the number of radial shape functions for each harmonic is fixed in a predetermined manner. Moreover, the airfoil theory includes the effects of reversed flow and large angles of attack, and the wake theory accounts for the finite number of blades (Ref. 18).

Results

We present data on run time and other parallel-performance measures as well as on computational reliability. These data are used to compare parallel fast Floquet analysis with three other analyses: parallel analysis based on classical Floquet theory (Ref. 6) and sequential analyses based on classical and fast Floquet theories (Refs. 5 and 7). The run-time data include run time and its variation with the number of states and processors; they are directly measured quantities. In addition to run time, the parallel-performance data also include speedup, efficiency, sequential and parallel fractions, and the rate of change of efficiency with respect to the number of processors; they are computed from Eqs. (36), (37), (43) and (45), respectively. Similarly, the computational-reliability data comprise the condition number of the Jacobian in Newton iteration in the converged cycle, the eigenvalue condition numbers and the residual errors of the corresponding eigenpairs, computed from Eqs. (32), (34) and (35), respectively.

The run time refers to the elapsed time in predicting trim and stability of isolated rotors in trimmed flight. We consider both rigid flap-lag as well as elastic flap-lag-torsion models with dynamic stall and wake; in each case three- and four-bladed rotors with identical blades are treated. The model order is increased by increasing the number of wake harmonics, aerodynamic elements per blade and the number of modes (for the elastic blade model only). For example, for the rigid-blade model, the order varies from 94 for a three-bladed rotor with five aerodynamic elements per blade and a first-harmonic wake distribution to 430 for a four-bladed rotor with 10 elements per blade and a wake distribution with 19 harmonics. The results are addressed with respect to the number of states or the order of the model, and the number of blades ($Q = 3$ or 4) per se is not explicitly mentioned; however, it is accounted for in computing the number of states. The bulk of the results in Figs. 2–8 is based on the simpler rigid flap-lag model (Figs. 2–5, 7 and 8) with only a passing reference to the elastic flap-lag-torsion model (Fig. 6), which corroborates the results based on the simpler model. The computations are done on a massively parallel MP-1 system with 8192 processors. For a model of fixed order, the number of processors are selected automatically by the compiler system. In some cases, the number of processors is stipulated through a program directive; such cases are mentioned explicitly. If not stated otherwise, the results refer to a rigid flap-lag model with dynamic stall and wake for the following parameters: $\mu = 0.3$, $\gamma = 5$, $P_\beta = 1.15$, $\omega_\zeta = 1.14$, $\sigma = 0.05$, $a = 6.28$, $C_{d_0} = 0.0079$, $C_w = 0.00375$, and $\overline{f} = 0.01$; any deviation is so stated explicitly.

Figure 2 shows a comparison of run times from two parallel analyses based on classical- and fast-Floquet theories; that is, parallel Floquet analysis of Ref. 6 and parallel fast-Floquet analysis of the present study. The model order varies from 94 to 430. Throughout, parallel fast-Floquet analysis provides nearly a Q-fold reduction in run time. As for the specifics of the run-time variation, both the parallel analyses show jump-type increases around $M = 128, 192, \cdots$, *etc.*, and nearly constant variation within the intervals $64 \leq M \leq 128$, $129 \leq M \leq 192$ for relatively small order models and linear variation with a small slope within the intervals $193 \leq M \leq 256$, $257 \leq M \leq 384$ for relatively large order models. The jump-type variation and the linear variation within certain intervals is due to the virtualization of arrays larger than the physical processor-element array (the actual number of processors available) and the subsequent increase in the number of memory layers used. In MasPar, the processors are arranged in a 64×128 two-dimensional mesh grid. When the dimension of a variable array exceeds 64, the additional data of the variable array are stored in different memory layers. The total number of memory layers utilized increases with increasing model order, which in turn increases the communication overhead in accessing data stored in different layers. As an example, for $M = 301$, the number of memory layers created is 15 (number of layers in the x-direction $\times$ number of layers in the y-direction; that is, $301/64 \times 301/128$

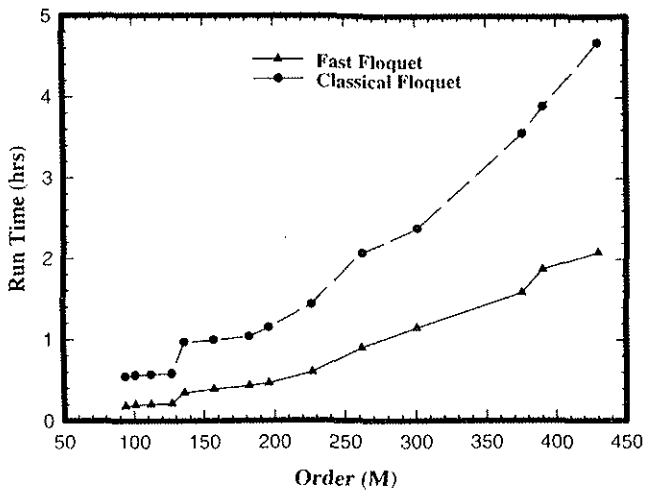

Figure 2: Run-Time Variations for Trim and Stability from Parallel Analyses Based on Classical and Fast-Floquet Theories.

leads to 5×3 on the MasPar MP-1 with a 64×128 array). For $M = 430$, this number increases to 28 ($= 7 \times 4$).

Figure 3 shows a comparison of run times for sequential and parallel analyses, both based on the fast-Floquet theory. The sequential code is run on a VAX 4320 mainframe computer. Admittedly, the run times are from two different machines and a direct comparison between them is not meaningful. However, of considerable significance is the growth of run time with the order since this growth roughly indicates the feasibility of the sequential and parallel fast-Floquet codes for comprehensive analyses that require models with thousands of states. In Fig. 3a, the order varies from 94 to 169; the corresponding run time varies from 6 hours 45 minutes to 2 days, 11 hours and 12 minutes. Indeed, the run time grows between quadratically and cubically with order ($\approx M^{2.4}$), and thus, the results are limited to relatively small-order models ($94 \leq M \leq 169$). Despite this limitation, Fig. 3a does demonstrate that it is not practical to treat models with thousands of states by the sequential fast-Floquet code. Recall that compared to classical Floquet theory, the fast-Floquet theory brings in nearly Q-fold reduction in run times in both the sequential and parallel approaches.

Figure 3b shows both the sequential and parallel run times versus order, which varies from 94 to 430. The parallel run time varies more or less linearly with a very small slope, although some of the finer details such as jumps of the type observed in Fig. 2 are masked by the scaling adopted along the ordinate. Thus, Fig. 3b shows the feasibility of treating models with thousands of states by the parallel fast-Floquet code and thereby the potential practical utility of the parallel fast-Floquet approach in comprehensive analyses.

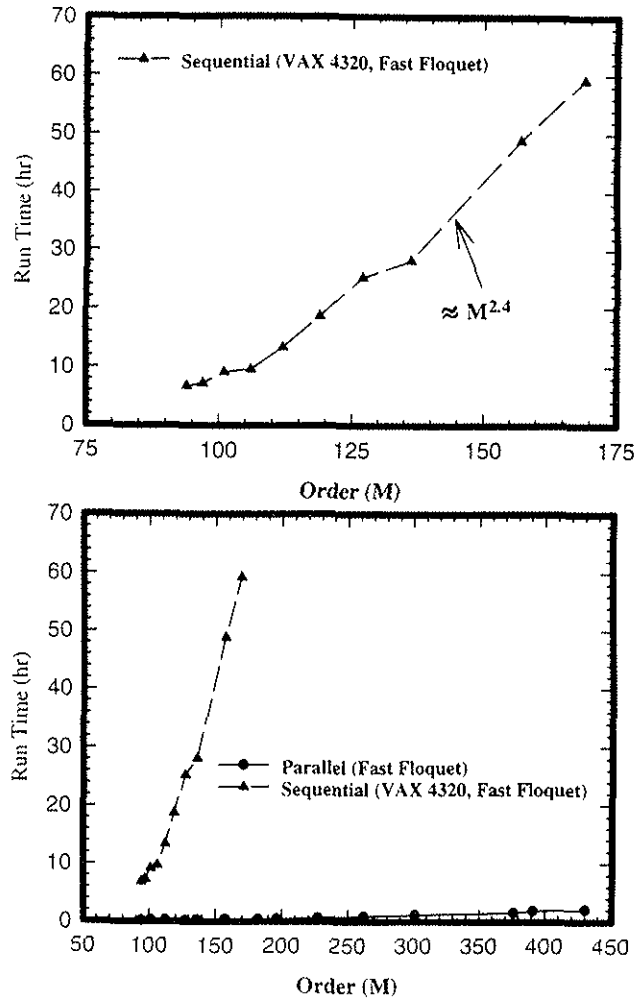

Figure 3: Run-Time Variations for Trim and Stability from Sequential and Parallel Analyses Based on the Fast-Floquet Theory.

Figure 4 shows the speedup S_p, which is the most widely used parallel-performance measure (Ref. 10). As seen therein, the speedup increases with the order for $94 \leq M \leq 376$ and decreases for $376 < M \leq 430$. Compared to the predicted uniprocessor run time, the parallel run time is dramatically reduced. For example, for $M = 376$, the speedup is 2000; that is, for the same parallel code, the predicted uniprocessor run time is 2000 times the observed parallel or multiprocessor run time. Overall, as seen from Fig. 4, the effectiveness of parallelism increases with increasing problem size or model order, and the decrease in speedup for $M > 376$ is perhaps associated with increased overhead such as that due to interprocessor communications. For a more complete understanding of speedup, it is necessary to study the corresponding number of processors and the variation of run time with the order. This is done in Fig. 5 for the rigid-blade model and in Fig. 6 for the elastic flap-lag-torsion model.

Figure 5 shows speedup, efficiency and run time versus the number of processors for four models of order $M = 94, 262, 376, 430$. In the massively parallel MP-

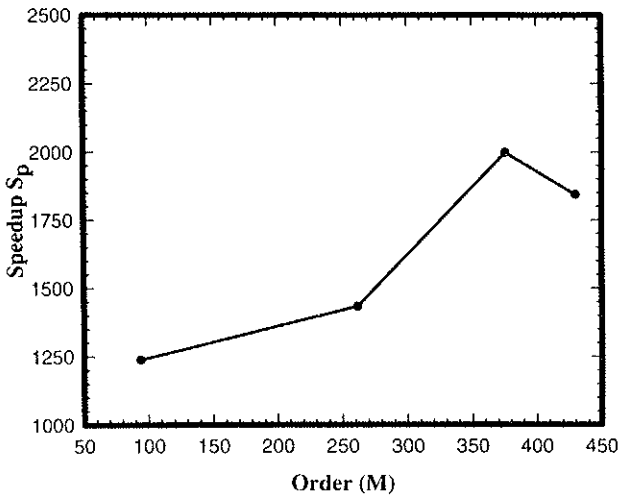

Figure 4: Speedup Versus Order for Trim and Stability from the Parallel Fast-Floquet Analysis.

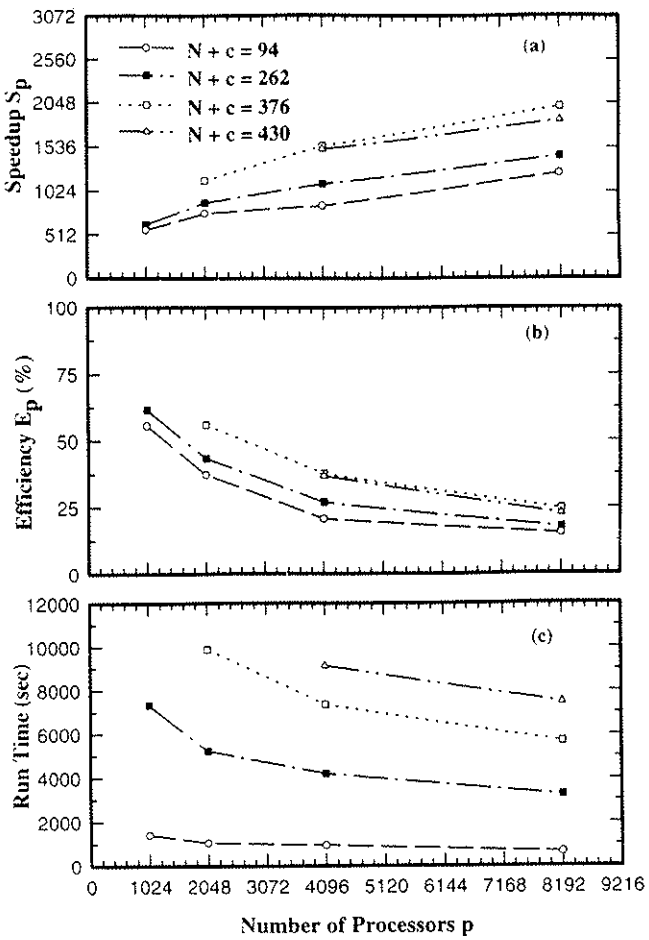

Figure 5: Variations of Speedup, Efficiency and Run Time with the Order and Number of Processors for Trim and Stability from Parallel Fast-Floquet Analysis.

1 computer with 8192 processors, we can solve a problem with 1024, 2048, 4096 and 8192 processors. The model order and number of processors are identified in the figure. As expected, for a model of fixed order and with increasing number of processors, while the speedup increases, the efficiency and run time decreases. Similarly, with a fixed number of processors, the speedup, efficiency and run time increase with increasing order. As an example, for a model of order 376 and with 2048 processors, the speedup and efficiency are 1147 and 56%, and the run time is 2 hrs, 44 mins and 32 secs (= 9872 secs). For the same model, the run time can be reduced by nearly 26% (= 2518 secs) by doubling the number of processors but with reduced efficiency; that is, the efficiency comes down to nearly 38%. These figures should be interpreted in a relative sense of trade off between how fast is fast enough and how busy the processors are kept. Figure 5 also shows that using a large number of processors for relatively small problems results in severe underutilization; for example with $M = 96$ and $p = 8192$, efficiency $E_p = 25\%$ and the run time remains virtually constant for $p > 2048$. This reflects the fact that the problem is not large enough to exploit reasonably the available computing power. Figure 6, based on the elastic flap-lag-torsion model, essentially corroborates the results in Fig. 5, which is based on the rigid flap-lag model, except that speedup, efficiency and run time are higher for a fixed value of the number of processors. Stated otherwise, compared to the rigid blade model, the number of concurrent computations is much more extensive and thereby the available computing resources are used more effectively.

The last two figures demonstrate the dominant influence of serial fraction f (or equivalently parallel fraction 1-f) on speedup and efficiency and thereby on the overall effectiveness of parallelism. Figure 7 shows f and $(1-f)$ versus the order and Fig. 8 shows the variation of $1/E_p$ with p. To help present these data and appreciate their significance, we revisit Eqs. (43–45). For example, Eqs. (43) and (44) show that the speedup is bounded by $1/f$ no matter how many processors are used, and that it grows quadratically with the number of processors. Similarly, Eq. (45) shows that the rate of change of $1/E_p$ with respect to the number of processors is equal to f, thus providing a measure of the connection among the number of processors, serial fraction and efficiency; also, see comments following Eq. (45). Figure 7 shows that serial fraction f generally decreases with increasing order but is accompanied by some localized variations. For example, serial fraction f, which is close to 0.0008 for $M = 94$, decreases to 0.0004 for $M = 376$. Thus, the upper bound of S_p increases from 1250 for $M = 94$ to 2500 for $M = 376$. The increasing degree of parallelism with decreasing f is well reflected in Fig. 8, which shows that the linearity of the $1/E_p$-versus-p curve increases with increasing order. Stated otherwise, the decrease in serial fraction f with increasing

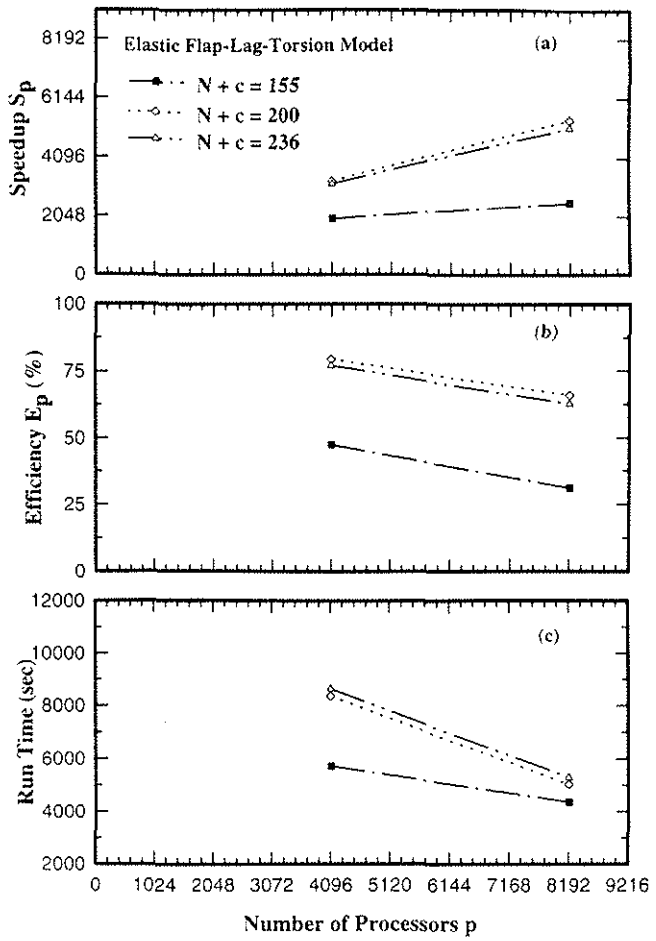

Figure 6: Variations of Speedup, Efficiency and Run Time with the Order and Number of Processors for Trim and Stability from Parallel Fast-Floquet Analysis of Flap-Lag-Torsion Models.

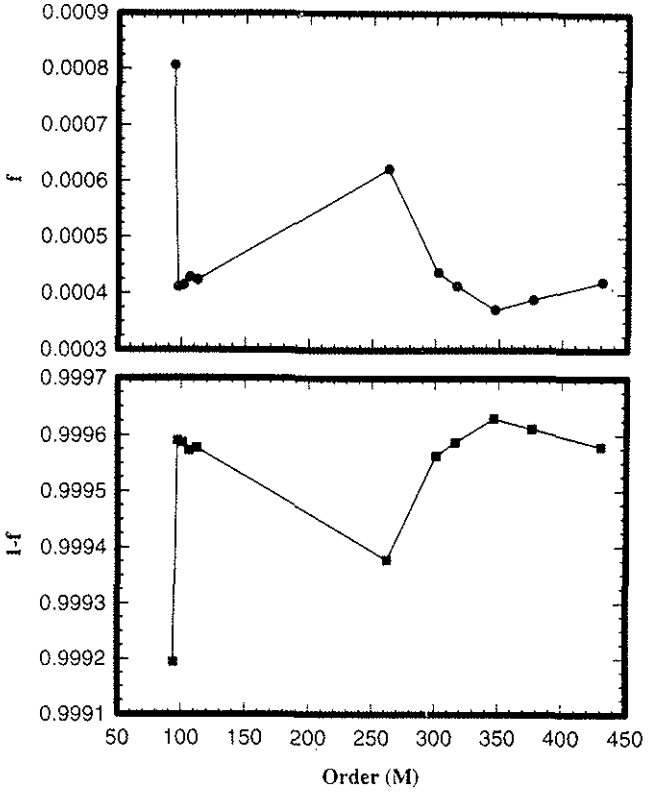

Figure 7: Variations of Sequential and Parallel Fractions with the Order for Trim and Stability from Parallel Fast-Floquet Analysis.

model order means increasing degree of parallelism and increasing linearity of the $1/E_p$-versus-p curve. Thus, in summary, Figs. 7 and 8 demonstrate the fact that the shooting method by the fast Floquet theory is tailored to parallelism, the higher the model order, the better the degree of parallelism.

Table 1 gives a sample of the computational reliability parameters for all four types of analysis — sequential and parallel analyses based on classical and fast Floquet theories. The results are based on the rigid flap-lag model with dynamic wake. For a given problem size or model order, the condition number of the Jacobian in the converged cycle, eigenvalue condition number of the lag-regressive mode and residual error of the corresponding eigenpair are shown. We mention in passing that the eigenvalue condition number of the lag-regressive mode often corresponds to the maximum value of the eigenvalue condition number. The condition numbers of the Jacobian are acceptable and the eigenvalue condition numbers are close to the ideal value of unity (Refs. 2 and 13). Moreover, the residual errors of the eigenpairs are indeed negligible. Overall, the computational reliability figures of all four analyses are comparable.

Concluding Remarks

The parallel fast Floquet analysis exploits the fast-Floquet theory and parallelism to predict trim and stability of large helicopter models ($N > 100$). It applies to single-rotor and multirotor models, each rotor having identical blades. Its computational reliability is compared with three other analyses: sequential analysis based on classical and fast-Floquet theories and parallel analysis based on classical Floquet theory; the reliability figures are comparable. Both parallel analyses based on classical and fast-Floquet theories are nearly identical with respect to parallel-performance metrics of speedup, efficiency and parallel fraction. However, the fast-Floquet theory brings in an additional Q-fold reduction in run time.

With increasing model order, while the run time for parallel fast-Floquet analysis remains nearly constant, the corresponding sequential run time even with the fast-Floquet theory is much longer and rapidly increases. Furthermore, the parallel fast-Floquet analysis should prove to be more effective to multirotor models owing to the necessity of finding a common period, which in turn should bring in a higher-than-Q-fold reduction in run time.

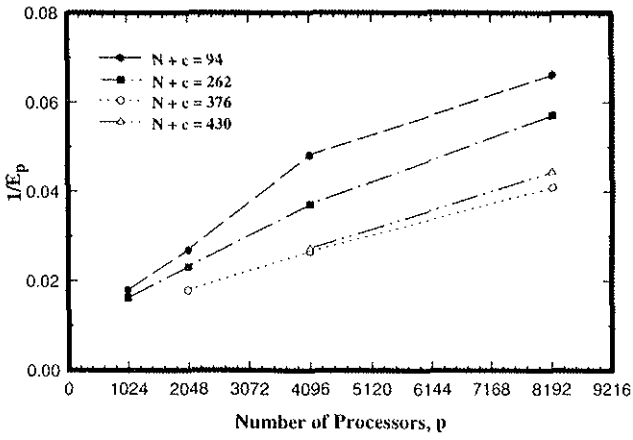

Figure 8: Variation of Efficiency E_p with the Number of Processors p for Models of Varying Order.

To sum up: The parallel fast-Floquet analysis dramatically increases the saving in run time and demonstrates its feasibility to models with thousands of states, and thereby its applications to comprehensive and design analyses offer promise.

Acknowledgment

This work is sponsored by the Army Research Office (Grant DAAH04-94-G0185). We are also grateful to MasPar Computer Corporation, Sunnyvale, California, for providing free computer time on the MasPar MP-1 system with 8192 processors.

References

1. Gaonkar, G. H. and Peters, D. A., "Review of Floquet Theory in Stability and Response Analysis of Dynamic Systems with Periodic Coefficients," The R. L. Bisplinghoff Memorial Symposium Volume on Recent Trends in Aeroelasticity, Structures and Structural Dynamics, University Presses of Florida, 1987, pp. 101–119.

2. Achar, N. S. and Gaonkar, G. H., "Helicopter Trim Analysis by Shooting and Finite Element Methods with Optimally Damped Newton Iterations," *Journal of the American Institute of Aeronautics and Astronautics*, Vol. 31, (2), Feb 1993, pp. 225-234.

3. Achar, N. S. and Gaonkar, G. H., "An Exploratory Study of a Subspace Iteration Method as an Alternative to the QR Method for Floquet Eigenanalysis," *Journal of Mathematical and Computer Modeling*, Vol. 19, (3/4), Apr 1994, pp. 69-73.

4. Nagabhushanam, J. and Gaonkar, G. H., "Automatic Identification of Modal Damping from Floquet Analysis," *Journal of the American Helicopter Society*, Vol. 40, (2), Apr 1995, pp. 39-42.

5. Chunduru, S. J., "*Dynamic Stall and Three-Dimensional Wake Effects on Trim, Stability and Load of Hingeless Rotors with Fast Floquet Theory*, Ph. D. Thesis, College of Engineering, Florida Atlantic University, Boca Raton, FL, Dec 1995.

6. Subramanian, S., Gaonkar, G. H., Nakadi, R. M. and Nagabhushanam, J., "Parallel Computing Concepts and Methods for Large Scale Floquet Analysis of Helicopter Trim and Stability, *Journal of the American Helicopter Society*, Vol. 41, (4), Oct 1996.

7. Subramanian, S., *Dynamic Stall and Three-Dimensional Wake Effects on Isolated Rotor Trim and Stability with Experimental Correlation and Parallel Fast Floquet Analysis*, Ph. D. Thesis, College of Engineering, Florida Atlantic University, Boca Raton, FL, Aug 1996.

8. Peters, D. A., "Fast Floquet Theory and Trim for Multi-Bladed Rotorcraft," *Journal of the American Helicopter Society*, Vol. 40, (3), Jul 1995, pp. 82-89 (Also see Proceedings of the 51st Annual Forum of the American Helicopter Society, Fort Worth, TX, May 9-11, 1995).

9. McVicar, J. S. G. and Bradley, R., "A Generic Tilt-Rotor Simulation Model with Parallel Implementation and Partial Periodic Trim Algorithm," Proceedings of the Eighteenth European Rotorcraft Forum, Avignon, France, Sep 15-18, 1992, pp. 138.1-138.19.

10. Morse, S. H., *Practical Parallel Computing*, Academic Press, New York, 1994, Chapter 2.

11. Lewis, T. G., "Where is Computing Headed?" *Computer*, Vol. 27, (8), Aug 1994, pp. 59-63.

12. Anderson, E., et al., *LAPACK User's Guide*, SIAM Publication, pp. 7-43, 1992.

13. Ravichandran, S., Gaonkar, G. H., Nagabhushanam, J. and Reddy, T. S. R., "A Study of Symbolic Processing and Computational Aspects in Helicopter Dynamics," *Journal of Sound and vibration*, Vol. 137, (3), Mar 1990, pp. 495-507.

14. Crowl, L. A., "How to Measure, Present and Compare Parallel Performance," *IEEE Parallel and Distributed Technology: Systems and Applications*, Spring 1994, pp. 9-25.

15. Karp, A. H. and Flatt, H. P., "Measuring Parallel Processor Performance," *Communications of the ACM*, Vol. 33, (5), May 1990, pp. 539-543.

16. Peters, D. A., "Toward a Unified Lift Model for Use in Rotor Blade Stability Analysis," *Journal of the American Helicopter Society*, Vol. 30, (3), Jul 1985, pp. 32-42.

17. Petot, D., "Differential Equation Modeling of Dynamic Stall," ONERA, Technical Note No. 5, 1989.

18. Peters, D. A., Boyd, D. D. and He, C. J., "Finite-State Induced-Flow Model for Rotors in Hover and Forward Flight," *Journal of the American Helicopter Society*, Vol. 34, (4), Oct 1989, pp. 5-17.

Table 1: A Sample of Computational Reliability Parameters for Flap-Lag Stability with Dynamic Wake.

S: sequential; P: parallel

System Order, $N + c$	Condition number of the Jacobian matrix for the converged cycle, Classical (Fast) Floquet Theories	Eigenvalue condition number for Lag Regressive mode damping, Classical (Fast) Floquet Theories	Residual error of the corresponding Eigenpair, Classical (Fast) Floquet Theories
107 (S)	562.79(592.83)	1.17658(1.1768)	1.462E-13(2.588E-14)
107 (P)	603.27(628.78)	1.17658(1.1709)	1.908E-13(2.150E-13)
136 (S)	560.17(603.88)	1.17638(1.1792)	2.033E-13(3.155E-14)
136 (P)	598.62(629.79)	1.17638(1.1769)	1.786E-13(3.642E-13)