NASA Technical Paper 3519

PLATSIM: An Efficient Linear Simulation and Analysis Package for Large-Order Flexible Systems

Peiman G. Maghami and Sean P. Kenny
Langley Research Center • Hampton, Virginia

Daniel P. Giesy
Lockheed Martin Engineering & Sciences • Hampton, Virginia

National Aeronautics and Space Administration
Langley Research Center • Hampton, Virginia 23681-0001

September 1995

Available electronically at the following URL address: http://techreports.larc.nasa.gov/ltrs/ltrs.html

Printed copies available from the following:

NASA Center for AeroSpace Information
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650

Contents

Chapter 1

Introduction

The software package PLATSIM provides time and frequency domain analysis of generic space platforms that can be modeled as linear, time-invariant systems. PLATSIM can perform open-loop analysis or closed-loop analysis with different linear time-invariant spacecraft control systems, such as attitude control systems with or without flexible body controls, active isolation systems, and payload or instrument local control systems. In the time domain analysis, PLATSIM simulates the response of the space platform to disturbances and calculates the jitter and stability values from the response time histories. In the frequency domain analysis, PLATSIM calculates frequency response function matrices and provides the corresponding Bode plots. While PLATSIM was designed for analyzing space platforms, it only assumes that it has a finite element model of a structure that is being excited by force and/or torque inputs. Thus, any structure (e.g., aeronautical, automotive, structural, or mechanical) that fits this model can be analyzed by PLATSIM.

Several novel algorithmic features have been developed and are incorporated in PLATSIM. These features result in a significant increase in the computational efficiency for all analyses. PLATSIM exploits the particular form of sparsity (block diagonal with 2×2 blocks) of the plant matrices both in the continuous form used in the frequency analysis and in the discretized form used in the time simulation. A new and original algorithm for the efficient computation of closed-loop (as well as open-loop) frequency response functions for large-order systems has been developed and implemented within PLATSIM. This algorithm is an enabling technology for the analysis of large-order systems. Furthermore, a novel and efficient jitter analysis routine that determines jitter and stability values from time simulations in an efficient manner has been developed for and is incorporated in the PLATSIM package (ref. 1). This routine increased the computational speed by more than 1600 times in typical examples over the brute force approach of sweeping minima and maxima.

PLATSIM allows the user to maintain a database of performance measurement outputs on the space platform and a database of disturbance scenarios. An individual run of PLATSIM can use all performance outputs or a user-selected subset, and the user selects one disturbance scenario for each run. Output options for time domain analysis include on-screen plots of time histories at user-selected output locations (e.g., instrument boresight), encapsulated Adobe PostScript (EPS) files of these plots, tables of jitter values for user-provided time window sizes, and files containing the time history data in either compressed or full form. Output options for frequency domain analysis include on-screen Bode plots, EPS files of these plots, and files containing the plot data.

PLATSIM requires MATLAB® version 4.1 or higher. MATLAB, a product of The MathWorks, Inc., is a technical computing environment for high-performance numeric computation and visualization (ref. 2). PLATSIM also uses the Control System Toolbox, which is another product of The MathWorks, Inc. User input to PLATSIM is provided in MATLAB readable data files and MATLAB function M-files.

PLATSIM uses six main modules: a plant definition module, a spacecraft control system module (user-supplied), a disturbance module (user-supplied), a simulation module, a frequency analysis module, and a jitter analysis module. All modules are written as MATLAB M-files, which are compatible with MATLAB version 4.1 or higher. However, the jitter analysis module, the frequency analysis module, and the simulation module have also been rendered as FORTRAN 77 MATLAB MEX-files to improve the execution time for those modules.

PLATSIM uses a sparse matrix formulation of the spacecraft modal model (which differs from MATLAB's built-in sparse utilities) for the spacecraft dynamic model. Utilizing sparseness makes the time simulations and frequency analysis efficient, particularly when large numbers of modes are required to capture the true dynamics of the spacecraft. The user provides a linear, time-invariant, continuous-time state space model of the spacecraft control system. PLATSIM performs time simulation of the system by first discretizing the spacecraft dynamics (taking advantage of the sparsity in the model) and the spacecraft control system and then carrying out an algebraic state propagation for discrete time steps. Frequency domain analysis consists of evaluating the system transfer function over a user-specified range of frequencies and then extracting gain and phase information.

PLATSIM requires the following user inputs: modal data of the spacecraft as generated by finite element analysis, damping ratios for flexible modes, information about control actuators, measurement feedback sensors, and performance instrument outputs (e.g., boresight measurements), spacecraft disturbance data, and spacecraft control system matrices. The finite element data, including the modal frequencies and mode shapes at required grid points, must be provided either in ASCII files `omega.dat` and `phi.dat`, or MATLAB binary files `omega.mat` and `phi.mat`, respectively. The damping ratios for the structural modes are defined in the MATLAB function `mkdamp`. The data for instrument connectivity and input/output selection must be provided through the MATLAB function `instdata`. The required data on spacecraft disturbances, such as disturbance profiles, elements, and integration step sizes, must be provided through the MATLAB function `distdata`. In the EOS (Earth Observing System) example used throughout this manual, `distdata` calls other user-defined MATLAB routines, which perform tasks such as calculating individual disturbance profiles. The matrices modeling the space platform control system (SCS) must be output by the MATLAB function `formscs`. Any spacecraft control system can be implemented by PLATSIM as long as it is stable and has a linear time-invariant model. The space platform control system can be a basic spacecraft attitude control system with possible augmentations by additional control systems to control local vibrations, isolate payloads, or articulate payloads.

The program can be used in three modes, a GUI mode, a command-driven mode, or a batch mode. The three modes differ in the way the required and optional flags and parameters that control execution are defined. In the GUI mode, the parameters and flags are chosen from pop-up MATLAB menus with a keypad and a mouse. In the command-driven mode, the required parameters and flags are defined through keyboard responses to PLATSIM questions; however, any optional parameters are defined with MATLAB command lines. In the batch mode, all flags and parameters are defined in MATLAB command lines, which can be placed in an ASCII input file.

Although PLATSIM was developed to analyze generic space platforms, the Earth Observing System EOS-AM-1 (ref. 3) is used throughout this manual as an example. Furthermore, several M-files and data files included in the PLATSIM distribution, some of which are listed in appendix A, correspond to the EOS-AM-1 spacecraft. These files are the spacecraft control system defined in `formscs.m`, the instrument types and connectivity data defined in `instdata.m`, the finite element data defined in `omega.mat` and `phi.mat`, the damping schedule defined in `mkdamp.m`, and the spacecraft disturbance data defined in `distdata.m` and its supporting routines. These files can serve as templates for the user-supplied files for other space platform applications.

The MATLAB convention for naming MEX-files is to append to the function name an extension that starts with the characters `.mex` and may have additional characters depending on the computer or operating system. For example, in this document, the MEX-file for function jitter is referred to as `jitter.mex*`. The `*` can stand for 0 or more additional characters in accordance with the UNIX or DOS wild card character convention.

Chapter 2

Mathematical Formulation

Second-Order Modal Equations

The dynamics of the spacecraft can be written in a second-order form as

$$
\begin{aligned}
M\ddot{x} + D\dot{x} + Kx &= Hu + H_d w \\
y &= C_p x + C_r \dot{x} \\
y^{pr} &= C_p^{pr} x + C_r^{pr} \dot{x} + C_a^{pr} \ddot{x}
\end{aligned}
$$

where M, D, and K are the $n \times n$ mass, damping, and stiffness matrices, respectively; x is the $n \times 1$ position and/or attitude vector; u is the $m \times 1$ spacecraft control input vector, which includes the attitude control system inputs along with any additional augmenting controllers; w is the $r \times 1$ disturbance vector; H is the $n \times m$ control input influence matrix; and H_d is the $n \times r$ disturbance influence matrix. In addition, y is the $q \times 1$ measurements output vector and y^{pr} is the $l \times 1$ performance output vector; C_p and C_r are $q \times n$ measurement output influence matrices; and C_p^{pr}, C_r^{pr}, and C_a^{pr} are the $l \times n$ performance output influence matrices.

The elements of the control input and the disturbance influence matrices depend on how the control inputs and the disturbances are applied to the spacecraft. In general, element (i, j) of matrix H represents the influence of the control input j on the position coordinate i (the ith element of position vector x). For example, if the second control input applies a force and/or torque solely on position coordinate 15, then all elements of the second column of matrix H would be 0 except its 15th element, which would be 1. Generally, if the jth control input applies a force or torque to only s position coordinates $x_{l_1}, x_{l_2}, \ldots, x_{l_s}$, each with a distribution factor of α_i with $i = 1, \ldots, s$, then the elements of column j of matrix H would all be 0 except for the elements $l_1, l_2, \ldots, l_s$, which take values $\alpha_1, \alpha_2, \ldots, \alpha_s$, respectively. The elements of the disturbance influence matrix are defined in the same manner.

The elements of the measurement output (and the performance output) influence matrices depend on the measurement process. In general, element (i, j) of matrix C_p represents the contribution of the response at position coordinate j to measurement i, and element (i, j) of C_r represents the contribution of the rate response at position coordinate j to measurement i. For example, if measurement output k has contributions only from the response at t position coordinates $x_{o_1}, x_{o_2}, \ldots, x_{o_t}$, each with a contribution factor θ_i with $i = 1, \ldots, t$, then the elements of row k of matrix C_p would all be 0 except for elements $o_1, o_2, \ldots, o_t$, which take values $\theta_1, \theta_2, \ldots, \theta_t$, respectively. Moreover, all elements of row k of matrix C_r would be 0. The elements of the performance output influence matrices C_p^{pr}, C_r^{pr}, and C_a^{pr} can be defined in a similar manner.

If the second-order system is transformed into normal mode coordinates, and p of the normal modes are retained to capture the relevant dynamics of the spacecraft, then the system equations can be written in a modal form as

$$
\begin{aligned}
\bar{M}\ddot{q} + \bar{D}\dot{q} + \bar{K}q &= \bar{H}u + \bar{H}_d w \\
y &= \bar{C}_p q + \bar{C}_r \dot{q} \\
y^{pr} &= \bar{C}_p^{pr} q + \bar{C}_r^{pr} \dot{q} + \bar{C}_a^{pr} \ddot{q}
\end{aligned}
$$

where $\bar{M}$, $\bar{D}$, and $\bar{K}$ are the $p \times p$ modal mass, damping, and stiffness matrices, respectively; q is the $p \times 1$ vector of modal coordinates; and $\bar{H}$ and $\bar{H}_d$ are the $p \times m$ spacecraft control input and the $p \times r$ disturbance influence matrices in modal coordinates, respectively. Futhermore, $\bar{C}_p$ and $\bar{C}_r$ are $q \times p$ measurement output influence matrices in modal coordinates, and $\bar{C}_p^{pr}$, $\bar{C}_r^{pr}$, and $\bar{C}_a^{pr}$ are $l \times p$ performance output influence matrices in modal coordinates.

PLATSIM assumes that the mode shapes are normalized with respect to the mass matrix and damping is modal. These assumptions result in $\bar{M} = I_{p \times p}$, $\bar{D} = \mathrm{diag}\,(2\zeta_1\omega_1, 2\zeta_2\omega_2, \ldots, 2\zeta_p\omega_p)$, and $\bar{K} = \mathrm{diag}\left(\omega_1^2, \omega_2^2, \ldots, \omega_p^2\right)$, where ω_i and ζ_i are the open-loop frequencies and damping ratios.

The spacecraft control input and disturbance influence matrices are defined as follows:

$$
\begin{aligned}
\bar{H} &= \Phi^T H \\
\bar{H}_d &= \Phi^T H_d
\end{aligned}
$$

The measurement and performance output influence matrices are given by

$$\begin{aligned}
\bar{C}_p &= C_p\Phi \\
\bar{C}_r &= C_r\Phi \\
\bar{C}_p^{pr} &= C_p^{pr}\Phi \\
\bar{C}_r^{pr} &= C_r^{pr}\Phi \\
\bar{C}_a^{pr} &= C_a^{pr}\Phi
\end{aligned}$$

The columns of matrix Φ are the p retained mode shapes:

$$\Phi = [\phi_1, \phi_2, \ldots, \phi_p]$$

First-Order Modal Equations

The second-order modal equations can be rewritten in a first-order form as follows:

$$\left.\begin{aligned}
\dot{x}_s &= A_s x_s + B_s u + B_d w \\
y &= C x_s \\
y^{pr} &= C_1^{pr} x_s + C_2^{pr} \dot{x}_s
\end{aligned}\right\} \tag{1}$$

The vector x_s is the plant state vector whose components are as follows:

$$x_s = \begin{Bmatrix} q_1 \\ \dot{q}_1 \\ q_2 \\ \dot{q}_2 \\ \vdots \\ \vdots \\ q_p \\ \dot{q}_p \end{Bmatrix}$$

Furthermore, the vectors y and y^{pr} are the plant (spacecraft) measurement and performance outputs, respectively. The matrix A_s is the plant state matrix and has the following form:

$$A_s = \begin{bmatrix} A_s^1 & 0 & \ldots & 0 \\ 0 & A_s^2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & A_s^p \end{bmatrix} \tag{2}$$

where

$$A_s^i = \begin{bmatrix} 0 & 1 \\ -\omega_i^2 & -2\zeta_i\omega_i \end{bmatrix} \tag{3}$$

The matrix B_s is the control input influence matrix, which is formed by setting its odd-numbered rows to 0 and by using the rows of $\bar{H}$ for its even-numbered rows as follows:

$$B_s = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ \bar{H}_{11} & \bar{H}_{12} & \ldots & \bar{H}_{1m} \\ 0 & 0 & \ldots & 0 \\ \bar{H}_{21} & \bar{H}_{22} & \ldots & \bar{H}_{2m} \\ \vdots & \vdots & \ldots & \vdots \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 0 \\ \bar{H}_{p1} & \bar{H}_{p2} & \ldots & \bar{H}_{pm} \end{bmatrix} \tag{4}$$

in which $\bar{H}_{ij}$, for example, represents element (i, j) of matrix $\bar{H}$. (In MATLAB notation, $B(1 : 2 : 2p,\ 1 : m) = \text{zeros}(p, m)$ and $B(2 : 2 : 2p,\ 1 : m) = \bar{H}$.) The matrix B_d is formed from $\bar{H}_d$ in the same manner.

The measurement output influence matrix C is defined by setting the odd-numbered columns of C to the columns of $\bar{C}_p$ and by setting the even-numbered columns of C to the columns of $\bar{C}_r$ as follows:

$$C = \begin{bmatrix} \bar{C}_p(1,1) & \bar{C}_r(1,1) & \bar{C}_p(1,2) & \bar{C}_r(1,2) & \ldots & \ldots & \bar{C}_p(1,p) & \bar{C}_r(1,p) \\ \bar{C}_p(2,1) & \bar{C}_r(2,1) & \bar{C}_p(2,2) & \bar{C}_r(2,2) & \ldots & \ldots & \bar{C}_p(2,p) & \bar{C}_r(2,p) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{C}_p(q,1) & \bar{C}_r(q,1) & \bar{C}_p(q,2) & \bar{C}_r(q,2) & \ldots & \ldots & \bar{C}_p(q,p) & \bar{C}_r(q,p) \end{bmatrix}$$

where $\bar{C}_p(i,j)$ and $\bar{C}_r(i,j)$ denote element (i, j) of matrix $\bar{C}_p$ and $\bar{C}_r$, respectively. (In MATLAB notation, $C(1 : q,\ 1 : 2 : 2p) = \bar{C}_p$ and $C(1 : q,\ 2 : 2 : 2p) = \bar{C}_r$.)

Furthermore, C_1^{pr} is defined from $\bar{C}_p^{pr}$ and $\bar{C}_r^{pr}$ in the same fashion, and C_2^{pr} is defined by setting the odd-numbered columns of C_2^{pr} to 0 and the even-numbered columns of C_2^{pr} to the columns of $\bar{C}_a^{pr}$. (In MATLAB notation, C_2^{pr} $(1 : l,\ 1 : 2 : 2p) = \text{zeros}(l,\ p)$ and C_2^{pr} $(1 : l,\ 2 : 2 : 2p) = \bar{C}_a^{pr}$.)

By substituting the first equation of equations (1) into the third, the acceleration term can be replaced by feedthrough as follows:

$$\left.\begin{aligned} \dot{x}_s &= A_s x_s + B_s u + B_d w \\ y &= C x_s \\ y^{pr} &= C^{pr} x_s + D_u^{pr} u + D_w^{pr} w \end{aligned}\right\} \tag{5}$$

The performance output influence matrix is given by

$$C^{pr} = C_1^{pr} + C_2^{pr} A_s$$

The performance feedthrough matrices are

$$D_u^{pr} = C_2^{pr} B_s$$

$$D_w^{pr} = C_2^{pr} B_d$$

Notice that if no performance acceleration output exists $(C_a^{pr} = 0)$, then $\bar{C}_a^{pr} = 0$ and $C_2^{pr} = 0$. Thus, both feedthrough matrices D_u^{pr} and $D_w^{pr} = 0$.

Control System Equations

In this development, the space platform is assumed to be controlled by a linear time-invariant attitude control system. However, additional augmenting controllers that are also linear time invariant can be implemented along with the attitude control system and analyzed by the PLATSIM program. The model of a linear time-invariant spacecraft control system for a typical space platform can be written as

$$\left.\begin{aligned} \dot{x}_c &= A_c x_c + B_c y \\ u &= -C_c x_c \end{aligned}\right\} \tag{6}$$

where x_c denotes the spacecraft control system states; A_c, B_c, and C_c represent the spacecraft control system state matrix, input influence matrix, and output influence matrix, respectively; and y is the measurement output vector defined in the previous section. Typical measurement outputs can consist of roll, roll rate, pitch, pitch rate, yaw, and yaw rate measurements taken at the spacecraft navigational unit.

Discrete Time Simulation

The simulation technique used here is a discrete closed-loop or open-loop simulation of the spacecraft plant and the spacecraft control system. To perform this simulation, the plant matrices and the control system matrices are separately discretized for the given integration step size, and then the discretized dynamics are combined for closed-loop simulation through algebraic state propagation. Equation (5) discretizes as

$$\left.\begin{aligned} x_s(k+1) &= \bar{A}_s x_s(k) + \bar{B}_s u(k) + \bar{B}_d w(k) \\ y(k) &= C x_s(k) \\ y^{pr}(k) &= C^{pr} x_s(k) + D_u^{pr} u(k) + D_w^{pr} w(k) \end{aligned}\right\} \tag{7}$$

The spacecraft control system, equation (6), discretizes as

$$\left.\begin{aligned} x_c(k+1) &= \bar{A}_c x_c(k) + \bar{B}_c y(k) \\ u(k) &= -C_c x_c(k) \end{aligned}\right\} \tag{8}$$

The overbars indicate the discretized system matrices. The discrete plant state matrix $\bar{A}_s$ has the following form:

$$\bar{A}_s = \begin{bmatrix} \bar{A}_s^1 & 0 & \dots & 0 \\ 0 & \bar{A}_s^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \bar{A}_s^p \end{bmatrix}$$

where the square partition matrix $\bar{A}_s^i$ is the discretized counterpart of the partition matrix A_s^i, which can be easily computed through the zero-order hold function in MATLAB (`c2d`). The remaining plant and spacecraft control system matrices can also be transformed into discrete forms through the same MATLAB function.

Frequency Domain Equations

For the open-loop plant, y and u of equation (5) are nonexistent; thus, the equations reduce to the following:

$$\dot{x}_s = A_s x_s + B_d w$$

$$y^{pr} = C^{pr} x_s + D_w^{pr} w$$

The open-loop transfer function from the disturbances w to the performance output y^{pr} is given by

$$T(s) = C^{pr} (sI - A_s)^{-1} B_d + D_w^{pr} \tag{9}$$

The closed-loop system is more complicated. By using equations (5) and (6), the closed-loop dynamics of the controlled spacecraft can be written as

$$\left.\begin{aligned} \dot{x} &= \tilde{A}x + \tilde{B}_d w \\ y^{pr} &= \tilde{C}^{pr} x + D_w^{pr} w \end{aligned}\right\} \tag{10}$$

where x represents the closed-loop state vector defined as

$$x = \begin{Bmatrix} x_s \\ x_c \end{Bmatrix} \tag{11}$$

The matrices $\tilde{A}$, $\tilde{B}$, and $\tilde{C}^{pr}$ are the closed-loop state matrix, disturbance influence matrix, and output influence matrix, respectively. These matrices are defined as follows:

$$\left.\begin{aligned} \tilde{A} &= \begin{bmatrix} A_s & -B_s C_c \\ B_c C & A_c \end{bmatrix} \\ \tilde{B}_d &= \begin{bmatrix} B_d \\ 0 \end{bmatrix} \\ \tilde{C}^{pr} &= \begin{bmatrix} C^{pr} & -D_u^{pr} C_c \end{bmatrix} \end{aligned}\right\} \tag{12}$$

The closed-loop transfer function from the disturbances w to the performance output y^{pr} is given by the following:

$$\tilde{T}(s) = \tilde{C}^{pr} (sI - \tilde{A})^{-1} \tilde{B}_d + D_w^{pr} \tag{13}$$

Chapter 10 contains the efficient methods used by PLATSIM to calculate the transfer functions in equations (9) and (13).

Jitter and Stability Measurement

One measure of pointing jitter in a time series has been described in an internal report by Martin Marietta as the worst case peak-to-peak variation of the actual pointing direction over relatively short time intervals; pointing stability has the same definition except that the time intervals are relatively long. This measure of jitter (or stability) is the one calculated by PLATSIM. These time intervals are referred to here as “windows.”

Recall that $y^{pr}(j)$ is the value at time step j of vector y^{pr} of the performance outputs. A generic component of this vector is denoted by z such that $z(j)$ denotes the (scalar) value of a typical performance output at time step j. Suppose that the vector z has length n and that the window (time interval) that is slid along z during the measurement of jitter is of such a length as to cover k points of z. (If the length of the window is T_w and the time increment between points of z is ΔT, then k is the largest integer such that $(k-1)\Delta T \le T_w$.) Now define $z_{\max}$ and $z_{\min}$ as follows:

$$\begin{aligned} z_{\max}(j) &= \max\,[z(i) \mid j \le i \le j + k - 1] \\ z_{\min}(j) &= \min\,[z(i) \mid j \le i \le j + k - 1] \end{aligned}$$

Then jitter is given by

$$\text{Jitter} = \max\,[z_{\max}(j) - z_{\min}(j) \mid 1 \le j \le n - k + 1]$$

The straightforward way to calculate jitter has computational complexity $O[k(n-k+1)]$ as measured by the number of references into the z vector. A novel algorithm with computational complexity $O(n)$ is implemented in PLATSIM[1] to perform this calculation. If jitter or stability is to be calculated for the same time signal for several windows, additional time savings is realized by using the information in $z_{\max}$ and $z_{\min}$ for a smaller window to calculate $z_{\max}$ and $z_{\min}$ for a larger window. This calculation also has $O(n)$ complexity but with a smaller O constant than the jitter calculation for the smallest window.

[1] A paper by Daniel P. Giesy describing this, titled “Efficient Calculation of Jitter,” has been submitted for publication.

Chapter 3

Input Files

To construct the data needed to use the theory of chapter 2, PLATSIM requires the following information:

1. Information to calculate the first-order dynamic plant equation matrices in equation (5)
2. Matrices for the space platform control system in equation (6)
3. Sampling period to calculate the matrices in the discrete forms of equations (7) and (8)
4. Disturbance time histories (the *w* of eq. (7))
5. Windows for jitter-stability analysis
6. Character information to label outputs and menu buttons

PLATSIM obtains this information from input files that define the modal model of the spacecraft, the spacecraft damping schedule for its elastic modes, the instrument types and connectivity data, the spacecraft disturbances, and the spacecraft control system.

The finite element modal data are read from two MATLAB readable data files, one for frequencies and one for mode shapes. These data files can be either in ASCII format or MATLAB binary MAT-file format. PLATSIM checks to see whether the MATLAB binary file is present. If it is, PLATSIM loads it. Otherwise, the ASCII file is loaded.

The remaining input data are transmitted to PLATSIM by the execution of user-supplied MATLAB function M-files. In these files, the user can load the data needed by PLATSIM from data files, set the output variables to values hardcoded into the M-file, or perform more extensive internal calculations to determine the data. Examples of the latter two approaches are included in the EOS example files distributed with the PLATSIM software. The user must conform exactly to the specifications for input and output parameters for these files.The remainder of this chapter is devoted to presenting the details of these input files.

Finite Element Data

The finite element modal frequency and mode shape data are provided in ASCII files `omega.dat` and `phi.dat` or MATLAB binary files `omega.mat` and `phi.mat`, respectively. While the ASCII files are more convenient for moving the data from a finite element modeling program to PLATSIM, using the binary MAT-files to reduce the loading time for these files is more efficient, particularly for the mode shape data. The user may want to keep both and regenerate the MAT-files from the ASCII files only when the ASCII files change. The data in both ASCII files are assumed to have a free format. Note that both these files should be placed in MATLAB's directory path. They are accessed by loading them into the MATLAB environment in a file named `formplnt.m`.

The file `omega.dat` contains one frequency per line as follows:

$$\begin{matrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_p \end{matrix}$$

where p is the number of modes in the spacecraft model. If p frequencies are defined in `omega.dat`, modal amplitudes on exactly p modes should be defined in `phi.dat` and the ith mode in `phi.dat` should correspond to frequency ω_i.

The file `omega.mat` should contain a variable `omega`, which is a $p \times 1$ vector containing the frequencies from `omega.dat`, as described previously. Such a variable can be generated in MATLAB from the `omega.dat` file by the command:

```
>> load omega.dat
```

The `omega.mat` file can then be generated by the MATLAB command:

```
>> save omega omega
```

Use s to denote the number of grid points at which mode shape data is to be given in file `phi.dat` and label these grid points with grid numbers $N_1, N_2, \ldots, N_s$. Then `phi.dat` has $(p \times s)$ lines in the format shown in figure 1, where X_j^i, Y_j^i, and Z_j^i are the modal translations in X, Y, and Z, and θ_j^i, ϕ_j^i, and ψ_j^i are rotations about X,

$$
\begin{array}{ccccccc}
N_1 & X_1^1 & Y_1^1 & Z_1^1 & \theta_1^1 & \phi_1^1 & \psi_1^1 \\
N_2 & X_2^1 & Y_2^1 & Z_2^1 & \theta_2^1 & \phi_2^1 & \psi_2^1 \\
 & & & \vdots & & & \\
N_s & X_s^1 & Y_s^1 & Z_s^1 & \theta_s^1 & \phi_s^1 & \psi_s^1 \\
\hdashline
N_1 & X_1^2 & Y_1^2 & Z_1^2 & \theta_1^2 & \phi_1^2 & \psi_1^2 \\
N_2 & X_2^2 & Y_2^2 & Z_2^2 & \theta_2^2 & \phi_2^2 & \psi_2^2 \\
 & & & \vdots & & & \\
N_s & X_s^2 & Y_s^2 & Z_s^2 & \theta_s^2 & \phi_s^2 & \psi_s^2 \\
\hdashline
 & & & \vdots & & & \\
 & & & \vdots & & & \\
 & & & \vdots & & & \\
 & & & \vdots & & & \\
\hdashline
N_1 & X_1^p & Y_1^p & Z_1^p & \theta_1^p & \phi_1^p & \psi_1^p \\
N_2 & X_2^p & Y_2^p & Z_2^p & \theta_2^p & \phi_2^p & \psi_2^p \\
 & & & \vdots & & & \\
N_s & X_s^p & Y_s^p & Z_s^p & \theta_s^p & \phi_s^p & \psi_s^p
\end{array}
$$

Figure 1. Format of `phi.dat`.

Y, and Z for mode i at grid point N_j. Note that grid points that are not needed to model instruments, the control system, or disturbances do not need to be included in file `phi.dat`. However, if they are included, care must be taken to ensure that each mode block contains data for exactly the same grid points.

The file `phi.mat` should contain a variable `phi`, which is a $(p \times s) \times 7$ matrix containing the grid numbers and mode shape data from `phi.dat`, as described previously. This variable can be generated in MATLAB from the file `phi.mat` by the command

```
>> load phi.dat
```

The file `phi.mat` can then be generated by the MATLAB command

```
>> save phi phi
```

Modal Damping Schedule

PLATSIM determines the damping ratios of the flexible spacecraft modes by a calling the user-supplied MATLAB function `mkdamp` in file `mkdamp.m`. The first line of `mkdamp.m` has the following form:

```
function [d]=mkdamp(omega)
```

The input `omega` is a vector containing the modal frequencies $\omega_1, \omega_2, \ldots, \omega_p$. The user must set the output `d` to a vector containing the corresponding damping ratios. Appendix B contains a sample file `mkdamp.m` for the EOS-AM-1 mission. This file defines a damping ratio of 0.2 percent for structural modes with frequencies under 15 Hz, a damping ratio of 0.25 percent for structural modes with frequencies between 15 Hz and 50 Hz, and damping ratio of 0.3 percent for structural modes

with frequencies over 50 Hz. The user running PLATSIM in its graphical user interface (GUI) mode can use Modify Damping under the Options menu to change the damping schedule. (See chapter 6.)

Effector and Sensor Data

The term *effectors* in a system modeled by PLATSIM refers to the control system actuators and the disturbances denoted by u and w, respectively, in equation (5). The term *sensors* refers to the measurement and performance outputs, denoted by y and y^{pr}, respectively. A commonalty exists in the structure of the data needed by PLATSIM to calculate the input influence matrices B_s and B_d and the output influence and feedforward matrices C_p, C_y, C_p^{pr}, C_y^{pr}, D_u^{pr}, and D_w^{pr}, which operate on the states and effectors to calculate their influence on the sensors.

Common Data Structure

The simplest effector or sensor to model is located at a grid point of the structure and either produces or measures linear motion parallel to an axis or rotational motion about an axis. This information is transmitted by passing PLATSIM the grid number (which must appear in file `phi.mat` or `phi.dat`) and a direction number whose value is an integer between 1 and 6. Direction number 1 represents a force that is parallel to the *X*-axis for an effector, or it represents a linear displacement that is parallel to the *X*-axis for a sensor. Direction numbers 2 and 3 serve the same purpose for the *Y*- and *Z*-axes, respectively. Direction number 4 represents torque about the *X*-axis for an effector or angular displacement about the *X*-axis for a sensor. Direction numbers 5 and 6 serve the same purpose for the *Y*- and *Z*-axes, respectively. A sensor can measure linear or angular displacements or it can measure the rate or acceleration of this displacement. Thus, its type is specified to PLATSIM in an additional type number that has the value 0 for displacement, the value 1 for rate, or the value 2 for acceleration. In addition, each effector and sensor is assigned an identification number, which should be a positive integer. This number also tells PLATSIM the order of effectors and sensors in vectors. PLATSIM arranges the effectors and sensors so that their identification numbers are in increasing order. Thus, the user must define identification numbers such that the order of entries in the control input vector u and the measurement outputs vector y connects properly with the control system model. The identification number is also used to form variable names for MATLAB variables in some output files and to form file names on some computer systems. A weighting factor is also included in the common data structure. One use of this weighting factor is for conversion of units. For example, if a performance output is a pointing error that is being calculated in radians and the user wants the results to be in arc-seconds, then a weighting factor of $3600 \times 180/\pi$ would accomplish this conversion.

The weighting factor can also be used to model effectors and sensors that might be located off any grid point or oriented in a nonaxial direction. Before explaining this use of the weighting factor, this document explains how the data are passed to PLATSIM.

The grid number, direction number, identification number, and weighting factor (and type number for sensors) are stacked in a column vector. The necessary number of column vectors are assembled into a matrix, which is then passed to PLATSIM. A single effector or sensor can be modeled by information in more than one of these columns. The user simply uses the same identification number in all columns that refer to a single effector or sensor. Each single effector or sensor described by the grid number and the direction number of each column (and the type number for sensors) is multiplied by its weighting factor, and all these weighted columns with the shared identification number are combined into a single effector or sensor in the PLATSIM model of the plant.

For example, suppose that an *X*-axis thruster is located halfway between grid points 1211 and 1212. This location is identified by including two columns in the `act` matrix (see section entitled “Instrument Data”) as follows:

$$\begin{bmatrix} 1211 & 1212 \\ 1 & 1 \\ 550 & 550 \\ 0.5 & 0.5 \end{bmatrix}$$

The duplicate use of identification number 550 indicates that one actuator is being described by these two columns. The weighting factor 0.5 and the direction number 1 mean that half the force of actuator 550 will be applied in the *X*-direction at each of grid points 1211 and 1212. This arrangement gives the effect of placing the actuator half way between these grid points.

For another example, suppose that the user wants to assess the linear acceleration performance at grid point 277 in a direction lying in the first quadrant of the local *Y*-*Z* plane 30° from the positive *Y*-axis. This arrangement is identified to PLATSIM by including two columns in

the `pout` matrix (see section entitled "Instrument Data") as follows:

$$\begin{bmatrix} 277 & 277 \\ 2 & 3 \\ 4 & 4 \\ 0.866 & 0.500 \\ 2 & 2 \end{bmatrix}$$

The two columns are again linked by using the same identification number in row 3. Row 1 shows that both columns deal with grid point 277. Row 2 shows that translational information in the *Y*- and *Z*-directions is being combined. The weighting factors in row 4 are the direction cosines for the desired direction. Row 5 is added to the sensor data and, in this example, indicates that acceleration information is desired.

These two applications of the weighting factor can be combined by multiplying the weighting factors from each individual use. For example, suppose grid points 515 and 516 are 10 in. apart and a roll attitude sensor is modeled, which has the identification number 300 and is located between these grid points 3 in. from grid point 515. Furthermore, the output is to be converted from radians to arc-seconds. This conversion is accomplished by including two columns in the `pout` matrix as follows:

$$\begin{bmatrix} 515 & 516 \\ 4 & 4 \\ 300 & 300 \\ 1.4439 \times 10^5 & 0.6188 \times 10^5 \\ 0 & 0 \end{bmatrix}$$

The fourth row of weighting factors is found by multiplying the vector (0.7, 0.3), which is used to weight the roll attitude from grid points 515 and 516 by the unit conversion factor $3600 \times 180/\pi$.

Instrument Data

This section explains how to input data into PLATSIM to define the instruments on the space platform, to provide names for the performance outputs, and to specify the jitter-stability windows to be used in the analysis. These instruments are the actuators, measurement output sensors, and performance outputs, corresponding to variables u, y, and y^{pr}, respectively, of equation (5). The data should be provided through a user-supplied MATLAB function `instdata` in a file named `instdata.m`. The first line of `instdata.m` must have the following form:

```
function [act,mout,pout,instr,...
          window]=instdata
```

The following is a description of the user-defined output parameters in file `instdata`. (See appendix B for a sample file `instdata.m` for the EOS-AM-1 spacecraft.)

1. Parameter `act` is a matrix with four rows and as many columns as necessary to represent the actuators (vector u of eq. (5)) in the format given in the section entitled "Common Data Structure."

2. Parameter `mout` is a matrix with five rows and as many columns as necessary to represent the measurement outputs (vector u of eq. (5)) in the format given in the section entitled "Common Data Structure." PLATSIM is only programmed to handle displacement and rate sensors as input to the control system; thus, entries in row five, the type numbers, are limited to 0's and 1's. However, acceleration feedback to the control system can be implemented by the addition of a judiciously selected prefiltering of a velocity signal (e.g., a differentiator followed by a roll-off filter of relative degree of at least two to eliminate feedforward and fit into the format of eq. (6)).

3. Parameter `pout` is a matrix with five rows and as many columns as necessary to represent the performance outputs (vector y^{pr} of eq. (5)) in the format given in the section entitled "Common Data Structure." When PLATSIM is run, the user is given an opportunity to select a subset of the performance measurements presented in the matrix `pout`. Thus, when writing the user-supplied MATLAB function `instdata`, the user can include any performance sensor that may be desired in any analysis. This inclusion should reduce the need to edit this file when changing analytical focus. **WARNING:** PLATSIM does not permit the user to mix the acceleration type with displacement and rate types in a single (multicolumn) performance output instrument.

4. Parameter `instr` is a matrix of character strings that provides names for the performance outputs. The number of rows in `instr` must equal the number of performance outputs, that is, the number of distinct entries in the third row of `pout`. Each element of `instr` is a string consisting of three or four fields separated from each other by the vertical bar (|) character. The first field is an integer that must correspond to one of the performance output identification numbers defined in the third row of `pout`. Furthermore, every identification number in that row must be referenced in `instr`. The second

field contains one or more alphanumeric words that provide a unique name for the corresponding performance output. These names are used in PLATSIM on menu labels and in output tables and graphs. These names are also used as parts of file names after imbedded blanks have been replaced by underscores. For this naming convention, the user should be aware of the following:

- No special characters (except blanks which are replaced by underscores) should be used which would confuse the operating system when used in a file name.
- When determining whether character strings are distinct, embedded blanks should be considered to be the same as underscores (e.g., SWIR roll and SWIR_roll should be considered to be the same).
- When determining whether character strings are distinct, the case sensitivity of the operating system should be considered (e.g., swir_roll and SWIR_roll can be considered distinct under UNIX, but should be considered the same under DOS).

The third field is used to name the menu option under which this instrument is found in menu mode. By using the same name in the third field of several entries, the names are grouped in a submenu under the same menu heading. For example, if three instruments are named VNIR roll, VNIR pitch, and VNIR yaw in their respective second fields of their `instr` entries, then by putting VNIR in the third field of all three, all three are found as submenu items under the menu option VNIR. If the optional fourth field is present, PLATSIM adds it to the second field in labeling the *Y*-axis of time history plots. This field can be used, for example, to give the units of the output.

5. Parameter `window` is a row vector defining the window sizes (in the same time units, usually seconds, used for parameter `period` in function `distdata`, which is described in the next section) for the jitter analysis.

Disturbance Data

The PLATSIM package was developed with the assumption that spacecraft simulations and analyses would be performed for a variety of different disturbance scenarios, and users would maintain their own disturbance database. Easy interactive access to the disturbance database is provided in PLATSIM through a graphical user interface (GUI). The operation and capabilities of the disturbance GUI are described in the “Disturbance” section in chapter 6.

The PLATSIM disturbance data are communicated to the main package through a user-supplied MATLAB function `distdata` in file `distdata.m`. This user-supplied data file is used to provide a complete description of all spacecraft disturbance scenarios, which can consist of several disturbance events. A disturbance event is a force or torque time history acting on the platform. This event is modeled in PLATSIM by a pure force or torque effector (see section entitled “Common Data Structure”) and a discretized force or torque time history stored as a column vector, which is to be applied to the platform through this effector.

The actual structure of the user disturbance data is described in the remainder of this section. The first line(s) of file `distdata.m` must have the following form:

```
function [dist,w,period,cnames,...
         dnames instdat,mapping]...
         =distdata(casenum)
```

The data the user returns in the first three parameters (`dist`, `w`, and `period`) depends on `casenum`, and the data returned in the last four parameters (`cnames`, `dnames`, `instdat`, and `mapping`) must be the same independent of the value of `casenum`. The data returned in `dist`, `w`, and `period` model the one or more disturbance events of the disturbance scenario identified by `casenum`.

The input for `distdata.m` is as follows:

- `casenum`: If PLATSIM calls `distdata` with `casenum = 0`, then the user need only return the last four parameters (`cnames`, `dnames`, `instdat`, and `mapping`). If `casenum` is input as a positive integer between the values of 1 and the number of rows in character matrix `dnames`, then all seven parameters should be returned with the values in the first three corresponding to the disturbance scenario whose name is in row `casenum` of `dnames`. All other values of `casenum` are invalid.

The output for `distdata.m` is as follows:

- `dist`: A matrix of four rows and as many columns as necessary to describe one disturbance effector (see section entitled “Common Data Structure”) for each column of the matrix `w`.
- `w`: A matrix of force and/or torque disturbance time history profiles. The number of rows in `w` is the number of time steps, and the number of columns in `w` is the number of disturbance events. During simulation, the time history in the first column of `w` is

applied to the disturbance effector in `dist` with the smallest identification number, the second column of `w` to the disturbance with the second smallest identification number, and so on.

- `period`: A scalar time step value used to define the disturbance profile and also used as the sample period for time discretization of the space platform control system and plant matrices. The same discretization period is used for all the events in a single disturbance scenario. The parameter `period` should be returned in the same time units, usually seconds, as the parameter `window` returned by the function `instdata` described previously.

- `cnames`: A string matrix that contains the menu heading labels for the disturbance GUI. The disturbance GUI creates a pull-down menu option for each row in `cnames`. The entries in `cnames` are used for menu labeling only and do not appear in output documentation.

- `dnames`: A string matrix of disturbance scenario case names. The rows in `dnames` are used for labeling pull-down menu items in the disturbance GUI, for labeling plot figures, for constructing file names for the PostScript and ASCII jitter table output files, and for constructing file names for the frequency domain analysis output files containing transfer function and Bode plot data. When a disturbance scenario is selected, PLATSIM calls `distdata` with `casenum` set equal to a number that correctly corresponds to a row in the matrix `dnames`. The file `distdata.m` must return the disturbance case that corresponds to the disturbance scenario defined by `casenum`. Individual disturbance scenarios can be "commented out" by editing `distdata.m` to return the corresponding row of `dnames` with an asterisk (*) as the first character. The scenario still appears on the disturbance GUI menu, but in a distinctive gray type, and it is not selectable. However, PLATSIM does not guard against the selection of a "commented out" disturbance scenario in batch mode or in command mode when the text version of the disturbance selection menu is being used. (This alternate menu is used when the UNIX environment variable `DISPLAY` is determined by the MATLAB function `getenv` to be undefined.)

- `instdat`: The variables `instdat` and `mapping` together define which `dnames` row entries appear under a GUI pull-down menu for a particular `cnames` disturbance. The vector `instdat` can be any length and its entries can be in any numerical order. The only limitation on `instdat` entries is that they must be valid `dnames` matrix row numbers. Remember that the entries in `instdat` correspond to row numbers in the `dnames` string matrix.

- `mapping`: The vector `mapping`, which must contain nonnegative integers, defines the partitioning of the `instdat` vector required to map the entries in `dnames` to the disturbance GUI labels created from `cnames`. The number of entries in the `mapping` variable must equal the number of rows in `cnames`. Also, the sum of all entries in `mapping` must equal the number of elements in `instdat`. For example, if `instdat` has 30 elements then, the MATLAB command `sum(mapping)` must also equal 30.

As an example to illustrate the use of the parameters `cnames`, `dnames`, `instdat`, and `mapping`, suppose the file `distdata.m` contains the following commands:

```
cnames = str2mat('Calibration','Tracking');

dnames = str2mat('Solar Array',...
'Telescope calib','Antenna calib');

dnames = str2mat(dnames,...
'Telescope track', 'Antenna track',...
'Antenna sweep');

instdat=[1 2 3 1 4 5 6];

mapping=[3 4];
```

This disturbance database contains six disturbances whose names are given in the six lines of the `dnames` matrix. When PLATSIM is run in GUI or command line mode, two options on the disturbance menu are labeled Calibration and Tracking. The Calibration submenu has three items (`mapping(1) = 3`), which are from the first three rows of the `dnames` matrix (the first three entries of `instdat` are `[1 2 3]`). The Tracking submenu has four items (`mapping(2) = 4`), which are taken from rows 1, 4, 5, and 6 of `dnames` (the next four entries of `instdat` are `[1 4 5 6]`). Note that the same disturbance scenario (solar array in this example) can be included in more than one category. The MATLAB function `str2mat` can be used to build these character arrays.

An example of a typical disturbance data structure matrix that is returned with proper execution of a `distdata.m` disturbance file is given as follows. The example is a disturbance scenario that consists of two distinct disturbance events. The first disturbance event is a roll torque (coordinate direction 4) that is equally distributed across two FEM grid points. The second disturbance event is a yaw torque (coordinate direction 6) that is applied at a single grid point.

$$\texttt{dist} = \begin{bmatrix} 329526 & 329527 & 329526 \\ 4 & 4 & 6 \\ 100 & 100 & 150 \\ 0.5 & 0.5 & 1 \end{bmatrix}$$

Note that the number of distinct elements in the third row of `dist`, corresponding to the disturbance numbers, and the column dimension of the disturbance `w` must be the same. In the previous example, `w` must have two equal length columns. Furthermore, the mapping between the disturbance numbers and the columns of `w` is based upon the sorted order (from lowest to highest) of the disturbance numbers. That is, the first disturbance `w`(:,1) is applied at the finite element grid point corresponding to the smallest disturbance number. In the previous example, `w`(:,1) corresponds to disturbance 100, and `w`(:,2) corresponds to disturbance 150. The PLATSIM package is not capable of detecting whether or not the user complies with this sorted-order mapping convention.

The input variable `casenum` to the user-provided file `distdata.m` is used to select a particular disturbance scenario from the user-provided database. The value of `casenum` should correspond to a valid row number in the `dnames` string matrix. For all nonbatch mode operations, the value of `casenum` is set by the disturbance module's GUI. In batch mode, the value of `casenum` is set in the batch mode input file. The value `casenum=0` is reserved for internal use.

Space Platform Control System

The space platform control system is provided by a call to MATLAB function `formscs` in a user-supplied file `formscs.m`. This routine assumes that the spacecraft control system of the platform is represented by linear time-invariant dynamics. Note that routine `formscs` can be used to model the attitude control system as well as any additional control system that augments the attitude control system for enhanced performance, for example, flexible body control, local payload isolation, or disturbance rejection. The first line of file `formscs.m` must have the following form:

```
function [ascs,bscs,cscs]=formscs
```

where the output parameters `ascs`, `bscs`, and `cscs` denote the state matrix, the input influence matrix, and the output influence matrix of the space platform control system, respectively. For time domain analysis, these matrices are subsequently discretized by the MATLAB routine `c2d` by using a sampling period equal to the integration step size, which PLATSIM gets from parameter period of user-supplied file `distdata.m`.

Currently, the contents of file `formscs.m` in appendix B correspond to the attitude control system of the EOS-AM-1 spacecraft, with the exception that reaction wheel dynamics or output quantization are not included. The user must replace or modify the contents of `formscs.m` to model control systems for platforms other than EOS-AM-1. However, the user has to ensure that the calling sequence in the revised `formscs.m` remains the same; that is, the order and number of the output variables remain unchanged. The following assumptions are made in `formscs`:

1. PLATSIM assumes that the space platform control system is implemented with a negative feedback connection. (See eq. (6).) The number of space platform control system inputs are assumed to equal the number of measurement outputs of the plant. Moreover, a straight connection exists between the measurement outputs of the plant and the inputs of the space platform control system.

2. The number of space platform control system outputs are assumed to equal the number of control inputs to the plant. Moreover, a straight connection exists between the outputs of the space platform control system and the inputs of the plant.

Chapter 4

PLATSIM Output

PLATSIM returns its results to its users in two general manners—interactive and permanent. Interactive output includes MATLAB workspace variables that are set to the results of PLATSIM calculations and plots and tables displayed in MATLAB figure windows. Permanent output consists of a variety of files. The most useful of the PLATSIM workspace variables can be captured in MATLAB MAT-files. Plots can be output as encapsulated PostScript files. Tables of jitter values are output as both ASCII and PostScript files. In the following sections, general file naming conventions are given first and the modifications necessary for PC file-naming conventions are given at the end of each section.

Time Domain Analysis

Full Time Histories

When PLATSIM is performing time domain analysis, it first calculates the time histories of the selected performance outputs in response to the selected disturbance scenario. The user has two choices. Either a separate simulation is performed for each individual disturbance event in the scenario (this is the default), or a simulation is performed showing the response when all events of the disturbance scenario are applied simultaneously. These simulations have the potential to produce a large amount of data. The results of each simulation are contained in a variable *y*, which contains one column for each performance measurement output and one row for each row in the user-supplied disturbance time histories used to drive the simulation.

When the PLATSIM script has finished the time domain analysis, if multiple simulations have been performed, only the most recent one (corresponding to the disturbance event with the highest identification number) remains in the variable *y*. The others have been overwritten to prevent excessive use of computer memory. If the user needs these full time histories, the user can elect to have them written to MAT-files. File `y1.mat` contains the time history for the first disturbance event (which is the default case) or it contains the time history for all events simultaneously (if that choice is made). If more than one disturbance event is used and they are being run individually, the additional time histories are in files `y2.mat`, `y3.mat`,..., up to as many as are needed. Each file contains the time history in variable *y* as previously stated, a scalar variable period that contains the time increment between discrete points of the simulation, and a character array `instr` whose rows contain the names of the performance outputs that are being simulated. Once again, the user is cautioned to be careful about requesting these files because depending on the numbers of performance outputs, time steps of simulation, and disturbance events, they can be large and require significant disk space.

Time History Plots

The user can elect to have the time histories plotted on screen and also elect to write EPS files of the on-screen plots (this is the default). If the latter is chosen, then a reduced form of the data is plotted. Although some simulations have involved a hundred thousand or even a million time steps, the reduced data typically involves vectors of length less than ten thousand. The reduction is performed in such a manner (see chapter 15) that the visual effect of plotting the reduced data is virtually identical to that of plotting the full time history. Thus, the results are faster on-screen plotting, faster writing of EPS files, smaller PostScript files, and faster printing of these files by a PostScript printer. (The time required for a printer to process the PostScript file has been seen to decrease from more than 30 min when printing a plot made from a full time history of 100000 points to less than 2 min to print a plot made from the reduced data.)

If each disturbance event is simulated individually, then one time history is generated for each combination of performance measurement output and disturbance event. If the disturbance events are run simultaneously, then one time history exists per performance measurement output. The reduced data from a typical time history are contained in a MATLAB variable with a name such as `p342_1`. The number 342 is a performance output identification number, taken from the third row of the user-supplied matrix `pout`. (See chapter 3.) The number 1 indicates that this is the response to the first event in the chosen disturbance scenario when the disturbance events are run separately, or it indicates that this is the response when the disturbance events are run simultaneously. If more than one event occurs in the disturbance scenario and if they are run separately, then the reduced

response to the second event has a number 2 after the underscore, and so on. The variable p342_1 (or like variables) is a two-column matrix, with the number of rows being somewhere between 2 and 4 times the value of execution time parameter nplpts. (See chapter 15 and appendix C.) The plots are made by plotting the second column of this variable as a function of the first column.

All these variables are saved in a MATLAB MAT-file with a name such as MODIS_static_imbalance_1_time.mat. The MODIS_static_imbalance part of the name comes from replacing any blanks with underscore characters in the name for the disturbance used in this simulation. This name was given in the array dnames returned by user-supplied MATLAB function distdata. The number 1 is used if this data comes from the first disturbance event in the disturbance scenario when events are being run separately and is also used if the events are run simultaneously. If the events are being run separately, the second event would have a number 2. The time in the name distinguishes this file from a similarly named file from the frequency analysis part of PLATSIM.

If the user chooses the option Plot with Hardcopy, PLATSIM writes EPS files of the plots, which the user can send to the printer. These files have names such as MODIS_Yaw_1_time.eps. The MODIS_Yaw part of the name comes from replacing any blanks with underscore characters in the name for the performance output used in this simulation that was given in the second field of the appropriate entry of the array instr returned by user-supplied MATLAB function instdata. The number 1 and time follow the same conventions described in the previous paragraph.

Jitter

If the calculation of jitter is elected (this is the default), then the results of the jitter calculation are available to the user in several forms. If disturbance events are run separately, then the results of the jitter calculation for the first event are contained in the MATLAB variable JITTER1, those for the second event (if any) are in JITTER2, etc. Each of these variables is a matrix with one column for each jitter window and one row for each performance output. (The order of the rows is determined by the numerical order of the identification numbers of the performance outputs.) In addition, the total jitter variable, JITTER, is just the sum of the individual jitter calculations. If the disturbance events are run simultaneously, the results are contained in the variables JITTER1 and JITTER, which in this case are identical.

The variable JITTER1 is preserved in the same file as the variables containing the reduced time histories for first event (or combined events). If the user chooses not to plot the time histories but wants the jitter values, the variable JITTER1 is the only thing in this file. Continuing the example started in the section on time history plots, JITTER1 would be in file MODIS_static_imbalance_1_time.mat. If a second disturbance event was being used, JITTER2 would be in file MODIS_static_imbalance_2_time.mat. In addition, a file named MODIS_static_imbalance_time.mat would be written containing only JITTER.

The table of jitter values is displayed on the screen. If the events are run separately, a table is presented for each event and another for the jitter totals. The worst value in each column is displayed in a contrasting color for emphasis. These tables are also written in both PostScript and ASCII files. The PostScript format tries to emulate what is shown on the screen; however, if too many performance measurement outputs exists, the PostScript file prints two or more pages. The ASCII format is more suitable for importing into a document. If disturbance events are run separately, the PostScript files for the individual disturbance event jitter results are saved in files with names such as MODIS_static_imbalance_1_jitr.ps, MODIS_static_imbalance_2_jitr.ps. The table with the overall totals is saved in MODIS_static_imbalance_jitr.ps. For the ASCII files, the names are the same except they have the extension out instead of ps. If the simulation is for a single event in the disturbance scenario or the disturbance events are run simultaneously, then only one jitter result occurs. This result is written to the files with names without event numbers. From the previous example, these names would be MODIS_static_imbalance_jitr.ps and MODIS_static_imbalance_jitr.out.

Example

As an example, suppose that PLATSIM is run with the default values of all execution time parameters and the example data distributed with the PLATSIM software, which is based on the EOS-AM-1 spacecraft. Suppose further that the disturbance scenario selected is MODIS static imbalance. Then the following files are generated by PLATSIM:

```
MODIS_static_imbalance_1_time.mat          MODIS_static_imbalance_2_time.mat
MODIS_static_imbalance_time.mat
CERES1_Pitch_1_time.eps     CERES1_Pitch_2_time.eps     CERES1_Roll_1_time.eps
CERES1_Roll_2_time.eps      CERES1_Yaw_1_time.eps       CERES1_Yaw_2_time.eps
CERES2_Pitch_1_time.eps     CERES2_Pitch_2_time.eps     CERES2_Roll_1_time.eps
CERES2_Roll_2_time.eps      CERES2_Yaw_1_time.eps       CERES2_Yaw_2_time.eps
MISR_Pitch_1_time.eps       MISR_Pitch_2_time.eps       MISR_Roll_1_time.eps
MISR_Roll_2_time.eps        MISR_Yaw_1_time.eps         MISR_Yaw_2_time.eps
MODIS_Pitch_1_time.eps      MODIS_Pitch_2_time.eps      MODIS_Roll_1_time.eps
MODIS_Roll_2_time.eps       MODIS_Yaw_1_time.eps        MODIS_Yaw_2_time.eps
MOPITT_Pitch_1_time.eps     MOPITT_Pitch_2_time.eps     MOPITT_Roll_1_time.eps
MOPITT_Roll_2_time.eps      MOPITT_Yaw_1_time.eps       MOPITT_Yaw_2_time.eps
NAVBASE_Pitch_1_time.eps    NAVBASE_Pitch_2_time.eps    NAVBASE_Roll_1_time.eps
NAVBASE_Roll_2_time.eps     NAVBASE_Yaw_1_time.eps      NAVBASE_Yaw_2_time.eps
SWIR_Pitch_1_time.eps       SWIR_Pitch_2_time.eps       SWIR_Roll_1_time.eps
SWIR_Roll_2_time.eps        SWIR_Yaw_1_time.eps         SWIR_Yaw_2_time.eps
TIR_Pitch_1_time.eps        TIR_Pitch_2_time.eps        TIR_Roll_1_time.eps
TIR_Roll_2_time.eps         TIR_Yaw_1_time.eps          TIR_Yaw_2_time.eps
VNIR_Pitch_1_time.eps       VNIR_Pitch_2_time.eps       VNIR_Roll_1_time.eps
VNIR_Roll_2_time.eps        VNIR_Yaw_1_time.eps         VNIR_Yaw_2_time.eps
MODIS_static_imbalance_1_jitr.out          MODIS_static_imbalance_2_jitr.out
MODIS_static_imbalance_jitr.out
MODIS_static_imbalance_1_jitr.ps           MODIS_static_imbalance_2_jitr.ps
MODIS_static_imbalance_jitr.ps
```

The reduced time histories for the EOS-AM-1 instruments, namely, CERES1, CERES2, MISR, MODIS, MOPITT, NAVBASE, SWIR, TIR, and VNIR, are provided in the corresponding EPS files for roll, pitch, and yaw axes with the extension `.eps`. For example, the file `CERES1_Roll_1_time.eps` contains the roll response of the CERES1 instrument due to the first MODIS static imbalance disturbance element (vector), and `VNIR_Yaw_2_time.eps` contains the yaw response of the VNIR instrument due to the second MODIS static imbalance disturbance element (vector).

A typical time history output in file `MISR_Roll_2_time.eps` is shown in figure 2. This time history was generated with the 703-mode model of the EOS-AM-1 spacecraft being distributed with the PLATSIM software.

The files `MODIS_static_imbalance_1_time.mat` and `MODIS_static_imbalance_2_time.mat` are MATLAB binary files that include the reduced time history data for all EOS-AM-1 instruments and the corresponding jitter values (in variables JITTER1 and JITTER2) for the first and the second MODIS static imbalance disturbance elements (vectors), respectively. File `MODIS_static_imbalance_time.mat` is a MATLAB binary file that has the overall jitter values (in variable JITTER). The files `MODIS_static_imbalance_1_jitr.out` and `MODIS_static_imbalance_2_jitr.out` are ASCII files that contain the jitter values, in a tabular form, for the first and the second MODIS static imbalance disturbance elements (vectors), respectively. The file `MODIS_static_imbalance_jitr.out` is an ASCII file containing the total jitter values in a tabular form. These values are obtained through the direct summation of jitter values in files `MODIS_static_imbalance_1_time.mat` and `MODIS_static_imbalance_2_time.mat`. The following is an example of the jitter data written to file `MODIS_static_imbalance_1_jitr.out`:

Disturbance Source: MODIS static imbalance(1)

Window	1.00	1.80	9.00	55.00	420.00	480.00	1000.00
NAVBASE Roll	3.19	3.82	7.55	12.50	12.50	12.50	12.50
NAVBASE Pitch	0.42	0.55	1.40	2.20	2.20	2.20	2.20
NAVBASE Yaw	3.97	4.96	14.19	26.88	26.88	26.88	26.88
CERES1 Roll	3.19	3.82	7.55	12.50	12.50	12.50	12.50
CERES1 Pitch	0.42	0.55	1.40	2.20	2.20	2.20	2.20
CERES1 Yaw	3.97	4.96	14.19	26.88	26.88	26.88	26.88
CERES2 Roll	3.19	3.82	7.55	12.50	12.50	12.50	12.50
CERES2 Pitch	0.42	0.55	1.40	2.20	2.20	2.20	2.20
CERES2 Yaw	3.97	4.96	14.19	26.88	26.88	26.88	26.88
MISR Roll	3.19	3.82	7.55	12.50	12.50	12.50	12.50
MISR Pitch	0.42	0.55	1.40	2.20	2.20	2.20	2.20
MISR Yaw	3.97	4.96	14.19	26.88	26.88	26.88	26.88
MODIS Roll	3.19	3.82	7.55	12.50	12.50	12.50	12.50
MODIS Pitch	0.42	0.55	1.40	2.20	2.20	2.20	2.20
MODIS Yaw	3.97	4.96	14.19	26.88	26.88	26.88	26.88
MOPITT Roll	3.19	3.82	7.55	12.50	12.50	12.50	12.50
MOPITT Pitch	0.42	0.55	1.40	2.20	2.20	2.20	2.20
MOPITT Yaw	3.97	4.96	14.19	26.88	26.88	26.88	26.88
SWIR Roll	3.19	3.82	7.55	12.50	12.50	12.50	12.50
SWIR Pitch	0.42	0.55	1.40	2.20	2.20	2.20	2.20
SWIR Yaw	3.97	4.96	14.19	26.88	26.88	26.88	26.88
TIR Roll	3.19	3.82	7.55	12.50	12.50	12.50	12.50
TIR Pitch	0.42	0.55	1.40	2.20	2.20	2.20	2.20
TIR Yaw	3.97	4.96	14.19	26.88	26.88	26.88	26.88
VNIR Roll	3.19	3.82	7.55	12.50	12.50	12.50	12.50
VNIR Pitch	0.42	0.55	1.40	2.20	2.20	2.20	2.20
VNIR Yaw	3.97	4.96	14.19	26.88	26.88	26.88	26.88

RUN DATE: 29-Apr-94

The `MODIS_static_imbalance_1_jitr.ps`, `MODIS_static_imbalance_2_jitr.ps`, and `MODIS_static_imbalance_jitr.ps` files are equivalent to their `.out` counterpart files except that they are in a PostScript format.

File-Naming Conventions for PC's

PC's running under DOS (disk operating system) limit which character strings can be file names. Letters are mapped to uppercase. The file name can have at most eight characters, which are optionally followed by a period and an extension of one to three characters. PLATSIM knows it is running on a PC when the first two characters of the character string returned by the MATLAB function `computer` are PC. If so, then the output files are given alternate names that conform to PC DOS restrictions.

The file that was named `MODIS_static_imbalance_1_time.mat` is now named `JITTER1.MAT`, and the file `MODIS_static_imbalance_time.mat` is now named `JITTER.MAT`. In addition, `MODIS_static_imbalance_1_jitr.out` is now `JITR_1.OUT` and `MODIS_static_imbalance_jitr.out` is `JITR.OUT`. The parallel `.ps` files are similarly renamed. The `CERES2_Yaw_1_time.eps` file in the EOS-AM-1 example is now named `P9_2_T.EPS`. The number 9 comes from the identification number used for the instrument named `CERES2 Yaw`. Double-digit identification numbers are also used completely as in `P19_1_T.EPS` for `SWIR_Roll_1_time.eps`, but only the low-order two digits of larger identification numbers are used. Thus, the PC user must ensure that instruments are uniquely identified by the two low-order digits of their identification numbers.

Frequency Domain Analysis

Transfer Function Evaluation

When PLATSIM performs frequency domain analysis, it first calculates an open- or closed-loop transfer function from disturbances to performance outputs, which can have multiple inputs or outputs. PLATSIM generates the column vector `w` of frequencies from information contained in some execution control parameters. For transfer function inputs, the user first specifies a

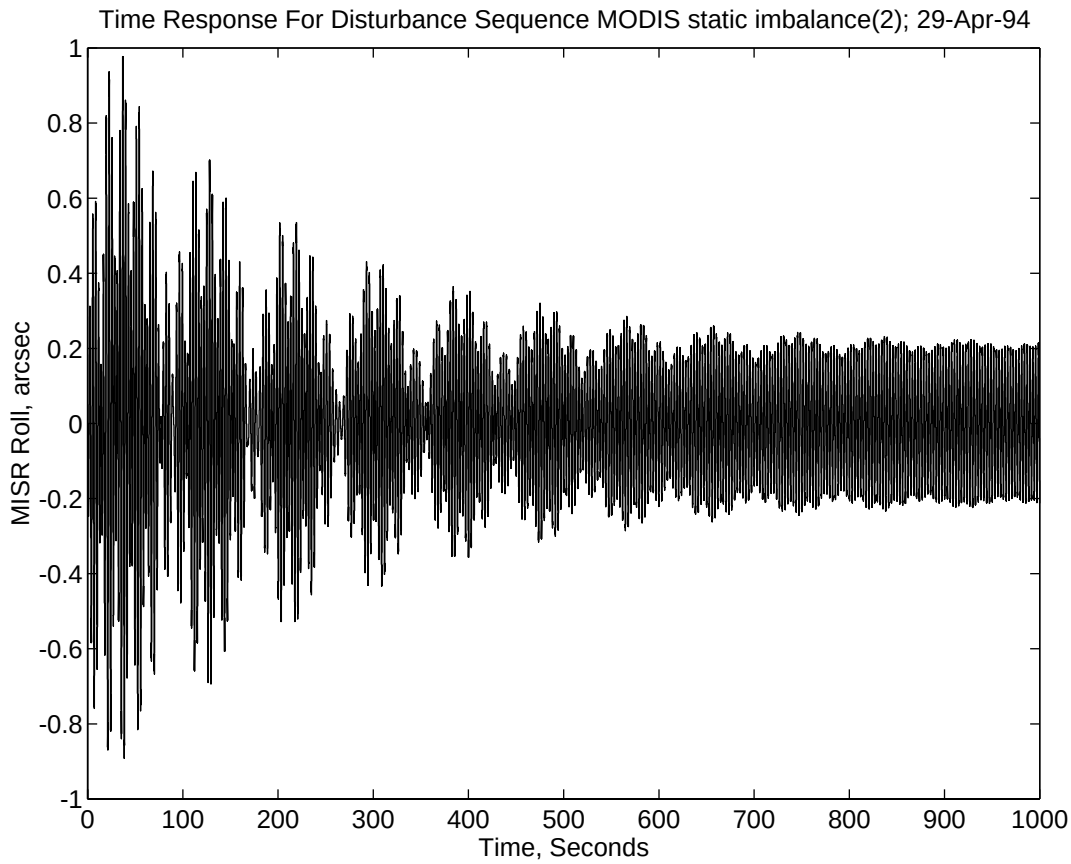

Figure 2. Example of time-history plot.

disturbance scenario and PLATSIM uses all (the default) or a user-specified subset of the events of that scenario as inputs. The outputs are either all (the default) of the performance output instruments or a user-specified subset.

The natural object to contain the results of these calculations is a complex array with three indices (subscripts), one each for the inputs, outputs, and frequency values. MATLAB handles the complex part naturally, but only accepts two indices. To overcome this limitation, some data manipulation is used. Suppose that there are `i` inputs and `j` outputs, and that the transfer function is evaluated at the frequency in `w(k)` and placed in the temporary MATLAB variable `gt`, which is then a complex matrix with `j` rows and `i` columns. PLATSIM then transfers all information from the temporary matrix `gt` into row `k` of the output variable `g` by a statement with the form

```
g(k,:) = reshape(gt,1,j*i);
```

In other words, the columns of `gt` are stacked into a single column vector and the transpose of that vector is placed in the row of `g` corresponding to the row of `w` from which the frequency value came. Then the gain in decibels `m` and the phase in degrees `p` are calculated as follows:

```
m=20*log10(g);

p=(180.0/pi)*angle(g);
```

The variables `w`, `g`, `m`, and `p` are MATLAB workspace variables that are available to the user when PLATSIM completes the frequency domain calculation. They are also written in MATLAB binary form to a MAT-file with a name such as `MODIS_static_imbalance_freq.mat`. The `MODIS_static_imbalance` part of the name comes from replacing any blanks with underscore characters in the name for the disturbance used in this analysis. This name was given in the `dnames` array returned by the user-supplied MATLAB function `distdata`. The variables `ndist` and `instr` are also in this file. The variable `ndist` is a vector of integers that tells which events of the disturbance scenario were used in the calculation as inputs. The variable `instr` is a matrix of characters, each row of which names a performance instrument that was used as an output in the calculation.

Bode Plots

If plotting is requested, the Bode plots for the selected performance outputs in response to the selected disturbances are plotted on screen. If "plot with hard-copy" is selected, in addition to on-screen plots PLATSIM writes EPS files of the plots. These files have names such as `MODIS_Yaw_1_freq.eps`. The `MODIS_Yaw` part of the name comes from replacing any blanks with underscore characters in the name for the performance output used in this analysis. This name was defined in the second field of the appropriate entry of the `instr` array returned by the user-supplied MATLAB function `instdata`. The number `1` means that this was the response to the first event in the chosen disturbance scenario. The `freq` part of the name distinguishes these plots from the time-history plots.

Example

As an example, suppose that PLATSIM is run with the default values of all execution time parameters except that `tdflag = 'no'` (which selects frequency domain analysis) and also uses the example data distributed with the PLATSIM software, which is based on the EOS-AM-1 spacecraft. Suppose further that the disturbance scenario selected is MODIS static imbalance. Then the following files are generated by PLATSIM.

The Bode plots for the EOS-AM-1 instruments, namely, CERES1, CERES2, MISR, MODIS, MOPITT, NAVBASE, SWIR, TIR, and VNIR, are provided in EPS form in the corresponding files for roll, pitch, and yaw axes with the extension `.eps`. For example, the file `CERES1_Roll_1_freq.eps` contains the Bode plot for the roll response of the CERES1 instrument due to the first MODIS static imbalance disturbance element (vector), and `VNIR_Yaw_2_time.eps` contains the Bode plot for the yaw response of the VNIR instrument due to the second MODIS static imbalance disturbance element (vector). Figure 3 shows a sample Bode plot output. This Bode plot was generated with the 703 mode model being distributed with the PLATSIM software. The file `MODIS_static_imbalance_freq.mat` is a MATLAB binary file containing the results of the frequency domain calculation in the form indicated at the beginning of this section.

```
MODIS_static_imbalance_freq.mat
CERES1_Pitch_1_freq.eps    CERES1_Pitch_2_freq.eps     CERES1_Roll_1_freq.eps
CERES1_Roll_2_freq.eps     CERES1_Yaw_1_freq.eps       CERES1_Yaw_2_freq.eps
CERES2_Pitch_1_freq.eps    CERES2_Pitch_2_freq.eps     CERES2_Roll_1_freq.eps
CERES2_Roll_2_freq.eps     CERES2_Yaw_1_freq.eps       CERES2_Yaw_2_freq.eps
MISR_Pitch_1_freq.eps      MISR_Pitch_2_freq.eps       MISR_Roll_1_freq.eps
MISR_Roll_2_freq.eps       MISR_Yaw_1_freq.eps         MISR_Yaw_2_freq.eps
MODIS_Pitch_1_freq.eps     MODIS_Pitch_2_freq.eps      MODIS_Roll_1_freq.eps
MODIS_Roll_2_freq.eps      MODIS_Yaw_1_freq.eps        MODIS_Yaw_2_freq.eps
MOPITT_Pitch_1_freq.eps    MOPITT_Pitch_2_freq.eps     MOPITT_Roll_1_freq.eps
MOPITT_Roll_2_freq.eps     MOPITT_Yaw_1_freq.eps       MOPITT_Yaw_2_freq.eps
NAVBASE_Pitch_1_freq.eps   NAVBASE_Pitch_2_freq.eps    NAVBASE_Roll_1_freq.eps
NAVBASE_Roll_2_freq.eps    NAVBASE_Yaw_1_freq.eps      NAVBASE_Yaw_2_freq.eps
SWIR_Pitch_1_freq.eps      SWIR_Pitch_2_freq.eps       SWIR_Roll_1_freq.eps
SWIR_Roll_2_freq.eps       SWIR_Yaw_1_freq.eps         SWIR_Yaw_2_freq.eps
TIR_Pitch_1_freq.eps       TIR_Pitch_2_freq.eps        TIR_Roll_1_freq.eps
TIR_Roll_2_freq.eps        TIR_Yaw_1_freq.eps          TIR_Yaw_2_freq.eps
VNIR_Pitch_1_freq.eps      VNIR_Pitch_2_freq.eps       VNIR_Roll_1_freq.eps
VNIR_Roll_2_freq.eps       VNIR_Yaw_1_freq.eps         VNIR_Yaw_2_freq.eps
```

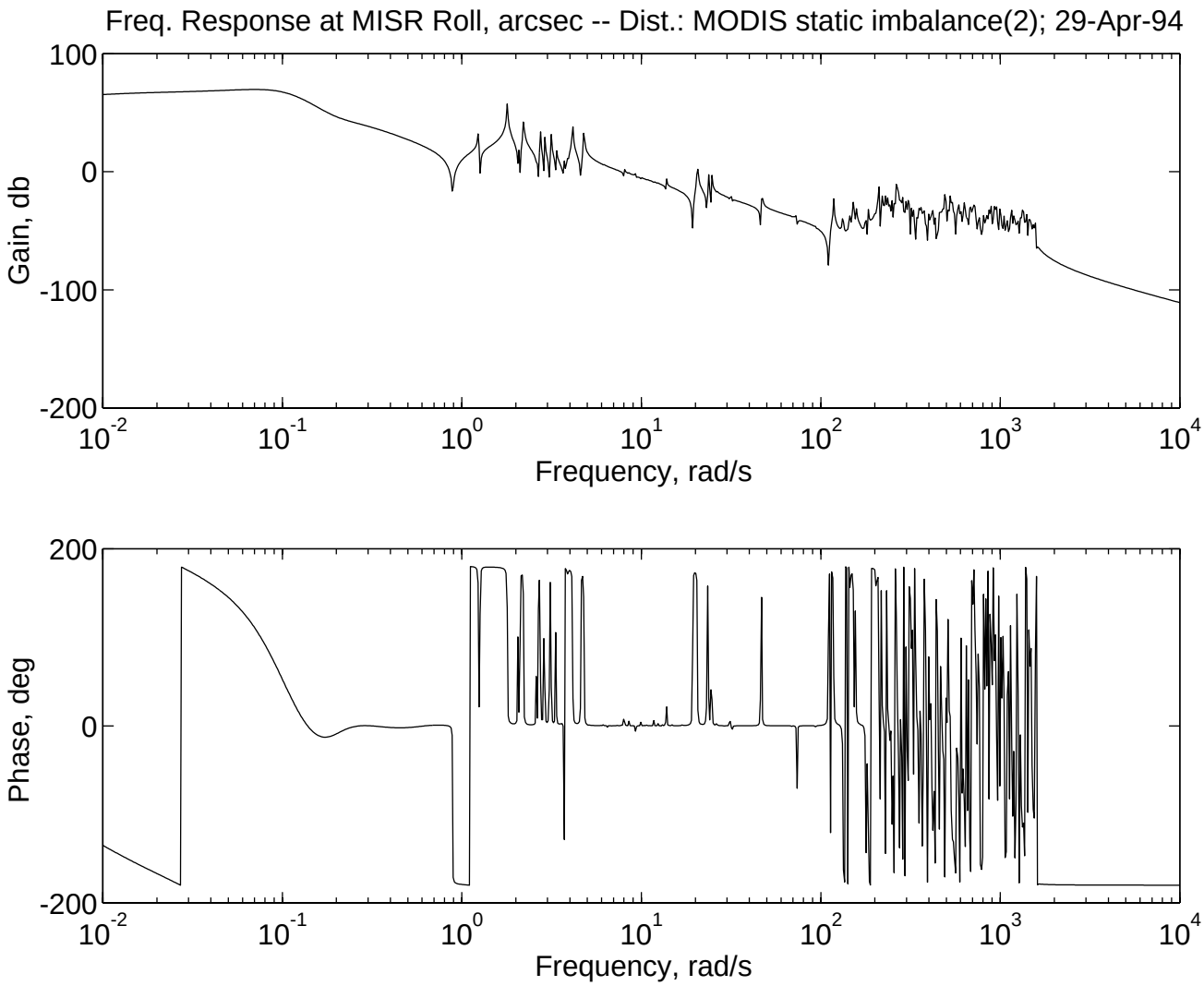

Figure 3. Example of Bode plot.

File-Naming Conventions for PC's

As in the section on time domain analysis, the file-naming conventions just given do not work for PC's. The conventions used on PC's are exemplified here. The file that was named `MODIS_static_imbalance_freq.mat` is `FREQ.MAT`. What was `CERES2_Yaw_1_freq.eps` in the EOS-AM-1 example is now named `P9_2_F.EPS`. The number 9 comes from the identification number used for the instrument named CERES2 Yaw. Double-digit identification numbers are used completely as in `P19_1_F.EPS` for `SWIR_Roll_1_freq.eps`, but only the two low-order digits of larger identification numbers are used. Thus, the PC user must ensure that instruments are uniquely identified by the two low-order digits of their identification numbers. This identification method allows for up to 99 events in a disturbance scenario.

Chapter 5

Program Execution Overview

Overview

As previously stated, PLATSIM operates in the MATLAB technical computing environment at MATLAB version 4.1 or higher. The Control System Toolbox, by MathWorks, Inc., must also be available. To run PLATSIM, the user must first start MATLAB, which needs to know where to find the PLATSIM software and the user-supplied data and M-files. At least one PLATSIM execution control parameter, `runmode`, must be set manually, and depending on the value of `runmode` and the user's desire to override default values, additional execution control parameters may need to be set manually as well. The user then invokes PLATSIM. If `runmode` is not `'batch'`, the user has an opportunity to set more of the execution control parameters. (See chapters 6 and 7 on execution in GUI mode and in command mode.) Finally, the time or frequency domain analysis requested by the user is performed, any requested plots are displayed, and any requested output files are created.

File Management

Of the many ways files can be managed, the authors give one suggestion here. Because most files that constitute the permanent record of PLATSIM's output have names that do not reflect the disturbance scenario being used, each disturbance scenario should be run in its own subdirectory. Otherwise, the second disturbance scenario run uses many of the same file names as the first and thus write over output from the first disturbance scenario. Furthermore, if time domain simulations are to be run by using events in a disturbance scenario both individually and simultaneously, these runs should be made in separate directories to avoid file naming conflicts. The files from the PLATSIM distribution should be kept in a subdirectory of their own, and another directory should be maintained for the user-supplied files specific to the application being considered.

Suppose that PLATSIM is in directory `/usr/local/platdir`, the user-supplied platform model files are in `/astro1/usr2/jdoe/observer`, and the specific disturbance analysis is being run in `/astro1/usr2/jdoe/observer/calib`. Also suppose that a file named `startup.m` (ref. 2) is in this last directory and the file included the following lines:

```
path('/usr/local/platdir',path);

path('/astro1/usr2/jdoe/observer',path);
```

Then each time MATLAB is started in directory `/astro1/usr2/jdoe/observer/calib`, MATLAB executes file `startup.m` and then appropriately sets the MATLABPATH variable. When the PLATSIM `makefile` is executed with the UNIX `make` utility, a template `startup.m` file is created. The user can use this file to access the EOS-AM-1 example data included with PLATSIM or modify the second path statement to point to the user's own data.

Execution Control Parameters

The execution of PLATSIM is controlled by some 20 execution control parameters. One of these, `runmode`, must be set by the user before invoking PLATSIM. The acceptable values for `runmode` are `'GUI'`, `'command'`, and `'batch'`. (Only the first character is significant, and PLATSIM is insensitive to the case of characters in runmode.) Depending on which mode PLATSIM is run in and what default values the user wants to change, the user may need to set some other execution control parameters by MATLAB assignment statements before PLATSIM is invoked. The GUI mode (`runmode='GUI'`) and the command mode give the user a more user-friendly access to some of these parameters. Appendix C contains a complete list of execution control parameter names, functions, and default values.

On-Line Help

A small amount of on-line help is available. The About option on the GUI interface window brings up an overview of PLATSIM. The MATLAB command `help platdir` (assuming PLATSIM is installed in directory `platdir` as suggested in the appendix) provides a list of one-line descriptions of the various M-files that constitute PLATSIM.

Chapter 6

GUI Execution Mode

A graphical user interface has been developed for PLATSIM, which uses MATLAB's Handle Graphics available in MATLAB version 4. The objective of this GUI is to provide the user with a convenient and intuitive way to set PLATSIM execution control parameters. Figure 4 shows a screen image of the PLATSIM GUI. The basic interface consists of five menu options (MATLAB `uimenu` functions) and two slider controls (MATLAB `uicontrol` functions). In addition to setting execution control parameters, the GUI also provides the user with easy access to some common MATLAB workspace commands. The GUI Mode can be executed by typing `runmode='gui'` followed by `platsim` in the MATLAB command window. Normal execution of the PLATSIM GUI creates a file called `platsim.mat` that contains parameters associated with various GUI menus and sliders. A complete description of all PLATSIM GUI features is given in the following sections. If the user wants to override the default actions controlled by execution control parameters `phold`, `nplpts`, or `gad`, then these parameters must be set by hand in the MATLAB command window before PLATSIM is invoked.

Workspace

The Workspace menu has the following MATLAB function commands: New Figure, Clear & Reset Defaults, Who, Whos, Save, and Close Window. A

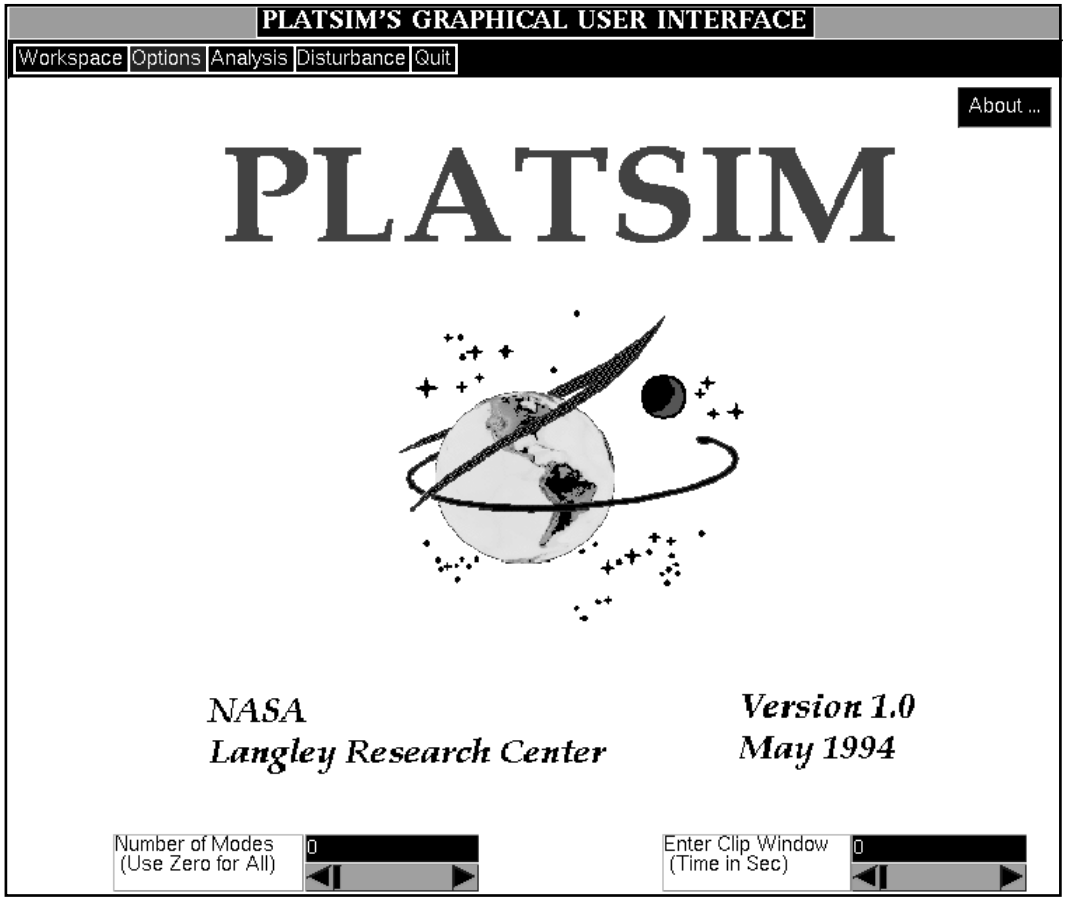

Figure 4. Graphical user interface for PLATSIM jitter analysis package.

complete description of all Workspace menu functions, with the exception of Clear & Reset Defaults, can be found in the *MATLAB Reference Guide* (ref. 2). The Clear & Reset Defaults function requires the existence of a file called `platsim.mat`, which is automatically created in the current working directory. Selecting the Clear & Reset Defaults menu executes various MATLAB commands, including `clear` and `load platsim.mat`. The result of this action effectively clears all variables except those needed by the GUI. In addition to clearing all variables, the values of all buttons and sliders are reset to their default values.

Options

The Options menu provides access to several PLATSIM analysis and documentation output functions. This menu allows the user to select the following PLATSIM features: performance output locations, interactive graphical modification of the structural model (frequencies and damping ratios), the level of simulation output documentation, feedback control, jitter analysis, and whether output time-history data should be saved. The following sections contain detailed descriptions of the Options menu features.

- The Performance Outputs menu provides access to the performance output selection figure window. The performance output window allows users to select a full or partial set of output locations from those defined in the user-defined function `instdata.m`. This utility allows for a large database of output locations to be maintained in function `instdata.m` without the computational expense of solving for all outputs in every analysis. The labeling and grouping of the menu items for the performance output window is determined by the third field in the workspace variable `instr`, which is defined in the user-defined function `instdata.m`. All entries with the same character string in the third field of `instr` are located under the same pull-down menu, and this common character string is used to title the menu group. The performance output selection figure window has three button functions. These buttons are used to select all outputs, deselect all outputs, or close the selection window, and they are labeled Set All, Clear All, and Close. A user can select performance outputs by using the Select All option for each group or the master Set All button for all groups. Any currently selected output can be deselected by simply reselecting the item or selecting the Clear All button. Currently, because of a documented program error in MATLAB 4.1, the Close button has been disabled to eliminate the occurrence of bus errors on some UNIX machines when PLATSIM is running under versions prior to 4.2. When the Close button is disabled, the performance output selection figure window can be closed by issuing a `close(perfig)` command from the MATLAB workspace.

- The Control menu allows the user to select between open-loop and closed-loop analysis of the spacecraft. The space platform control system must be defined in the user-changeable file `formscs.m`. The default setting for the Control menu is closed-loop analysis.

- The Plotting & Printing menu provides the user with the same options as given for the Command-Driven mode, as described in the following chapter. The default settings for this menu are `pltflag='yes';` and `prtflag='yes';`.

- The Perform Jitter Analysis menu allows the user to select whether jitter analysis is performed (this is the default) or not.

- The Save Output Time History menu provides the user with the option to save the complete output time history data. In other words, variables `y` and `instr` are saved if this option is set to yes. However, selecting yes for this option has the potential of creating a large file and therefore should be used with caution. The output data are saved in one file per disturbance event; these files are labeled `y1.mat`, `y2.mat`, `y3.mat`, etc. The default setting for this option is `saveflag='no'`. This variable is optional; hence, the user is not prompted when using PLATSIM in the command-line driven mode. In this mode, issuing the MATLAB command `saveflag='yes'` before issuing `platsim` saves the output time-history data.

- The Modify Plant Model menu has two menu options: Frequencies and Damping Ratios. These menu options allow the user to graphically modify or define both frequency uncertainties and modal damping ratios. Details for using these options follow.

Frequency Ratio Modification

The Frequencies option of the Modify Plant Model menu provides the user access to a graphical interface tool, which is shown in figure 5. This interface tool has a variety of options for adding modal uncertainties to the plant structural model. Two basic types of uncertainties are considered: a constant scaling and a random scaling. The constant scaling uncertainties are defined as follows:

$$\omega_i^* = \omega_i^o + \omega_i^o \times S$$

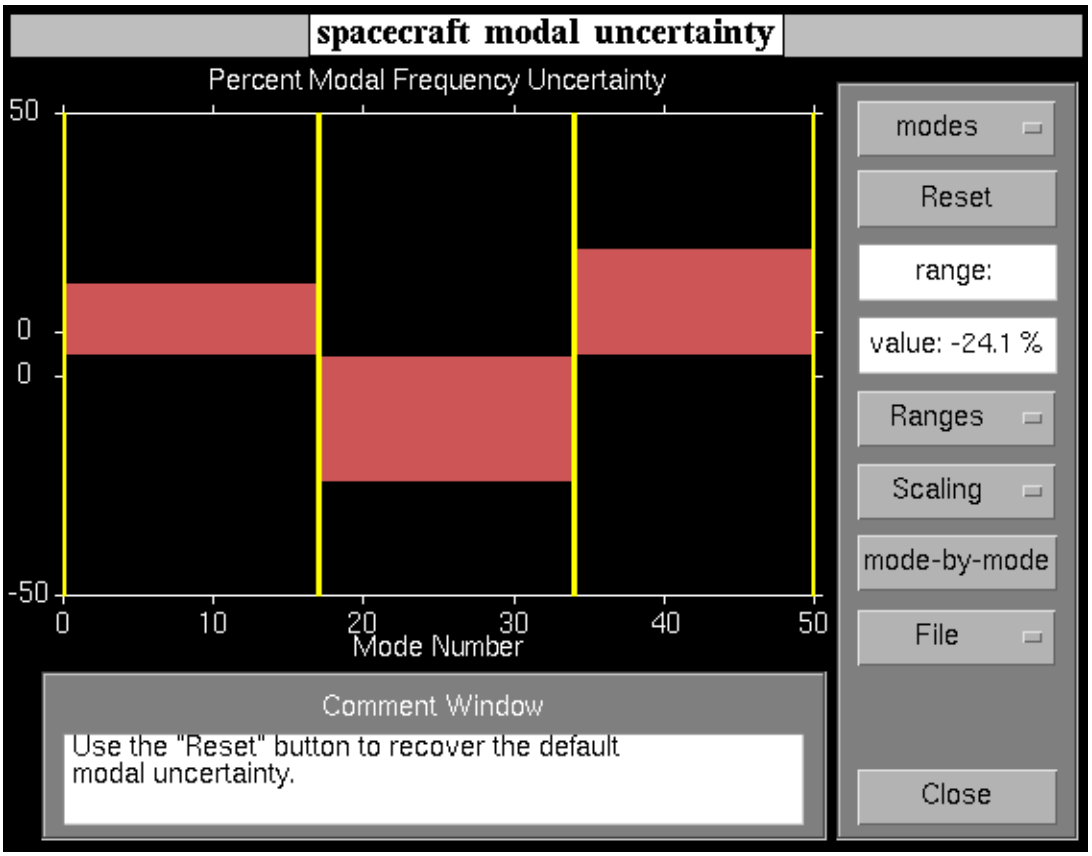

Figure 5. Spacecraft modal uncertainty graphical user interface.

The random scaling uncertainties are defined as follows:

$$\omega_i^* = \omega_i^o + \omega_i^o \times R_i \times S$$

The variables S and R_i in the previous equations represent a user-defined constant scale factor and a set of random numbers uniformly distributed over the interval (0, 1) (by the MATLAB random number generator `rand`). The frequency spectrum, as defined in files `omega.dat` or `omega.mat`, can be divided into intervals (ranges) to allow for different levels of uncertainties for different ranges of modes within the frequency spectrum. The user can change the number of ranges by selecting the interface button labeled Ranges. A range, as defined by the interface, is denoted by the interval (a, b], where a and b are the lower and upper bounds on the interval, respectively. For example, if a = 10 and b = 20, then all modes from 11 through 20 (including mode 20) have the same S factor. The lower and upper bounds (range) for the interval can be changed by using a mouse to click and drag on the vertical lines that represent interval ranges. The scale factors (value) can also be changed by using a mouse to click and drag on the shaded regions between the vertical lines. Both range and value can be changed with keyboard entries once the corresponding graphical element has been activated. A range or value element is activated by using the mouse to click on a vertical line or shaded region, respectively. Once a graphical element has been activated, its numeric value is displayed in the range or value text area of the graphical interface.

A user can choose between constant or random scaling by using the Scaling button (constant is the default setting). The frequency spectrum axis can be changed from Mode Number (default setting) to Frequency (Hz) by using the top button on the interface. Other features include Reset, File, and Mode-by-Mode. The Reset button resets the frequency uncertainty scale factor to 0 for all modes. The File button allows the user to either save the current frequency uncertainties to a `.mat` file or load a preexisting `.mat` file containing frequency uncertainties. The Mode-by-Mode button provides access to a graphical interface that allows convenient mode-at-a-time frequency modifications.

Figure 6 shows a screen image of the Mode-by-Mode graphical interface. This interface gives the user a list of frequencies of all modes displayed in a read-only text window on the right side of the interface. The user can use the Change Selection buttons to scroll through

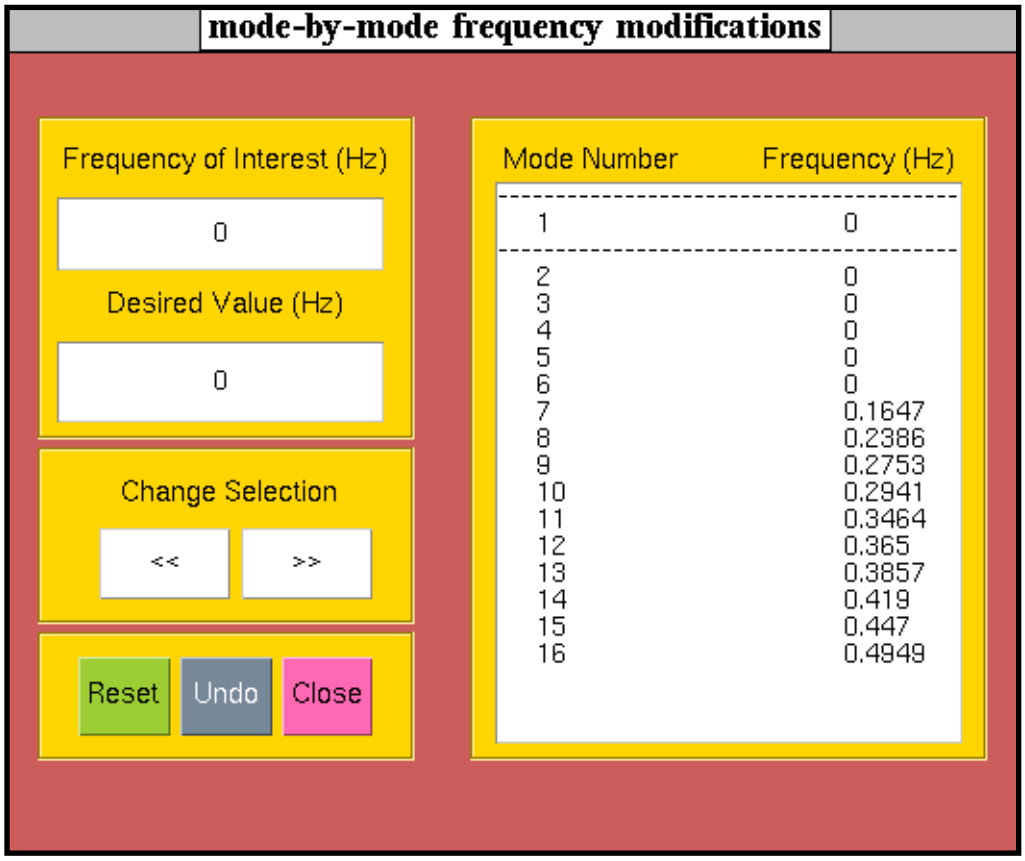

Figure 6. Mode-by-Mode frequency modification graphical user interface.

the list of frequencies or use the Frequency of Interest text field to enter a particular frequency region of interest. The value in hertz of the currently selected mode is displayed in the Desired Value text field. The currently selected mode is shown between the two horizontal dashed lines. The value of this mode can be changed by either deleting the text and reentering a new value or by scaling it with any arithmetic operator, for example, +, –, * for multiplication, or / for division. Additional options on the mode-by-mode graphical interface include Reset, Undo, and Close. The Reset button restores the modal frequencies to their nominal values as defined with the interface in figure 5, and the Undo button clears only the last frequency modification. The Close button closes the interface and accepts all frequency changes. A final note on frequency modifications is that each time the Frequencies submenu is selected, the frequencies defined in files `omega.dat` or `omega.mat` are used as the initial values.

Damping Ratio Modification

The Damping Ratios option from the Modify Plant Model menu provides the user access to a graphical tool for defining or modifying the spacecraft's modal damping schedule. The operation of the damping schedule graphical interface is similar to the frequency modification graphical interface. The graphical damping tool allows users to easily modify elements within the structural damping matrix. Structural damping ratios and ranges are specified by using a mouse to click and drag on graphical elements that represent modal damping ratios and ranges. Keyboard entries are also permitted once a modal damping range or value element has been activated by a mouse click. (See previous frequency modification description.) Damping schedules can also be saved or loaded by using the File button. This GUI feature uses, as its default damping schedule, the schedule defined in the user-changeable file `mkdamp.m`. If `mkdamp.m` has not been defined, then a default damping schedule of 0.25 percent is assumed for all modes.

Analysis

The Analysis menu allows the user to select between either time domain or frequency domain analysis modes. The time domain and frequency domain modes are mutually exclusive options; that is, PLATSIM does not allow for simultaneous time and frequency domain analyses. Selecting the button labeled frequency domain analysis

changes the functionality of the slider in the lower right corner of the main PLATSIM GUI and displays an additional figure window for setting frequency domain parameters. The additional figure window is closed when the frequency domain analysis is started or when frequency domain is no longer the mode of analysis.

Additional options under the Analysis top-level menu are Display Parameters, Progress Meter, and Begin Analysis. Selecting the Display Parameters button echoes the current values of PLATSIM parameters in the MATLAB command window. Progress Meter is selected by default and can be deselected by clicking on it with the mouse. During time domain analysis, if Progress Meter is selected, a graphic meter is displayed that shows the percent completion of the simulation response to the current disturbance event. Selecting Begin Analysis executes the MATLAB script file `plattime.m` or `platfreq.m` for time or frequency domain analyses, respectively. Once an analysis, time or frequency domain, has been started, all GUI functions are disabled. User control of the GUI is not returned until the analysis is finished. As of MATLAB version 4.2, the user can terminate the analysis at any time by sending an interrupt signal (Control C on most UNIX workstations) from the MATLAB command window unless both the Progress Meter is turned on and the analysis is performed by MEX-file (as opposed to M-file). Even if MATLAB is unresponsive to the interrupt signal, the system responds to the suspend signal (typically, control Z). The user resorting to this expedient should remember to kill the suspended process.

Disturbance

The Disturbance menu allows the user to select the disturbance model to be used in the PLATSIM analysis as well as select how multielement disturbance scenarios are handled in time simulations. The submenu Select Disturbances provides the user with access to the disturbance module, which is described in detail in chapter 11. The submenus Run Disturbances Separately and Run Disturbances Together control whether several simulations are run, one for each individual disturbance event, or whether the result of all disturbance events acting simultaneously is simulated. Typically, jitter analysis is performed by using individual disturbance events (i.e., run separately), and the contribution of all responses are summed together. However, to determine overall spacecraft response at various output locations, the Run Disturbances Together option provides a more useful approach. Selecting Run Disturbances Together sets the workspace variable `multflag` to yes (the default is no). The `multflag` variable is considered to be an optional time domain execution control parameter; hence, the user is not prompted for its value when using PLATSIM in the command-line driven mode. To run the disturbances together when using PLATSIM in the command-line mode, the user should issue the following MATLAB command before issuing `platsim`:

```
>> multflag='yes'
```

Quit

The Quit button simply quits the current MATLAB session without saving any workspace variables.

Number of Modes Slider

A slider control can be used to set the number of modes to be included in the analysis. The slider allows the user to use the mouse to move a sliding bar that sets the numeric input for the value in the text field of the slider. The slider moves according to the instructions given in the MATLAB 4.1 Release Notes: "When you mouse on an arrow, the slider position indicator (and the associated slider value) moves in the indicated direction by a value of 1/100th of the total range. Clicking the mouse while the pointer is in the trough moves the slider 1/10th of the total range." Currently, the total range for the number of modes is set to 600. Therefore, one click on the position indicator of the mode slider sets the number of modes equal to six.

The values of the sliders can also be changed by clicking the mouse in the text field and typing the desired scalar or vector values. A scalar entry of 100 is equivalent to `[1:100]` in MATLAB notation; that is, the first 100 modes are used in the simulation. Vector entries are used when multiple mode ranges are desired or when mode range starting points do not include the first mode. All vector entries must follow MATLAB's syntax. For example, to capture the rigid body response plus the flexible body response between modes 100 and 200, the user would enter `[1:6 100:200]` in the slider text field. Negative scalar values are also permitted and used to indicate that a single mode is used in the simulation. For example, if a text field entry of –100 is used, only mode 100 is used in the simulation. A slider value of 0 (the default) implies that all available modes are used in the simulation. The user must enter mode numbers in the text field for modes greater than 600.

Clip Window Slider/Range of Disturbance Events

This GUI slider serves a dual role. In time domain analysis, the Clip Window Slider is used to define the clip window (`tclip`), and in frequency domain

analysis, it is used to define the subset of disturbance events inputs (`ndist`) for a given disturbance scenario. The functionality of this slider is similar to that of the mode slider. Definition of the `tclip` and `ndist` execution control parameters are provided in appendix C. The default value of the Clip Window Slider is 0.

Chapter 7

Command-Driven Execution Mode

In command-driven mode, the user can set many of the execution control parameters by responses to questions from PLATSIM. To access and execute PLATSIM in command-driven mode, one must be running MATLAB 4.1 or higher. The MATLABPATH variable must be set properly so that MATLAB knows where to find PLATSIM and the user's model of the platform. In MATLAB, the user must issue the command `runmode='command';` set any PLATSIM execution control parameters that are not accessible through responses to PLATSIM questions, and issue the command `platsim` to MATLAB. These execution control parameters used by both time and frequency domain analyses are as follows: `phold` to set the maximum time that any given plot is held on the screen; `gad` to adjust the gains matrix, and `desint` to select a proper subset of the performance output instruments defined in the user-supplied file `instdata.m`. Complete definitions of all execution control parameters are given in appendix C.

Time Domain Analysis

Additional execution control parameters that apply to time domain analysis are as follows: `multflag`, which must be set to yes if multiple events in a disturbance scenario are to be run simultaneously; `saveflag`, which must be set to yes if the full time histories are to be written to MAT-files; `jtrflag`, which must be set to no if no jitter analysis is desired; `tclip`, which must be set to a positive value in the time units used in the time-history analysis if the user wants to disregard an initial segment of the time history during jitter analysis; and `nplpts`, which must be set to a value if the user does not want the default value for the number of points to use in reducing time-history data for plotting. Additionally, the performance meter, which is on by default, can be turned off by setting `pmflag` to no.

After the user has set the desired execution control parameters, the `platsim` command is issued. The user then performs the following steps:

1. Accept the default value, yes, for the time domain analysis prompt by entering a carriage return.

2. With the left mouse button, the user should choose a disturbance sequence from the MATLAB pop-up menu that appears on the screen. The menu disappears after a choice has been made. (See chapter 11 for a full description of the disturbances menu.) If PLATSIM has reason to believe that the user's terminal does not support graphics, then a text menu with numbered options is used instead of the GUI menu.

3. After receiving the prompt `Enter number of modes to use (enter 0 for all) [0]?`, the user should provide the number or range of spacecraft modes to be used in the analysis. To use all modes given in the file `omega.mat` (or `omega.dat`), enter `0` or accept the default with a carriage return. To use the first *k* modes in the analysis, enter `k`. To use a mixed range of modes (e.g., modes 4 to 8, 300, and 450 to 480), enter a MATLAB vector `[4:8,300,450:480]`. To use a single mode, such as mode k, enter `-k`.

4. After receiving the prompt `Plot results [yes]?`, enter `yes`, `y`, or carriage return for the time histories to be reduced and plotted on the graphics display. To obtain jitter results without plotting, enter `no` or `n`.

5. After requesting plotting (item 4) and receiving the prompt `Print results [yes]?`, enter `yes`, `y`, or carriage return for the plots to be printed to an encapsulated PostScript file. Otherwise, enter `no` or `n`.

6. After receiving the prompt `Closed-loop Analysis [yes]?`, enter `yes`, `y`, or carriage return for a closed-loop analysis with the spacecraft control system. Otherwise, enter `no` or `n`.

A typical PLATSIM session in a command-driven mode is presented in the following computer listing for an EOS-AM-1 disturbance scenario. The commands and command responses have been highlighted in bold. Anything not in bold was generated by the program. A carriage return used to accept a default value is shown as **<CR>**.

```
>> runmode='c';
>> desint=13:15;
>> pmflag='n';
>> platsim
```

```
Time Domain Analysis [yes]? <CR>

Form the disturbance vector and
 instrument data

Selected disturbance case:
 MODIS static imbalance
Task completed in 7.45 seconds

Form the continuous plant matrices

Enter range of modes to use (enter 0
 for all) [0]? [1:6,13:20,25]
Task completed in 5.417 seconds

Discretize the plant dynamics
Task completed in 0.1167 seconds

Plot results [yes]? <CR>

Print results [yes]? n

Perform linear discrete simulation

Closed-loop Analysis [yes]? n

Initiating simulation for disturbance
 vector 1
Task completed in 21.02 seconds

Perform jitter analysis
Task completed in 6.45 seconds
Current plot held

Truncate, plot, and print time
 histories
Task completed in 7.317 seconds

Simulation for disturbance vector 1
 completed

Initiating simulation for disturbance
 vector 2
Task completed in 19.53 seconds

Perform jitter analysis
Task completed in 6.417 seconds
Current plot held

Truncate, plot, and print time
 histories
Task completed in 7.467 seconds

Simulation for disturbance vector 2
 completed

Current plot held

PROGRAM COMPLETED
Total cpu time = 86.65 seconds
```

In this example, setting the execution control parameter `desint` to `13:15` before invoking PLATSIM simulates the response of only three performance outputs that were assigned identification numbers 13, 14, and 15. These identification numbers are defined in the user-supplied MATLAB function in file `instdata.m`. The first command response chooses time domain analysis by accepting the default. Note, the second command response, which is used to choose a disturbance scenario, is a mouse command from the menu shown in figure 7, and thus has not been included in this session. PLATSIM is then asked to build a model of the plant with modes 1 through 6, 13 through 20, and 25. The default to plot results is accepted; thus, the data are reduced for plotting, plots are displayed on the screen, and the plot data are written to a MATLAB MAT-file. The user declines print results; thus, the EPS files of the plots are not created. The last user response selects open-loop analysis.

Frequency Domain Analysis

To perform frequency domain analysis, after setting any desired execution control parameters and issuing the `platsim` command, the user performs the following steps:

1. Respond with `no` or `n` to the time domain analysis prompt. This response informs PLATSIM that frequency domain analysis will be performed.
2. With the left mouse button, the user should choose a disturbance scenario from the MATLAB pop-up menu that appears on the screen. The menu disappears after a choice has been made. (See chapter 11 for a full description on the disturbances menu.) If PLATSIM has reason to believe that the computer terminal does not support graphics, then a text menu with numbered options is used instead of the GUI menu.

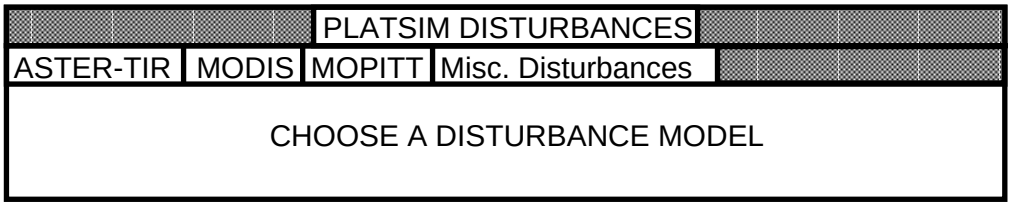

Figure 7. Disturbance module pop-up menu.

3. After receiving the prompt `Enter number of modes to use (enter 0 for all) [0]?`, the user should provide the number or range of spacecraft modes to be used in the analysis. To use all modes given in the file `omega.mat` (or `omega.dat`), enter `0` or accept the default with a carriage return. To use the first *k* modes in the analysis, enter `k`. To use a mixed range of modes (e.g., modes 4 to 8, 300, and 450 to 480), enter a MATLAB vector `[4:8,300,450:480]`. To use a single mode, such as mode k, enter `-k`.

4. After receiving the prompt `Enter number of disturbances to use (enter 0 for all) [0]?`, provide the number or range of disturbance events of the chosen disturbance scenario to be used in the frequency analysis. To use all events in the disturbance scenario, enter `0` or a carriage return to accept the default. To use the first *k* events in the analysis, enter `k`. To use a mixed range of events, (e.g., events 1, 3, 5, 6, and 7), enter a MATLAB vector `[1,3,5:7]`. To use a single disturbance event, such as mode k, enter `-k`.

5. After receiving the prompt `Plot results [yes]?`, enter `yes`, `y`, or carriage return for the Bode logarithmic gain and phase diagrams to be plotted on the graphics display. Otherwise, enter `no` or `n`.

6. After requesting plotting (item 5) and receiving the prompt `Print results [yes]?`, enter `yes`, `y`, or carriage return for the plots to be printed to an EPS file. Otherwise, enter `no` or `n`.

7. After receiving the prompt `Closed-loop Analysis [yes]?`, enter `yes`, `y`, or carriage return for a closed-loop frequency analysis with the spacecraft control system. Otherwise, enter `no` or `n`.

8. After receiving the prompt `Input the lower bound value for frequency in power of 10 [-2]?`, enter the desired lower bound frequency value (decimals are accepted, carriage return accepts the default). Note that the default for the lower bound value is 0.01 (or 10^{-2}).

9. After receiving the prompt `Input the upper bound value for frequency in power of 10 [4]?`, enter the desired upper bound frequency value (decimals are accepted, carriage return accepts the default). Note that the default for the upper bound value is 10000.0 (or 10^{4}).

10. After receiving the prompt `Input the number of points to be used in the frequency plots [1000]?`, enter the desired number of points to be used in the generation of the frequency response function matrix (carriage return accepts the default). Note, the default number of points is 1000.

A typical PLATSIM session in a command-driven mode is presented in the following computer listing for an EOS-AM-1 disturbance sequence. The commands and command responses have been highlighted in bold. Anything not in bold was generated by the program. A carriage return used to accept a default value is shown as **<CR>**.

```
>> runmode='c';
>> platsim

Time Domain Analysis [yes]? n

Form the disturbance vector and
 instrument data

Selected disturbance case:
 MODIS static imbalance
Task completed in 7.417 seconds

Form the continuous plant matrices

Enter range of modes to use (enter 0
 for all) [0]? 40
Task completed in 7.2 seconds

Enter range of disturbance events to
 use (enter 0 for all) [0]? <CR>

Plot results [yes]? <CR>

Print results [yes]? <CR>

Closed-loop Analysis [yes]? <CR>

Input the lower bound value for fre-
 quency in power of 10 [-2]? <CR>

Input the upper bound value for fre-
 quency in power of 10 [4]? 3

Input the number of points to be used
 in the frequency plots [1000]? 501

Form the Spacecraft Control System
 (SCS) matrices
Task completed in 1.067 seconds

Initiating Bode Plot for the distur-
 bance set
Task completed in 11.33 seconds

Plot and print Bode plots
Task completed in 89.75 seconds
```

```
PROGRAM COMPLETED
Total cpu time = 117 seconds
```

The first command response selects frequency domain analysis. Note that the second command response, which is used to choose a disturbance sequence, is a mouse command from the menu shown in figure 7 and thus has not been shown. In this example, modes 1 through 40 are used in the plant model, all disturbance elements of the disturbance sequence are chosen for frequency domain analysis. Both plotting and printing options are chosen; a closed-loop analysis is requested; and a frequency range between 0.01 (10^{-2}) and 1000.0 (10^{3}) with 501 point discretization is chosen.

Chapter 8

Batch Mode

In batch mode, PLATSIM execution control parameters are either set by the user with MATLAB assignment statements before the command `platsim` is entered or used with their default value. These assignments can be done in an interactive MATLAB window, but they are primarily intended to allow the user to run PLATSIM in the background with commands from a prewritten script. With two exceptions, if the user has not defined an execution control parameter when the command `platsim` is given, PLATSIM assigns a default value to the variable. One of these exceptions, `runmode`, must be set before PLATSIM is called. The other exception is `casenum`. If PLATSIM is running in GUI or command mode, `casenum` is set by user responses to the Disturbances menu. In batch mode, `casenum` must be set before PLATSIM is called. Appendix C contains specifications for the individual execution control parameters.

Batch Mode Operation

This section uses two examples to demonstrate the batch mode operations.

Example 1. MATLAB is Started in a Directory Containing Two Files

```
:::::::::::::
startup.m
:::::::::::::
format compact
path('/scb3/usr5/eos/platdir',path)
path(...
'/scb3/usr5/eos/platdir/eos_eg',path)
:::::::::::::
runplat.m
:::::::::::::
diary          % generate a diary of
               % the run

runmode='b';   % run in batch mode

tdflag = 'n'   % perform frequency
               % domain analysis

nmode = 40;    % use first 40 modes

clflag = 'n';  % run Bode plot open
               % loop

casenum = 7;   % use 'MODIS static
               % imbalance' disturbance
               % from the EOS-AM-1
               % example

prtflag = 'n'; % do not write .eps
               % files

desint = 14;   % only output is 'MODIS
               % Pitch'

ndist = -2;    % use only the second
               % event from the 'MODIS
               % static imbalance'
               % disturbance

phold = 120;   % give the user at least
               % 2 minutes to look at
               % plot

platsim        % This should now run a
               % single input/single
               % output Bode plot
               % using 1000 points
               % logarithmically spaced
               % between .01
               % and 10,000.
```

MATLAB is started and the command `>> runplat` is given. The necessary calculations are performed and the Bode plot is displayed. To exit from MATLAB, the command `>> quit` is given. Now two new files are in the directory—`MODIS_static_imbalance_freq.mat`, which contains the results of the Bode calculation in MATLAB readable form, and `diary`, which contains a record of events. The following shows the contents of `diary`.

```
Selected disturbance case:
 MODIS static imbalance

Form the continuous plant matrices
Task completed in 3.333 seconds

Form the Spacecraft Control System
 (SCS) matrices
Task completed in 0.6833 seconds

Initiating Bode Plot for the distur-
 bance set
Task completed in 0.55 seconds
```

```
Plot and print Bode plots
Task completed in 39.35 seconds

PROGRAM COMPLETED
Total cpu time = 43.98 seconds
>> quit

1471741 flops.
```

Example 2

This example uses a batch operation utility such as the UNIX `at` command. First a command script is prepared and written to a file, say `doit`, as follows. (Line numbers are not present in the script, they have been added here for reference purposes.)

```
1. setenv DISPLAY
2. /usr/local/matlab-4.1/bin/matlab\
   << EOF > mat-out
3. diary
4. runmode='b';
5. casenum=7;
6. pmflag='n';
7. platsim
8. quit
9. EOF
```

The UNIX command `at -m 20:00 doit` waits until 8 p.m. and then starts executing the commands in `doit`. Line 1 of `doit` corrects a problem MATLAB has on some systems with trying to run plotting commands without being run from a logged-in terminal. Line 2 starts MATLAB running and uses the remaining lines as input to MATLAB until it finds the EOF in line 9. MATLAB then executes lines 3 through 8 causing it to run PLATSIM in batch mode with disturbance number 7 (MODIS static imbalance) and with defaults for all other run parameters except `pmflag`. When the job is done, the user can obtain results from the files created by PLATSIM.

Chapter 9

Time Domain Analysis Module

The file `plattime.m` is the main driver for the linear time domain simulation and jitter analysis program. It connects the plant definition module, the spacecraft control system module, the disturbances module, the simulation module, and the jitter analysis module. PLATSIM takes advantage of the sparsity of the spacecraft dynamic model system matrix to perform efficient open-loop and closed-loop simulations. To take advantage of the open-loop sparsity, PLATSIM individually discretizes the plant and the controller models and then combines them in a feedback loop. This loop is generally not restrictive because almost all modern control systems are digitally implemented. The `plattime` module calls various routines to form the open-loop plant and disturbance matrices, to discretize the system matrices, to form the spacecraft control system matrices, to obtain time simulations of the performance outputs, to perform jitter analysis, and finally to document the results in MATLAB binary, tabular ASCII, and graphical PostScript forms. These routines are described in the order used as follows:

1. Routine `distrbs` (in file `distrbs.m`) is used to call user-supplied routine `distdata`. From the returned information, `distrbs` can obtain the disturbance profiles and the corresponding grid points and directions for those disturbances as well as the integration step size.

2. Routine `instrmts` (in file `instrmts.m`) is used to call user-supplied routine `instdata`. From the returned information, `instrmts` can define instrument types, locations, directions, and names as well as the window sizes for the jitter analysis.

3. Routine `rminstr` (in file `rminstr.m`) is used to redefine the performance output data according to parameter `desint`, which identifies those performance outputs to be used in simulations and jitter analysis.

4. Routine `formplnt` (in file `formplnt.m`) is used to generate the sparse open-loop plant matrices.

5. Routine `discrtz` (in file `discrtz.m`) is used to discretize the sparse system matrices with a sampling period equal to the integration step size.

6. User-supplied routine `formscs` (in file `formscs.m`) is used to generate the spacecraft control system matrices. These matrices are subsequently discretized by the MATLAB routine `c2d` with a sampling period equal to the integration step size.

7. Routine `simuc` (in file `simuc.m` or `simuc.mex*`) is used to perform the closed-loop simulations for the discrete system.

8. Routine `simuo` (in file `simuo.m` or `simuo.mex*`) is used to perform the open-loop simulations for the discrete system.

9. Routine `jitter` (in file `jitter.m` or `jitter.mex*`) is used to compute jitter values from the time histories computed by `simuc` or `simuo` for the time windows defined by variable `window` in file `instdata.m`.

10. Routine `jitterds` (in file `jitterds.m`) is used to generate the ASCII files that contain the jitter results in tabular form.

11. Routine `table` (in file `table.m`) is used to generate the PostScript files that contain the jitter results in tabular form.

12. Routine `trplot4` (in file `trplot4.mex*`) is used to condense the time-history data in such a way that hardcopy plots of the condensed time histories have the same visual effect as plots of the full time history.

The jitter and simulation analysis is started by calling the jitter and simulation analysis module that resides in a file named `plattime.m`. The optional parameters `gad`, `tclip`, `desint`, `saveflag`, and `multflag` are used within `plattime` to provide additional capabilities. The error and diagnostic messages generated by `plattime` (in file `plattime.m`) are as follows:

1. In the command-line and GUI execution modes, if a correct disturbance `casenum` is not selected from the disturbances menu, then the following message appears in the MATLAB window:

   ```
   A proper disturbance casenum was
   not specified, please select one
   now.
   ```

Solution: Choose a proper disturbance `casenum` from the pop-up menu. This error also stops execution in batch mode.

2. If the scalar parameter `tclip` defined is greater than the simulation time and the execution mode is command driven or menu driven, then the following message appears in the MATLAB window:

   - `tclip is larger than tfinal;`
   - `input a new value for tclip.`

   If the scalar parameter `tclip` defined is greater than the simulation time and the execution mode is batch, then the following message appears:

   - `tclip is larger than tfinal;`
   - `tclip is set to 0.5*tfinal.”`

   Solution: Adjust `tclip` or increase the simulation time.

Chapter 10

Plant Definition Module

Continuous Model

The Plant Definition module generates the open-loop state matrix, the control input influence matrix, the measurement and performance output influence matrices, and the disturbance influence matrix. These matrices are generated by routine `formplnt`. As mentioned earlier, the plant dynamics in a first-order state space form is given in equation (5) as

$$\begin{aligned} \dot{x}_s &= A_s x_s + B_s u + B_d w \\ y &= C x_s \\ y^{pr} &= C^{pr} x_s + D_u^{pr} u + D_w^{pr} w \end{aligned}$$

where x_s is the plant state vector, u is the spacecraft control system input vector, w is the disturbance vector, A_s is the plant state matrix, B_s is the input influence matrix, B_d is the disturbance influence matrix, C and C^{pr} are the measurement and performance output influence matrices, and D_u^{pr} and D_w^{pr} are the control input and disturbance feedthrough matrices. The respective structure of these vectors and matrices are defined in chapter 2. However, because of the sparse nature of the open-loop state matrix, this matrix is stored by `formplnt` in the following compact form:

$$A_s = \begin{bmatrix} 0 & 1 \\ -\omega_1^2 & -2\zeta_1\omega_1 \\ 0 & 1 \\ -\omega_2^2 & 2\zeta_2\omega_2 \\ \vdots & \vdots \\ \vdots & \vdots \\ 0 & 1 \\ -\omega_p^2 & 2\zeta_p\omega_p \end{bmatrix}$$

In this form, ω_i and ζ_i represent the modal frequency and modal damping ratio of mode i. The symbol p represents the number of modes included in the model. This compact form reduces the storage requirements from $p \times p$ to $2 \times p$ and prepares the state matrix for PLATSIM sparse computations. The routine `formplnt` requires the following user-supplied inputs or input files:

- In the command-driven mode, `formplnt` prompts the user for the range of structural modes to be used in the analysis. The form of this input is discussed in chapter 7.
- For the modal frequencies, `formplnt` first looks for a MATLAB binary file `omega.mat`. If such a file does not exist, `formplnt` then attempts to load the ASCII file `omega.dat`.
- For the structural mode shapes, `formplnt` first looks for a MATLAB binary file `phi.mat.` If such a file does not exist, `formplnt` then attempts to load the ASCII file `phi.dat`.
- As mentioned previously, `formplnt` uses routine `mkdamp` (in the user-supplied file `mkdamp.m`) to generate the modal damping ratios.

The error and diagnostic messages generated by `formplnt` are as follows:

- If any of the mode numbers chosen for analysis is greater than the number of modes available in the file `omega.dat` or `omega.mat`, then the following message appears in the MATLAB window and program execution stops:

  ```
  An error has been detected in file formplnt.m; maximum mode number chosen is greater than number of modes available; program termination in formplnt.
  ```

  Solution: Check the mode numbers and restart the program.

- If the number of rows of array `phi` (in file `phi.mat` or `phi.dat`) is not an integer multiple of the number of modal frequencies defined in file `omega.mat` or `omega.dat`, then the following error message appears in the MATLAB window and program execution stops:

  ```
  The number of rows of phi (in file phi.mat or phi.dat) must be an integer multiple of the number of elements of omega (in file omega.mat or omega.dat). Program termination in formplnt.
  ```

Solution: Make the `phi` and `omega` files consistent and restart the program.

- If a grid point defined in file `instdata.m` for the spacecraft control system input is not defined in array `phi` (in file `phi.mat` or `phi.dat`), then the following error message appears in the MATLAB window and program execution stops:

  ```
  An error has been detected in file formplnt.m. A FEM space platform control system input grid number as referenced in file instdata.m is not available in file phi.dat or phi.mat. Program termination in formplnt.
  ```

  Solution: Check the first row of parameter `act` in file `instdata.m`, check file `phi.dat` or `phi.mat`, and restart the program.

- If a grid point used in file `instdata.m` for the measurement outputs is not defined in array `phi` (in file `phi.mat` or `phi.dat`), then the following error message appears in the MATLAB window and program execution stops:

  ```
  An error has been detected in file formplnt.m. A FEM measurement output grid number as referenced in file instdata.m is not available in file phi.dat or phi.mat. Program termination in formplnt.
  ```

  Solution: Check the first parameter `mout` in file `instdatas.m`, check file `phi.dat` or `phi.mat`, and restart the program.

- If a grid point used in file `instdata.m` for the performance outputs is not defined in array `phi` (in file `phi.mat` or `phi.dat`), then the following error message appears in the MATLAB window and program execution stops:

  ```
  An error has been detected in file formplnt.m. A FEM performance output grid number as referenced in file instdata.m is not available in file phi.dat or phi.mat. Program termination in formplnt.
  ```

  Solution: Check the first row of parameter `pout` in file `instdata.m`, check file `phi.dat` or `phi.mat`, and restart the program.

- If a grid point used in file `distdata.m` for a disturbance sequence is not defined in array `phi` (in file `phi.mat` or `phi.dat`), then the following error message appears in the MATLAB window and program execution stops:

  ```
  An error has been detected in file formplnt.m. A FEM disturbance grid number as referenced in file distdata.m is not available in file phi.dat or phi.mat Program termination in formplnt.
  ```

  Solution: Check file `distdata.m` and file `phi.dat` or `phi.mat` and restart the program.

Discrete Model

Routine `discrtz` (in file `discrtz.m`) is used to discretize the plant dynamics, with a zero-order hold, for use in the simulation of the open-loop and closed-loop response. This routine takes advantage of the sparsity of the open-loop state matrix of the plant and discretizes the dynamics of each mode separately. For example, for mode i the discretized dynamics is obtained by forming

$$P^i \equiv \begin{bmatrix} A_s^i & B_s^i & B_d^i \\ 0 & 0 & 0 \end{bmatrix}$$

where A_s^i is the ith 2×2 partition of state matrix A_s, B_s^i is a matrix formed by rows $2i-1$ and $2i$ of the control input influence matrix B_s, B_d^i is a matrix formed by rows $2i-1$ and $2i$ of the disturbance influence matrix B_d, and the rows of 0's are added to produce a square matrix P^i. Then, the exponential of P^i is computed as follows:

$$\bar{P}^i = e^{P^i} \equiv \begin{bmatrix} \bar{A}_s^i & \bar{B}_s^i & \bar{B}_d^i \\ 0 & 0 & 0 \end{bmatrix}$$

where $\bar{A}_s^i$, $\bar{B}_s^i$, and $\bar{B}_d^i$ are appropriate partitions of $\bar{P}^i$. These partitions represent the discretized state matrix, control input influence matrix, and disturbance influence matrix for mode i. The discretized plant matrices can be written as

$$\bar{A}_s = \begin{bmatrix} \bar{A}_s^1 \\ \bar{A}_s^2 \\ \vdots \\ \bar{A}_s^p \end{bmatrix} \qquad \bar{B}_s = \begin{bmatrix} \bar{B}_s^1 \\ \bar{B}_s^2 \\ \vdots \\ \bar{B}_s^p \end{bmatrix} \qquad \bar{B}_d = \begin{bmatrix} \bar{B}_d^1 \\ \bar{B}_d^2 \\ \vdots \\ \bar{B}_d^p \end{bmatrix}$$

where $\bar{A}_s$, $\bar{B}_s$, and $\bar{B}_d$, respectively, represent the discretized state matrix in compact form, control input influence matrix, and disturbance influence matrix of the plant.

Chapter 11

Disturbance Module

The PLATSIM disturbance module provides the capability to interactively select various disturbance scenarios with a graphical user interface. The disturbance module also offers a command-line mode when MATLAB is run in the terminal graphics mode. This module also allows batch mode simulations. (See chapter 8.)

The main PLATSIM GUI provides access to three disturbance functions under the pull-down menu labeled Disturbance. The three disturbance functions are labeled Select Disturbances, Run Disturbances Separately, and Run Disturbances Together. When the pull-down menu labeled Select Disturbances is selected, an additional figure window with a top-level menu appears. Figure 7 shows this menu for the EOS example. The disturbance window top-level menu displays labels pertaining to disturbance scenario groups that consist of one or more disturbance scenarios. A single instrument or system can have multiple disturbance scenarios. For example, a scanning telescope can have a disturbance scenario that describes the forces and/or torques related to the motion of a scanning mirror, and it can have a separate disturbance scenario for calibration operations.

The entries in the `cnames` and `dnames` matrices provide the names for the disturbance scenario groups and the disturbance scenarios, respectively. (See section on “Disturbance Data” of chapter 3.) Each disturbance scenario (a row entry in `dnames`) is a separate submenu item under the top-level menu and can consist of single or multiple disturbance events. String variables describing the disturbance scenarios are used for labeling time history plots, submenus, items, and jitter file names. The disturbance scenario group names are used only as labeling for menu buttons. A disturbance scenario can be selected from the pull-down menu with a single click from a mouse. Upon selection, the appropriate disturbance models, as defined in the user-provided file `distdata.m`, are executed and instrument or system disturbance data are loaded into the MATLAB workspace. This procedure can take several seconds to complete; actual time depends on the number of data points and individual force or torque profiles that comprise the disturbance scenario.

If the user attempts to run a simulation without first selecting a disturbance model, a warning message is displayed with instructions to select a disturbance. Then the disturbance module figure window is displayed.

The PLATSIM disturbance module consists of two internal functions (`distrbs.m` and `newmenu.m`) and a user-provided file `distdata.m`. The file `distrbs.m` is the disturbance module executive file and is used to call both `distdata.m` and `newmenu.m`. In batch mode, the disturbance figure window is not displayed, and disturbance selection is accomplished by predefining the variable `casenum`. The batch mode variable `casenum` must be assigned in the batch mode input file to a valid disturbance scenario case number (i.e., a valid `dnames` row number). As described in the batch mode operation section, the workspace variable `runmode='batch'` is reserved as an indicator of batch mode operation.

The function `newmenu.m` controls all GUI functions for the PLATSIM disturbance module. This function displays a figure window with a top-level menu. The labels for this menu correspond to the disturbance scenario group labels stored in variable `cnames`. Each top-level menu has several submenus corresponding to the disturbance scenarios as referenced in their respective rows of the matrix `dnames`. The output argument `casenum` refers to a row number in `dnames` and is passed back into the file `distdata.m`.

The PLATSIM disturbance options labeled Run Disturbances Separately and Run Disturbances Together determine how simulation results are generated. If Run Disturbances Separately is selected, the time history response (including jitter results) for each disturbance event in a particular disturbance scenario is calculated separately. The total jitter results are then determined by adding the contribution from each disturbance event. When the overall dynamic response of the systems is desired (i.e., the response of the system due to all disturbance events applied simultaneously), then Run Disturbances Together should be selected.

Chapter 12

Simulation Module

The simulation routine named `simuc` (in file `simuc.m` or `simuc.mex*`) takes advantage of the sparsity of the plant system matices to perform efficient time simulations for the closed-loop system. The open-loop simulations are performed by the routine `simuo` (in file `simuo.m` or `simuo.mex*`). These routines use the sparse discrete model of the plant and the discrete model of the spacecraft control system. A discrete simulation of the open-loop and closed-loop behavior of the system is performed through algebraic state propagation. The discretized dynamics of the plant and the spacecraft control system from equation (7) on mathematical formulations are as follows:

$$\begin{aligned} x_s(k+1) &= \bar{A}_s x_s(k) + \bar{B}_s u(k) + \bar{B}_d w(k) \\ y(k) &= C x_s(k) \\ y^{pr}(k) &= C^{pr} x_s(k) + D_u^{pr} u(k) + D_w^{pr} w(k) \end{aligned}$$

where $\bar{A}_s$ refers to the discrete form of the block diagonal form of chapter 2, and

$$\begin{aligned} x_c(k+1) &= \bar{A}_c x_c(k) + \bar{B}_c y(k) \\ u(k) &= -C_c x_c(k) \end{aligned}$$

The simulation routines take advantage of the sparsity of discretized state matrix $\bar{A}_s$. The $2 \times p$ packed version of the discrete system matrix is now denoted by $\bar{A}_s$. Then the simulation routines propagate the plant states as follows:

$$\begin{aligned} x_s(k+1;1{:}2{:}2p) &= \bar{A}_s(1{:}2{:}2p, 1) .* x_s(k;1{:}2{:}2p) \\ &\quad + \bar{A}_s(1{:}2{:}2p, 2) .* x_s(k;2{:}2{:}2p) \end{aligned}$$

$$\begin{aligned} x_s(k+1;2{:}2{:}2p) &= \bar{A}_s(2{:}2{:}2p, 1) .* x_s(k;1{:}2{:}2p) \\ &\quad + \bar{A}_s(2{:}2{:}2p, 2) .* x_s(k;2{:}2{:}2p) \end{aligned}$$

$$x_s(k+1) = x_s(k+1) + \bar{B}_s * u(k) + \bar{B}_d * w(k)$$

where, for example, $x_s(k;1{:}2{:}p)$ uses MATLAB notation to represent a vector whose elements are the first, third, fifth, etc., elements of $x_s(k)$; $\bar{A}_s(2{:}2{:}p, 2)$ represents a vector using every other element from the second column of $\bar{A}_s$ starting with the second row; and $\bar{A}_s$, $\bar{B}_s$, $\bar{B}_d$, u, and w denote the discrete plant state matrix, the discrete spacecraft control input influence matrix, the discrete disturbance influence matrix, the spacecraft control input vector, and the disturbance vector, respectively. Symbols `*` and `.*` denote matrix multiplication and element-by-element vector multiplication, respectively.

With this method of propagation, the matrix vector multiplication $\bar{A}_s x_s(k)$ requires only about $4p$ floating-point operations as compared with the $4p^2$ floating-point operations for the full matrix-vector multiplication. Thus, this method results in significant reduction in computational time, particularly as the number of modes in model increases ($p > 100$). Currently, the simulation routine `simuc` does not accommodate feedthrough terms (associated with measurement outputs not performance outputs) for the plant or the spacecraft control system. However, feedthrough terms for the plant or the spacecraft control system can easily be implemented with a slight modification of the routine.

To increase the computational speed of the discrete simulation beyond MATLAB routines `simuc` and `simuo`, the routines have been implemented in FORTRAN 77 with the same discrete architecture. The calling statement of the MEX-file version of the open-loop and closed-loop simulation routines are the same as their M-file counterparts. Appendix A explains how to create the MEX-files from the FORTRAN 77 code distributed with PLATSIM.

Chapter 13

Frequency Domain Analysis Module

PLATSIM accomplishes frequency domain analysis by calling the MATLAB script `platfreq` (in M-file `platfreq.m`) from the `platsim.m` script. The `platfreq` script obtains values for any PLATSIM run-time parameters that have not been previously set, calls a frequency domain analysis routine to calculate the Bode plot data, and forms the plots.

The frequency domain analysis routines are named `sbodeo` (in files `sbodeo.m` and `sbodeo.mex*`) for open-loop analysis and `sbodec` (in files `sbodec.m` and `sbodec.mex*`) for closed-loop analysis. These routines evaluate the transfer function matrix from the selected events of the chosen disturbance scenario to the selected performance outputs at each point of a user-specified vector of frequency values. Efficiency is achieved in the calculation by taking advantage of special characteristics of the plant model.[2] The open-loop system matrix is sparse (see chapter 10), and the odd-numbered rows of the control and disturbance influence matrices are 0.

Open-Loop Calculation

Recall that the open-loop transfer function from the disturbances w to the performance output y^{pr} is given in equation (9) as

$$T(s) = C^{pr}(sI - A_s)^{-1}B_d + D_w^{pr}$$

where, for purposes of this chapter, B_d and D_w^{pr} contain only the columns requested by execution control parameter `ndist`. The symbol r denotes the number of columns in these matrices.

By taking advantage of sparsity in the open-loop system matrix, the computational complexity of calculating $(sI - A_s)^{-1}B_d$ is much less than it is for the full matrix techniques. Recall from equations (2) and (3) that

$$A_s = \begin{bmatrix} A_s^1 & 0 & \dots & 0 \\ 0 & A_s^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_s^p \end{bmatrix}$$

where

$$A_s^i = \begin{bmatrix} 0 & 1 \\ -\omega_i^2 & -2\zeta_i\omega_i \end{bmatrix}$$

Then $(sI - A_s)^{-1}$ is block diagonal with the ith block being

$$\left(sI_2 - A_s^i\right)^{-1} = \left\{1 \big/ \left(s^2 + 2\zeta_i\omega_i s + \omega_i^2\right)\right\} \begin{bmatrix} s + 2\zeta_i\omega_i & 1 \\ -\omega_i^2 & s \end{bmatrix}$$

$$(i = 1, 2, \dots, p)$$

where I_2 denotes the 2×2 identity matrix. Furthermore, as noted in the discussion following equation (4) in chapter 2, the odd-numbered rows of B_d are 0. If the row i of B_d is denoted by $b_d^{(i)}$, $Q(s)$ is used to represent $(sI - A_s)^{-1}B_d$, and $Q(s)$ is partitioned as

$$Q(s) = \begin{bmatrix} Q_1(s) \\ Q_2(s) \\ \vdots \\ Q_p(s) \end{bmatrix} \tag{14}$$

where each partition matrix $Q_i(s)$ is a 2 by r matrix, then

$$Q_i(s) = \left\{1 \big/ \left(s^2 + 2\zeta_i\omega_i s + \omega_i^2\right)\right\} \begin{bmatrix} b_d^{(2i)} \\ s b_d^{(2i)} \end{bmatrix}$$

$$(i = 1, 2, \dots, p) \tag{15}$$

The calculation is completed in the following manner:

$$T(s) = C^{pr}Q(s) + D_w^{pr}$$

[2] A paper by Peiman G. Maghami and Daniel P. Giesy describing this, titled "Efficient Computation of Closed-Loop Frequency Response Matrices for Large Order Flexible Systems," has been submitted for publication.

The gain and phase angle data (Bode plot data) are then computed directly from the frequency response function matrix. If no acceleration performance measurements exist, then the feed-forward term is 0 and the software bypasses the step where D_w^{pr} is added.

The standard, full matrix way to calculate $Q(s)$ involves first performing an LU decomposition of $sI - A_s$ followed by a backward and then forward solution of the triangular systems of equations using the columns of B_d as right-hand sides. The FLOP (floating point operations) count for this is $O(p^3) + O(p^2 r)$. When the solution is calculated as in equations (14) and (15), the FLOP count is $O(pr)$. This count represents a substantial savings, particularly when a large number of modes is necessary to model the desired phenomena.

Closed-Loop Calculation

Recall from equations (10) to (12) that the closed-loop dynamics of the controlled spacecraft can be written as

$$\dot{x} = \tilde{A}x + \tilde{B}_d w$$

$$y^{pr} = \tilde{C}^{pr} x + D_w^{pr} w$$

where

$$x = \begin{Bmatrix} x_s \\ x_c \end{Bmatrix}$$

and

$$\tilde{A} = \begin{bmatrix} A_s & -B_s C_c \\ B_c C & A_c \end{bmatrix} \qquad \tilde{B}_d = \begin{bmatrix} B_d \\ 0 \end{bmatrix} \qquad \tilde{C}^{pr} = \begin{bmatrix} C^{pr} & -D_u^{pr} C_c \end{bmatrix}$$

Given the closed-loop state matrix $\tilde{A}$, the block diagonal form of the open-loop plant has obviously been destroyed by the coupling generated by the feedback connection of the plant and the spacecraft control system. However, the initial sparsity of the open-loop state matrix is still intact. Now, this sparsity is exploited to develop an efficient method for the computation of the closed-loop frequency response function matrix of the controlled flexible spacecraft. Recall from equation (13) that the closed-loop transfer function from the disturbances to the performance outputs is

$$\tilde{T}(s) = \tilde{C}^{pr} (sI - \tilde{A})^{-1} \tilde{B}_d + D_w^{pr}$$

If sparsity is not exploited and many structural modes are modeled, then from equation (13), a large computational effort is required to calculate the closed-loop frequency response function matrix because the matrix term $(sI - \tilde{A})^{-1} \tilde{B}_d$ needs $s = j\omega$ to be computed for all desired frequency values.

The matrix term in equation (13) can be written as

$$(sI - \tilde{A}) = \begin{bmatrix} sI_s - A_s & B_s C_c \\ -B_c C & sI_c - A_c \end{bmatrix} \equiv \begin{bmatrix} E_{11}(s) & E_{12} \\ E_{21} & E_{22}(s) \end{bmatrix} \tag{16}$$

where I_s and I_c are identity matrices of orders equal to the size of plant state vector and controller state vector, respectively. Now let the elements of the matrix term $(sI - \tilde{A})^{-1}$ be defined as

$$\begin{bmatrix} X_{11}(s) & X_{12}(s) \\ X_{21}(s) & X_{22}(s) \end{bmatrix} \equiv (sI - \tilde{A})^{-1} = \begin{bmatrix} E_{11}(s) & E_{12} \\ E_{21} & E_{22}(s) \end{bmatrix}^{-1} \tag{17}$$

A formula for this inverse is given in section 5.3g of reference 4 as follows:

$$\left.\begin{aligned} \Delta &= E_{22} - E_{21} E_{11}^{-1} E_{12} \\ X_{11} &= E_{11}^{-1} + E_{11}^{-1} E_{12} \Delta^{-1} E_{21} E_{11}^{-1} \\ X_{12} &= -E_{11}^{-1} E_{12} \Delta^{-1} \\ X_{21} &= -\Delta^{-1} E_{21} E_{11}^{-1} \\ X_{22} &= \Delta^{-1} \end{aligned}\right\} \tag{18}$$

where it is called Schur's identity. Using equations (12) and (17) in equation (13), the closed-loop transfer function from the disturbances to the performance outputs is reduced to the following:

$$\begin{aligned} \tilde{T}(s) &= \begin{bmatrix} C^{pr} & -D_u^{pr} C_c \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} B_d \\ 0 \end{bmatrix} + D_w^{pr} \\ &= C^{pr} X_{11} B_d - D_u^{pr} C_c X_{21} B_d + D_w^{pr} \end{aligned}$$

Using equation (18) this equation becomes

$$\begin{aligned} \tilde{T} = {} & C^{pr} E_{11}^{-1} B_d + C^{pr} E_{11}^{-1} E_{12} \Delta^{-1} E_{21} E_{11}^{-1} B_d \\ & + D_u^{pr} C_c \Delta^{-1} E_{21} E_{11}^{-1} B_d + D_w^{pr} \end{aligned}$$

Using equation (16), replace E_{12} and E_{21} in the expression for Δ in equation (18) and the previous equation. This substitution produces the following:

$$\left.\begin{aligned} \Delta &= E_{22} + B_c C E_{11}^{-1} B_s C_c \\ \tilde{T} &= C^{pr} E_{11}^{-1} B_d - C^{pr} E_{11}^{-1} B_s \left(C_c \Delta^{-1} B_c C E_{11}^{-1} B_d \right) \\ &\quad - D_u^{pr} \left(C_c \Delta^{-1} B_c C E_{11}^{-1} B_d \right) + D_w^{pr} \end{aligned}\right\} \quad (19)$$

In this form, the following computational efficiencies are observed:

- Because $E_{11} = (sI - A_s)$, B_s and B_d have 0's in the odd-numbered rows, the subexpressions $E_{11}^{-1} B_s$ and $E_{11}^{-1} B_d$ can be computed with the techniques presented for efficient computation of the open-loop transfer function.
- The computation of $\Delta^{-1} B_c$ requires a full matrix computation. However, it should not be costly to compute because Δ is of the same order as the spacecraft control system, which is usually small compared with the order of the analysis model of the plant.
- Common subexpressions, such as those mentioned in the previous items and those enclosed in parentheses in equation (19), are computed once, saved, and reused.
- Because of the expected shapes of the matrices and the exploitation of common subexpressions, precomputing certain matrix products in equation (19) that are independent of the frequency parameter s is not advisable.
- If no acceleration sensors exist in the performance outputs so that the feed-forward matrices are 0, the software can bypass the second line of the $\tilde{T}$ calculation in equation (19).

Now, having defined $\tilde{T}(s)$, the frequency response function matrix of the closed-loop system is evaluated for various values of $s = j\omega$, with ω taking on the user-specified frequency values. The closed-loop gain and phase plots (Bode plots) are then computed directly from the frequency response function matrix.

Software Implementation

The evaluation of the open-loop or closed-loop transfer function is accomplished by MATLAB function M-files or MEX-files. The M-files contain straightforward implementations of the calculations presented in the preceding two subsections.

The FORTRAN 77 source code for the MEX-files uses the Basic Linear Algebra Subprograms (BLAS) to perform vector-vector, vector-matrix, and matrix-matrix operations. (See refs. 6 to 12.) In addition, the LAPACK (ref. 12) subroutine ZGESV, a complex double precision linear equation solver, is used to calculate $\Delta^{-1} B_c$.

Chapter 14

Jitter Analysis Module

The jitter analysis module named `jitter` (in file `jitter.m` and (optionally) `jitter.mex*`) performs efficient jitter analysis of discrete time histories represented in MATLAB as a vector of points.

The level of jitter in a given time history is defined as a function of a user-supplied jitter window, which is just a time interval whose duration is no more than the duration of the time history. Suppose that the discrete time history z contains n points $z(1), \ldots, z(\mathrm{n})$ at equal time increment) and the length of the jitter window can cover k of these points at a time with $k \le n$. First define $z_{\max}$ by setting

$$z_{\max}(i) = \max_i \; [z(i), z(i+1), \ldots, z(i+k-1)]$$

and define $z_{\min}$ by setting

$$z_{\min}(i) = \min_i \; [z(i), z(i+1), \ldots, z(i+k-1)]$$

The components of the vectors $z_{\max}$ and $z_{\min}$ give the maximum and minimum values in z for each distinct position of the window as it slides along the vector z. Jitter is then defined by

$$\text{jitter} = \max [z_{\max}(1) - z_{\min}(1), \ldots, z_{\max}(n-k+1) - z_{\min}(n-k+1)]$$

which is the maximum peak-to-peak excursion in z over the jitter window in any position. In a typical application, jitter needs to be calculated for several different jitter windows (i.e., several different values of k).

If this definition is turned into a simple computer program, the calculation is inefficient because $k(n-k+1)$ references into the z vector must be made. In typical applications, n is 100000 or more, and k can run from several hundred through several tens of thousands up to as large as n. Several ideas are implemented to improve the efficiency of this calculation.

For the smallest jitter window, a single scan is made through z, and lists of pointers are kept that point to locations in z that might possibly contribute to $z_{\max}$ or $z_{\min}$. At each step, a check is made to determine each value that can be dropped from each list. Values can be dropped if the window has passed by the location it points to or if new information has superceded it. At each step, the value of $z_{\max}$ is $z(j)$, where j is the smallest pointer in the maximum list. The determination of $z_{\min}$ is analogous. A running tally is kept that yields jitter at the end of the sweep.

For each larger jitter window, further efficiency is gained by using the information stored in $z_{\max}$ and $z_{\min}$ from the previous jitter window. Suppose $k_1 < k_2$, the information stored in $z_{\max}$ and $z_{\min}$ correspond to a window containing k_1 points, and jitter is to be calculated for a jitter window containing k_2 points. To find $\max [z(i), \ldots, z(i+k_2-1)]$, the interval $[i, i+k_2-1]$ is first covered with intervals of length k_1 (overlap is permissible and can be used to advantage, but none of the short intervals can extend outside the long interval). Then, the maximum of the z data over each short interval is found from an appropriate entry of $z_{\max}$, and the maximum of the z data over the long interval is determined as the maximum of these numbers. By properly choosing this covering, the maximum can be determined for several additional values of i by moving only the left and right covering intervals and leaving the central group of covering intervals fixed. For these values of i, $\max [z(i), \ldots, z(i+k_2-1)]$ is computed by taking the maximum of just three numbers, that is, values from the $z_{\max}$ array for the left and right covering intervals and a value for the central group that was calculated at the initial placement of the covering intervals. Eventually, this moving of left and right intervals and leaving the central group fixed arrives at an unusable configuration, at which point a new start must be made with a new covering. The calculation of $\min [z(i), \ldots, z(i+k_2-1)]$ is performed in the same manner. With these values, $z_{\max}$ and $z_{\min}$ can be updated for the new jitter window length, and a value accumulated to find the jitter at the new jitter window length.

For some values of n, k_1, and k_2, the time to recompute the central group contribution each time a new start is made with a new covering becomes significant. Thus, using an adaptation of the technique for the first window, now applied to subsequences from the $z_{\max}$ and $z_{\min}$ arrays for the k_1 window, provides further efficiency in the k_2 window calculation.

The proposed jitter algorithm provides for substantial computational efficiency. In a typical case, with $n = 100001$ and four jitter windows giving values of

k = 500, 900, 4500, and 27500, the processing time from the techniques outlined previously is a factor of over 1600 times less than that obtained with the brute force method. Testing indicates that the present method is linear in problem size (i.e., the problem is twice as big when both n and k values are doubled), and the brute force method is quadratic in problem size.

The jitter analysis is performed with a statement of the form

```
[zjtr, zmax, zmin] = jitter(z, delt,...
win);
```

The required inputs are as follows:

`z` A vector containing the time history. The value of n above is `length(z)`.

`delt` A scalar containing time increment between points of `z`.

`win` A vector containing the lengths of the jitter windows (measured in time units, the values of k used above are in the vector `win/delt`).

The restrictions are as follows:

- The value in `delt` must be positive.
- The numbers in `win` must be in nondecreasing order, and the largest must be no bigger than `delt*(length(z) – 1)`, the total duration of the time history in z.

The outputs are as follows:

`zjtr` A vector the same length as `win`. The value returned in `zjtr(i)` is the jitter in `z` for the jitter window `win(i)`. This output is required.

`zmax`, `zmin` These are optional outputs, originally included for debugging purposes. These are the same as the z_{max} and z_{min} in the definition above for the longest jitter window in `win`.

The file `jitter.m` contains help information for the jitter function and does the jitter calculation if no MEX-file is available. The jitter calculation however can be performed substantially faster with the file `jitter.mex*`. Appendix A explains how to create the MEX-files from the FORTRAN 77 code distributed with PLATSIM.

The FORTRAN 77 code for jitter analysis is contained in six subprograms that are distributed in the files `jitter.for`, `jitter.f`, `jitr1b.f`, `jitr1d.f`, `jitrnb.f`, `jitrnd.f`, and `jitterg.f`. The file `jitterg.f` contains the MATLAB MEX-file gateway routine. This routine manages the transfer of data between MATLAB and FORTRAN. The remaining five routines can be used as a package in FORTRAN 77 applications for jitter analysis.

The main routine is subroutine `JITTER`. This routine calculates the number of points in each jitter window, then it selects from among the remaining four modules to complete the jitter calculation.

Two of the modules, `JITR1B` and `JITR1D`, perform the calculation for the smallest window. They are identical in function; both take the time history vector z and the number k of points in the jitter window as input and return the amount of jitter in z for this window and the z_{max} and z_{min} arrays. The difference is in the algorithm used. `JITR1B` implements the brute force calculation, and `JITR1D` uses the more sophisticated version. Both algorithms are included because for values of k close to 1 or n, the brute force algorithm is better than the more sophisticated algorithm, which has some nontrivial $O(n)$ overhead. The decision as to which module to use is based on comparing the estimated timings of the two modules. These timing estimates are based on timing formulas that were empirically determined by timing a variety of problems over a range of n and k values on a Sun Sparc 10/512 workstation. A weighted least-squares fitting technique is used on the resulting data to provide coefficients in timing formulas. The form of these timing formulas is based on an analysis of the algorithms.

Once the first window calculation has initialized the z_{max} and z_{min} arrays, further calculation is performed by either `JITRNB` or `JITRND`. These are also identical in function; they take as input z_{max}; z_{min}; n; k_1, (the number of points in the window used in calculating the input values of z_{max} and z_{min}), and k_2, (the number of points in the window over which jitter is now to be calculated). The output is the value of jitter for this new window and updated values in the arrays z_{max} and z_{min} for this new window. The difference between `JITRNB` and `JITRND` is in whether the calculation of the maximum and minimum of y over the central group of short input windows is done in a brute force fashion or by the more sophisticated algorithm. Once again, the choice of which module to use is determined by comparing the estimated timings based on empirical formulas.

Chapter 15

Data Reduction

Plotting and printing time histories from a PLATSIM simulation can place a heavy burden on computational resources. These time histories are normally a hundred thousand or even a million points long. Plotting a time history of 100000 points produces an EPS file of about half a megabyte, which takes a typical PostScript printer about 9.5 min. to print. The EOS-AM-1 study produced 24 output time histories that typically contained a million points for each of the 59 disturbance events. The burden this amount places on disk space and printer time makes it virtually impossible to produce hardcopies of all these plots.

The goal is to use MATLAB to reduce and plot time histories calculated by the linear time simulation and make hard copies in a reasonable amount of time with a reasonable amount of disk storage. One solution is to plot fewer points by sampling. This solution is fine as long as the plot is smooth, but the plot looses visual information when a lot of jitter occurs in the signal. In the worst case, major jitter components can be aliased out. At the very least, fine detail of the jitter envelope is lost.

The solution presented here is based on physical characteristics of the plot device. Typical printers today are raster devices. The printers used in the development of PLATSIM are laser printers with a resolution of 300 dpi (dots per inch). A default size MATLAB plot printed in landscape mode uses an *X*-axis of 8.5 in, which is 2550 pixels long. Plotting 1000001 points over this axis means that 392 or 393 points are plotted over each pixel of the *X*-axis. Thus, the pen moves vertically up and down about 392 times before it undergoes a motion with a minimal horizontal movement. The idea is to calculate the net effect of all that vertical movement and then plot it with a minimum of pen strokes.

The data points to be plotted are divided into `nplpts` (one of the execution control parameters) non-overlapping groups of consecutive points, the groups containing, as nearly as possible, the same number of points. A polygonal path is computed that covers the range of each group vertically at a single representative *x*-coordinate (which is accomplished with a single vertical line segment) together with vertical and sideways positioning moves to transition from one group of data to the next. To make the resulting graph print as fast as possible, logic was introduced to minimize the number of line segments in the plot. When this technique was implemented in an M-file, it involved a lot of if-then-else logic and was slow. Thus, this technique was also implemented as a MEX-file with a speed increase of about 80 times.

The principle module for fast plotting is `trplot4` (in file `trplot4.m` and (optionally) `trplot4.mex*`). This routine is used as follows:

```
>>[a,b]=trplot4(delt,y,nplpts)
```

The vector `y` is assumed to be a time history with `y(i)` occurring at time `(i-1)*delt`. If `nplpts` is set to a number reflecting that pixel count (3600 has been found to work for full page plots in landscape orientation on 300 dpi printers and is the default value of optional variable `nplpts`), then `plot(a,b)` gives nearly the same picture as `plot(t,y)` (where `t=delt*(0:length(y)-1))` with `2*nplpts` ≤ `length(b)` ≤ `4*nplpts-2`.

Appendix A explains how to create the MEX-files from the FORTRAN 77 code distributed with PLATSIM.

Five files support the fast plotting module: `trplot4.for`, `trplot4.f`, `trplot4g.f`, `xn.f`, and `trplot4.m`. The file `trplot4.for` is simply a concatenation of the three `.f` files. The file `trplot4.f` is the main computational module for `trplot4.mex*`. The file `trplot4g.f` is the gateway routine for transforming data between MATLAB format and FORTRAN 77 format. The file `xn.f` contains a utility subroutine. The file `trplot4.m` is a MATLAB M-file function that is logically equivalent to `trplot4.mex*` but much slower. Even if the MEX-file is used, `trplot4.m` is included to provide help.

NASA Langley Research Center
Hampton, VA 23681-0001
June 12, 1995

Appendix A

Installation Instructions

UNIX Systems

PLATSIM is distributed in a UNIX tar file, which may be in compressed format. The PLATSIM code is in directory `platdir`, and the files pertaining to the EOS-AM-1 example are in the subdirectory `platdir/eos_eg`.

If PLATSIM comes in a compressed file (i.e., the last letter of filename is Z), it should be uncompressed with the UNIX utility `uncompress`. The user should then change (`cd`) to the directory which will be the parent of directory `platdir` and untar the tar file. (The command `tar -xf platsim_ tar_file_name` works on a Sun SPARCstation.)

Next, change the directory to `platdir` and edit `makefile` according to the directions in the `makefile` comments. This step is necessary to customize `makefile` to the user's computing environment. The MATLAB MEX-files and file `startup.m` are then created by the UNIX command `make`. If the user does not want to compile MEX-files but still wants the template file `startup.m`, then use the command `make startup`.

Other Systems

For systems other than UNIX, the user can install the PLATSIM files on the system by following the directions included with the distribution medium. The directory `platdir` includes six files with `.for` extensions. Each of these contains the FORTRAN 77 code for the corresponding MEX-file. To obtain MEX-files the user needs to compile each of these files individually with the script `fmex`, which is supplied with the MATLAB. To provide MATLAB with the location of PLATSIM and the user-supplied model, the user needs to edit the `startup.m` file that is distributed with PLATSIM.

Appendix B

Listing of User-Supplied Routines for EOS-AM-1 Example

The following listings are examples of the routines to be supplied to PLATSIM by the user. These examples are based on the EOS-AM-1 spacecraft and are distributed with PLATSIM. The source listing for PLATSIM and its supporting routines are available through NASA's software technology transfer center COSMIC.[3] The user may want to use these examples as templates for writing the user-supplied routines for the platform that the user wishes to analyze.

mkdamp.m

```
 function [d]=mkdamp(omega)
%
% function [d]=mkdamp(omega)
%
% purpose: to assign modal damping ratios
%
% input variables:
%
% omega : vector containing the natural frequencies
%
% output variables:
%
% d : vector of damping ratios
%
%

% Author: Peiman G. Maghami
%         Spacecraft Controls Branch
%         NASA Langley Research Center
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% damping schedule for the EOS-AM-1 Spacecraft
%
% damping ratio = 0.2% for modes with frequency less than 15Hz
% damping ratio = 0.25% for modes with frequency greater than 15Hz
%                 but less than 50Hz
% damping ratio = 0.3% for modes with frequency greater than 50Hz
for i=1:max(size(omega));
%
%
if omega(i)< 30.0*pi; d(i)=0.002;
elseif omega(i)>=30.0*pi&omega(i)<100.0*pi; d(i)=0.0025;
else;d(i)=0.003;end;end;d=d’;
```

---

[3] COSMIC (Computer Software Management and Information Center), 382 E. Broad Street, University of Georgia, Athens, GA 30602.

instdata.m

```
  function [act,mout,pout,instr,window]=instdata
%
% function [act,mout,pout,instr,window]=instdata
%
% purpose: a user-defined routine to provide the grid point numbers, directions
%          distribution/contribution factors, and identification numbers for
%          the spacecraft instruments. It provides names for the prformance
%          outputs, as well as, window time sizes for jitter/stability analysis.
%
%
% output variables:
%
%
% act    : control input information matrix
% mout   : measurement output information matrix
% pout   : performance output information matrix
% instr  : list of names for performance outputs
% window : list of time window sizes for jitter analysis
%
%
%

% Author: Peiman G. Maghami
%         Spacecraft Controls Branch
%         NASA Langley Research Center
%         August, 1993
%
% modified: P. G. Maghami
%           March, 1994
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% The following parameters are associated with EOS-AM-1 Spacecraft.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% define the spacecraft control input information matrix (the ACS input at
% the RWA)
%
act=[155003,155003,155003;
     4,     5,     6;
     1,     2,     3;
     1.0,   1.0,   1.0];
%
% define grid points for the measurement feedbacks at the NAVBASE
%
mout=[111091,111091,111091,111091,111091,111091;
      4,     4,    5,     5,     6,     6;
      1,     2,    3,     4,     5,     6;
      1.0,   1.0,  1.0,   1.0,   1.0,    1.0;
      0,     1,    0,     1,     0,      1];
%
% define grid points for performance outputs
```

```
% [NAVBASE, CERES1, CERES2, MISR, MODIS-N, MOPITT, SWIR, TIR, VNIR]
%
pout=[111091,111091,111091,350420,350420,350420,...
       351420,351420,351420,333498,333498,333498,361203,361203,361203,...
       396400,396400,396400,329722,329722,329722,326989,326989,326989,...
       325647,325647,325647;
       4,5,6,4,5,6,4,5,6,4,5,6,4,5,6,4,5,6,4,5,6,4,5,6,4,5,6;
       1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27;
       1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
       0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
%
% convert the performance output units from rads to arcsec
%
pout(4,:)=(180.0*3600.0/pi)*pout(4,:);
%
%
% define the performance output names, display and unit labels
%
instr=str2mat('1|NAVBASE Roll|NAVBASE|arcsec','2|NAVBASE Pitch|NAVBASE|arcsec');
instr=str2mat(instr,'3|NAVBASE Yaw|NAVBASE|arcsec');
instr=str2mat(instr,'4|CERES1 Roll|CERES|arcsec','5|CERES1 Pitch|CERES|arcsec');
instr=str2mat(instr,'6|CERES1 Yaw|CERES|arcsec');
instr=str2mat(instr,'7|CERES2 Roll|CERES|arcsec','8|CERES2 Pitch|CERES|arcsec');
instr=str2mat(instr,'9|CERES2 Yaw|CERES|arcsec');
instr=str2mat(instr,'10|MISR Roll|MISR|arcsec','11|MISR Pitch|MISR|arcsec');
instr=str2mat(instr,'12|MISR Yaw|MISR|arcsec');
instr=str2mat(instr,'13|MODIS Roll|MODIS|arcsec','14|MODIS Pitch|MODIS|arcsec');
instr=str2mat(instr,'15|MODIS Yaw|MODIS|arcsec');
instr=str2mat(instr,'16|MOPITT Roll|MOPITT|arcsec','17|MOPITT Pitch|MOPITT|
      arcsec');
instr=str2mat(instr,'18|MOPITT Yaw|MOPITT|arcsec');
instr=str2mat(instr,'19|SWIR Roll|ASTER|arcsec','20|SWIR Pitch|ASTER|arcsec');
instr=str2mat(instr,'21|SWIR Yaw|ASTER|arcsec');
instr=str2mat(instr,'22|TIR Roll|ASTER|arcsec','23|TIR Pitch|ASTER|arcsec');
instr=str2mat(instr,'24|TIR Yaw|ASTER|arcsec');
instr=str2mat(instr,'25|VNIR Roll|ASTER|arcsec','26|VNIR Pitch|ASTER|arcsec');
instr=str2mat(instr,'27|VNIR Yaw|ASTER|arcsec');
%
% set window sizes for the jitter analysis
%
window=[1.0,1.8,9.0,55.0,420.,480.0,1000.00];
%
```

distdata.m

```
function [dist,w,dt,cnames,dnames,instdat,mapping]=distdata(casenum)
% function [dist,w,dt,cnames,dnames,instdat,mapping]=distdata(casenum)
%

% Author:  Sean P. Kenny
%          NASA Langley Research Center
%          Spacecraft Controls Branch
% Created: 2/14/94
%
%-----------------------------------------------------------------------
```

```
if nargin<1,
    casenum=0;
end;
%
%
% Individual Instrument Disturbances (IID)
%
% Defines labels for pull-down menus, also labels for jitter tables,
% and time history plots.
%
% An IID GUI menu item can be disabled by including a preceding asterisk
% in the string variable, e.g., s23='* High Gain Antenna', is
% displayed, but cannot be selected with the mouse. This feature
% pertains to GUI mode ONLY ! Batch mode or terminal display allows
% the selection of all entries.
%
%
s1='TIR repoint';
s2='TIR calibrate';
s3='TIR scan';
s4='TIR chopper';
s5='TIR cryocooler LDE';
s6='MODIS scan mirror';
s7='MODIS static imbalance';
s8='MODIS dynamic imbalance';
s9='MOPITT mirror scan';
s10='MOPITT cryocooler LDE';
s11='MOPITT pressure modulated cells';
s12='Reaction Wheel Assembly case 1';
s13='* Reaction Wheel Assembly case 2';
s14='Solar Array Drive';
s15='Solar Array Thermal Snap';
%
%
% Combine IID's into a matrix form (a maximum of 10 arguments are allowed)
% Each string variable represents a row entry within the dnames string matrix.
dnames=str2mat(s1,s2,s3,s4,s5,s6,s7,s8,s9,s10);
dnames=str2mat(dnames,s11,s12,s13,s14,s15);
%
%
%-----------------------------------------------------------------------
%
%    Major Instrument Disturbance Case (MIDC)
% (labels for top-level menu items on pop-up figure)
%
ss1='ASTER-TIR';
ss2='MODIS';
ss3='MOPITT';
ss4='Misc. Disturbances';
%
% Combine MIDC's into a matrix form
cnames=str2mat(ss1,ss2,ss3,ss4);
%
%
% Setup mapping between MIDC's and IID case numbers.
```

```
%
% instdat(i) corresponds to ss(i), e.g., instdat5 are all IID’s that
% belong to MIDC ss5. The elements of the instdat(i) vectors are the
% row indices of the IID’s within the dnames string matrix. For example,
% if the 4th,5th, and 10th row entries in dnames correspond to instdat5,
% then instdat5=[4,5,10].
%
%
  mapping=[];
instdat1=[1,2,3,4,5];
  linst=length(instdat1);
  mapping=[mapping,linst];
instdat2=[6,7,8];
  linst=length(instdat2);
  mapping=[mapping,linst];
instdat3=[9,10,11];
  linst=length(instdat3);
  mapping=[mapping,linst];
instdat4=[12,13,14,15];
  linst=length(instdat4);
  mapping=[mapping,linst];
%
instdat=[instdat1,instdat2,instdat3,instdat4];
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% Based upon menu selection, create the proper disturbance
% input vector(s).
%
%
%  TIR Mirror Repointing
%
if (casenum == 1)
% 326990 ==> scanner
igrid=[326990];
idir=[4];
inum=[1];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,torque] = tir1;
w(:,1)=torque’;
%
%  TIR Mirror Calibration
%
elseif (casenum == 2)
% 326944 ==> chopper
igrid=[326990];
idir=[4];
inum=[1];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,torque] = tir2;
w(:,1)=torque’;
%
```

```
%  TIR Scanner
%
elseif (casenum == 3)
% 326990 ==> scanner
igrid=[326990 326990 326990 326990 326990 326990];
idir=[1 2 3 4 5 6];
inum=[1:6];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,torque] = tirscan1;
w(:,1)=torque’;
[dt,torque] = tirscan2;
w(:,2)=torque’;
[dt,torque] = tirscan3;
w(:,3)=torque’;
[dt,torque] = tirscan4;
w(:,4)=torque’;
[dt,torque] = tirscan5;
w(:,5)=torque’;
[dt,torque] = tirscan6;
w(:,6)=torque’;
%
%  TIR Chopper
%
elseif (casenum == 4)
% 326944 ==> chopper
igrid=[326944 326944];
idir=[1 2];
inum=[1:2];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,torque] = tirchop1;
w(:,1)=torque’;
[dt,torque] = tirchop2;
w(:,2)=torque’;
%
%  TIR Compressor/Displacer Low Distortion Electronics (LDE)
%
elseif (casenum == 5)
% 326992 ==> compressor
% 326993 ==> displacer
igrid=[326992 326992 326992 326992 326992 326993 326993 326993 326993 326993];
idir=[1 2 4 5 6 2 1 4 5 6];
inum=[1:10];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,torque] = tirca1;
w(:,1)=torque’;
[dt,torque] = tirca2;
w(:,2)=torque’;
[dt,torque] = tirca3;
w(:,3)=torque’;
[dt,torque] = tirca4;
w(:,4)=torque’;
[dt,torque] = tirca5;
```

```
w(:,5)=torque’;
[dt,torque] = tirda1;
w(:,6)=torque’;
[dt,torque] = tirda2;
w(:,7)=torque’;
[dt,torque] = tirda3;
w(:,8)=torque’;
[dt,torque] = tirda4;
w(:,9)=torque’;
[dt,torque] = tirda5;
w(:,10)=torque’;
%
%
%  MODIS scan mirror
%
elseif (casenum == 6)
% 3601 ==> averaged interface
% 361203 ==> scan mirror center
% 361342 ==> scan mirror motor/encoder
% 355349 ==> solar door 8/12/93 spk
igrid=[361342];
idir=[4];
inum=[1];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,torque] = modis1;
w(:,1)=torque’;
%
%  MODIS static imbalance
%
elseif (casenum == 7)
% 361342 ==> scan mirror motor/encoder
igrid=[361342 361342];
idir=[2 3];
inum=[1:2];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,for1,for2] = modis2;
w(:,1)=for1’;
w(:,2)=for2’;
%
%  MODIS dynamic imbalance
%
elseif (casenum == 8)
% 361342 ==> scan mirror motor/encoder
igrid=[361342 361342];
idir=[5 6];
inum=[1:2];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,tor1,tor2] = modis3;
w(:,1)=tor1’;
w(:,2)=tor2’;
%
%  MOPITT mirror scan
```

```
%
elseif (casenum == 9)
% 396400 ==> scan motor 1
% 396403 ==> scan motor 2
% 3608 ==> Avg. interface
igrid=[396400];
idir=[4];
inum=[1];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,torque] = mopitt;
w(:,1)=torque’;
%
%  MOPITT Compressor/Displacer Low Distortion Electronics (LDE)
%
elseif (casenum == 10)
% 396416 ==> compressor
% 396417 ==> displacer
igrid=[396416 396416 396416 396417 396417 396417];
idir=[2 1 5 2 1 5];
inum=[1:6];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,torque] = mopittc1;
w(:,1)=torque’;
[dt,torque] = mopittc2;
w(:,2)=torque’;
[dt,torque] = mopittc3;
w(:,3)=torque’;
[dt,torque] = mopittd1;
w(:,4)=torque’;
[dt,torque] = mopittd2;
w(:,5)=torque’;
[dt,torque] = mopittd3;
w(:,6)=torque’;
%
%  MOPITT pressure modulated cell (PMC)
%
elseif (casenum == 11)
% 396412 ==> PMC #1
% 396413 ==> PMC #2
igrid=[396412 396413];
idir=[1 1];
inum=[1:2];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,pmc1,pmc2] = mopmc;
w(:,1)=pmc1’;
w(:,2)=pmc2’;
%
%
%  Reaction Wheel Assembly (RWA) Imbalance: Case 1
%
% ** static imbalance in wheel #1 **
%
```

```
elseif (casenum == 12)
% 50600 ==> RWA averaged interface
% 155003 ==> RWA 1 (farthest from C.G.)
igrid=[155003 155003 155003];
idir=[1 2 3];
inum=[1:3];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,rwax,rway,rwaz] = rwa1;
w(:,1)=rwax’;
w(:,2)=rway’;
w(:,3)=rwaz’;
%
% Reaction Wheel Assembly (RWA) Imbalance: Case 2
%
elseif (casenum == 13)
% ==>
%igrid=[        ];
%idir=[1 2 3];
%inum=[1:3];
%ifac=ones(size(inum));
%dist=[igrid;idir;inum;ifac];
disp(’THIS CASE IS NOT CURRENT as of 8/11/93 SPK’);
%[dt,rwax,rway,rwaz] = rwa2;
%w(:,1)=rwax’;
%w(:,2)=rway’;
%w(:,3)=rwaz’;
%
% Solar Array Harmonic Drive
%
elseif (casenum == 14)
% 69090 ==> solar array drive (SAD)
igrid=[69090];
idir=[5];
inum=[1];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,torque] = sadhd;
w(:,1)=torque’;
%
% Solar Array Thermal Snap
%
elseif (casenum == 15)
% 69090 ==> SAD
% 60024 ==> SA
igrid=[69090 60024 60024];
idir=[4 1 3];
inum=[1:3];
ifac=ones(size(inum));
dist=[igrid;idir;inum;ifac];
[dt,torque] = sa1;
w(:,1)=torque’;
[dt,torque] = sa2;
w(:,2)=torque’;
[dt,torque] = sa3;
```

```
w(:,3)=torque';
%
end
%
% End distdata.m
```

formscs.m

```
  function [aacs,bacs,cacs]=formscs

%
% function [aacs,bacs,cacs]=formscs
%
% Purpose: To Form the continuous-time spacecraft control system (SCS)
%          for the space platform
%
%          Currently, this function forms the attitude control system
%          for the EOS-AM-1 Spacecraft.
%
%
% output variables:
%
% aacs : the SCS state matrix (continuous)
% bacs : the SCS input influence matrix (continuous)
% cacs : the SCS output influence matrix
%

% Author: P. G. Maghami
%         Spacecraft Controls Branch
%         NASA Langley Research Center
%         December, 1992
%
% Modified: March, 1994
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% third-order Butterworth filter
kf=0.25005;
af=1.0;
bf=1.2600;
cf=0.7938;
df=kf;
numf=kf;denf=[af,bf,cf,df];
% wide-band notch filter
an=0.52335;
bn=0.19678;
cn=1.0;
kn=0.57408;
numn=[an,bn,cn];denn=[an,kn,cn];
% rate gyro
kg=1.0;
wg=12.5664;
zg=0.7071;
numg=[kg*wg*wg];deng=[1.0,2.0*zg*wg,wg*wg];
% zero order hold
taw=0.512;
% numh=taw;denh=[0.125*taw*taw,taw,1.00];
```

```
numh=1.00;denh=[0.125*taw*taw,taw,1.00];
% delay
numd=[0.125*taw*taw,-taw,1.00];dend=[0.125*taw*taw,taw,1.00];
%
krp=[4284.6;6696.4;8322.5];kri=[30.604;40.761;105.68];
%
% rate loop compensator-roll
numrr=[krp(1),kri(1)];denrr=[1.00,0.00];
%
% rate loop compensator-pitch
numrp=[krp(2),kri(2)];denrp=[1.00,0.00];
%
% rate loop compensator-yaw
numry=[krp(3),kri(3)];denry=[1.00,0.00];
%
kp=[0.051604;0.049064;0.047159];ki=[1.5199e-4;1.6984e-4;3.3011e-4];
%
% position loop compensator-roll
numpr=[kp(1),ki(1)];denpr=[1.00,0.00];
%
% position loop compensator-pitch
numpp=[kp(2),ki(2)];denpp=[1.00,0.00];
%
% position loop compensator-yaw
numpy=[kp(3),ki(3)];denpy=[1.00,0.00];
%
% rate loop total compensation
%
%
% transform (numg,deng) to state-space
%
[a1,b1,c1,d1]=tf2ss(numg,deng);
%
%
% tranform the position-loop combined TFs to state-space
%
[ap1,bp1,cp1,dp1]=tf2ss(numpr,denpr);
[ap2,bp2,cp2,dp2]=tf2ss(numpp,denpp);
[ap3,bp3,cp3,dp3]=tf2ss(numpy,denpy);
%
%
% combine the position loop and rate loop filters
%
[ac1,bc1,cc1,dc1]=append(ap1,bp1,cp1,dp1,a1,b1,c1,d1);
[ac2,bc2,cc2,dc2]=append(ap2,bp2,cp2,dp2,a1,b1,c1,d1);
[ac3,bc3,cc3,dc3]=append(ap3,bp3,cp3,dp3,a1,b1,c1,d1);
%
cc1=cc1(1,:)+cc1(2,:);dc1=dc1(1,:)+dc1(2,:);
cc2=cc2(1,:)+cc2(2,:);dc2=dc2(1,:)+dc2(2,:);
cc3=cc3(1,:)+cc3(2,:);dc3=dc3(1,:)+dc3(2,:);
%
% combine the butterworth filter and the notch filter
%
[num1,den1]=series(numf,denf,numn,denn);
%
```

```
% combine the rate loop compensator in series with the notch filter
%
[num31,den31]=series(numrr,denrr,num1,den1);
[num32,den32]=series(numrp,denrp,num1,den1);
[num33,den33]=series(numry,denry,num1,den1);
%
% add the zero order hold in series
%
[num41,den41]=series(num31,den31,numh,denh);
[num42,den42]=series(num32,den32,numh,denh);
[num43,den43]=series(num33,den33,numh,denh);
%
% add the time delay in series
%
[numinr,deninr]=series(num41,den41,numd,dend);
[numinp,deninp]=series(num42,den42,numd,dend);
[numiny,deniny]=series(num43,den43,numd,dend);
%
% transform the rate loop TFs to state-space
%
[ar1,br1,cr1,dr1]=tf2ss(numinr,deninr);
[ar2,br2,cr2,dr2]=tf2ss(numinp,deninp);
[ar3,br3,cr3,dr3]=tf2ss(numiny,deniny);
%
%
% Form the SCS state-space model
%
%
[aacs1,bacs1,cacs1,dacs1]=series(ac1,bc1,cc1,dc1,ar1,br1,cr1,dr1);
[aacs2,bacs2,cacs2,dacs2]=series(ac2,bc2,cc2,dc2,ar2,br2,cr2,dr2);
[aacs3,bacs3,cacs3,dacs3]=series(ac3,bc3,cc3,dc3,ar3,br3,cr3,dr3);
%
[aacs,bacs,cacs,dacs]=append(aacs1,bacs1,cacs1,dacs1,aacs2,bacs2,cacs2,dacs2);
[aacs,bacs,cacs,dacs]=append(aacs,bacs,cacs,dacs,aacs3,bacs3,cacs3,dacs3);
%
% apply the scale factor KW of the reaction wheels (units in lb-in/ssc)
%
kw=0.22*12.00;
cacs=cacs*kw;
dacs=dacs*kw;
%
```

Appendix C

Execution Control Parameters

PLATSIM has 20 execution control parameters, which are MATLAB workspace variables. Some of these parameters have meaning to both time and frequency domain analysis, some only to time domain, and some only to frequency domain. With two exceptions, if the user has not defined an execution control parameter when the command `platsim` is given, PLATSIM assigns a default value to the variable. One of these exceptions, `runmode`, must be set before PLATSIM is called. The other exception is `casenum`. If PLATSIM is running in GUI or command mode, `casenum` is set by user responses to the Disturbances menu. In batch mode, `casenum` must be set before PLATSIM is called. Specifications for the individual execution control parameters follow:

Execution Control Parameters

Name	Default	Used by	Description
`runmode`	Undefined	Both	Determines what mode PLATSIM runs in. Must be set to one of `'GUI'`, `'command'`, or `'batch'` before the `platsim` command is entered. Only the first character of `runmode` is significant and may be of either case.
`tdflag`	Yes	Both	Defaults to time domain analysis. Set `tdflag = 'no'` for frequency domain analysis. Can be set from GUI Analysis menu and command mode.
`casenum`	None	Both	Which disturbance to use. PLATSIM inputs this value to the user-supplied MATLAB function `distdata`. Can be set from GUI and command mode Disturbance menu.
`nmode`	0	Both	Which structural modes to model. The default `0` means all modes. A positive integer `n` means modes 1 through *n*. A negative integer `-n` means only mode *n*. A MATLAB vector like `[1:6,10,13:16]` means use exactly the mode numbers in the vector. Can be set from GUI mode by slider or data entry box and from command mode.
`clflag`	Yes	Both	Defaults to closed loop analysis. For open loop analysis, set `clflag='no'`. Can be set from GUI mode Control submenu of Options menu and from command mode.
`desint`	Undefined	Both	Determines which performance outputs will be modeled. If left undefined, all will be used. To use, set `desint` to a vector of instrument identification numbers (these appear in the third row of the `pout` matrix returned by user-supplied MATLAB function `instdata`). Can be set from GUI mode Performance Output submenu of Options menu. Note 1 applies.
`pltflag`	Yes	Both	In time domain analysis, causes reduction of time history data for plotting and writing of MAT-file with reduced

			time histories. In both analyses, causes plots (time history or Bode) to be displayed on screen. Disable by setting `pltflag='no'`. Can be set from GUI mode Plotting & Printing submenu of Options menu and from command mode.
`prtflag`	Yes	Both	If `pltflag` is set to yes, this causes the encapsulated PostScript forms of the plots to be written to files. Disable by setting `prtflag='no'`. Can be set from GUI mode Plotting & Printing submenu of Options menu and from command mode.
`gad`	Identity	Both	This is used for adjusting the spacecraft control system inputs. Parameter `gad` should be defined as an $m \times m$ matrix, where m is the number of spacecraft control system inputs in the vector u. The scaled control inputs are given by $u_{\text{new}} = \texttt{gad} \times u_{\text{old}}$. Note that the default matrix for `gad` is an identity matrix. This is actually implemented by adjusting the spacecraft control system output influence and feedthrough matrices by `gad`. For open-loop analysis, this parameter is ignored. This option is usually used when inertia changes require gain adjustments for the spacecraft system. Notes 1 and 2 apply.
`phold`	20	Both	A mnenmonic for plot hold, `phold` is the number of seconds a time history or Bode plot or jitter analysis table will remain on screen before being cleared for the next plot. Notes 1 and 2 apply.
`jtrflag`	Yes	Time	Defaults to perform jitter analysis on time histories. Disable by setting `jtrflag='no'`. Can be set from GUI mode Options menu. Note 1 applies.
`multflag`	No	Time	Defaults to run a separate time simulation for each event in the disturbance scenario. To simulate the effect of all disturbances simultaneously, set `multflag='yes'`. Can be set from GUI mode Disturbance menu. Note 1 applies.
`saveflag`	No	Time	Defaults to not save full time histories in MAT-files. To save full time histories, set `saveflag='yes'`. Can be set from GUI mode Options menu. Note 1 applies.
`tclip`	0	Time	The parameter `tclip` is used for clipping the time histories. If the value entered is not 0 (the default), any data point corresponding to a time before `tclip` will be removed, that is, not used for jitter computation. Units of `tclip` must match the units of parameter `period` returned by user-supplied MATLAB function `distdata`. This option is useful in the jitter analysis of steady-state disturbance sequences such as the EOS-AM-1 solar array drive disturbance and cryocooler disturbances. May be set from GUI mode slider or data entry box. Note 1 applies.
`pmflag`	Yes	Time	Defaults to display during simulation a performance meter that shows what percent of the simulation calculation is completed. Can be set from GUI mode Analysis

			menu. To turn off the performance meter in command or batch modes, set `pmflag='no'`. Note 1 applies.
`nplpts`	3600	Time	The parameter `nplpts` is used in reducing the time histories for plotting. Refer to chapter 15 for a more complete description of this parameter. Notes 1 and 2 apply.
`ndist`	0	Frequency	This parameter determines which events of the disturbance scenario to use in calculating the frequency response function matrix. The default `0` means all events. A positive integer `n` means events 1 through n. A negative integer `-n` means only event n. A MATLAB vector like `[1:2,4,7]` means use exactly the event numbers in the vector. Can be set from GUI mode by slider or data entry box and from command mode.
`il`	–2	Frequency	The smallest frequency at which frequency domain analysis will be done is 10^{il}. May be a decimal. Can be set from GUI mode in a data entry box in the Frequency Parameters window, which opens when Frequency Domain Analysis is selected and from command mode.
`iu`	4	Frequency	The largest frequency at which frequency domain analysis will be done is 10^{iu}. May be a decimal. Can be set like `il`.
`npts`	1000	Frequency	The number of frequency points at which frequency domain analysis will be done is `npts`. Can be set like `il`.

Note 1: To run PLATSIM in command mode with a nondefault value for this parameter, it must be set by MATLAB assignment statement prior to invoking `platsim`.

Note 2: To run PLATSIM in GUI mode with a nondefault value for this parameter, it must be set by MATLAB assignment statement prior to invoking `platsim`.

References

1. Belvin, W. Keith; Maghami, Peiman; Tanner, Cheri; Kenny, Sean; and Cooley, Vic: Evaluation of CSI Enhancements for Jitter Reduction on the EOS AM-1 Observatory. *Dynamics and Control of Large Structures—Proceedings of the Ninth VPI & SU Symposium*, L. Meirovitch, ed., May 1993, pp. 255–266.

2. MATLAB® Reference Guide—High-Performance Numeric Computation and Visualization Software. The MathWorks, Inc., 1992.

3. Asrar, Ghassem; and Dokken, David Jon, eds.: *1993 Earth Observing System Reference Handbook.* NASA NP-202, 1993.

4. Marcus, Marvin: *Basic Theorems in Matrix Theory.* Nat. Bur. Stand., Appl. Math. Ser. 57, U.S. Dep. of Commerce, Jan. 22, 1960. (Reprinted 1964.)

5. Lawson, C. L.; Hanson, R. J.; Kincaid, D. R.; and Krogh, F. T.: Basic Linear Algebra Subprograms for Fortran Usage. *ACM Trans. Math. Softw.*, vol. 5, no. 3, Sept. 1979, pp. 308–323.

6. Lawson, C. L.; Hanson, R. J.; Kincaid, D. R.; and Krogh, F. T.: Algorithm 539—Basic Linear Algebra Subprograms for Fortran Usage [F1]. *ACM Trans. Math. Softw.*, vol. 5, no. 3, Sept. 1979, pp. 324–325.

7. Dongarra, J. J.; Moler, C. B.; Bunch, J. R.; and Stewart, G. W.: *LINPACK Users' Guide.* SIAM, 1979.

8. Dongarra, Jack J.; Du Croz, Jeremy; Hammarling, Sven; and Hanson, Richard J.: An Extended Set of FORTRAN Basic Linear Algebra Subprograms. *ACM Trans. Math. Softw.*, vol. 14, no. 1, Mar. 1988, pp. 1–17.

9. Dongarra, Jack J.; Du Croz, Jeremy; Hammarling, Sven; and Hanson, Richard J.: Algorithm 656—An Extended Set of Basic Linear Algebra Subprograms: Model Implementation and Test Programs. *ACM Trans. Math. Softw.*, vol. 14, no. 1, Mar. 1988, pp. 18–32.

10. Dongarra, Jack J.; Du Croz, Jeremy; Hammarling, Sven; and Duff, Iain: A Set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Softw.*, vol. 16, no. 1, Mar. 1990, pp. 1–17.

11. Dongarra, Jack J.; Du Croz, Jeremy; Hammarling, Sven; and Duff, Iain: Algorithm 679—A Set of Level 3 Basic Linear Algebra Subprograms: Model Implementation and Test Programs. *ACM Trans. Math. Softw.*, vol. 16, no. 1, Mar. 1990, pp. 18–28.

12. Anderson, E.; Bai, Z.; Bischof, C.; Demmel, J.; Dongarra, J.; Du Croz, J.; Greenbaum, A.; Hammarling, S.; McKenney, A.; Ostrouchov, S.; and Sorensen, D.: *LAPACK Users' Guide*. SIAM, 1992.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
September 1995

3. REPORT TYPE AND DATES COVERED
Technical Paper

4. TITLE AND SUBTITLE
PLATSIM: An Efficient Linear Simulation and Analysis Package for Large-Order Flexible Systems

5. FUNDING NUMBERS
WU 233-01-01-05

6. AUTHOR(S)
Peiman G. Maghami, Sean P. Kenny, and Daniel P. Giesy

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER
L-17437

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TP-3519

11. SUPPLEMENTARY NOTES
Maghami and Kenny: Langley Research Center, Hampton, VA; Giesy: Lockheed Martin Engineering & Sciences, Hampton, VA.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited
Subject Category 18
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

PLATSIM is a software package designed to provide efficient time and frequency domain analysis of large-order generic space platforms implemented with any linear time-invariant control system. Time domain analysis provides simulations of the overall spacecraft response levels due to either onboard or external disturbances. The time domain results can then be processed by the jitter analysis module to assess the spacecraft's pointing performance in a computationally efficient manner. The resulting jitter analysis algorithms have produced an increase in speed of several orders of magnitude over the brute force approach of sweeping minima and maxima. Frequency domain analysis produces frequency response functions for uncontrolled and controlled platform configurations. The latter represents an enabling technology for large-order flexible systems. PLATSIM uses a sparse matrix formulation for the spacecraft dynamics model which makes both the time and frequency domain operations quite efficient, particularly when a large number of modes are required to capture the true dynamics of the spacecraft. The package is written in MATLAB script language. A graphical user interface (GUI) is included in the PLATSIM software package. This GUI uses MATLAB's Handle graphics to provide a convenient way for setting simulation and analysis parameters.

14. SUBJECT TERMS
PLATSIM; Frequency domain analysis; Time domain analysis; Jitter analysis; Flexible structures; Large-order systems; Controlled systems

15. NUMBER OF PAGES
74

16. PRICE CODE
A04

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT