44th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, and Materials Confere
7-10 April 2003, Norfolk, Virginia

AIAA 2003-1532

AIAA-

Fast Time Domain Simulation for Large Order Linear Time-Invariant Systems

Kin Cheong Sou* and Olivier L. de Weck†
Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA

Time domain simulation is an essential technique in multidisciplinary design optimization (MDO). Unfortunately it can be time consuming and cause memory saturation problems when systems get large, sampling rates are high and input time histories are long. In this paper, an efficient simulation scheme is presented to simulate large order continuous-time linear time-invariant (LTI) systems. The A matrix (assumed to be block-diagonalizable) of the system is first diagonalized. Then, subsystems of manageable dimensions and bandwidth are formed and multiple sampling rates can be chosen to associate with the subsystems. Each subsystem is then discretized using a O(n_s) discretization scheme, where n_s is the number of state variables. Next, a sparse matrix recognizable O(n_s) discrete-time system solver (i.e. matrix-vector product solver) is employed to compute the history of the state and the output. Finally, the response of the original system is obtained by superposition and interpolation of the subsystem responses. In practical engineering applications, closing feedback loops, cascading filters (shaping filters or other feedforward compensators) and other structures can hinder the efficient use of the simulation scheme (e.g. by destroying the block diagonal structure of the A matrix). Solutions to these problems are also addressed in the latter part of the paper. The simulation scheme, implemented as a MATLAB script `newlsim.m`, is compared with the established LTI system simulator `lsim.m` in MATLAB and is shown to be superior in the case of medium and large order systems. The 2184 state variable Space Interferometry Mission (SIM v2.2) simulation is enabled with the proposed technique (standard simulation fails due to excessive memory requirements) and a computer time savings factor of ≈ 50 is demonstrated without loss of accuracy. Aside from handling realistic applications the simulator brings time domain simulation on par with frequency domain and Lyapunov analyses, while allowing transient response computation.

Nomenclature

A, B, C, D	=	State space matrices
BZ	=	Block size of a subsystem
$\mathbb{C}$	=	The set of all complex numbers
DSF	=	Downsampling factor
EM	=	Ending mode
I	=	Identity matrix
$\mathbb{R}$	=	The set of all real numbers
SM	=	Starting mode
SS	=	Subsystem
TM	=	Total number of modes
T	=	CPU time
T	=	Sampling time period
$\mathbb{Z}$	=	The set of all integers
m	=	Number of input channels
n	=	Number of samples
n_s	=	Number of state variables
p	=	Number of output channels
u	=	Control input
v	=	Discrete-time signal
w	=	Disturbance input
x	=	State vector
y	=	Measurement output
z	=	Performance output
σ	=	Root-mean-square
θ	=	Attitude angles
$\cdot/$	=	Matrix entry-wise division

Subscripts

d	=	Discrete-time / Disturbance
i	=	i-th subsystem
k	=	Controller
o	=	Optical controller
p	=	Plant
$\underline{(\cdot)}$	=	Lift operator

Superscripts

-1	=	Matrix inversion
T	=	Matrix transpose

*Graduate Research Assistant, Space Systems Laboratory.
†Assistant Professor, Space Systems Laboratory, Department of Aeronautics and Astronautics, Engineering Systems Division, Member AIAA, email: deweck@mit.edu

1 Introduction

SIMULATION is an essential tool for the design of complex engineering systems, as the models of these systems become more and more complex. Fig. 1 shows some example systems and their complexity in terms of model dynamic bandwidth (normalized) and size. We have previously discussed the need for efficiency in large system time domain simulation and rationalized the distinction between large systems ($n_s \geq 500$), medium systems ($100 < n_s < 500$) and small systems ($n_s \leq 100$).[1]

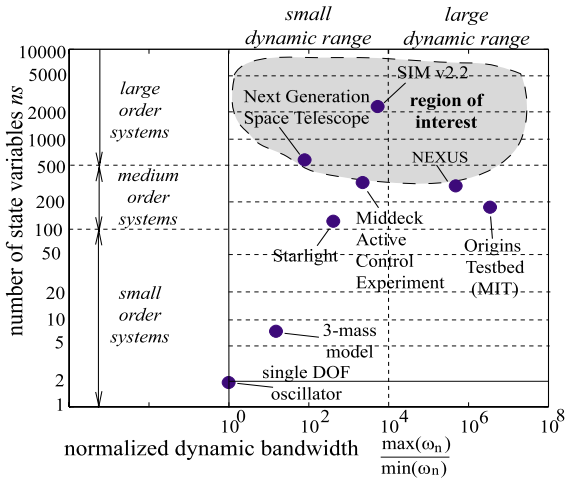

Fig. 1 Examples of size and dynamic bandwidth for LTI models of various space systems. The x-axis is the ratio between the highest and lowest natural frequencies in the model. The y-axis is the number of state variables of the model. The highlighted region motivates the presented simulation scheme.

Model dimension (or size/order) is a delicate issue in systems design because of the tradeoff between model fidelity and the number of designs explored within a limited time budget, $T_{tot} = N \times T_{cpu}$, where N is the number of simulations or designs explored and T_{cpu} is the runtime of a single simulation. The dilemma is shown in Fig. 2. It is clear from Fig. 2 that

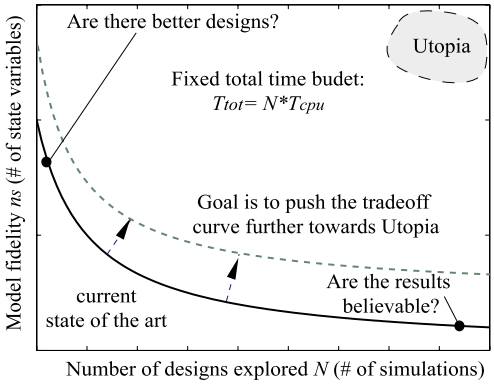

Fig. 2 Notional tradeoff curve between number of designs explored and model fidelity. The solid line is the curve allowed by current techniques, based on a $T_{cpu} \propto n_s^3$ scaling relationship, see de Weck et al.[1] The dashed line is the objective of this work. The upper right corner is the utopia point.

the more efficient a simulation is for a given level of model fidelity, the better a position system designers find themselves in. In addition, efficient simulation can facilitate the performance evaluation step of multidisciplinary design optimization (MDO), whose prospect and significance have been reported by Giesing et al.,[2] Sobieski et al.[3] and Anderson[4] among others. In Gutierrez,[5] de Weck et al.[1] and de Weck,[6] three methods for performance evaluation of dynamic systems are summarized: 1) Time domain simulation, 2) Frequency domain analysis and 3) Lyapunov analysis. In these papers, improvements for frequency domain and Lyapunov methods were proposed with the derivation of new algorithms such as `newlyap.m` and modified internally balanced model reduction (see also Mallory[7]) with apriori error bounds, but the problem of time domain simulation remained unsolved. While frequency domain analysis and Lyapunov analysis can provide critical performance metrics such as steady state root-mean-square (RMS) values of performances, they cannot provide information on the transient response of the systems, which is sometimes required (e.g. in designing control systems). Also, while model reduction can result in smaller systems with acceptable accuracy, it suffers from the fact that arbitrary decisions must be made as to the amount of reduction and the balancing process itself requires substantial resources in computer time and memory, even though there are exciting improvements in this area, see for example, Willcox et al.[8] and Beran.[9] The problem just mentioned can be relieved without parallelization, if not entirely solved, by the capability of efficient time domain simulation of large order systems, which is the topic of this paper.

The proposed simulation algorithm, `newlsim.m`, first decouples the original dynamical system by implementing a block diagonalization[1] of the A matrix, it then forms fictitious subsystems with lower, and thus manageable dimensions and narrower bandwidths. After that, the subsystems are discretized so that efficient computation of state transition can be realized. Finally the responses of the subsystems are interpolated and superposed to form the response of the original large-order system.

The organization of the paper is as follows: In Section 2 the technical time domain simulation problem will be discussed. Then in Section 3 the flow chart and some important details of the operations are presented. After describing the algorithm, Section 4 studies the simulation problems and solutions with various kinds of control loops. In Section 5 simulation results applied to a high fidelity Space Interferometry Mission (SIM) model, version 2.2, are given and some of these results are compared with those obtained by the standard MATLAB LTI systems simulator, `lsim.m`, as well as other techniques. A summary and conclusions are discussed in Section 6.

[1]For a diagonalizable matrix, the transformed matrix can be strictly diagonal but the result is usually complex, which is not very useful in practice. However, for a real diagonalizable matrix, the complex diagonal matrix can be converted into a real block diagonal one. Also, whenever the word "diagonal" is used in this paper, it means real block diagonal, unless noted otherwise.

2 Time Domain Simulation Problem

The time domain simulation problem of an LTI system can be defined as follows: Given a system in (1)

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{1}$$

where $x(t)$ is the state, $u(t)$ is the input, $y(t)$ is the output and the matrices are of appropriate dimensions. The solution to this problem is the unique $y(t)$, given an external input $u(t)$ and initial conditions $x(t_0)$. There are at least two ways to solve the problem: 1) Standard ordinary differential equation (ODE) solvers like Runge Kutta, and 2) The state transition formula in (2)

$$x(t) = \underbrace{e^{A(t-t_0)}x(t_0)}_{x_H} + \underbrace{\int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau}_{x_p}, \tag{2}$$

where t_0 denotes the initial time instant, τ is a dummy variable, $e^{A(t-\tau)}$ is the matrix exponential of the matrix $A(t-\tau)$ and $u(t)$ is the external input to the system, x_H and x_P denote the homogenous and particular solutions, respectively. The state transition method just mentioned is equivalent to transforming the original problem (1) into its discretized version in (3), provided that a further assumption is made on $u(t)$ (e.g. zero order hold (ZOH)).

$$\begin{aligned} &x[n+1] = A_d x[n] + B_d u[n] \\ &y[n] = Cx[n] + Du[n], \\ &\text{where,} \\ &A_d = e^{AT}, \\ &B_d = \textstyle\int_{KT}^{(K+1)T} e^{A(KT+T-\tau)}Bd\tau = \int_0^T e^{A\tau}Bd\tau, \\ &\text{and } T \text{ is the sampling period and } n \in \mathbb{Z}. \end{aligned} \tag{3}$$

The state transition method serves the current problem better in that it requires less computation and it maps left half s-plane poles to inside the unit circle in the z-plane, no matter how large the discretization time step, T, is. Nevertheless, the benefits of the state transition method do not come for free in that the following problems must be addressed: The first problem is the computational expense of e^{AT} in (3). The cost of this operation is $O(n_s^3)$ with n_s being the number of state variables.[10] The second problem is memory saturation, if the computation requires the whole history of states before the history of the output can be computed (see `lsim.m` for such an algorithm[11]), in that case the simulation might halt because of memory saturation. The solutions to these problems will be given in Section 3.

3 Fast Time Domain Simulation Algorithm

In this section, the steps and implementation issues of the presented algorithm will be addressed. Before the details are discussed, an overview of the algorithm is given by the flowchart in Fig. 3. In this flow chart,

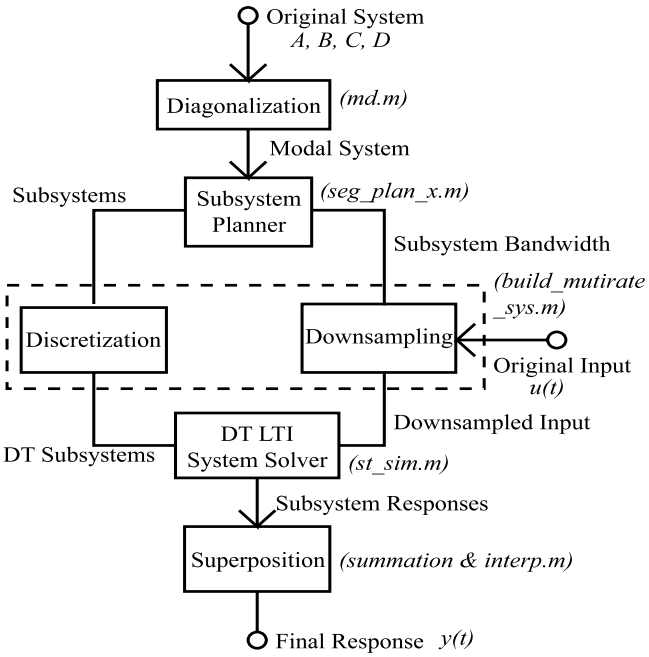

Fig. 3 Flow chart of the fast time domain simulation algorithm, `newlsim.m`.

the circles correspond to data such as the original system A, B, C, D matrices, the external input and the computed response (output). Each block represents an operation or process with the actual MATLAB implementation label next to it (The name of the m-file is in italic font).

Diagonalization

Diagonalization is a similarity transform of the original system into a system that has a block diagonal A matrix. That is,

$$A = S\Lambda S^{-1},$$

where S is an invertible similarity transform matrix and Λ is a real block diagonal matrix. This similarity transform can be an eigenvalue decomposition or state variable reordering, whereby the latter is less expensive. Additionally, the corresponding subroutine of the presented simulation scheme sorts the modes of the diagonalized system in ascending natural frequencies. This diagonalization step is necessary in order to enable the following implementations:

- *Decoupling the dynamics and forming fictitious subsystems.* This can relieve the problem of memory saturation since the original large problem is

divided into smaller subproblems. Also, this gives rise to the potential for parallel computation.

- *Applying multiple sampling rates.* The reason for this implementation is twofold. First, simulation can be facilitated by the application of multiple sampling rates. Secondly and more importantly, this can pave the way to solving multiple time scale dynamics problems, see Reich[12] for such a problem in molecular dynamics.

- *Exploiting the sparsity resulting from the diagonal structure of the A matrix.* The number of nonzero entries of an ordinary dense matrix $A \in \mathbb{R}^{n \times n}$ is n^2, but the number of nonzero entries in the diagonalized matrix is less then or equal to $2 \times n$. This sparsity is important in the matrix-vector product computation to be discussed later.

Subsystem Planning

The subsystem planning subroutine is the key decision element in the algorithm. The objective of this function is to form fictitious subsystems and to assign them appropriate sampling rates. The assumption in this subsection is that the plant is already block diagonal and that the modes are sorted (i.e. after the preceding diagonalization step). There are two considerations for the subsystem planning.

The first issue is the size (number of state variables) of each subsystem. In terms of floating point operations (FLOPS), the size of the subsystems is not very important because the number of FLOPS for simulating discrete-time systems (after diagonalization) depends linearly on the number of state variables. According to Sou[10] the approximate cost is[2],

$$\text{FLOPS} = 2 \times (2 + m + p) \times n_s \times n \qquad (4)$$

where m is the number of input channels, p is the number of output channels, n_s is the number of state variables and n is the number of samples to be processed. This is in contrast to non-sparse continuous time system simulation, where the cost is proportional to the number of states to the third power.[1] Nevertheless, simulating subsystems that are too large and too small is not efficient because of memory saturation and size independent overhead, respectively.

The other issue is the sampling rate for each simulation. The minimum sampling rate is given by Nyquist's sampling theorem (e.g. see Oppenheim[13]) but it is usually insufficient for computer simulations. As a rule of thumb, the sampling rate should be four to ten times the system bandwidth. The issue of appropriate sampling rate selection has been extensively discussed by Franklin[14] and Åström[15] among others.

Taking into account the aforementioned issues, the corresponding subsystem planning subroutine has the following features:

- It automatically chooses an appropriate subsystem block size by considering a lower bound, upper bound and the effect of the number of input and output channels.

- It automatically suggests downsampling based on an estimate of the ratio between the high frequency components and low frequency components of the output.

A flow chart of this subroutine is given in Fig. 4.

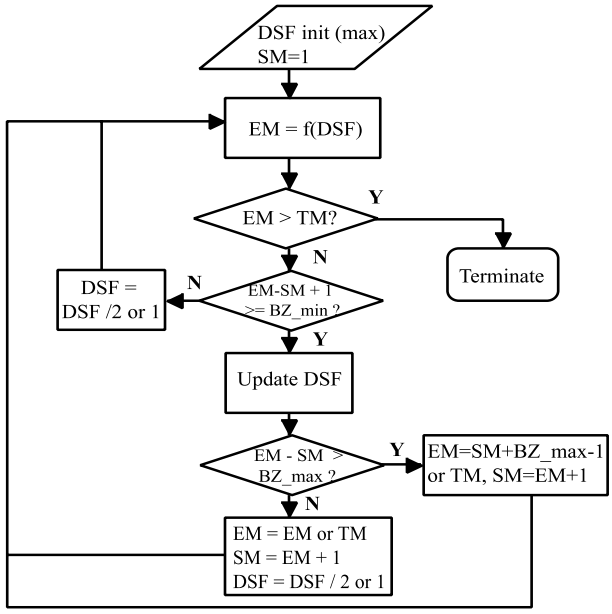

Fig. 4 Flow chart of the subsystem planning subroutine. DSF denotes downsampling factor (the ratio between the original sampling rate and the reduced sampling rate). SM is the starting mode of the subsystem. EM is the ending mode of the subsystem. TM is the total number of modes of the original system. This serves as a termination criterion. BZ is the currently chosen block size of each subsystem.

In this flowchart, the f(DSF) is implemented as a bisection process that determines the ending mode (EM) of each subsystem. As an example, the subsystem planning of the SIM v2.2 flexible mode model was considered with the planning result and downsampling factors given in Fig. 5.

Discretization

As can be seen in (3), the bottleneck of discretization is the matrix exponential. Fortunately the diagonalization described earlier relieves the problem. By exploiting the sparsity of the block diagonal A matrix structure, a $O(n_s)$ matrix exponential algorithm can be realized. This is obviously seen if the matrix exponential A_d in (3) is expressed as follows (see Chen,[16] for example):

$$A_d = e^{AT} = I + AT + \frac{A^2T^2}{2} + \frac{A^3T^3}{3!} + \cdots . \qquad (5)$$

[2]The FLOPS due to feedthrough are ignored.

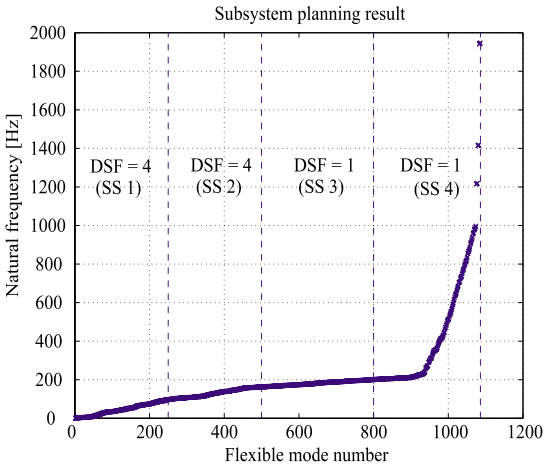

Fig. 5 Subsystem planning result of SIM v2.2 flexible mode model. The original sampling rate is 4096 Hz.

By noticing the fact that

$$A^n T^n = \begin{bmatrix} A_1 T & 0 & \cdots \\ 0 & A_2 T & \ddots \\ \vdots & \ddots & \ddots \end{bmatrix}^n = \begin{bmatrix} A_1^n & 0 & \cdots \\ 0 & A_2^n & \ddots \\ \vdots & \ddots & \ddots \end{bmatrix} T^n, \tag{6}$$

where $A_i \in \mathbb{R}^{2\times 2}\ \forall\, i \in \{1, 2, \ldots\}$ and applying (6) to (5), the following equality holds:

$$\exp\left(\begin{bmatrix} A_1 T & 0 & \cdots \\ 0 & A_2 T & \ddots \\ \vdots & \ddots & \ddots \end{bmatrix} \right) = \begin{bmatrix} e^{A_1 T} & 0 & \cdots \\ 0 & e^{A_2 T} & \ddots \\ \vdots & \ddots & \ddots \end{bmatrix}. \tag{7}$$

Equation (7) is the basis of the proposed $O(n_s)$ discretization scheme. As a result, a generic procedure for the fast discretization can be given as:

```
for i = 1 to total number of subsystems
   index = location of subsystem ;
   DT_system(index) = c2d(subsystem) ;
end
```

Here the size of each subsystem is not fixed and a theoretical optimal block size can be found. Nevertheless, the optimality is not an important issue since the difference between optimal and suboptimal strategies is not obvious here[3].

A simple example as follows should reveal the fact[4]:

```
% create a random diagonal matrix
>> A = diag(randn(1000,1));
% dense matrix exponential
>> tic; Aexp = expm(A); toc
% computation time
elapsed_time = 27.1090
% diagonal matrix exponential
>> tic; Aexp1 = diag(exp(diag(A))); toc
% computation time
elapsed_time = 0.0310
% error checking
>> norm(Aexp - Aexp1)/length(Aexp)^2
ans = 3.7303e-020
```

That is, the computation of the diagonal matrix exponential is 874 times faster than that of the dense matrix, and the difference between the results is negligible. Although in the example just shown, the A matrix is strictly diagonal, it can be concluded that a significant amount of computation is saved even in the case of a block diagonal A matrix, since the comparison is between $O(n^3)$ and $O(n)$ operations. Nevertheless, the advantage of fast discretization comes with the expense of the diagonalization step itself, which is also $O(n^3)$. Fortunately, the computation of eigenvalues is usually more efficient than that of the matrix exponential, as Table 1 shows.

Table 1 Computation time of eigenvalues (`eig`) and matrix exponential (`expm`) in seconds with test matrices of size n. The specifications of the test machine are given in footnote 4.

Size (n)	100	200	400	600	1000
`eig`	0.0310	0.2340	1.9850	6.4530	29.4530
`expm`	0.0310	0.3280	2.6250	9.0620	40.9530

Downsampling

Downsampling the input is required whenever the assigned sampling rate of a particular simulation is lower than that of the input (this is true if the multiple sampling rates strategy is employed). The straightforward way to achieve the goal is to low-pass filter the signal and then downsample it (see Oppenheim, for example[13]). That is,

$$x_d[n] = x[Mn],$$

where $x[n]$ has been low-pass filtered and M is the downsampling factor.

In addition to the direct method just discussed, there is another way to downsampling. Recall the state transition formula (or the state equations for the discretized system),

$$\begin{aligned} x[n+1] &= A_d x[n] + B_d u[n] \\ y[n] &= C x[n] + D u[n]. \end{aligned} \tag{8}$$

The first equation in (8) is certainly satisfied at the $n+1$ instant,

$$x[n+2] = A_d x[n+1] + B_d u[n+1]. \tag{9}$$

[3]This is in contrast to the case of `newlyap.m` discussed in de Weck et al.[1]

[4]The test machine is a Pentium 4 PC with a 1.5GHz CPU and 128MB RAM. Any other computations in this report were performed on the same machine

If (8) is substituted into in (9), then the following results

$$\begin{aligned} x[n+2] &= A_d\{A_d x[n] + B_d u[n]\} + B_d u[n+1] \\ &= A_d^2 x[n] + A_d B_d u[n] + B_d u[n+1]. \end{aligned} \quad (10)$$

Equation (10) can be generalized for N steps,

$$x[n+N] = A_d^N x[n] + A_d^{N-1} B_d u[n] + \cdots + B_d u[n+N-1] \quad (11)$$

If the following new matrices are defined

$$\begin{aligned} \underline{u}[n] &= \left[\begin{array}{ccc} u[n] & \cdots & u[n+N-1] \end{array}\right]^T \\ \underline{A_d} &= A_d^N \\ \underline{B_d} &= \left[\begin{array}{ccc} A_d^{N-1}B_d & \cdots & B_d \end{array}\right] \\ \underline{C} &= C \\ \underline{D} &= \left[\begin{array}{cccc} D & 0 & \cdots & 0 \end{array}\right], \end{aligned}$$

then the N-step propagation version of (8) is

$$\begin{aligned} x[n+N] &= \underline{A_d} x[n] + \underline{B_d}\underline{u}[n] \\ y[n] &= \underline{C} x[n] + \underline{D}\underline{u}[n]. \end{aligned} \quad (12)$$

By employing a long time step state transition, the calculation of the unwanted intermediate states and outputs can be avoided. This downsampling scheme essentially downsamples the output instead of the input and the accuracy is much better than the first direct approach. However, the efficiency gain achieved by the second method is less than that by the first method. Compare the FLOPS for the direct downsampling (13) and those of the second method (14):

$$\text{FLOPS} = 2 \times \frac{(2+m+p) \times n_s \times n}{\text{DSF}}, \quad (13)$$

$$\text{FLOPS} = 2 \times \frac{(2+DSF \times m+p) \times n_s \times n}{DSF}. \quad (14)$$

Here $DSF \geq 1$ is the downsampling factor. The meanings of other parameters in (13) and (14) are referred back to (4). In conclusion, the second method is more conservative. The current version of `newlsim.m` adopts the second method, also referred to as lifting, if downsampling is to be realized.

Simulation, Interpolation and Superposition

With the subsystems formed and discretized, the burden on the simulator (the actual code that computes the states and outputs) is lighter compared to the original ODE problem and it now becomes a much simpler problem of matrix-vector multiplication (cf. (3)). Taking into account the sparsity of the A matrix, the matrix-vector multiplication (state transition) can be realized in $O(n_s n)$ (n_s is the number of state variables and n is the number of samples to be simulated) FLOPS. The required features of the simulator are summarized as follows:

- The simulator must be memory conscious. It cannot request any amount of memory (in bytes) proportional to $n_s n$ or more.
- The simulator must be able to recognize the zero pattern of the A matrix, otherwise, the advantage of the sparsity will be lost and the computation effort estimate in (4) will not be achieved.

Taking into account the above considerations, MATLAB SIMULINK's[17] discrete-time system solver "discrete state-space" is chosen, instead of the conventional choice of `ltitr`.

Interpolation is needed whenever the output is downsampled. Because the downsampling instants are evenly spaced, the downsampling can be viewed as the resampling of discrete-time signals and efficient interpolation methods in signal processing (i.e. inserting zeros and then low-pass filtering) can be employed. In fact, `newlsim.m` applies the MATLAB command `interp.m`,[18] which does exactly this.

With the responses of the subsystems computed, it is finally possible to form the response of the original system due to the original input. This is allowable because of the linearity property of LTI systems.

The commands mentioned above are very MATLAB specific because the current version of `newlsim.m` is implemented in MATLAB. Nevertheless, the idea to take advantage of problem specific insights and structure extends naturally to more general platforms.

4 Closed Loop Systems Issues

In this section, issues concerning the implementation of the `newlsim.m` algorithm are addressed. The first problem is due to closing a feedback loop (e.g. attitude control systems (ACS) for a satellite). The second problem is due to cascade connections between the plant and some other filters (e.g. noise shaping filter and/or optical controller). These two problems will be studied in two separate subsections.

Simulation with Feedback Loop

The block diagram of the problem is given in Figure 6.

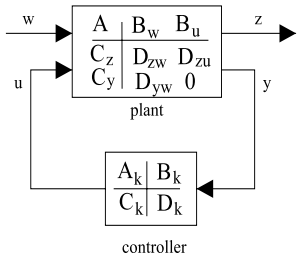

Fig. 6 Block diagram of the plant with a feedback controller.

Suppose the open loop system is in modal form (i.e. the A matrix is block diagonal) and the state-space

form is[5]

$$\left[\begin{array}{c|cc} A & B_w & B_u \\ \hline C_z & D_{zw} & D_{zu} \\ C_y & D_{yw} & 0 \end{array}\right]. \tag{15}$$

The feedback controller has the following state-space realization:

$$\left[\begin{array}{cc} A_k & B_k \\ C_k & D_k \end{array}\right], \tag{16}$$

where subscripts w and u denote quantities related to disturbance input and control input, respectively. Subscripts z and y are related to performance output and measurement output, and subscript k denotes quantities related to the controller. The closed loop system has the state space form

$$\left[\begin{array}{cc|c} A + B_u D_k C_y & B_u C_k & B_w + B_u D_k D_{yw} \\ B_k C_y & A_k & B_k D_{yw} \\ \hline C_z + D_{zu} D_k C_y & D_{zu} C_k & D_{zw} + D_{zu} D_k D_{yw} \end{array}\right]. \tag{17}$$

The off-diagonal terms, $B_u C_k$ and $B_k C_y$ in the "A" matrix of (17) are not expected to be zero, otherwise there will be no control effect at all. Now the problem is: Even if the open loop system is in modal form (i.e. the A matrix is block diagonal), the closed loop system will not be so because of the dynamics coupling (off diagonal terms in "A" in (17)). Another problem arises if the feedback controller is a discrete-time system[6] and this causes the closed loop system to be hybrid[7]. For the problem of dynamics coupling, two solutions are proposed:

Rediagonalization by eigenvalue decomposition. An eigenvalue decomposition is applied to the system in (17). This is the most straightforward way but the computation can be expensive if the systems considered are large-order, because of the eigenvalue problem involved.

Forced decoupling. This is a heuristic method in that some of the entries of C_y in (17) are set to zero. In other words, some of the measurements are regarded as insensitive to some of the state variables. Suppose for simplicity that all D matrices are zero and the state variables are reordered in such a way that the following equalities hold (A is assumed to be block diagonal and C_y is partitioned into two blocks):

$$\begin{aligned} A &= \left[\begin{array}{cc} A_1 & 0 \\ 0 & A_2 \end{array}\right] \\ B_u &= \left[\begin{array}{c} B_{1u} \\ B_{2u} \end{array}\right] \\ B_w &= \left[\begin{array}{c} B_{1w} \\ B_{2w} \end{array}\right] \\ C_y &= \left[\begin{array}{cc} 0 & C_{2y} \end{array}\right] \\ C_z &= \left[\begin{array}{cc} C_{1z} & C_{2z} \end{array}\right]. \end{aligned}$$

Then the state-space representation of the closed loop system (17) is as follows:

$$\left[\begin{array}{ccc|c} A_1 & 0 & B_{1u} C_k & B_{1w} \\ 0 & A_2 & B_{2u} C_k & B_{2w} \\ 0 & B_k C_{2y} & A_k & 0 \\ \hline C_{1z} & C_{2z} & 0 & 0 \end{array}\right]. \tag{18}$$

It can be verified that the system in (18) can be decomposed into two subsystems. The first subsystem includes controller dynamics and is subject to disturbance input only,

$$\left[\begin{array}{cc|c} A_2 & B_{2u} C_k & B_{2w} \\ B_k C_{2y} & A_k & 0 \\ \hline C_{2z} & 0 & 0 \end{array}\right] \tag{19}$$

and the second subsystem evolves in time with disturbance input and control input that is determined by solving the first subsystem (as the output signal of the controller)

$$\left[\begin{array}{c|cc} A_1 & B_{1u} & B_{1w} \\ \hline C_{1z} & 0 & 0 \end{array}\right]. \tag{20}$$

It can be seen that if the dimension of A_2 in (19) is much smaller than that of A_1 in (20) and if A_1 is block diagonal, then the bottleneck of diagonalization can be avoided. The justification of this method hinges upon the ability to find the state variables that are insensitive to sensor measurements and the relative significance of the contributions of the ignored measurements to the total measurements. The determination of the "important" state variables can be quite case specific. For example, in the study of a space structure with an attitude control system (ACS), if the measurements are attitude angles, then it is natural that the rigid body modes are far more important than other flexible modes. In order to quantify the error induced by the forced decoupling method, it is possible to compute the ratio

$$E = \frac{\sigma_1}{\sigma_2}, \tag{21}$$

where σ_1 and σ_2 are the open (feedback control) loop RMS values of the contributions of the unimportant

[5] The D_{yu} is missing here to avoid an algebraic loop, i.e. the coexistence of feedthroughs in the plant and the controller.

[6] In practical implementations, controllers are usually digital, which implies that the signals, as well time instants, are discrete. The discrete-time assumption here is merely for the ease of analysis.

[7] Here the term "hybrid" is used in the very specific sense that the system contains both continuous-time and discrete-time states. There is no discrete state involved.

and important dynamics to the measurement. If E is smaller than some tolerance, then the forced decoupling heuristic is justified, otherwise rediagonalization by eigenvalue decomposition must be applied. The computation of the ratio E in (21) can be very efficient if the open loop system is already in modal form. For example, the Lyapunov analysis with `newlyap.m` in de Weck et al.[1] can be applied here. As an example, consider the 2184 state variable SIM model v2.2 with three unstable (or marginally stable) rotational rigid body modes and assume the attitude angles (proportional to rigid body mode angles, together with some additional contributions from other flexible modes) are measured directly. The numeric values of E in (21) in this example are summarized in Table 2.

Table 2 E: The ratio (%) between the RMS attitude angle of the flexible mode subsystem over the rigid body mode subsystem.

θ_1	θ_2	θ_3
0.1021	0.2201	0.1950

The results in Table 2 shows that E is quite small if the forced decoupling heuristic is used. To verify the prediction, the actual results from the two methods are computed: The RMS values of the performance outputs by eigenvalue decomposition method and the forced decoupling heuristic are 1.8275×10^{-5} and 1.8276×10^{-5}, respectively. Their difference is 1.8135×10^{-9}, which amounts to about 0.0099 % of the performance given by the eigenvalue decomposition method (chosen as a reference here). In conclusion, the forced decoupling heuristic is not exact but can be fairly accurate if properly applied.

For the problem of "hybrid" closed loop systems, there are two options suggested:

Continuous-time approximation. If the sampling rate of the control is high enough, then the digital controller can actually be approximated by a continuous-time system using techniques like zero order hold (ZOH) or bilinear (Tustin) transform. The accuracy of this approximation can be found in any common digital (or computer) control text.

Lifting. This is a useful approach to deal with sampled-data control systems analysis problems. For example, see Chen[19] and Yamamoto.[20] Suppose a discrete-time signal $v[n]$ is defined as

$$v = \{v[0], v[1], v[2], \ldots\}.$$

The lifted version of the signal $\underline{v}$ can be expressed as

$$\underline{v} = \left\{ \begin{bmatrix} v[0] \\ v[1] \\ \vdots \\ v[n-1] \end{bmatrix}, \begin{bmatrix} v[n] \\ v[n+1] \\ \vdots \\ v[2n-1] \end{bmatrix}, \cdots \right\},$$

where $n \in \mathbb{Z}$. If an LTI system is represented as

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right],$$

then its lifted version is defined such that the original inputs and outputs signals are lifted. That is,

$$\left[\begin{array}{c|cccc} A^n & A^{n-1}B & A^{n-2}B & \cdots & B \\ \hline C & D & 0 & \cdots & 0 \\ CA & CB & D & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{n-1} & CA^{n-2}B & CA^{n-3}B & \cdots & D \end{array}\right] \tag{22}$$

The lifting procedure in (22) basically reduces sampling rate at the expense of an increase in input and output dimensions. Equivalently this procedure can be viewed as one of the applications of the state augmentation technique. The main application of this method in the algorithm is to convert a multiple sampling rates[8] system into a single rate (slow rate) LTI system without losing the effect of fast dynamics. The drawback of this method is the high resulting dimensionality. Nevertheless, this method can work well in conjunction with the forced decoupling method discussed previously if the subsystem coupled with the controller has low dimension.

Simulation with Feedforward Controller

The block diagram is depicted in Figure 7. Even

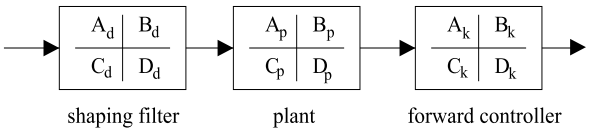

Fig. 7 Block diagram of the plant with shaping filter and feedforward controller appended.

if the open loop plant is in modal form (block diagonal A matrix), the cascade connection of the plant and other dynamic systems (e.g. noise shaping filter or feedforward controller) does not in general have a block diagonal A matrix. Suppose the plant dynamics is

$$\left[\begin{array}{c|c} A_p & B_p \\ \hline C_p & 0 \end{array}\right] \tag{23}$$

and the controller has the state-space representation as

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & 0 \end{array}\right], \tag{24}$$

[8]Sometimes the sampling rate of the plant is much higher than that of the controller, in order to represent the plant dynamics accurately, see Chen[19] for more detail. Note also that a multiple sampling rate system is time-variant.

then the cascade connection of these two systems will have the state-space representation as follows:

$$\left[\begin{array}{cc|c} A_p & B_pC_k & 0 \\ 0 & A_k & B_k \\ \hline C_p & 0 & 0 \end{array}\right]. \tag{25}$$

It is clear that even if A_p is block diagonal, the A matrix of the closed loop system will not be so because of B_pC_k. Therefore, a rediagonalization is necessary. Nevertheless, it is not necessary to call the eigenvalue solver (e.g. `eig` in MATLAB) to redo the diagonalization if the plant and the controller do not have the same eigenvalues. That is,

$$\lambda(A_p) \cap \lambda(A_k) = \emptyset,$$

where $\lambda(\cdot)$ denotes the set of all eigenvalues of a matrix. This is true because of the following lemma (from Golub and Van Loan[21]):

Lemma 4.1 *Let $T \in \mathbb{C}^{n\times n}$ be partitioned as follows:*

$$T = \left[\begin{array}{cc} T_{11} & T_{12} \\ 0 & T_{22} \end{array}\right], \tag{26}$$

where $T_{11} \in \mathbb{C}^{p\times p}$ and $T_{22} \in \mathbb{C}^{q\times q}$. Define the linear transformation $\phi : \mathbb{C}^{p\times q} \to \mathbb{C}^{p\times q}$ by

$$\phi(X) = T_{11}X - XT_{22}, \tag{27}$$

where $X \in \mathbb{C}^{p\times q}$. Then ϕ is nonsingular if and only if $\lambda(T_{11}) \cap \lambda(T_{22}) = \emptyset$, where $\lambda(T_{11})$ and $\lambda(T_{22})$ are sets of eigenvalues of T_{11} and T_{22} respectively. If ϕ is nonsingular and Y is defined by

$$Y = \left[\begin{array}{cc} I_p & Z \\ 0 & I_q \end{array}\right] \qquad \phi(Z) = -T_{12},$$

then $Y^{-1}TY = diag(T_{11}, T_{22})$.
***Proof**: See Appendix.*

In the current context, A_p and A_k can be thought of as T_{11} and T_{22} in the lemma. It is assumed that the digonalization of A_k (or T_{22}) is possible and easy to find (e.g. controller A matrix). The implication of this lemma is that the eigenvalues and eigenvectors of the closed loop A matrix can be found without using general purpose eigenvalue solvers (e.g. `eig` in MATLAB), and thus the rediagonalization can be computed efficiently. The reason is as follows: Suppose $T_{11} \in \mathbb{R}^{p\times p}$ and $T_{22} \in \mathbb{R}^{q\times q}$ are block diagonal and diagonalizable, so there exist invertible matrices (eigenvector matrices) V_{11} and V_{22} such that $V_{11}^{-1}T_{11}V_{11} = D_{11}$ and $V_{22}^{-1}T_{22}V_{22} = D_{22}$, where $D_{11} \in \mathbb{C}^{p\times p}$ and $D_{22} \in \mathbb{C}^{q\times q}$ are both strictly diagonal. The matrix T in (26) can be expressed as

$$\left[\begin{array}{cc} V_{11}^{-1} & 0 \\ 0 & V_{22}^{-1} \end{array}\right] \left[\begin{array}{cc} D_{11} & D_{12} \\ 0 & D_{22} \end{array}\right] \left[\begin{array}{cc} V_{11} & 0 \\ 0 & V_{22} \end{array}\right], \tag{28}$$

where $D_{12} = V_{11}T_{12}V_{22}^{-1}$. The middle matrix in (28) can be transformed into block diagonal form by applying lemma 4.1. That is,

$$\left[\begin{array}{cc} D_{11} & D_{12} \\ 0 & D_{22} \end{array}\right] = \left[\begin{array}{cc} I & X' \\ 0 & I \end{array}\right] \left[\begin{array}{cc} D_{11} & 0 \\ 0 & D_{22} \end{array}\right] \left[\begin{array}{cc} I & -X' \\ 0 & I \end{array}\right], \tag{29}$$

where X' can be solved by the following equation (essentially (27))

$$-D_{12} = D_{11}X' - X'D_{22}.$$

It can be seen that

$$X' = -D_{12} \cdot / \left(D_{11}U - UD_{22}\right), \tag{30}$$

where U denotes a matrix whose entries are all one and $\cdot/$ denotes the operation of elementwise division. Combining (29) and (30), it can be seen that the eigenvector matrix of T in (26) is

$$\left[\begin{array}{cc} I & -X' \\ 0 & I \end{array}\right] \left[\begin{array}{cc} V_{11} & 0 \\ 0 & V_{22} \end{array}\right] = \left[\begin{array}{cc} V_{11} & -X'V_{22} \\ 0 & V_{22} \end{array}\right]. \tag{31}$$

With the eigenvalues and eigenvectors of T in (26) known, it is straightforward to compute the rediagonalization required in the beginning of this subsection.

5 Simulation Results

In this section, some application examples are given to show the potential value of the `newlsim.m` algorithm. The first example computes the performance RMS values of randomly generated stable SISO systems of different dimensions, n_s, driven by randomly generated input signals. As in de Weck et al.,[1] three methods are compared: Time domain method, frequency domain method and Lyapunov method. For the time domain method, the input signal is 10^5 samples long and for the frequency domain method, 10^5 frequency points (single sided) are computed. The results are summarized in Table 3.

In this table, n_s is the number of state variables. $\mathrm{T_{cpu}}$ is the CPU time (in seconds) of each computation and σ is the RMS value of each performance output (the actual unit is not of interest here). Methods: `freq` denotes the frequency domain method, `newlyap` denotes the fast Lyapunov method, `lsim` is the standard time domain simulation method provided by MATLAB. Finally, `newlsim` is the proposed time domain simulation method. Note that `freq` and `newlyap` are the fast implementations of frequency domain method and Lyapunov method respectively, see de Weck et al.[1] Note also that the time for `newlsim` includes the time to diagonalize. Since new data is generated in each case with a different n_s, the RMS values of different n_s cases differ accordingly and should be not compared.

The main point to illustrate here is that the performance RMS values computed by the three different

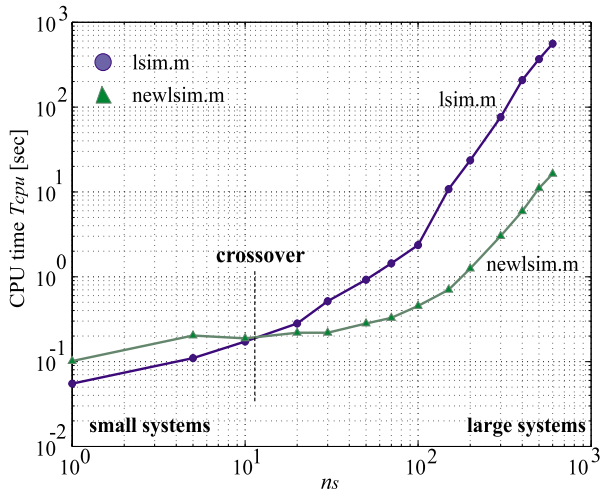

Fig. 8 Logarithmic plot of CPU times by `lsim.m` and `newlsim.m` versus number of state variables. Circle: `lsim.m` time. Triangle: `newlsim.m` time. Maximum savings factor $T_{lsim}/T_{newlsim} \simeq 50$. Crossover at around $n_s = 12$ state variables.

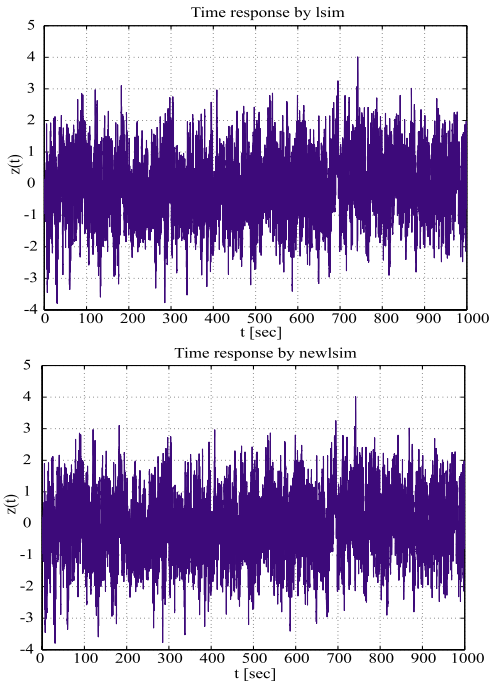

Fig. 9 Time responses for a $n_s = 200$ state system by `lsim.m` - $T_{cpu} = 71.1$ [sec] - and `newlsim.m` $T_{cpu} = 2.4$ [sec]. Both the system and the external input are randomly generated.

methods are quite close to each other. As shown in Table 3, time domain methods are generally not as efficient as other methods for small systems ($n_s < 100$). However, the situation changes when systems get larger ($n_s > 500$). The reason for this trend is that size-independent overhead of the time domain method is more significant in small system cases. It should also be noted that the computation time of the frequency domain and time domain methods varies with the number of samples evaluated. Nevertheless, it is this computation that provides the additional information that the Lyapunov method does not provide (i.e. time history and power spectral density). The reason why the results from `lsim` are unavailable (N/A) is that the computer ran out of memory, which means that `lsim` is not very suited for large-order systems simulation. The above example shows that `newlsim` can achieve efficiency similar to the fast implementations of other methods (frequency domain and Lyapunov methods) with acceptable accuracy when computing RMS values.

Computing output RMS values does not fully exemplify the advantage of `newlsim`. A time domain simulation scheme should be compared with another time domain simulation scheme. Therefore the MATLAB simulator, `lsim.m`, is chosen as a reference in the following example. In the example, a number of randomly generated systems with different sizes are simulated in the time domain with `lsim.m` and `newlsim.m`. The computation time of each simulation is given in Figure 8. The time responses of a sample system by `lsim.m` and `newlsim.m` are shown in Figure 9.

The example shows that `newlsim.m` is more efficient than `lsim.m`, while the error is insignificant.

The remaining examples are concerned with control tuning of the Space Interferometry Mission (SIM) model v2.2 (see Figure 10 for its finite element model) that is enabled by `newlsim.m`.

The SIM v2.2 model presents significant challenges

Table 3 Comparison between frequency domain, Lyapunov and time domain methods for randomly generated systems of different size. N/A indicates memory saturation.

n_s	results	`freq`	`newlyap`	`lsim`	`newlsim`
50	T_{cpu}[s]	0.2960	0.0780	1.3750	0.6880
	σ	1.714E-6	1.713E-6	1.722E-6	1.722E-6
100	T_{cpu}[s]	0.6410	0.2340	4.5310	1.0150
	σ	1.167E-5	1.159E-5	1.187E-5	1.187E-5
200	T_{cpu}[s]	2.0000	1.375	71.078	2.3900
	σ	1.516E-5	1.448E-5	1.463E-5	1.463E-5
500	T_{cpu}[s]	15.3280	13.859	N/A	15.2030
	σ	3.006E-5	3.002E-5	N/A	3.061E-5
1000	T_{cpu}[s]	104.25	98.562	N/A	96.0320
	σ	5.452E-5	5.184E-5	N/A	5.228E-5
1500	T_{cpu}[s]	425.3	393.3	N/A	382.7
	σ	1.346E-5	1.325E-5	N/A	1.334E-5
2000	T_{cpu}[s]	1046.4	949.3	N/A	934.4
	σ	2.869E-5	2.714E-5	N/A	2.733E-5

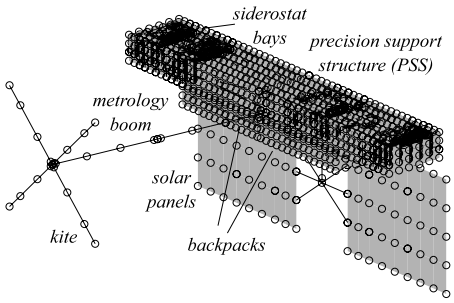

Fig. 10 Finite element model of SIM v2.2.

to time domain simulation because of its high dimensionality (2184 state variables) and wide dynamic range ($\omega_{min}/\omega_{max} \approx 4700$)[9]. For more information on SIM, see JPL.[22] Due to the scientific purpose, stringent design requirements are imposed on the SIM model. For example, see Table 4 for some requirements from Miller et al.[23] and Laskin.[24]

Table 4 SIM opto-mechanical performance requirements.

performance z_i	units	requirement
Starlight OPD	nm	10 (RMS)
Internal Metrology OPD	nm	20 (RMS)
Starlight WFT	asec	0.210 (RSS)

In order for the SIM system to work properly, two control systems are needed. One is the attitude control system (ACS) and the other is the optical control system. The overall system configuration is given in Fig. 11.

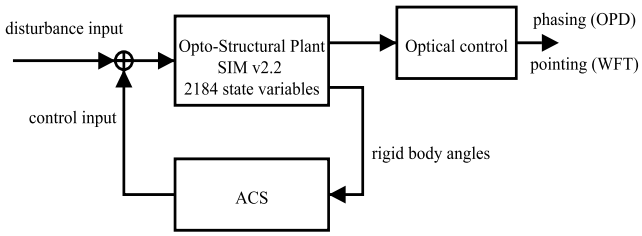

Fig. 11 Appended LTI system dynamics of SIM. The opto-structural plant is obtained by combining the optics and the finite element models (FEM). The attitude control system (ACS) is necessary to stabilize the open-loop unstable rigid body modes. The optical control system is added to improve optical performance.

The ACS in Fig. 11 stabilizes the open loop unstable rigid body modes of the SIM model. It can be designed by classical methods like PID, lead-lag or modern control techniques such as LQG. The optical control here is modelled as a second order high-pass filter and the transfer function of one channel is

$$K_o(s) = \frac{s^2}{s^2 + 2\zeta_o\omega_o s + \omega_o^2}, \tag{32}$$

where ζ_o is the damping ratio of the controller and is set to 0.707 and ω_o is the corner frequency, which is treated as a variable design parameter.

The first example is a parameter study of optical controller corner frequency ω_o [rad/s] (or f_o [Hz]). The system consists of the open loop SIM model (2184 state variables), an ACS designed by the standard LQG approach (e.g. Bélanger[25]) and the optical controller as given in (32). The ACS loop is closed by a rediagonalization by eigenvalue decomposition and the optical control path is closed by the method prescribed in Section 4. There are six input channels (three forces and three torques), which are driven by the six channels of Magellan reaction wheel assembly disturbance data (see Elias[26]). The outputs are starlight optical path difference (OPD), internal metrology (IM) and starlight wavefront tilt (WFT). In the simulation runs, different closed loop systems with different optical controller corner frequencies (f_o) are formed and the RMS values of the performance outputs are recorded. The result is shown in Figure 12. The result in Figure 12 is

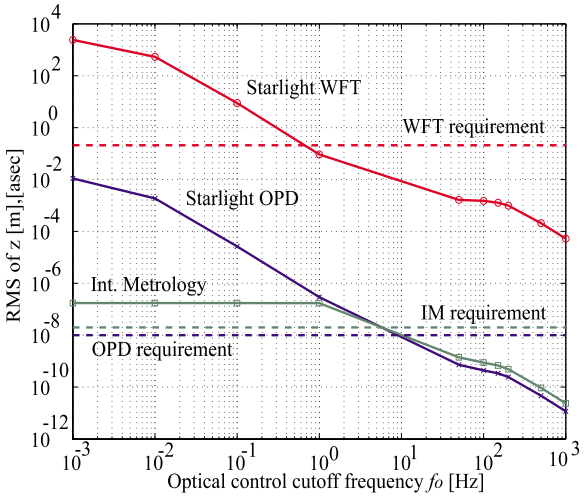

Fig. 12 RMS and RSS of the performance outputs. Circle: Starlight OPD #1. Asterisk: Internal Metrology OPD #1. Square: Starlight WFT #1.

consistent with the intuition that higher optical control bandwidth leads to better system performance. Nevertheless, it can be a problem of cost, implementation and stability margins if f_o is chosen too high. A controller cutoff frequency above 10 [Hz] appears to satisfy the requirements.

The second example is the tuning of the attitude control system (ACS). The performance outputs here are the three attitude angles (θ_x, θ_y and θ_z). In this

[9] ω_{min} is the minimum flexible mode natural frequency. The zero natural frequency of rigid body modes is not counted.

tuning, two typical scenarios are shown. One is the cheap control case and the other is the expensive control case. These cases are determined by the weights on the state and control when the ACS controller is designed. The unit step transient responses of the three attitude angles are shown in Figure 13.

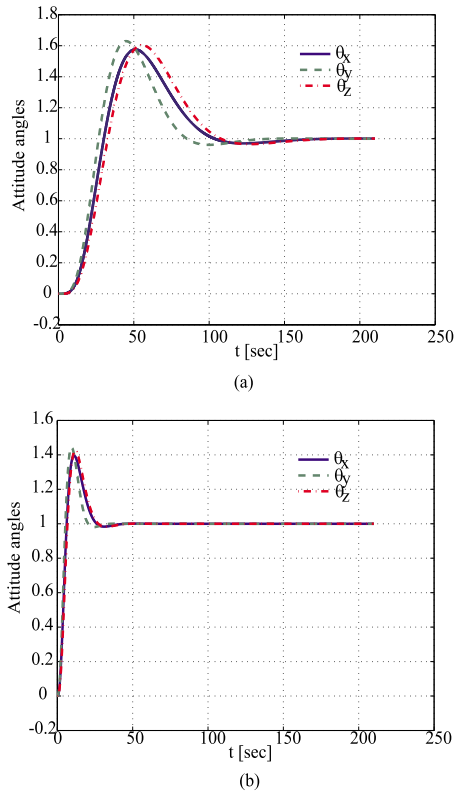

Fig. 13 Unit step responses of the attitude angles. (a) Expensive control case. (b) Cheap control case.

It can be seen from Figure 13 that while the settling time reduces with the increase of control effort, the overshoot remains large. The reason for this difficulty is the existence of non-minimum phase zeros of the open loop plant. In the simplest interpretation, the non-minimum phase zeros draw the closed loop root loci further to right half s-plane as control effort increases, and are thus reducing the stability margin. Therefore, if a structural design such that the non-minimum phase zeros are eliminated can be realized, then the transient responses of the model are expected to be much better. These transient analyses for large systems require fast time domain simulation algorithms such as `newlsim.m`.

6 Summary

Time domain simulation is an important technique for multidisciplinary design, analysis and optimization of dynamic systems. Unfortunately, as model fidelity and size get large one experiences excessive computation times and memory saturation problems. In this paper, the simulation scheme, `newlsim.m`, based on a block diagonalization pre-processing, is presented in response to this challenge. The targeted systems are large-order, diagonalizable continuous-time LTI systems. It has been found that diagonalization provides three benefits (dynamics decoupling, fast discretization and multiple sampling rates) that facilitate the simulation. In conjunction with the block diagonalization structure of the resultant A matrix, it has been shown that a sparse matrix recognizable state transition must be employed in order to achieve the $O(n_s n)$ state transition by taking advantage of the resultant sparsity. Problems with feedback and feedforward controllers are discussed and the corresponding solutions (e.g. forced decoupling and rediagonalization without using iterative eigenvalue solver) are proposed. It has been shown that `newlsim.m` can achieve similar efficiences as those achieved by fast implementations of frequency domain and Lyapunov methods (e.g. `newlyap.m`), while retaining the advantage of transient response calculations. Finally, applications enabled by `newlsim.m` are given as optical and ACS control tuning of the 2200 state SIM spacecraft system to illustrate the potential value of the simulation scheme.

Recommendations for future work include extensions of the algorithm to time-varying and weakly nonlinear systems as well implementation of distributed computation of the subsystem responses on parallel computers.

Acknowledgement

This research was funded by the NASA Jet Propulsion Laboratory under grant No. JPL 91123 and was monitored by Dr. Ipek Basdogan. The authors thank Prof. Wallace Vander Velde of MIT for his helpful comments.

Appendix: Derivation of Lemma 4.1

Suppose $\phi(X) = 0$ for $X \neq 0$ and that

$$U^H X V = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{matrix} r \\ p-r \end{matrix}$$
$$\begin{matrix} r & q-r \end{matrix}$$

is the SVD of X with $\Sigma_r = \mathrm{diag}(\sigma_i)$, $r = \mathrm{rank(X)}$. Substituting this into the equation $T_{11}X = XT_{22}$ gives

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix},$$

where $U^H T_{11} U = (A_{ij})$ and $V^H T_{22} V = (B_{ij})$. By comparing blocks we see that $A_{21} = 0, B_{12} = 0$, and $\lambda(A_{11}) = \lambda(B_{11})$. Consequently,

$$\emptyset \neq \lambda(A_{11}) = \lambda(B_{11}) \subseteq \lambda(T_{11}) \cap \lambda(T_{22}).$$

On the other hand, if $\lambda \in \lambda(T_{11}) \cap \lambda(T_{22})$ then we have nonzero vectors x and y so $T_{11}x = \lambda x$ and $y^H T_{22} =$

λy^H. A calculation shows that $\phi(xy^H) = 0$. Finally if ϕ is nonsingular then the matrix Z above exists and

$$\begin{aligned} Y^{-1}TY &= \begin{bmatrix} I & -Z \\ 0 & I \end{bmatrix} \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} \begin{bmatrix} I & Z \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} T_{11} & T_{11}Z - ZT_{22} + T_{12} \\ 0 & T_{22} \end{bmatrix} \\ &= \begin{bmatrix} T_{11} & 0 \\ 0 & T_{22} \end{bmatrix} \end{aligned}$$

Q.E.D.

References

[1]de Weck, O., Uebelhart, S., Gutierrez, H., and Miller, D., "Performance and Sensitivity Analysis for Large Order Linear Time-Invariant Systems," *9th AIAA/ISSMO symposium on multidisciplinary analysis and optimization*, Atlanta, Georgia, USA, September 2002.

[2]Giesing, J. and Barthelemy, J.-F., "A Summary of Industry MDO Applications and Needs," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Optimization and Analysis*, St. Louis, MO, September 1998, AIAA Paper No. 98-4737.

[3]Sobieszczanski-Sobieski, J. and Haftka, R. T., "Multidisciplinary Aerospace Design Optimization - Survey of Recent Developments," *AIAA 34th Aerospace Sciences Meeting and Exhibit*, Reno, NV, USA, January 1996.

[4]Anderson, M. R. and Mason, W. H., "An MDO approach to Control-Configured-Vehicle design," *the 6th AIAA, NASA, and ISSMO, Symposium on multidisciplinary Analysis and Optimization*, Bellevue, WA, USA, September 1996.

[5]Gutierrez, H. L., *Performance Assessment and Enhancement of Precision Controlled Structures During Conceptual Design*, Ph.D. thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 1999.

[6]de Weck, O. L., *Multivariable Isoperformance Methodology for Precision Opto-Mechanical Systems*, Ph.D. thesis, Massachusetts Institute of Technology, September 2001.

[7]Mallory, G. and Miller, D. W., "Increasing the Numerical Robustness of Balanced Model Reduction," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 3, May-June 2002, pp. 596–598, Engineering Notes.

[8]Willcox, K. and Peraire, J., "Balanced Model Reduction via the Proper Orthogonal Decomposition," *15th AIAA Computational Fluid Dynamics Conference*, Anaheim, CA, USA, June 2001.

[9]Beran, P. and Silva, W., "Reduced-Order Modeling: New Approaches for Computational Physics," *39th Aerospace Sciences Meeting and Exhibit*, Reno, NV, USA, January 2001.

[10]Sou, K. C., *Fast Time Domain Simulation for Large Order Hybrid Systems*, Master's thesis, Massachusetts Institute of Technology, Cambridge, MA 02139, May 2002, SSL Report # 11-02.

[11]Andrew, G., *Control system toolbox user's guide*, The MathWorks, Inc., 1996.

[12]Reich, S., "Multiple Time Scales in Classical and QuantumClassical Molecular Dynamics," *Journal of Computational Physics*, Vol. 151, No. 1, May 1999, pp. 49–73.

[13]Oppenheim, A., Schafer, R., and Buck, *Discrete-time Signal Processing*, Prentice Hall, 2nd ed., 1999.

[14]Franklin G.F., Powell J.D. and Workman M.L., *Digital Control of Dynamic Systems*, Addison Wesley, 2^{nd} ed., 1990.

[15]Åström, K. J. and Wittenmark, B., *Computer-Controlled Systems: Theory and Design*, Prentice-Hall, Inc., 1997.

[16]Chen, C., *Linear System Theory and Design*, Oxford University Press, 1999.

[17]The MathWorks Inc., *SIMULINK : a program for simulating dynamic systems, user's guide*, The MathWorks, Inc., 1992.

[18]John N, L., *Signal Processing Toolbox User's Guide*, The MathWorks, Inc., 1992.

[19]Chen, T. and Francis, B., *Optimal Sampled-Data Control Systems*, Communications and Control Engineering Series, Springer, London, 1995.

[20]Yamamoto, Y., "A function space approach to sampled data control systems and tracking problems," *IEEE transaction on automatic control*, Vol. 39, No. 4, April 1994, pp. 703–713.

[21]Golub, G. and Van Loan, C., *Matrix Computations*, The Johns Hopkins University Press, Baltimore and London, 3*rd* ed., 1996.

[22]NASA Jet Propulsion Laboratory, "http://sim.jpl.nasa.gov/," Internet link.

[23]Miller, D. W., de Weck, O. L., and Uebelhart, S. A., "*Integrated Dynamics and Controls Modeling for the Space Interferometry Mission (SIM)*," *IEEE Aerospace Conference*, Big Sky, Montana, USA, March 2001.

[24]Laskin, R. A., "SIM Dynamics & Control Requirements Flowdown Process," Presentation at the SIM Project Preliminary Instrument System Requirements Review, JPL.

[25]Bélanger, P. R., *Control Engineering: A Modern Approach*, Saunders College Publishing, 1995.

[26]Elias, L., A Structurally Coupled Disturbance Analysis Method Using Dynamic Mass Measurement Techniques, with Application to Spacecraft- Reaction Wheel Systems, Master's thesis, Massachusetts Institute of Technology, March 2001.