N94-14641

Simulation of Linear Mechanical Systems

S. W. Sirlin
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

INTRODUCTION

A dynamics and controls analyst is typically presented with a structural dynamics model and must perform various input/output tests and design control laws. The required time/frequency simulations need to be done many times as models change and control designs evolve.

This paper examines some simple ways that open and closed loop frequency and time domain simulations can be done using the special structure of the system equations usually available. Routines were developed to run under Pro-Matlab [1] in a mixture of the Pro-Matlab interpreter and Fortran (using the .mex facility). These routines are often orders of magnitude faster than trying the typical "brute force" approach of using built-in Pro-Matlab routines such as *bode*. This makes the analyst's job easier since not only does an individual run take less time, but much larger models can be attacked, often allowing the whole model reduction step to be eliminated.

I will first briefly discuss the standard model forms, then address each of the simulation cases separately.

LINEAR MECHANICAL SYSTEM MODELS

Linear mechanical system models have a special second order differential form. In general the various structural dynamics codes generate equations in the form:

$$M\ddot{q} + N\dot{q} + Kq = Bu + Gf, \tag{1}$$

$$y = C_p q + C_v \dot{q} + Du,$$

where $q \epsilon \mathrm{R}^n$ is the modal state, $u \epsilon \mathrm{R}^m$ is the control input, $y \epsilon \mathrm{R}^p$ is the output, and $f \epsilon \mathrm{R}^q$ is the disturbance input. Systems of the above form can almost always (generically) be put into modal form, and this modal form is typically what is given to the dynamics and controls analyst by the structural modellers (e.g. NASTRAN output):

$$\ddot{\eta} + 2\Sigma\dot{\eta} + \Lambda\eta = Bu + Gf, \tag{2}$$
$$y = C_p\eta + C_v\dot{\eta} + Du,$$

where $\eta \epsilon \mathrm{R}^n$ is the modal state, Σ and Λ are diagonal matrices, u and f are the control input and disturbance force respectively, and y is the output which can be position, velocity, input force, or some combination of these. The transformation to the form (2) is not easy for general damping, (N) but typically damping is so poorly known that simple modal damping models suffice. I will assume that (2) is available for the rest of the paper.

The standard analysis tools of Pro-Matlab require first order form, which can be done as follows:

$$\dot{x} = \bar{A}x + \bar{B}u + \bar{G}f, \tag{3}$$
$$y = \bar{C}x,$$
$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}, \qquad \bar{A} = \begin{bmatrix} 0 & I \\ -\Lambda & -2\Sigma \end{bmatrix},$$
$$\bar{B} = \begin{bmatrix} 0 \\ B \end{bmatrix}, \qquad \bar{G} = \begin{bmatrix} 0 \\ G \end{bmatrix}, \qquad \bar{C} = [\, C_p \quad C_v \,].$$

Note that the special structure does not fit any of the standard forms (block diagonal, triangular, banded, or Hessenberg).

To the above plant equations must be added typical control dynamics:

$$\dot{w} = A_c w + B_c(r - y), \tag{4}$$
$$u = C_c w + D_c(r - y),$$

where r is some reference command input. Again resorting to brute force we have the whole system:

$$\dot{z} = \hat{A}z + \hat{B}r + \hat{G}f, \tag{5}$$
$$y = \hat{C}z,$$
$$z = \begin{bmatrix} q \\ \dot{q} \\ w \end{bmatrix}, \qquad \hat{A} = \begin{bmatrix} \bar{A} - \bar{B}D_c\bar{C} & \bar{B}C_c \\ -B_c\bar{C} & A_c \end{bmatrix},$$
$$\hat{B} = \begin{bmatrix} BD_c \\ B_c \end{bmatrix}, \qquad \hat{G} = \begin{bmatrix} \bar{G} \\ 0 \end{bmatrix}, \qquad \hat{C} = [\, \bar{C} \quad 0 \,].$$

Note that now even the special structure of (3) is not present (in general) even in the upper left block of the $\hat{A}$ matrix. Due to the feedback we have lost all information regarding the eigenstructure. The system may not even be very sparse anymore.

FREQUENCY DOMAIN RESPONSE - OPEN LOOP

If we desire to compute the open loop frequency response for (2,3)

$$H(s) = \bar{C}(Is - \bar{A})^{-1}\bar{B},$$

there is the built-in *bode* command of Pro-Matlab, that uses the well known method of Laub [2]. While the method is efficient for the general case, given the special structure of (2) we can write down a much simpler solution:

$$H_{ij}(s_k) = \Sigma_{l=1}^{n} \frac{(C_{pil} + C_{vil}s)B_{lj}}{s_k^2 + 2\zeta_l\omega_l s_k + \omega_l^2}.$$

This can be easily implemented using just a few lines of the Pro-Matlab interpreter, the only trick being to come up with some way of representing a rank 3 array. The header of a routine that does this, *freqm2d*, is listed in the Appendix. Timing results are as one would expect. From Figure 1 (results obtained on a VAX 11/780), the brute force computation time increases roughly quadratically, while the direct solution requires only a linear cost with system order.

FREQUENCY DOMAIN RESPONSE - CLOSED LOOP

Once the system of (2,4) is put into the form (5) then the same standard tools apply as for the open loop case. As was pointed out above, the feedback destroys all of the open loop eigenstructure, hence there is no direct solution as in the open loop case. On the other hand consider Figure 2. Taking an "analog" approach to the analysis, given the individual transfer functions for the plant and the control one can combine them to get the overall transfer function via the usual algebra in the frequency domain:

$$H(s) = GF(I + GF)^{-1} = G(I + GF)^{-1}F = (I + GF)^{-1}GF.$$

In comparison to the brute force approach, there are a few issues:

- We can take advantage of any special forms for the individual blocks - this is clearly advantageous.

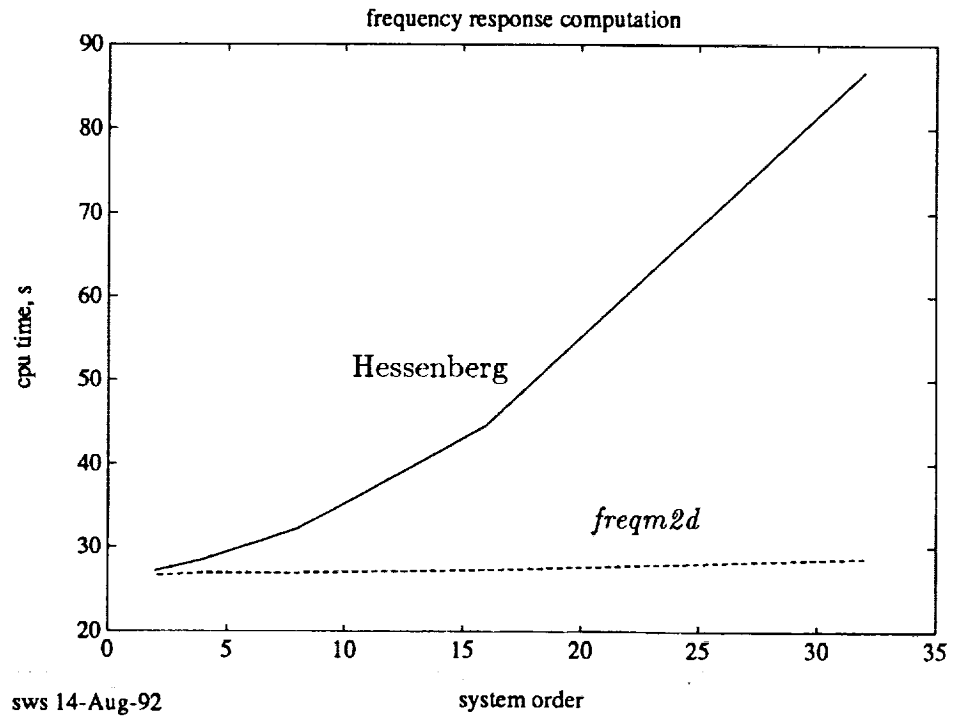

Figure 1. Computation time for open loop frequency response: Hessenberg versus *freqm2d* (specialized for mechanical systems). The test case is a second order system with a variable number of modes.

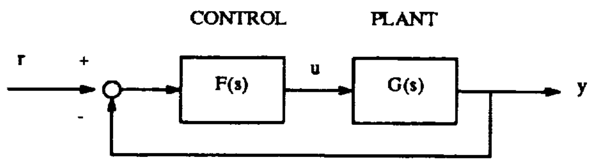

Figure 2. A simple closed loop system.

- We require matrix inversion of the order of the minimal input (to the plant or the control) - this is not clearly good or bad computationally. We're trading the usual n^2 cost for m^3, which is likely to be advantageous for large plants with only a few feedback loops, but a loss for full state feedback.

- The technique allows work directly with experimentally determined transfer functions - there is no need for system identification to proceed all the way to a state space model.

- We require storage of the intermediate results - this is an implementation issue. Currently the routines I have [3] first generate the individual transfer functions then combine the results. In some cases the memory to hold all the individual transfer functions can be quite high. This cost can be eliminated by moving the whole routine to Fortran however. The header of a routine that implements this, *feedbackmtf*, is listed in the Appendix.

As a simple example, consider a plant with 64 modes, 4 outputs, m inputs, and m pseudo-derivative feedback controllers. Timing results (again on a VAX 11/780) for a modest test case are shown in Figure 3.

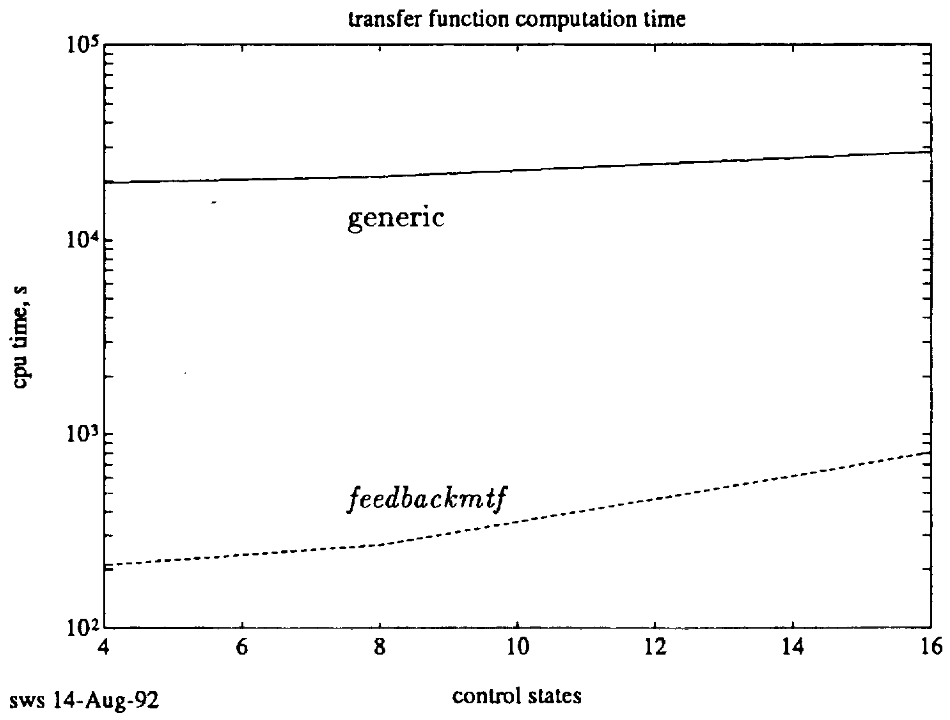

Figure 3. Computation time for closed loop frequency response: generic system approach versus subsystem approach (*feedbackmtf*). A 64 mode example with a variable amount of feedback.

A more stressing test, but one of practical interest was the NASA CSI (Control Structure Interaction) Focus Mission Interferometer (FMI) structural control [4]. The FMI is a $30m$ baseline interferometer, with the goal of controlling optical pathlength to the nanometer level. The plant model has 527 modes, with 17 rigid body modes and modal frequencies from 3.9 to $40000Hz$. To this we want to add 25 9-state controllers with displacement and force measurements, and force output. This gives a total system order of 1279. Use of the Hessenberg routines on the whole system is completely unrealistic in this case. Model reduction is a sensible approach, especially as modes above 100 Hz are not believable. On the other hand it is clear that model reduction is only required due to the numerical inefficiency of dealing with the problem as a whole block. Calculations of the individual components can be easily carried out using the full model without having to worry about truncation issues such as residual flexibility or the need for augmenting the reduced system with Ritz vectors [5]. The computation (for 1068 frequency points) took $4640s$ inverting the 25×25 input force matrix at every frequency. If this is to be done many times for the same system, then clearly model reduction is desirable, but if done only a few times (as was the case here), then model reduction isn't worthwhile. In this case only a 1×1 inverse is really required as the structural loops are all uncoupled, and so some time is certainly wasted in the (LINPACK) routine checking zeroes. Trying to construct more efficient routines I've run into some of the limitations of Pro-Matlab:

- Using an external (Fortran) routine many times (say for sequentially closing many loops) can lead to enormous allocation of unnecessary memory,

- Describing a system as separate blocks requires a data object much more complex than a matrix. What is needed is the concept of a data structure, for example that of the C language, or even better something that could be created, changed, and destroyed interactively such as the generalized arrays of APL2 or J[6]. Without this function, input lists must be long and specialized to particular cases.

In spite of these limitations, the tool is quite useful, enabling analysis for large systems and being fast for more modest systems.

TIME DOMAIN RESPONSE - OPEN LOOP

While the general solution to (3) is given by the well known matrix exponential, this is very difficult and expensive to compute in practice for general systems [7], often requiring a model reduction step.

On the other hand, the solution to (2) above is quite obvious since we just have a set of uncoupled second order equations. The solution is

$$\xi_k(t+\Delta t) = \Phi_k \xi_k(t) + \Gamma_k f_{Tk}(t), \qquad k = 1, \ldots n,$$

$$\xi_k(t) = \begin{bmatrix} \eta_k(t) \\ \dot{\eta}_k(t) \end{bmatrix}$$

$$f_T = Bu + Gf,$$

where the Φ_k and Γ_k matrices are easily calculated given the k^{th} eigenvalue (with various special cases depending on whether it is 0, real, imaginary, or complex). The computational cost increases linearly with state order. In addition the propagation from one time to the next can be done with a $2n \times 2$ matrix (n Φ_k's) versus the usual $2n \times 2n$ exponential matrix, saving on storage space.

Results for the Galileo spacecraft present another practical case of interest. The model at hand was built for investigating the possibility of finding ways to shake the spacecraft to free the stuck high gain antenna. The model had 142 modes, with 8 rigid body modes (6 for the whole plus the dual spin main bearing and a scan platform bearing), and with structural modal frequencies from 0.143 to $144Hz$. In the case shown the response of the system to the deployment and stow of a movable low gain antenna (LGA) was investigated. We've also looked at the system response to thruster firings and to torques at the spin bearing and scan platform bearing.

The routines were coded in Fortran, linked to Pro-Matlab, and run on a Sun SPARCstation 2 (Figure 4). The header for the main routine, *lsim2*, is listed in the Appendix. The computational gain in comparison to the standard built in Pro-Matlab c2d and lsim on the whole system is evident.

TIME DOMAIN RESPONSE - CLOSED LOOP

To see how to close the loop we look first at a simple coupled system with no external inputs:

$$\dot{x}_1 = A_1 x_1 + A_2 x_2,$$

$$\dot{x}_2 = B_1 x_1 + B_2 x_2.$$

The exact solution is again the matrix exponential

$$x(t) = \Phi(t) x_0 = e^{At} x_0, \qquad A = \begin{bmatrix} A_1 & A_2 \\ B_1 & B_2 \end{bmatrix},$$

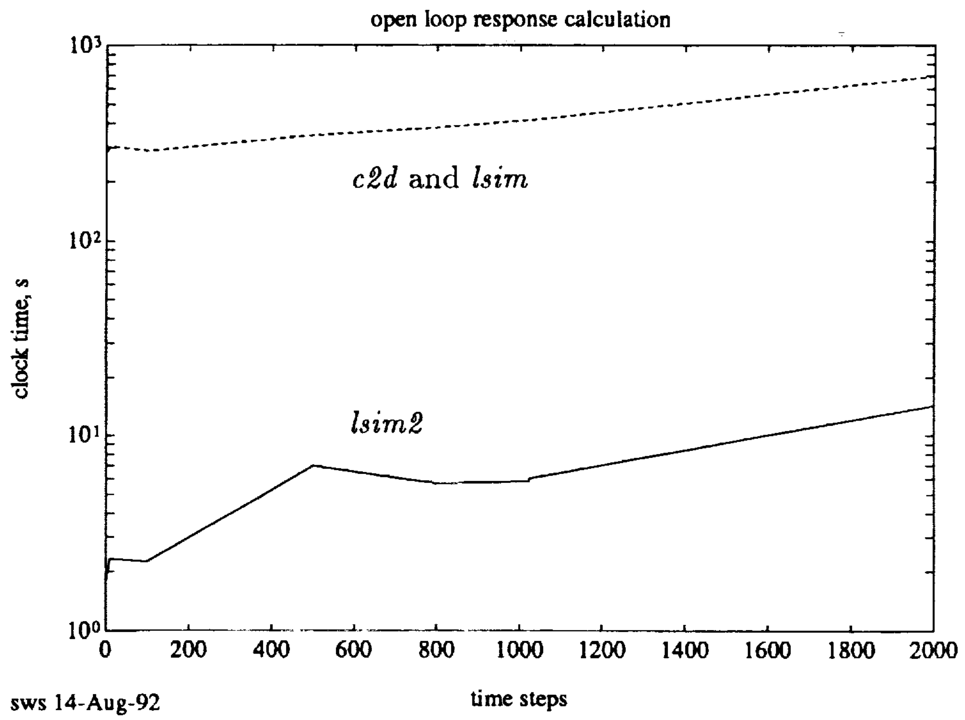

Figure 4. Galileo LGA deployment response calculation cost. Standard approach (*c2d* and *lsim*) versus approach specialized for mechanical systems (*lsim2*).

$$\Phi(t) = I + \begin{bmatrix} A_t & A_2 \\ B_1 & B_2 \end{bmatrix} t + \begin{bmatrix} A_1^2 + A_2 B_1 & A_1 A_2 + A_2 B_2 \\ B_1 A_1 + B_2 B_1 & B_1 A_2 + B_2^2 \end{bmatrix} \frac{t^2}{2} + \dots .$$

Unlike the frequency domain case, we can't easily find the response of the two systems separately and then combine algebraically (this would require a convolution in time). On the other hand if x_2 were controller states in a discrete controller, we would be perfectly justified in generating the responses for each system separately assuming a zero order hold for the other system, as this is exactly what really happens. Arguing that this must still be reasonable for low pass analog controllers, we have an approximate solution:

$$\hat{x}(t) = \hat{\Phi}(t) x_0 = \begin{bmatrix} \phi_1 & \gamma_1 \\ \gamma_2 & \phi_2 \end{bmatrix} x_0,$$

$$\phi_1 = e^{A_1 t}, \qquad \gamma_1 = \int_0^t e^{A_1(t-\tau)} d\tau A_2,$$

$$\phi_2 = e^{B_2 t}, \qquad \gamma_2 = \int_0^t e^{B_2(t-\tau)} d\tau B_1,$$

$$\hat{\Phi}(t) = I + \begin{bmatrix} A_1 & A_2 \\ B_1 & B_2 \end{bmatrix} t + \begin{bmatrix} A_1^2 & A_1 A_2 \\ B_2 B_1 & B_2^2 \end{bmatrix} \frac{t^2}{2} + \dots .$$

Comparing the approximate to the true solution, we see that

$$\Phi - \hat{\Phi} = \begin{bmatrix} A_2 B_1 & A_2 B_2 \\ B_1 A_1 & B_1 A_2 \end{bmatrix} \frac{t^2}{2} + \dots ,$$

so that the error $e(t) = ||x - \hat{x}||$ is of order t^2, so this method is of order 1. With this method we can use any special structure present in the individual subsystems, which can save considerable computational cost. On the other hand we must make sure to take a small enough time step. Currently a Fortran/Pro-Matlab implementation just uses a fixed step size. Local error estimates could be used to warn of possible trouble or change the step size. The routine ***lsim2fb*** (the header is listed in the Appendix) is the current implementation of these ideas.

Returning to the Galileo example, the previous section mentioned 8 rigid body modes, but the spin bearing and scan platform bearing (these are called the Clock and Cone degrees of freedom for the scan platform) are likely to be in a "caged" mode, actively controlled to have about $0.25Hz$ natural frequency with 70% damping ratio. A 2-state controller was added to implement this. Figure 5 shows the results of a Fortran/Pro-Matlab implementation, again comparing with the standard generic path in Pro-Matlab on a Sun SPARCstation 2. Note that the simulation times are very close to the open loop case. The difference in the system state was less than 4% for this case, in which the time step used was $11msec$. The time step was chosen based on the minimum discretization for thruster pulses rather than to minimize simulation error.

SUMMARY AND CONCLUSIONS

A set of tools has been built specifically for linear mechanical systems, open and closed loop, for use with Pro-Matlab. These tools use the pre-existing open loop eigenstructure typically available for structural dynamics models, which can save orders of magnitude in cpu time for typical problems.

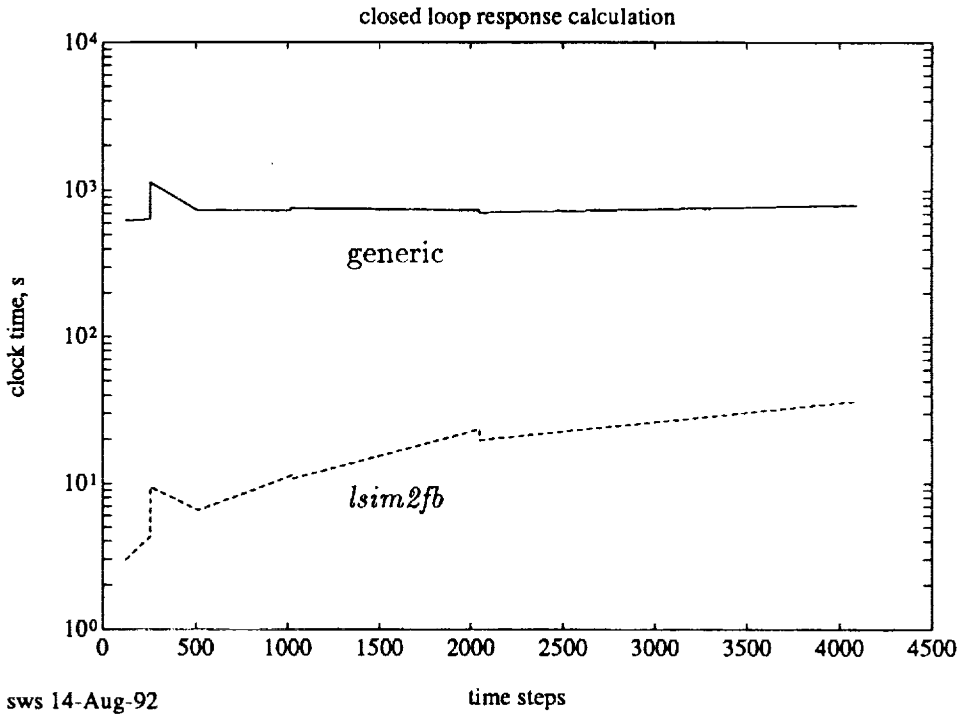

Figure 5. Galileo LGA response calculation cost given closed loop clock and cone control. Standard system approach versus approximate subsystem approach (*lsim2fb*).

For closed loop systems, treating the analysis of each block separately allows analysis of problems that might otherwise be too large, reducing or eliminating the model reduction step in the analysis.

ACKNOWLEDGEMENT

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

1. Pro-Matlab User's Guide and Control System Toolbox, The MathWorks, Inc., Natick, MA, 1990.

2. A. J. Laub, "Efficient Multivariable Frequency Response Computations," IEEE Transactions on Automatic Control, Vol. AC-26, No. 2, April 1981.

3. S. W. Sirlin, "Pro-Matlab Functions for Frequency Response," JPL EM 343-1163, December 1989 (JPL internal document).

4. S. W. Sirlin, "Active Structural Control for Damping Augmentation and Compensation of Thermal Distortion," Second Joint Japan-U.S.A. Conference on Adaptive Structures, Nagoya, Japan, November 1991.

5. C. C. Chu, M. H. Milman, "Computational Issues in Optimal Tuning and Placement of Passive Dampers," this proceedings.

6. R.K.W. Hui, K.E. Iverson, E.E. McDonnell, A. Whitney, "APL \ ?," APL Quote Quad, Volume 20, Number 4, July, 1990, pp. 192-200.

7. C. Moler, C. van Loan, "Nineteen Dubious Ways to Compute the Exponential of a Matrix," SIAM Review, Vol. 20, No. 4, October 1978.

APPENDIX
SOME SELECTED Pro-Matlab ROUTINES FOR SIMULATION

Below are included the selections from headers for the main routines used in the above examples.

I. Frequency domain

A. Open loop

```
function [ht] = freqm2d(sigma2,omega2,b,c,d,np, w)
% [h] = freqm2d(sigma2,omega2,b,c,d,np, w)
%
% MIMO tf calculation, given second order modal form
%
%    s^ 2 x(i) + sigma2(i) s x(i) +omega2(i) x(i) = b(i,:) u
%
%    y = [yp]
%        [yv]
%
%         yp = cp x    + dp u
%         yv = cv s x + dv s u
%
%              c = [cp]     d = [dp]  size np
%                  [cv]         [dv]
%
% The results are returned in a matrix
%
%       h = [h11, h12, ..., h1m, ]
%           [h21, ...            ]
%           [...
%
% given m inputs and q outputs,
% where each hij is an npts x 1 vector.
%
```

B. Closed loop

```
function ht=feedbackmtf(g,mg,ng,fg, h,mh,nh,fh, np)
% function ht=feedbackmtf(g,mg,ng,fg, h,mh,nh,fh, np)
%
%   Combine the frequency response of two systems in a feedback
```

```
loop.
% Two vector inputs and two vector outputs are allowed for:
%
%     r   +      ug
%     --Mg-->o----->G-----------Ng--> yg
%          - |               |
%            |               |
%            Fh              Fg
%            |               |
%            |               |+
%     <--Nh---------H<-------o<--Mh-- d
%      yh                uh      +
%
```

II. Time domain

A. Open loop

```
% function [x, dx]= lsim2(sigma,lambda,b,u,dt,x0, dx0);
%   Generate the time response of the system:
%
%     (d^2)x + 2 sigma dx + lambda x = b*u
%
```

B. Closed loop

```
%function [y,u,x,dx,maxle]= lsim2fb(sigma,lambda,b,g,f,cp,cv,...
%                          Ac,Bc,Cc,Dc, r,dt,x0, dx0);
%
% Generate the time response of a second order system with feedback:
%
%     (d/dt)^2 x + 2 sigma (d/dt)x + lambda x = b u + g f
%     y = cp x + cv (d/dt) x
%
%     (d/dt) w = Ac w + Bc (r-y)
%     u = Cc w + Dc (r-y)
%
```