1992

NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

Numerical Methods for the Analysis of Sampled-Data Systems and for the Computation of System Zeros

Prepared by:	A. Scottedward Hodel, Ph. D.
Academic Rank:	Assistant Professor
Institution	Department of Electrical Engineering Auburn University
NASA/MSFC:	
Office:	Structures and Dynamics Laboratory
Division:	Control System Division
Branch:	Mechanical Systems Control Branch
MSFC Colleague:	D. P. Vallely
Contract No:	NGT-01-008-021 The University of Alabama at Huntsville

1 Introduction

MARSYAS is a computer-aided control system design package for the simulation and analysis of dynamic systems. In the summer of 1991 MARSYAS was updated to allow for the analysis of sampled-data systems in terms of frequency response, stability, etc. This update was continued during the summer of 1992 in order to extend further MARSYAS commands to the study of sampled-data systems. Further work was done to examine the computation of `OPENAT` transfer functions, root-locii and $w-$plane frequency response plots.

2 Sampled-data systems with feed-forward coefficients

Consider a sampled data system whose state-space equations are

$$\dot{x}_c(t) = A_c x_c(t) + B_{cd} y_d(kT) + B_c u(t) \tag{2.1}$$
$$y(t) = C_c x_c(t) + D_{cd} y_d(kT) + D_c u(t) \tag{2.2}$$
$$u_d(t) = C_{dc} x_c(t) + D_{dd} y_d(kT) + D_{dc} u(t) \tag{2.3}$$

with corresponding discrete time subsystem

$$x_d(kT+T) = A_d x_d(kT) + B_d u_d(kT) \tag{2.4}$$
$$y_d(kT) = C_d x_d(kT) + D_d u_d(kT). \tag{2.5}$$

(Purely discrete time inputs $r_d(kT)$ to the discrete time system may be incorporated into the above equations by augmenting the input vector $u(t)$ and the feed-forward matrix D_{dc}.) The feed-forward coefficient D_d causes the states $x_c(t)$ to depend not only on their continuous values through the coefficient matrix A_c, but also on their sampled values through the coefficient matrix $B_{cd}D_dC_{dc}$. As of MARSYAS version 6.0.5, this dependence is dealt with by creating a matrix $ZX = B_{cd}D_dC_{dc}$ in the linearized continuous time dynamics.

Unfortunately, MARSYAS version 6.0.5 does not correctly treat the case where there are feed-forward coefficients in both the continuous time and discrete time blocks (i.e. $D_{dd} \neq 0$ and $D_d \neq 0$). In this case, MARSYAS infers an algebraic loop where there is none. For example, consider the sampled-data system with continuous time block

$$y(t) = y_d(kT) \quad u_d(t) = u(t)/2 - y_d(kT)$$

and discrete time block

$$y_d(kT) = k_d u_d(kT).$$

The code has no (explicit) states, yet because of the direct feed-forward gains (i.e., the "D" matrices) in the discrete and continuous time blocks of the linear system, a "hidden state" associated with the A/D converters in the system manifests itself at sampling times. The MARSYAS-6.0.5-generated system equations are

```
W [  1] = ( 2.500000E-01) * U [  1]
```

while the simulated output values display "spikes" at sampling times and the values between sampling times incorrect. There is a specific order in which the simulation updates must occur at each "scheduled" run of the discrete time systems: (1) Evaluate the continuous time system states and algebraic outputs. (2) Update the discrete time system states and algebraic outputs. (3) the algebraic outputs of the continuous time system to reflect the discrete time system changes. Once these three steps are completed, the simulation may proceed with the next integration step. Similarly, the analysis of discrete time systems must be modified as follows.

Lemma 2.6 *Let a sampled data system be defined by the equations (2.1)-(2.5) with sampling time T given. Then*

$$u_d(kT) = C_{dc} x_c(kT) + D_{dc} u(kT) + D_{dd} y_d(kT - T);$$

that is, the discrete-time output y_d becomes a system state when $D_{dd} \neq 0$.

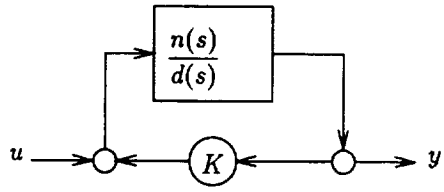

Figure 1: OPENAT linearized system

Based on the above operation, the vector $y_d(t)$ can be used as a vector state that replaces the "ZX" input used in MARSYAS version 6.0.5 as follows. Let $x_{dy}(kT) = y_d(kT - T)$. Then the overall system simplifies to

$$\begin{aligned}
\dot{x}_c(t) &= A_c x_c(t) + B_{cd} y_d(kT) + B_c u(t) \\
y(t) &= C_c x_c(t) + D_{cd} y_d(kT) + D_c u(t) \\
u_d(t) &= C_{dc} x_c(t) + D_{dc} u(t) \\
x_d(kT+T) &= A_d x_d(kT) + B_d u_d(kT) + B_d D_{dd} x_{dy}(kT) \\
x_{dy}(kT+T) &= C_{dc} x_d(kT) + D_d D_{dd} x_{dy}(kT) + D_d u_d(kT) \\
y_d(kT) &= C_d x_d(kT) + D_d D_{dd} x_{dy} + D_d u_d(kT).
\end{aligned}$$

3 OPENAT analysis: options and an example

The OPENAT command allows MARSYAS users to examine system robustness with respect to an individual parameter by breaking a specified signal path in a closed-loop system model and examining the poles/zeros of the newly created open loop system.

The current MARSYAS implementation (and the original MARSYAS implementation) treat the OPENAT command as a special case of the INOUT command. The manual for the original MARSYAS implementation describes an alternate technique for computing the OPENAT transfer function; this alternate technique forms the basis of the root locus calculation, and is thus of interest in both of these calculations. The two techniques are described below. A simple numerical example is given that indicates that algebraic loops (i.e., a non-zero D matrix in a linearized OPENAT system) can render the alternate algorithm inaccurate; it is advised that the INOUT approach be used in both OPENAT and root locus calculations.

The OPENAT command can be summarized as follows. Given a closed loop continuous time system $\dot{x} = f(x, w)$ with algebraic constraint $0 = g(x, w)$, OPENAT k selects a gain value in the system and linearizes about an operating point in order to obtain the SISO system shown in Figure 1. The OPENAT command identifies the poles and zeros of the transfer function $n(s)/d(s)$. The INOUT algorithm computes the poles and zeros by computing the matrices (A, B, C, D) that characterized a state-space realization of $n(s)/d(s)$ and then computing the (finite) generalized eigenvalues of the matrix pencil $\left(\begin{bmatrix} -A & -B \\ C & D \end{bmatrix} - \lambda \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}\right)$.

The alternate *closed-loop matrix method* is as follows For a fixed linearization, let $A(k)$ be the system Jacobian matrix evaluated with $K = k$; i.e. $A(k) = (A + k/(1 - Dk)BC)$. Then $\det(sI - A(k)) = a(k)(d(s) - kn(s))$ where $a(k)^{-1}$ is the leading coefficient of $(d(s) - kn(s))$. Clearly, the poles of the OPENAT system are the eigenvalues of $A(0)$, since $\det(sI - A(0)) = a(0)(d(s) - 0 \cdot n(s)) = d(s)$.

In order to obtain the zeros of the system, it is required to compute the algebraic gain D and the closed loop matrix $A(1) = (A(0) + 1/(1 - D)BC)$. Let $\hat{B}$, $\hat{C}$ be a column and row vector, respectively, such that $\hat{B}\hat{C} = (D - 1)(A(1) - A(0)$; then the zeros of the OPENAT transfer function may be obtained by computing

INOUT method			Closed-loop matrix method		
ϵ	$\lvert\lambda - \hat{\lambda}\rvert$	$\lvert\hat{D} - D\rvert$	ϵ	$\lvert\lambda - \hat{\lambda}\rvert$	$\lvert\hat{D}_2 - \hat{D}_1\rvert$
1	≈1.0e-13	1.9984e-14	1	1.0e-13	2*1.9984e-14
0.1000	≈1.0e-14	1.1102e-14	0.1000	≈1.0e-12	2*1.1102e-14
0.0100	≈1.0e-12	9.9920e-15	0.0100	≈1.0e-10	2*9.9920e-15
1.0000e-03	≈1.0e-12	9.9920e-15	1.0000e-03	≈1.0e-09	2*9.9920e-15
1.0000e-04	≈1.0e-11	9.9920e-15	1.0000e-04	≈1.0e-07	2*9.9920e-15
1.0000e-05	≈1.0e-11	9.9920e-15	1.0000e-05	≈1.0e-06	2*9.9920e-15
1.0000e-06	≈1.0e-10	9.9920e-15	1.0000e-06	≈1.0e-04	2*9.9920e-15
1.0000e-07	≈1.0e-09	9.9920e-15	1.0000e-07	≈1.0e-03	2*9.9920e-15

Figure 2: Results of tests of openat algorithms

the finite generalized eigenvalues of the matrix pencil $\left(\begin{bmatrix} -A & -\hat{B} \\ \hat{C} & D \end{bmatrix} - \lambda \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}\right)$. This is the *closed-loop matrix method* discussed in the manual for the original MARSYAS implementation.

The discussion of the closed-loop matrix method in the original MARSYAS manual did not correctly treat the case of $D \neq 0$. If $D \neq 0$, in particular, if $D \approx 1$, then the **INOUT** method is clearly superior to the closed-loop matrix method, as shown in the following example.

Example 3.1 Let the **OPENAT** linearized system have coefficient matrices $A = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $C = \begin{bmatrix} 2 & 1 \end{bmatrix}$, and $D = 1 + \epsilon$. The exact system zeros can be found to be

$$\lambda = \left(1 + \frac{2}{1+\epsilon}\right) \pm \sqrt{\left(1 + \frac{2}{1+\epsilon}\right)^2 - \left(2 + \frac{7}{1+\epsilon}\right)}.$$

Observe that as $\epsilon \to 0$, the zeros coincide at $\lambda = 3$.

A computed linearization $(\hat{A}, \hat{B}, \hat{C}, \hat{D}) = (A, B, C, D)$ will typically have roundoff noise in each matrix in proportion to the matrix norm; i.e., $\left\|\hat{A} - A\right\| \leq \mu \|A\|$ where μ is a function of machine precision and the algorithm condition. The example was selected so that matrix entries were "balanced;" that is, the matrix entries were of approximately the same magnitude so that round-off effects on problem condition would be reduced. For the purposes of a numerical experiment, small error ($\approx 10^{-14}$) was deliberately introduced into the problem in order to examine algorithm sensitivity. Since the computed value of **Ab** is near to the correct computed value, this method yields good results.

The closed-loop matrix method involves computing $A(0)$, $A(1)$, and the open-loop D-matrix. it was *not* assumed that the gain D involved in computing $A(1) = A - 1/(D-1)BC$ is exactly the same as that gained from an **INOUT** linearization; i.e., **INOUT** computes $(\hat{A}, \hat{B}, \hat{C}, \hat{D}_1) = (A, B, C, \hat{D})$, while $A(1)$ was computed as $A(1) = \hat{A} + 1/(\hat{D}_2 - 1)\hat{B}\hat{C}$. The respective results are shown in Figure 2. It is clear that the **INOUT** method yields superior accuracy; it is important to recognize that the difference in these two methods is a result of the difference between $1/(\hat{D}_1 - 1)$ and $1/(\hat{D}_2 - 1)$; since the estimated values of $\hat{B}$ and $\hat{C}$ in the closed loop matrix method will be scaled by the ratio $(\hat{D}_1 - 1)/(\hat{D}_2 - 1)$ from their "correct" values, it is to be expected that the closed-loop matrix method will yield poor results when $D \approx 1$.

□

On the basis of the above example and analysis, it is recommended that both **OPENAT** and root locus calculations be performed on the basis of the matrices (A, B, C, D) obtained from an **INOUT** linearization of the appropriate system. FORTRAN code for the **mvzero** routine and the associated numerical balancing procedure [2] have beed delivered to John Tiller, BCSS, this summer for use in MARSYAS.

The calculation of **OPENAT** transfer functions and root locus data become much more complicated in a sampled-data system. If the **OPENAT** operation is performed on a gain k in the discrete time portion of the system, then the **OPENAT** function may be executed in the normal fashion on the equivalent discrete time system. However, if the **OPENAT** is performed on a gain k in the continuous time part of the system, then the **OPENAT** function ceases to have a clear meaning. If an artificial *continuous time* input u and output y are placed around the gain, then there is no rational transfer function from u to y because of the time-varying dependence on the discrete time subsystem. Instead, *discrete time* input-output pair is inserted about the *continuous time* gain element K.

This system is not amenable to an **OPENAT** style analysis, since the transfer function varies in a transcendental (non-algebraic) fashion with the gain value K; closed form perturbation analysis of the equivalent discrete time system is intractable since the matrices A and BC do not commute in general. ($e^{ABC} = e^A e^{BC} = e^{BC} e^A \iff A(BC) = (BC)A$.) Hence a single "open-loop" **OPENAT** transfer function calculated with $K = 0$ is not meaningful to the user. However, a "transcendental root-locus" from u to y may be computed by computing the poles *and zeros* of an equivalent discrete-time plant for various values of k.

Closed-form analysis of transcendental root-locii is hindered by the property that $e^{(A_c + KB_cD_c)t}$ cannot be computed from $e^{A_c t}$ and $e^{(B_cC_c)t}$ except when A_c commutes with B_cC_c; this is clearly not the case in practice. It may be possible to gain some norm-bounds on the closed-loop eigenvalues of the overall discrete-time system in terms of the feedback gain K, but this is an open question.

4 W-plane analysis

MARSYAS currently provides z-plane frequency response plots of discrete-time systems. w-plane analysis is often used in practice to allow designers to employ continuous time design techniques to discrete-time systems; see, e.g., [1]. $G(z) \rightarrow G(w)$ by $w = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$. s-plane, z-plane, and w-plane frequency response plots are related by $z = e^{sT}$ and $w = \frac{2}{T}\frac{e^{sT}-1}{e^{sT}+1} = \frac{2}{T}\tanh\left(\frac{sT}{2}\right)$. So as $s = j\omega$ follows the imaginary axis, $z = e^{j\omega T}$ follows the unit circle (periodic with period $2\pi/T$), and

$$w = \frac{2}{T}\tanh\left(\frac{j\omega T}{2}\right) = \frac{2j}{T}\tan\left(\frac{\omega T}{2}\right)$$

follows the entire imaginary axis each time $z = e^{j\omega T}$ rotates about the unit circle The inverse bilinear transform from w to z is $z = \frac{1 + wT/2}{1 - wT/2}$; hence the Nyquist plots in the z and w plane will be identical. Magnitude and phase information in the z and w-planes will be the same except for a "warping" of the w-plane frequency variable $\nu \triangleq \tan(\omega T/2)$.

References

[1] G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital Control of Dynamic Systems.* Addison-Wesley, 2nd edition, 1990.

[2] A. Scottedward Hodel and John Tiller. Computation of system zeros with balancing. In *Proceedings of the 29th Allerton Conference on Comm., Contr., and Computing*, Monticello, Il., Oct 2-4 1991.