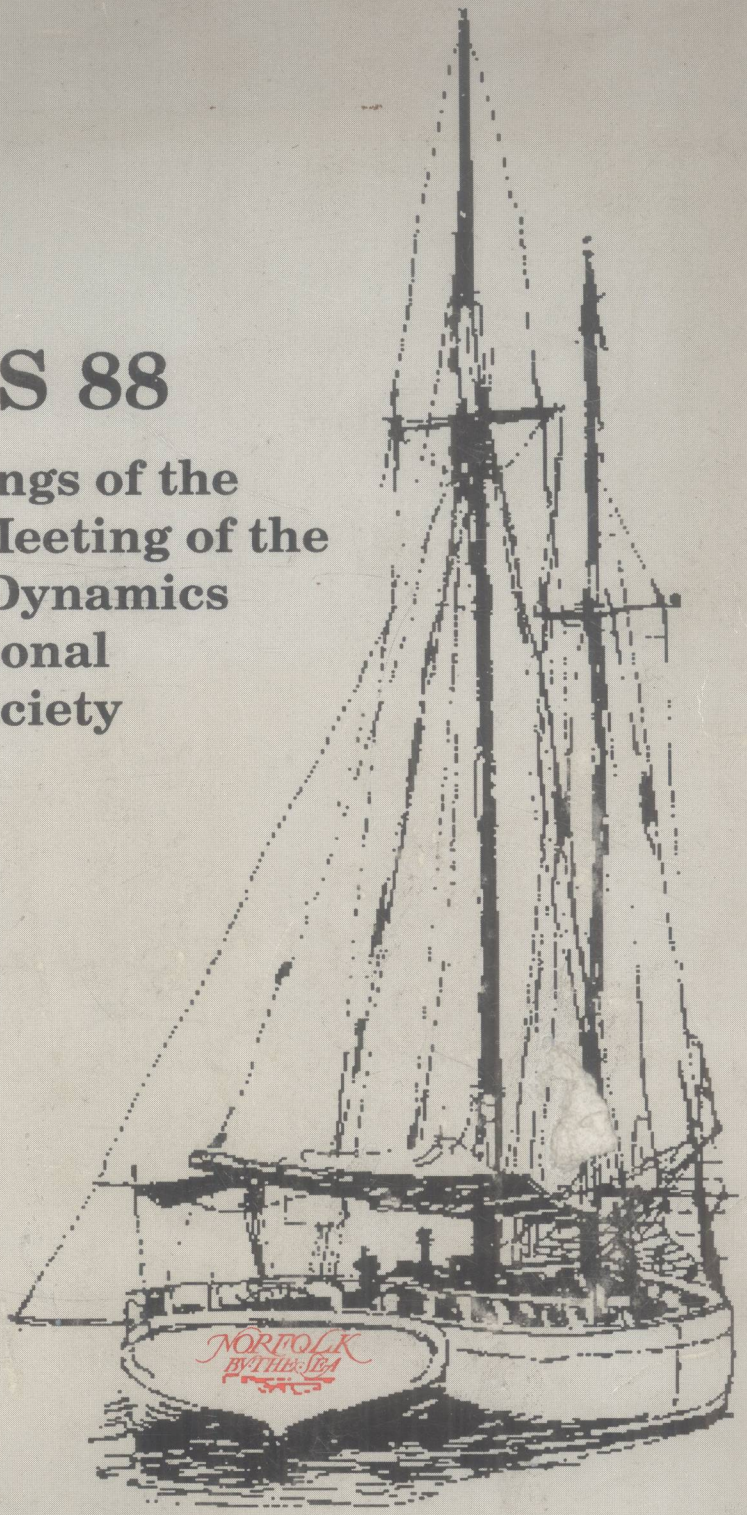
ADIUS 88

Proceedings of the Annual Meeting of the Applied Dynamics International Users' Society



Omni International Hotel
Norfolk, Virginia
June 19-22, 1988

PROCEEDINGS OF ADIUS 88

Annual Meeting of ADIUS, Inc.
(Applied Dynamics International Users' Society)

Omni International Hotel
Norfolk, Virginia
June 19–22, 1988

ADIUS DIRECTORS
1988—1989

Chairperson	**Robert Hickman** Aerojet Ordnance Company 2521 Michelle Drive Tustin, CA 92680 (714) 730-6004
Vice-Chairperson/ Program Chairperson	**Mark Bailey** U.S. Naval Air Test Center Code SY 32A, Building 2035 Patuxent River, MD 20670 (301) 862-7601
Secretary/Treasurer	**Linda Schiller** Applied Dynamics International, Inc. 3800 Stone School Road Ann Arbor, MI 48108-2499 (313) 973-1300
Director	**John Collins** Simulation and Computer Consultants Lavender Cottage, Mallows Yard Bozeat, Northamptonshire United Kingdom 0933-664822
Director	**Gottfried Hogh** Ford Motor Company Scientific Research Lab Control Systems Dept. P.O. Box 2053 Dearborn, MI 48121 (313) 337-5679
Director	**Wulfgang Wulff** Brookhaven National Laboratory Dept. of Nuclear Energy, Bldg. 475B Upton, NY 11973 (516) 282-2608

ADIUS 88
CONFERENCE PROGRAM

Sunday, June 19, 1988

3:00— 7:00 PM	Registration and ADI Reception

Monday, June 20, 1988

8:00— 9:00 AM	Continental Breakfast
9:00— 9:15 AM	Welcome and Announcements
9:15—10:15 AM	Keynote Speaker: Charles Cockrell, NASA Langley Research Center— *Computers as a Design Tool*
10:15—10:45 AM	Break
10:45—11:15 AM	Jim Ledin, U.S. Pacific Missile Test Center— *TRAN: A Program to Generate Transfer Function Models in ADSIM*
11:15—12:00 PM	Barry Powell, Ford Motor Company— *Real Time Integration of Orifice Mass Flow Rate Equations*
12:00— 1:00 PM	Buffet Luncheon
1:00— 1:10 PM	Board Buses for NASA Langley Research Center Tour
2:00— 4:00 PM	NASA Langley Research Center Tour
4:45— 5:30 PM	ADIUS Board Meeting

ADIUS 88
CONFERENCE PROGRAM

Tuesday, June 21, 1988

8:00— 9:00 AM	Continental Breakfast
9:00— 9:45 AM	Jim Annos, U.S. Naval Weapons Center—*Managing Multiple AD 100s*
9:45—10:15 AM	Norm Rodewald, U.S. Pacific Missile Test Center— *MPS 10 Under RSX-11M+*
10:15—10:30 AM	Break
10:30—11:15 AM	Loren Slafer, Space and Communications Group, Hughes Aircraft Company—*The Role of Simulation in Systems Engineering and its Application to the New HS601 Spacecraft Development*
11:15—12:00 PM	Tom Luther, Applied Dynamics International— *Digital's New Product Offerings*
12:00— 1:15 PM	Buffet Luncheon
1:15— 2:00 PM	Gene Graber and Ed Fadden, Applied Dynamics International— *ADI Overview and Direction*
2:00— 2:30 PM	John Collins, Simulation and Computer Consultants— *AAU—The ADSIM Array Utility*
2:30— 3:00 PM	Gottfried Hogh, Ford Motor Company— *GRF, An MPS-10 and ADSIM Linkable Graphics Program*
3:00— 3:15 PM	Break
3:15— 4:00 PM	Robert Hickman, Aerojet Ordnance Company—*Interfacing the ADI AD 100 to the APTEC IOC-24 Communications Computer with Graphics and Mass Storage Peripherals*
4:00— 5:00 PM	Workshops
6:30—10:00 PM	Conference Banquet

ADIUS 88
CONFERENCE PROGRAM

Wednesday, June 22, 1988

8:00— 9:00 AM	Continental Breakfast
9:00— 9:45 AM	John Defenbaugh, Sundstrand Corporation— *A Real-Time Simulation for a Vapor Cycle Cooling System*
9:45—10:15 AM	Larry Michaels, Applied Dynamics International— *FORTRAN Programming for the AD 100*
10:15—10:30 AM	Break
10:30—11:15 AM	ADIUS Business Meeting
11:15—12:00 PM	Edward J. Fadden, Applied Dynamics International— *Questions and Feedback*

TRAN: A PROGRAM TO GENERATE TRANSFER FUNCTION MODELS IN ADSIM

Jim Ledin
U.S. Pacific Missile Test Center
Point Mugu, California

TRAN: A Program to Generate Transfer Function Models in ADSIM

Jim Ledin
Pacific Missile Test Center
Point Mugu, California

Abstract

TRAN is a programming utility that takes as input an interactive description of an *s*-domain transfer function and produces as output a text file containing an optimized ADSIM source code model of the transfer function. An *s*-domain representation of the transfer function along with the time and date of model creation are included as comments. An optional one line user comment may also be entered.

TRAN is written in VAX C. The program source and executable files are available in the public domain. The executable file will work on any VAX system with VMS version 4.6 or higher regardless of whether or not VAX C is installed.

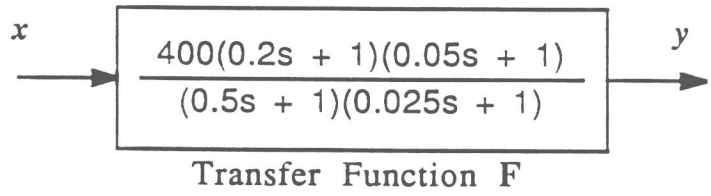
Introduction

In the simulation of dynamic systems, components are often modelled in the form of transfer functions in the *s*-domain. A transfer function is a Laplace transform of the ordinary differential equations that describe the ratio of output quantity to input quantity for a system functional block.

Transfer functions of linear system components are often in the form of a ratio of two polynomials in *s*. These polynomials may each be factored into a set of terms that are multiplied together. Also, transfer functions of components used in real systems usually have a numerator whose order is less than or equal to that of the denominator. TRAN can convert transfer functions that satisfy these conditions to ADSIM code.

ADSIM is a simulation programming language developed by Applied Dynamics International (ADI) for use on the ADI AD100 computer.

Figure 1 below shows an example of a transfer function that is suitable for TRAN. The polynomial ratio in the box is the transfer function, which will be referred to as **F**. The input to the system component is x and the output is y. Algebraically, $y = x\mathbf{F}$.



Transfer Function **F**

Figure 1

ADSIM cannot directly use transfer functions to model system components. In order for ADSIM to model the dynamics of a transfer function it must be converted to a set of first order differential equations. This conversion is the function of TRAN.

Algorithm Description

The process of converting a transfer function to differential equations consists of the following steps. The example transfer function **F** of Figure 1 will be used to demonstrate the algorithm.

1. Multiply out the factors (if there are more than one) in the transfer function so that the numerator and denominator are in polynomial form. Then divide all coefficients by the coefficient of the highest power of s in the numerator.

   The results of this step are shown in figure 2.

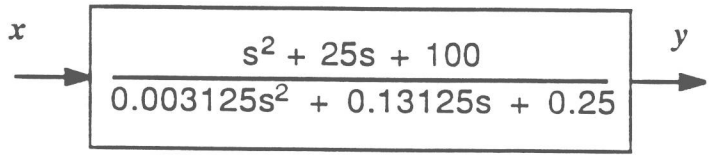


Figure 2

2. Split the transfer function of Figure 2 into two separate transfer functions with an intermediate state variable between them. The first transfer function is unity divided by the denominator of Figure 2. Its input is x and its output is the new intermediate variable that will be called z. This transfer function will be referred to as $\mathbf{F_1}$. The second transfer function is the numerator of Figure 2 and will be called $\mathbf{F_2}$. Its input is z and its output is y.

   Figure 3 shows the result of this manipulation.

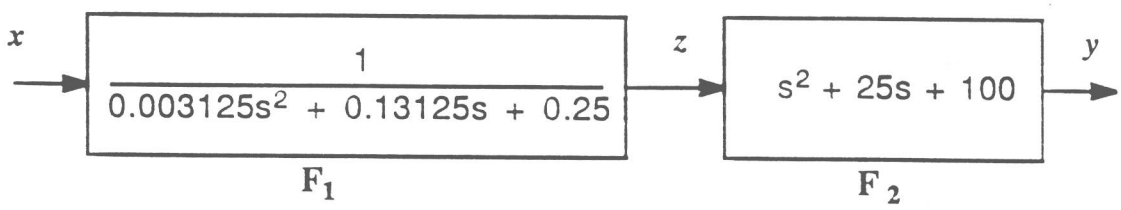


Figure 3

3. Convert $\mathbf{F_1}$ to a differential equation of order n where n is the order of s in $\mathbf{F_1}$.

   Observe that $z = x\mathbf{F_1}$. Rearranging this gives $x = z / \mathbf{F_1}$. Multiplying this expression out with the example $\mathbf{F_1}$ gives equation 1.

$$x = 0.003125s^2z + 0.13125sz + 0.25z \qquad (1)$$

   When a state variable z is multiplied by s^m it is equivalent to taking the mth order time derivative of z. Applying this fact to equation 1 gives the following differential equation:

$$x = 0.003125z'' + 0.13125z' + 0.25z \qquad (2)$$

   Solving equation 2 for the highest derivative of z produces a differential equation in standard form as shown in equation 3.

$$z'' = 320x - 42z' - 80z \qquad (3)$$

4. Convert this differential equation to a set of n first order differential equations.

   To do this the variable z and its derivatives will be given new names. z will become $z0$. The first derivative of z will become $z1$ and so on. Replacing these variables in equation 3 and adding an equation to solve for $z0$ gives the set of two differential equations 4 and 5.

$$z1' = 320x - 42z1 - 80z0 \quad (4)$$
$$z0' = z1 \quad (5)$$

   These equations are now in a form that can be coded in ADSIM.

5. Convert $\mathbf{F_2}$ into one algebraic equation that gives the output y as the sum of constant values multiplied by z and its derivatives.

   Equation 6 shows $\mathbf{F_2}$ multiplied through by its input, z.

$$y = s^2z + 25sz + 100z \quad (6)$$

   Again using the fact that each multiplication by s is equivalent to taking another time derivative, equation 6 is converted to a differential equation that gives y in terms of z and its derivatives as shown in equation 7.

$$y = z'' + 25z' + 100z \quad (7)$$

   Replacing z and its derivatives by the variables defined in step 4 above gives equation 8, which is in a form that can be coded in ADSIM.

$$y = z1' + 25z1 + 100z0 \quad (8)$$

6. TRAN carries out some optimizations to ensure that the ADSIM model generated is as computationally efficient as possible. In general the model generated by TRAN will have the minimum number of variables, integrations, multiplications, additions, subtractions, and assignments that

can be used to create an accurate model. There will never be any divisions.

The optimization steps taken by TRAN are as follows:

A. If the transfer function numerator contains only one power of s then then the variable zm (where m is the order of the numerator) is replaced by the variable y.

B. Variables that are multiplied by the same constant value in an equation are grouped together inside parentheses.

As neither of these conditions applies to the example, the combined equations 4, 5, and 8 give the complete transfer function model. These are shown in equations 9, 10 and 11 in ADSIM source code syntax as they would be written in the output file.

```
z1' = 320*x - 42*z1 - 80*z0          (9)
z0' = z1                             (10)
y   = z1' + 25*z1 + 100*z0           (11)
```

Appendix B contains the complete output file generated by this example.

Initial conditions for the differential equations in the model default to zero. If necessary, these may be changed by editing the output file and defining other initial conditions.

In general the algorithm will work with positive or negative coefficients in transfer functions of any order so long as the numerator order is less than or equal to the denominator order. High order transfer functions may run into a problem with lines that exceed the ADSIM limit of 80 characters per line. This situation may be remedied by editing the output file and splitting up long lines and inserting continuation characters as needed.

Program Operation

TRAN operates by prompting the user for all necessary information. Execution is started by issuing the DCL command RUN TRAN (assuming TRAN.EXE is in the default directory). The first prompt is for a model name up to 24 characters. The output file will also have

this name with a .ADS file type. Next a comment up to 78 characters will be prompted for. An empty carriage return may be entered if no comment is desired.

Next the numerator must be described. The first prompt is for the number of factors that are to be multiplied together. Then for each factor the following information must be entered:

A. The highest power of s in the factor.

B. The coefficients for each power of s from the highest power down to a constant.

If a power of s is not represented in a factor then a zero or an empty carriage return is entered.

The denominator is then entered in an identical fashion.

If an invalid response to a prompt is detected the prompt will be repeated.

After all responses are entered an output file will be created and the program will exit. If desired, this file may be edited with a standard text editor to include initial conditions for the state variables or more comments. This file may then be included in a main ADSIM program file or inserted into a library using the utility ADSIMLIB.

Appendix A contains a sample program run showing how a model of the example transfer function F would be created. Appendix B is a listing of the output file created by that program run.

Program Validation

TRAN was tested by selecting a group of 27 simple transfer functions that exercise all conditional paths through the ADSIM code-generating part of the program. The TRAN output models were then checked against hand calculated solutions.

Several other transfer functions were also tested. The most complex of these had a denominator order of 20 and a numerator order of 20.

No cases of incorrect ADSIM code generation have been found for any combination of input data. If the program receives bad input the worst that happens is that no output file is generated.

However, as software engineering is still an inexact science, it is possible that TRAN may contain some bugs. A report of any problems would be sincerely appreciated. Please call or mail bug reports, questions, or comments to the address given below.

Program Availability

A copy of TRAN source and executable files and related documentation may be obtained by sending a DEC compatible tape (reel or TK50) and a prepaid return mailer to:

Jim Ledin
Pacific Missile Test Center
Code 1056
Point Mugu, CA 93042

Phone: (805) 989-1388

All tapes will be in VMS Backup format. Reel tapes will be recorded at 1600 BPI.

If you have any interesting files that you would like to share with other users then put them on the tape you send in and they will be put on all copies sent out to other users. Please include a README.1ST text file describing the files and their use along with a written note.

Summary

TRAN is a useful tool for the simulation of system components that are modeled as *s*-domain transfer functions. It produces accurate models in ADSIM source code with the original *s*-domain transfer function included as a comment. The models that TRAN produces are as efficient as possible. TRAN performs some of the time consuming and error-prone calculations that need to be done in producing a system simulation and partially automates the process of generating ADSIM code.

References

Applied Dynamics International, "Reference Manual, ADSIM Version 5.0, Volume 1", Applied Dynamics International, Ann Arbor, Mich., 1987.

Franklin, Gene F., J. David Powell, and Abbas-Emami-Naeni: "Feedback Control of Dynamic Systems", Addison Wesley, Reading, Mass., 1986.

Holbrook, James G.: "Laplace Transforms for Electronic Engineers", Pergamon Press, New York, 1959.

McCollum, Paul A. and Buck F. Brown: "Laplace Transform Tables and Theorems", Holt, Rinehart and Winston, New York, 1965.

Appendix A - Sample Program Run

This appendix contains the terminal dialogue required to generate a model of the transfer function **F** used as an example in this paper. All responses entered by the user are shown in **boldface**.

```
$ RUN TRAN

     ************************************************
     * TRAN ** ADSIM Transfer Function Model Generator *
     ************************************************

Enter the model name [24 characters max]: example

Enter a comment up to 78 characters (time & date not needed):
By Jim Ledin - Example of a transfer function

Input the numerator -

Enter the number of polynomial factors: 3

Enter the order of polynomial factor number 1: 0
Enter the constant value for factor 1: 400

Enter the order of polynomial factor number 2: 1
Enter the coefficient of s for factor 2: .2
Enter the constant value for factor 2: 1

Enter the order of polynomial factor number 3: 1
Enter the coefficient of s for factor 3: .05
Enter the constant value for factor 3: 1

Input the denominator -

Enter the number of polynomial factors : 2

Enter the order of polynomial factor number 1: 1
Enter the coefficient of s for factor 1: .5
Enter the constant value for factor 1: 1

Enter the order of polynomial factor number 2: 1
Enter the coefficient of s for factor 2: .025
Enter the constant value for factor 2: 1

Model "example" completed. Output file is EXAMPLE.ADS

$
```

Appendix B - Sample Program Output

This is a listing of the output file created from the input dialogue of Appendix A.

```
!************************************************************
! Model example
! Created 10-JUN-1988 10:42
!
! By Jim Ledin - Example of a transfer function
!
! The transfer function modelled is:
!
! y    (400)(0.2*s + 1)(0.05*s + 1)
! - = ----------------------------
! x      (0.5*s + 1)(0.025*s + 1)
!************************************************************

MODEL example(y = x)

z1'    = 320*x - 42*z1 - 80*z0
z0'    = z1

y      = z1' + 25*z1 + 100*z0

END MODEL
```