TMS320 Algorithm Debugging Techniques

APPLICATION REPORT: SPRA084

Peter Robinson
Regional Technology Center
Waltham, Massachusetts
Texas Instruments

Digital Signal Processing Solutions

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

TMS320 Algorithm Debugging Techniques

Abstract

As DSP tasks become increasingly complex, the ability to debug a DSP algorithm as a discrete transfer function is becoming more critical.

This report analyzes a technique for debugging a coded transfer function in a purely software environment using traditional analog troubleshooting methods applied to analyze and debug DSP algorithms using Texas Instruments (TI™). TMS320.

Topics discussed include:

- An explanation of data logging and its advantages
- How to create an input file with Lotus and save it as ASCII
- How to run your program with data logging
- How to plot output data
- Examples of highly complex algorithms.

Accompanying the text are:

- Diagrams of DSP code division
- Sample input text editor file
- Spread sheet calculations of sine waves
- Lotus command structures
- Sample four Op-Amp block diagram
- Example of a four second-order IIR structure.

Product Support

World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. New users must register with TI&ME before they can access the data sheet archive. TI&ME allows users to build custom information pages and receive new product updates automatically via email.

Email

For technical issues or clarification on switching products, please send a detailed email to *dsph@ti.com.* Questions receive prompt attention and are usually answered within one business day.

Introduction

Debugging a DSP algorithm is becoming more of an issue as our DSP tasks increase in complexity. It is easy enough to verify program flow for a filter or FFT, but it is an entirely different task to evaluate a discrete transfer function, H(n), over time and frequency.

In this report, a technique for debugging a coded transfer function, h(n), in a purely software environment is presented. The technique shows how traditional analog trouble-shooting methods can be applied to analyze and debug DSP algorithms on an IBM/PC-based TMS320C2x Software Development system and/or any of the TMS320 simulators.

Data Logging

Data Logging is the ability to simulate an I/O device by using DOS files. Many, if not all, IBM/PC-based DSP software development tools and algorithm development packages, such as simulators, offer this ability or feature. In the case of the Texas Instruments TMS320C2x SWDS, the IN and OUT instructions can be equated or tied to a DOS file. Using files to simulate I/O devices can be extremely useful in analyzing performance of a transfer function such as a filter.

Assume that you have written an algorithm for the TMS320C25 for a first-order IIR filter, comprising one second-order element. Figure 1 shows how the code can be divided into four sections:

1. An initialization section where the coefficients are moved from program to data memory,
2. The aquisition of data, IN X0,PA1,
3. The IIR section, and
4. The output of y(n) via an OUT YN,PA0 (with a branch back to the IN instruction).

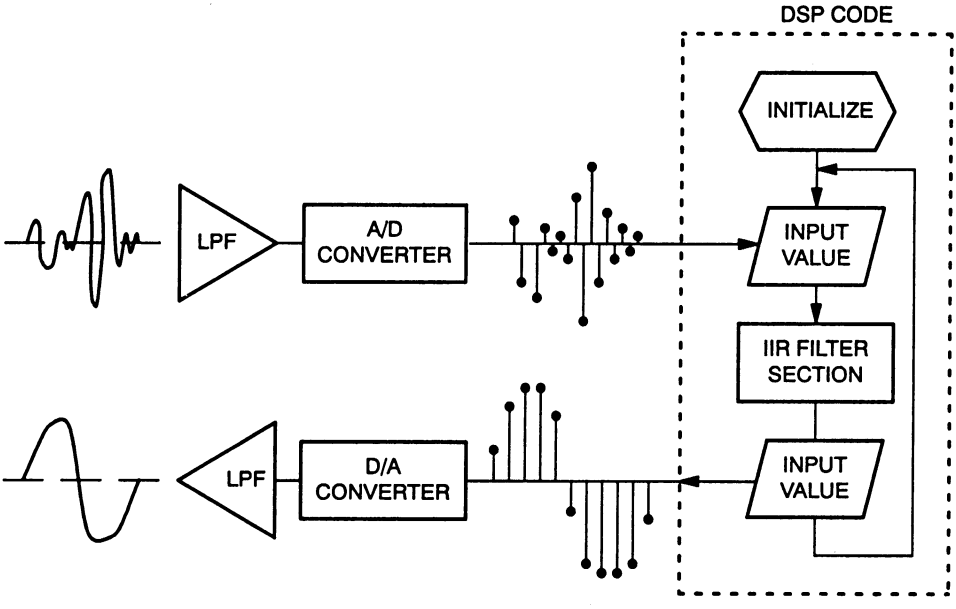


Figure 1. DSP Code Division

If this is an analog system made up of an op-amp, you can verify performance or response by sweeping the filter over frequency and observing the output on a spectrum analyzer. Since the example has no target system (this is entirely a software simulation), a file is used for both input (the A/D) and output (the D/A). To use the data logging feature as noted, you will need to create an input file.

Creating the Input File with Lotus

Before you start, determine the format of the INPUT DOS file structure. In the case of the TMS320C2x SWDS, input files must be represented by four ASCII HEX characters (a 16-bit field), followed by a carriage return and line feed, <CR><LF>. An example for a file containing an impulse, emulating a 12-bit, two's-complement A/D, is shown in Figure 2.

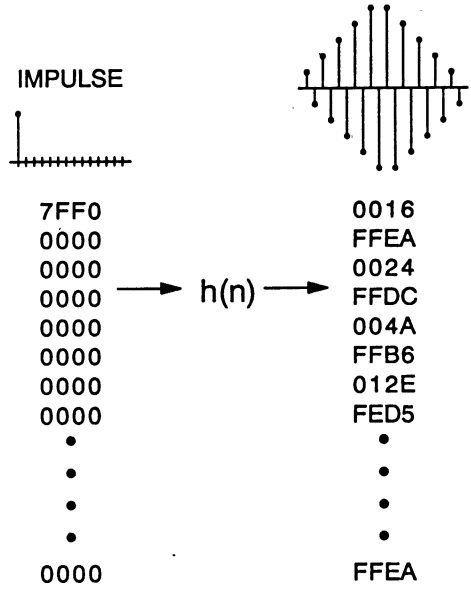
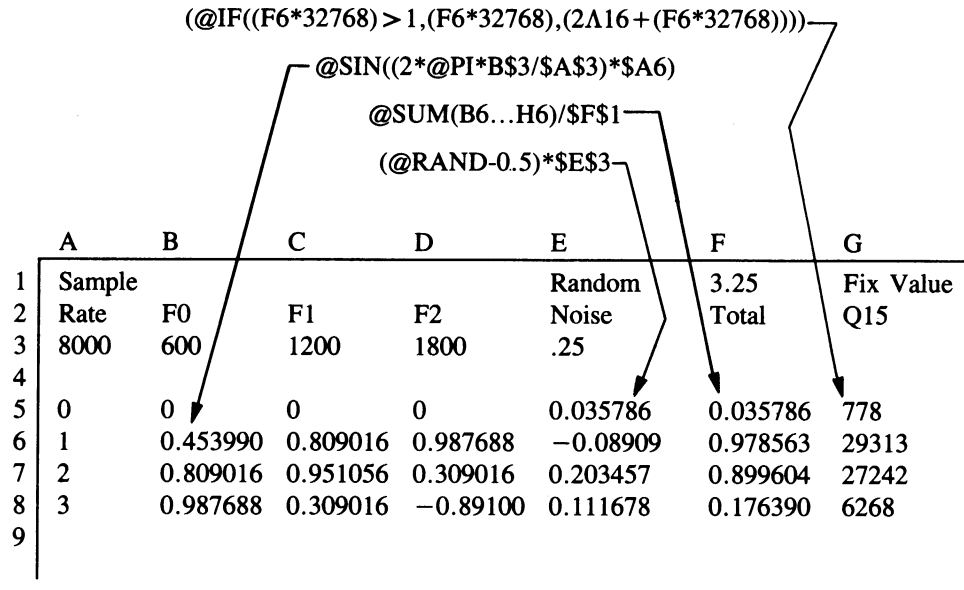


Figure 2. Input Text Editor File

Note: HEX values in this report are represented in Q15 format. In Q15 format, −1 = 8000h for both a 12- and 16-bit field, +1 = 7FF0h for a 12-bit field and 7FFFh for a 16-bit field. For further information on Qn notation, see Section 5.5.5 on page 5-33 of Reference [1].

The file shown in Figure 2 can easily be generated with a text editor and produces a near-ideal impulse response. This is seldom achieved with analog systems. However, what if you wanted to inject a more complex signal, such as several sine waves and/or random noise? Here, the generation of the input file can become a monumental task. One method of file generation, presented here, uses Lotus 1-2-3, a software package found on most PCs. Because Lotus 1-2-3 is a spreadsheet calculator, you can use a column to denote the input sequence, X(0), X(1), X(3) etc. Adjacent columns can be set up to calculate the desired x(N) values. An example of a spread sheet, which calculates three sine waves with a predetermined noise signal added in, is shown in Figure 3.



	A	B	C	D	E	F	G
1	Sample				Random	3.25	Fix Value
2	Rate	F0	F1	F2	Noise	Total	Q15
3	8000	600	1200	1800	.25		
4							
5	0	0	0	0	0.035786	0.035786	778
6	1	0.453990	0.809016	0.987688	−0.08909	0.978563	29313
7	2	0.809016	0.951056	0.309016	0.203457	0.899604	27242
8	3	0.987688	0.309016	−0.89100	0.111678	0.176390	6268
9							

Figure 3. Spread Sheet Calculation of Three Sine Waves with Added Noise Signal

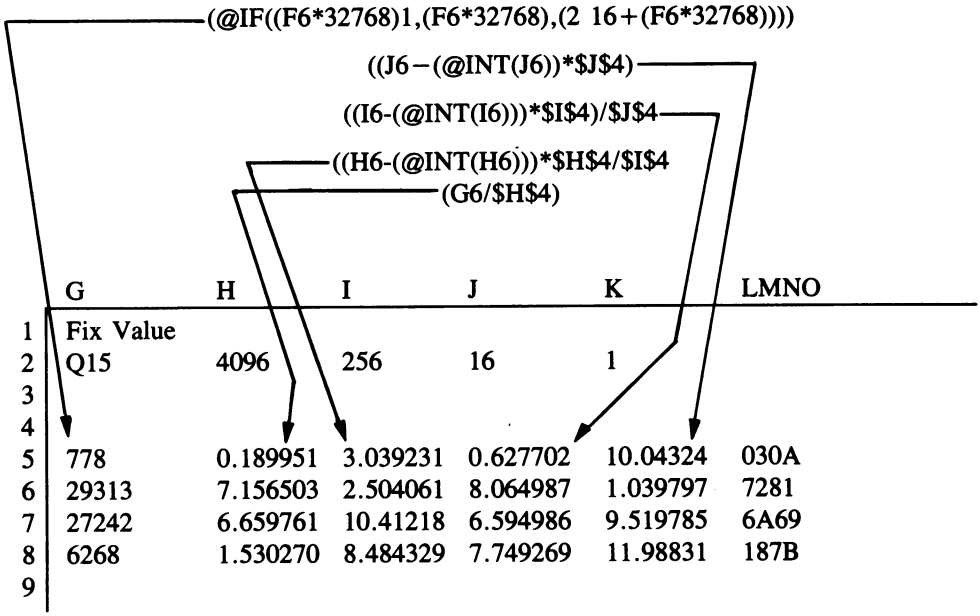
This spreadsheet approach allows you to specify a wide range of input conditions. You can add columns by copying previous column data and can extend the length of the array by copying rows. If you are going to use Lotus's random number generator for adding noise to your signal, you should note the following:

- Lotus's random number function generates uniform numbers or noise.
- Lotus appends an existing file when the print-to-file option is used.
- Each time you recalculate the spreadsheet with a random function, a new random seed is used, which will generate a new random array.

You can use this last feature to your advantage by writing to a file, recalculating the spreadsheet, then writing again. This sequence permits large input files with uniquely different file segments. You must exercise caution, however, to insure that all frequencies end at a zero crossing; if they don't, unwanted discontinuities will be introduced.

The end result of the above process is a column of decimal numbers scaled between −1 and +1. Using the Lotus graph utility, you can plot one or several full cycles of the wave form. When you get the desired results, you must convert the column data to a Qn HEX value of the form in the note that follows Figure 2. You can use the command structure noted in Figure 4.

@CHOOSE((@INT(H6)),"0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F","0")



	G	H	I	J	K	LMNO
1	Fix Value					
2	Q15	4096	256	16	1	
3						
4						
5	778	0.189951	3.039231	0.627702	10.04324	030A
6	29313	7.156503	2.504061	8.064987	1.039797	7281
7	27242	6.659761	10.41218	6.594986	9.519785	6A69
8	6268	1.530270	8.484329	7.749269	11.98831	187B
9						

Figure 4. Lotus Command Structure

Saving the Lotus 1-2-3 File as an ASCII Text File

You can save a spreadsheet range as an ASCII file by using the Lotus PRINT utility. If you select the FILE option under PRINT, with the left column margin set to zero, Lotus will write the columns to a file with the required <CR><LF> ending. Once this is written out, you must edit out the blank lines between pages so that the file content is continuous. You must also ensure that the data in the file is fully left-justified and starts at the top of the file (no blank lines). You can do this by editing the file with a text editor.

Running Your Program with Data Logging

After completion of the preceding steps, you can execute your program with data logging enabled. Name the input file INPUT.DAT, and the output file OUTPUT.DAT. Make certain the created LOTUS file is in the correct directory and is referenced by the correct file name; the DOS file name is the same as the data log file name."

Since each development tool has a unique procedure for enabling data logging, it is assumed that you know how to initialize file I/O. The data logging feature of the TMS320C25 SWDS is documented on page 3-29 of Reference [2], and the TMS320C30 Simulator on page 3-14 of Reference [3].

When your program is executed, the disk drive light will start to blink. With the TMS320C25 SWDS, each disk access is equal to 64 samples being written and/or read, while on the TMS320 simulators, there is one disk access for each sample. To control the number of samples written to and/or read from the file, you can either

1. Manually count the number of times the drive light flashes,
2. In the case of the TMS320C25 SWDS, use program control techniques, such as break points with count values (see page 4-108 of Reference [2], or
3. In the case of the TMS320C30 simulator, use the LOOP command (see page 5-103 of Reference [3].

When the above process is finished, there will be a new file in the working directory named OUTPUT.DAT (the name previously given to your file). This file contains a listing of *N* ASCII HEX character strings in which each line represents one output time sample, y(n). In the case of the first- and second-generation development tools, the file structure is identical to the input file structure: four HEX values represent a 16-bit field in which the sign bit is left-justified. However, the TMS320C30 Simulator outputs, and also requires for input, a form similar to the HEX syntax used in the C programming language. This is a 10-character HEX field with 0x as the first two characters. The impulse value shown in Figure 2 would be written as 0x7FF00000 for the TMS320C30 Simulator. You can generate an input file form, using Lotus 1-2-3, by simply adding two columns: a column containing the 0x prefix placed just before the calculated four-digit HEX field, and a column containing 0000 just after it.

Plotting the Output Data

Several software programs can easily read and plot the output file as a continuous time signal or frequency domain; Mat Lab, DAPiSP, ILS, Math Cad, and Hypersignal. For further information on any of these products, contact the companies shown in Appendix A. This report shows how Hyperceptions' Hypersignal package is used to debug DSP algorithms.

The Hypersignal program can acquire and display all types of TMS320 files. This permits viewing numerical file data (input and output) in both time and frequency representation. Hypersignal offers an extensive list of DSP utilities, such as

- Waveform display/edit
- FFT generation
- FIR and IIR filter construction and code generation (assembly and C)
- Convolution
- LPC autocorrelation
- Recursive filtering for IIR filter types

- Generation of user-defined difference equations (which can generate files for use as input to any of the TMS320 development tools)
- Digital Oscilloscope

Hypersignal has several other functions for analyzing data files and filters in the frequency domain with utilities for creating or displaying

- Filter/file magnitude display (both log or linear)
- Filter phase display
- 3-D and 2-D frequency vs time vs amplitude-spectrogram display
- Inverse FFT function
- Filter pole-zero plot (both in s and z domains)
- Power spectrum generation

Hypersignal's powerful functions permit the evaluation of DSP tasks. For a first time user, they can prove to be extremely helpful in establishing a base line knowledge of DSP.

Algorithms of High Complexity

Packages such as Hyperception's make DSP algorithm development manageable, even with N second-order cascaded sections. In Figure 3, there was one second-order section. If anything were to go wrong, it would do so within this section. How would data logging help if you have several cascade sections?

This can be answered by drawing an analogy between debugging a fourth-order analog system, which uses op-amps, and the equivalent DSP system implemented with four cascaded second-order IIR sections.

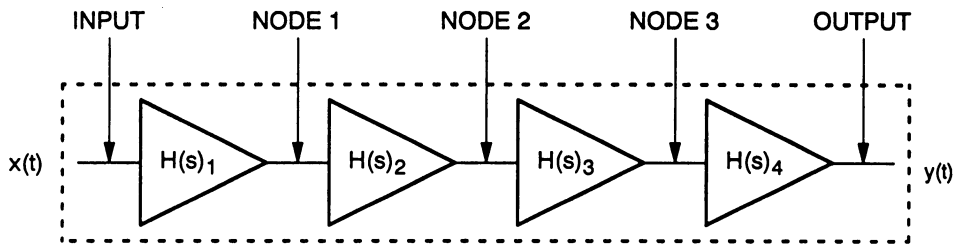


Figure 5. Four Op-Amp Block Diagram

Using traditional debugging techniques, i.e., an oscilloscope and function generator, you can examine the output versus the input on a stage-by-stage basis, correcting or adjusting each stage one-at-a-time. This process starts at node 1 and continues through to the output. When the system yields a satisfactory response for a given input condition

(not clipping, and amplifying at expected levels), use a spectrum analyzer to verify total frequency response. If the frequency response was not as expected, you can then examine each stage individually and adjust pole-zero placement to obtain the desired response.

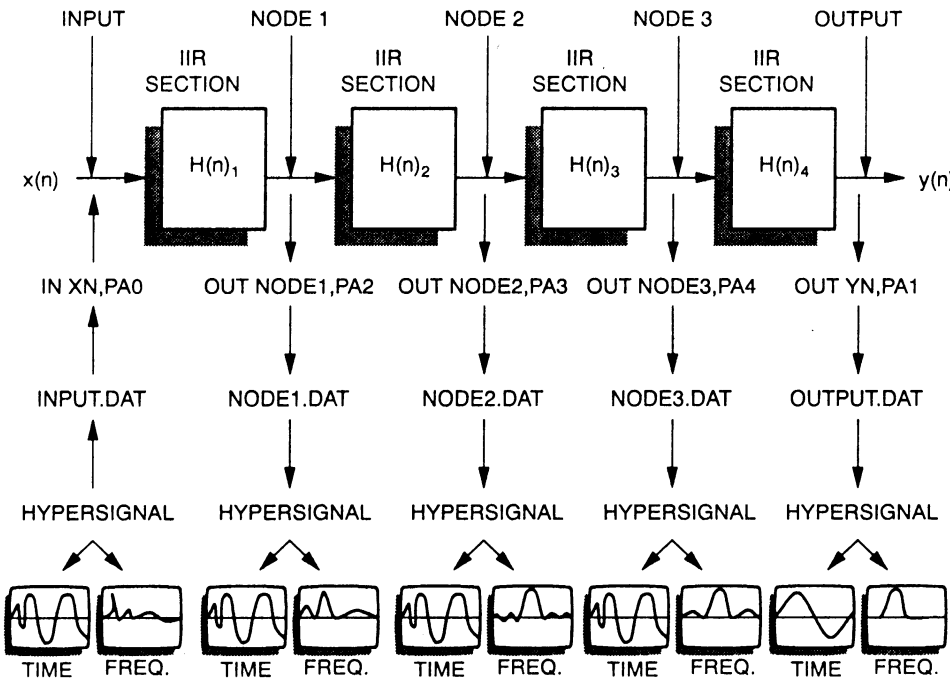


Figure 6. Four Second-Order IIR Structures

Figure 6 shows the same system as Figure 5 but uses four second-order IIR structures (a direct form II realization). When you use straight line code, it is a simple task to write a time sample to the DOS file by adding an OUT instruction. You can examine the feedback and/or feed forward signal within the IIR section as well. In addition to the obvious benefits of probing literally anywhere within the algorithm or system, there are some not-so-obvious benefits.

The Advantages of Data Logging:

1. You can assume any sample frequency. Sample frequency in a hardwarebased DSP system is a function of the A/D, the D/A, and the clock cycle of the DSP. With data logging, you can arbitrarily assign any frequency to the data samples within the files and can further assume any operating frequency for the DSP. It is therefore possible to specify devices with speeds in excess of any presently available speed if your algorithm so requires.

2. You can specify any input condition. If you are doing a modem design, you can use a real-time data sampler to acquire a REAL signal to use as an input file. It is also possible to use a numerically generated input signal supplied by Lotus or any other software system/utility such as Hypersignal, HLL programs, math packages, etc.

3. You can probe your system without having to observe any location restrictions. In hardware systems, you are restricted to available pins. With DSP code, an OUT instruction can be put anywhere.

4. You must use use a scope probe with analog systems, thus adding resistance and capacitance to the signal being examined. Data logging is a perfect observation utility, since it places no load on the signal.

5. You can examine the input and output signals with any level of desired granularity. If you intend to use a 12-bit A/D, you can examine the signal at 16 bits, then truncate the data to 12 bits and compare results. If you can get by with 10 or even 8 bits of granularity, you will reduce system cost.

6. You can print your results using plotting packages such as Hypersignal. Results can be printed for both frequency and time, thus providing a greater level of documentation.

7. You can archive input and output files as part of your total documentation package.

8. You can't get burned; there are no soldering irons involved.

Conclusion

DSP-based systems using data logging techniques demonstrate improved quality and shorter time to market. Using the TMS320 simulators and SWDS products in conjunction with graphic/data acquisition software packages, you can write and debug a large portion of an algorithm long before silicon or target platforms are available.

References

[1] *First-Generation User's Guide* (literature number SPRU013B), Texas Instruments, 1989.

[2] *TMS320 Family Simulator User's Guide* (literature number SPRU009B), Texas Instruments, 1988.

[3] *The TMS320C30 Simulator User's Guide* (literature number SPRU017), Texas Instruments, 1988.

Appendix A

Software Package Sources

DADiSP
DSP Development Corp
One Kendall Square, Cambridge, MA 02139
(617) 577-1133

Hypersignal
Hyperception
9550 Skillman-LB125 Dallas, TX 75243
(214) 343-8525

ILS
STI Signal Technology Inc
5951 Encina Road, Goleta, CA 93117
(805) 683-3771

Lotus 1-2-3
Lotus Development Group
55 Wheeler St. Cambridge, MA 02138
(617) 492-7171

Math CAD
MathSoft
One Kendall Sq., Cambridge, MA 02139
(617) 577-1017

MATLAB
The Math Works Inc
South Natick, MA 01760
(508) 653-1415