System Identification: A Practical Tool from a Fiddler's Paradise

Measuring the frequency response of a linear system is a typical first step in characterizing it's dynamic behavior. Requirements in control design and signal processing often demand information beyond the frequency response. Underlying differential and/or difference equations provide a concise mathematical description defining all other dynamic behavior. The task of estimating these equations from measurements of input and output signals is referred to in the control literature as the process of System Identification (SID). Despite the complexity of the signal processing involved, the complete identification process can be integrated into an easy to use GUI based application.

Overview

Where are We Going?

In 1971 two Swedish researchers, Åström and Eykhoff, published a paper that referred to the field of System Identification (SID) as a "Fiddler's Paradise."[1] Although many advances have been made in unifying the underlying theories, there continues to be a wide variety of approaches promoted in technical journals, text books, and conferences twenty-four years after this astute observation. This article is not intended to illuminate the SID expert, but rather the practicing engineer or researcher desiring a working knowledge of:

- What the process of SID is
- Why SID is useful
- Data acquisition requirements and suggestions
- Some real-world examples of SID
- A detailed description of a time-tested SID algorithm
- The integration of the data acquisition and SID algorithm into an easy-to-use GUI based application

The order may seem somewhat unorthodox, (i.e., examples before the discussion of the underlying algorithm), but if the examples are irrelevant to the reader, the details of the algorithms will be of little consequence.

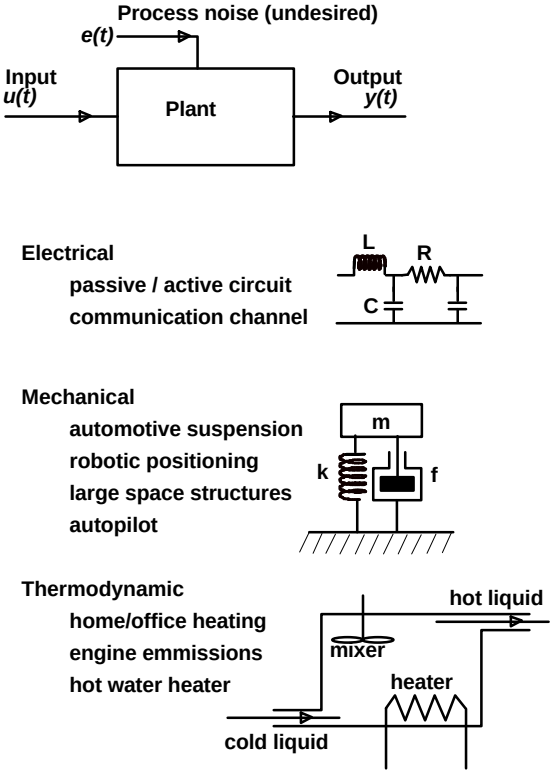


Figure 1 - Examples of Typical Plants

Plant..... What's a Plant?

The bulk of the theory and applications of SID evolved from control systems research. In control parlance, the dynamic system being identified (or modeled) is called the **plant**. The plant can take on a wide variety of forms. Generally speaking, a plant is an object or collection of objects, produces an output by modifying an input. The input to the plant could be temperature, pressure, voltage, current, position, velocity,

acceleration, and so on. The input can also contain more than one variable (e.g., several signals from transducers measuring various conditions of an engine). The output from the plant can be in the form of any of the inputs (temperature, pressure, position, etc.) and it also can contain several signals.

This article will focus on single-input, single-output (**siso**) plants since they are the most prevalent. The siso models can be combined to create multi-input, multi-output models if necessary.

Figure 1 depicts some examples of electrical, mechanical, and thermal plants. The common underlying link between all of these examples is that they can be accurately described mathematically by linear constant/coefficient, differential equations. A plant described by these equations is said to be Lumped-Linear-Time-Invariant (LLTI). Even plants that are mildly non-linear, slowly time-varying, and possibly distributed in nature (e.g., those require partial differential equations for a full description) can often be adequately modeled using these LLTI assumptions.

When to Use System Identification Techniques

If you require a mathematical model of a component or subsystem, SID may be the most accurate and direct path to this information. For instance, control systems engineering using state/space techniques require mathematical descriptions beyond the system's transfer function estimate.

If a graphical representation (e.g., frequency response) of the plant dynamics is sufficient for your purposes, then the more difficult task of system identification may not be worth the effort.

Why Not Analyze the Plant by Applying Physical Principals?

There are two reasons for using SID:

1. Complete knowledge of the physical parameters of the plant may not be known. Therefore, it is impossible to analytically determine the underlying differential equations.

2. Even with all the given physical parameters, the complexity of the plant may preclude an accurate analysis.

Mechanical plants are often good examples of how a seemingly simple physical system can have a remarkably complex dynamic description. This, for the most part, is due to the fact that mechanical systems are often distributed in nature rather than "lumped" like the simple spring-mass-dashpot system in Figure 1.

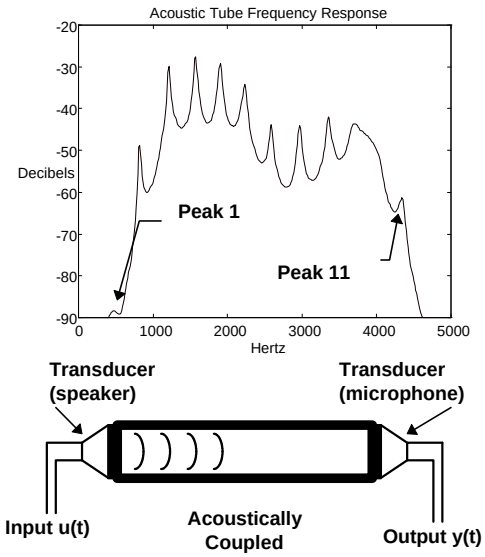


Figure 2 - A "Simple"Acoustic System

Figure 2 illustrates this point. Two audio transducers are coupled by a length of PVC tube. The plot illustrates the frequency response (relationship between response y and input u) of this simple electro-mechanical plant. A frequency response analysis over the 0-5000 Hz band shows 11

distinct peaks. This implies that a differential equation of order 22 or greater is necessary to describe the dynamics of this "simple" system. Not only are the acoustic wave properties of the PVC tube involved, but the dynamic characteristics of the transducers are as well. Analytically determining this plant's dynamic equations is next to impossible.

The Steps to System Identification

We will address the task of "off-line" identification using time-domain data. This means that measurements are made on the plant (input-output time histories are acquired), and subsequently a software algorithm is applied to this data to estimate a plant model. It is assumed that an excitation can be applied to the plant and the resulting response measured.

There are three primary steps to the off-line SID task:

1. Collect input-output data from the plant.
2. Analyze this data with a SID algorithm.
3. Validate the model.

We will focus on these three steps.

Step 1: Data Acquisition

Garbage In - Garbage Out

This time-worn axiom is applicable to the process of SID. The most sophisticated SID algorithms will produce results are no better than the data submitted to them. To perform input-output measurements on the plant modeled, we will be dealing with a sampled data acquisition system.

Table 1 outlines the features required by the data acquisition system for use in SID. The "poor" column represents hardware will not do a good job, while the "best" column has all the required features to support SID.

Feature	**Poor**	**Best**
Excitation:		
• burst pseudo random	Yes	Yes
• band limiting	No	Yes
• band translation	No	Yes
Acquisition:		
• Fs >2•Fmax	Yes	Yes
• lowpass alias filters	No	Yes
• bandpass alias filters	No	Yes
• pre-triggering	Yes	Yes
• time averaging	Yes	Yes
• high linearity	Yes	Yes

Table 1 - SID Excitation and Data Acquisition Requirements

Let's go over the list in a bit more depth to discuss why these features are important. Start with the excitation. In order to stimulate the plant, you need a source of excitation. The excitation must be capable of stimulating the entire range of frequencies over the plant is to be modeled.

A repeating pseudo random burst is a good choice for an excitation since it covers a band of frequencies. It is spectrally rich and unlike a chirp signal, it has a low sample-to-sample correlation. The best SID estimates will be obtained when the initial conditions of the plant states are known. The only practical way to establish the initial conditions of the plant is by forcing them to zero. The burst pseudo random excitation helps us here. A small amount of pre-trigger delay captures the output (and input) to the plant before the onset of the burst. The burst terminates and remains off for a period of time long enough to insure the states of the plant being identified have decayed to a value approaching zero.

It is beneficial to control the frequency range of this excitation, hence the band limiting feature listed in the table. In modeling an electro-mechanical component over the dc to 1 kHz range, there is no

benefit in exciting it over a dc to 100 kHz range. In fact, it is usually detrimental since you must drive the plant 100 times harder to get the same energy density over the 1 kHz range.

Likewise, if you are modeling a band-pass device, there is no benefit in exciting it beyond the bandwidth of the desired model, hence the band translation capability listed in the table is desirable. You can get by without excitation bandwidth control, but it is far better to focus the spectral energy to the bandwidth of interest.

The analysis frequency range will be determined by the model's characteristics as well as the intended application. For instance, the analysis of a residential telephone communication channel would span the dc-5 kHz range since most of the interesting dynamics of the channel are contained in this range.

On the other hand, components in a mechanical control system could have complex dynamic behavior into the 10s of kHz region. Due to the data acquisition used in the controller design and the control system design goals, a frequency range of only a few hundred Hz is of interest.

Now, on to the acquisition requirements. First, optimal SID requires programmable bandwidth alias filters. Well designed alias filters allow the acquisition sampling frequency to be set to 2.56 times the filter bandwidth.

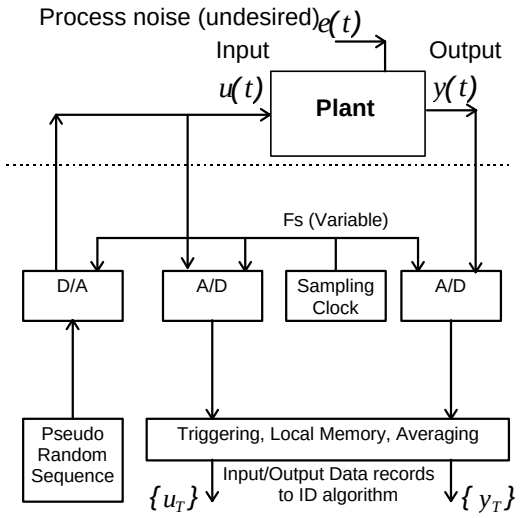


Figure 3 - Typical Data Acquisition Configuration

These filters guarantee the data will be free of aliases over the selected filter bandwidth.

The best data acquisition systems also have programmable bandpass filters. These are useful when analyzing plants with closely spaced resonances. These filters should be on both the excitation and acquisition channels.

Triggering allows the input-output plant time histories to be averaged. Averaging is beneficial when there is noise generated by the plant. Figure 1 shows a noise signal *e(t)* injected into the plant. This noise may be modified by the plant dynamics and coupled to the observed output signal *y(t)*.

Repeating the excitation (identical pseudo random burst noise excitation sequences) and averaging the input-output time histories will reduce the undesired noise component of the output signal due to *e(t)*. Assuming the noise term is uncorrelated with the averaging process, averaging can provide a significant reduction in this error at the expense of measurement time.

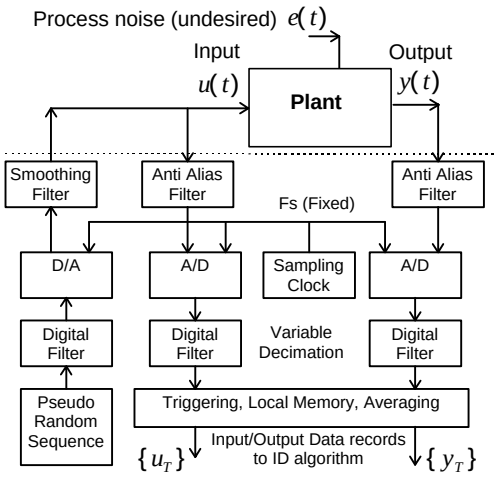


Figure 4 - Optimal Data Acquisition Configuration

Finally, it is important to have a highly linear acquisition system. Errors in the acquisition gain or dc offset can be easily corrected in software, but linearity errors are difficult (if not impossible) to remove. A good indicator of acquisition linearity is the spurious free dynamic range (sfdr) of the data acquisition system. Acquisition systems with 60 dB sfdr are linear to about 0.1%. Those with 80 dB or better sfdr have excellent linearity (approximately 0.01%).

As for notation, a continuous time signal will be referred to as y(t). A sequence of samples of this signal at times will be referred to as time history $\{y_T\}$. A single sample from this sequence at time will be referred to as X_T.

A Tale of Two Measurements

To further amplify the importance of proper data acquisition, a simple example is in order. We make two FFT-based transfer function estimates on a physical plant. The first estimate uses the hardware configuration in Figure 3. The features in this configuration are listed in the "poor" column of Table 1. Figures 5a and b show the measured input and output (averaged) time histories of this physical plant, with the resulting frequency response estimate computed from this data in Figure 5c.

Figures 5d and e show the same measurements but performed with the hardware in Figure 4. Notice the time histories in 5d and e are similar to those plotted in 5a and b. However, there is an obvious difference in the frequency response estimates shown in 5c and 5f. What's going on here? Aliasing.

Since aliased data creates serious errors in even the relatively trivial FFT-based frequency response estimate, would one expect the more sophisticated SID estimators to be immune to this corrupted data?

Why is the data infected by aliases? First, it must be understood that aliasing will occur in any sampled data system if frequency components of the digitized signal exceed one half the digitizer sampling rate (the Nyquist frequency).

The acquisition sampling rate was set to 12.8 kHz. The pseudo random excitation was not bandlimited. The test article has significant response to the excitation well beyond the Nyquist frequency of 6.4 kHz. Consequently significant aliasing occurred on both the input and output time history measurements. These aliases are not evident by inspecting the time history plots of Figures 5a and b. They manifest themselves as what might appear to be random noise on the transfer function estimate. The error is not random and averaging will not remove or reduce it. The time history data are garbage.

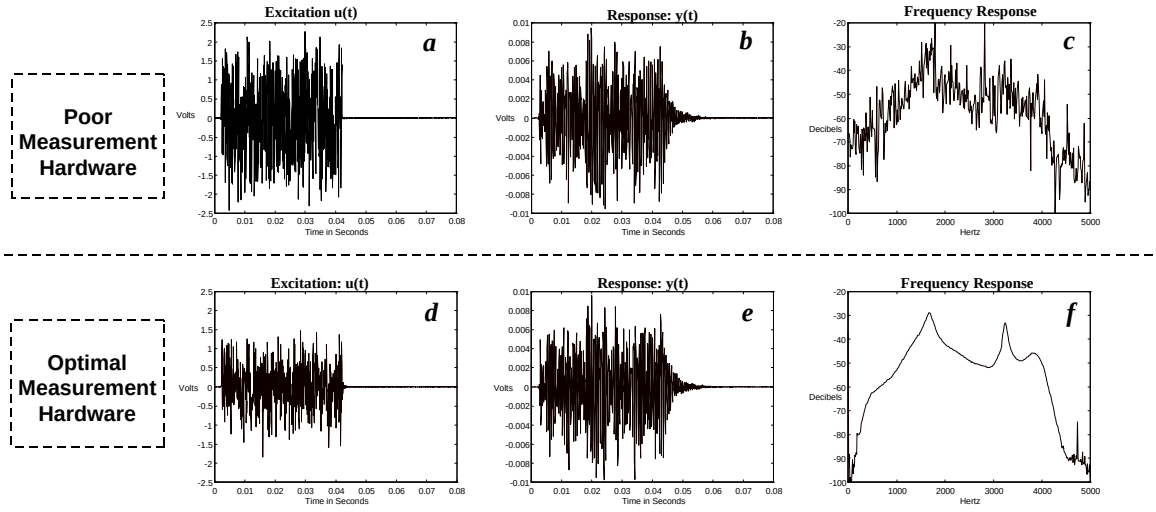


Figure 5 - Excitation, response, and frequency response estimates. Measurements in the upper row (a,b) were made the with hardware shown in Figure 3. Measurements in lower row (d,e) were made with the hardware shown in Figure 4. Note the similarity in the time-domain plots, but the poor frequency response estimate in c compared with the estimate in plot f. The transfer function estimates in plots c and f were computed from:

$$Y(f) = 20 \cdot \log 10(abs(fft(\{y\}_T)./fft(\{u\}_T)))$$

Step 2: Data Analysis

The Basic Concept

As implied by the title, there is more than one way (a fiddler's paradise) to create a model from measurements on the plant. The techniques fall into two major groups: time domain and frequency domain.[2] A time-domain technique will be discussed here.

Time-domain techniques have the advantage of not requiring yet another estimation technique (the transfer function estimate) as a front end to the algorithm. Input-output time history measurements of the plant are all that are necessary.

Figure 6 illustrates the basic concept behind time-domain SID. The LLTI plant is excited with bandlimited burst pseudo random noise as in the previous examples.

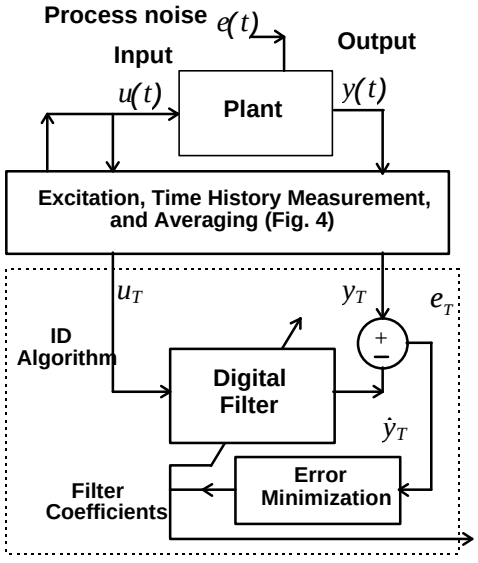


Figure 6 - Conceptual Analysis of SID Algorithm

The excitation and response from the plant is bandlimited, digitized, and averaged on a burst-by-burst basis by the data acquisition system to reduce the effect of the process noise $e(t)$. The resulting averaged input-output discrete time histories $\{u_T\}$ and $\{y_T\}$ (plotted in Figure 5d & e) are analyzed on a sample-by-sample basis by the (conceptualized) ID algorithm illustrated in the box formed by the broken line in Figure 6.

The samples, u_T, from the digitized input time history $\{u_T\}$, are the input to a digital filter. The coefficients of this filter are adjusted by the error minimization routine with the objective of making each output sample of the filter, $\hat{y}_T$, match each output time sample of the plant y_T. Of course, due to noise $e(t)$, plant non-linearities, and other inaccuracies, you will never get perfect agreement.

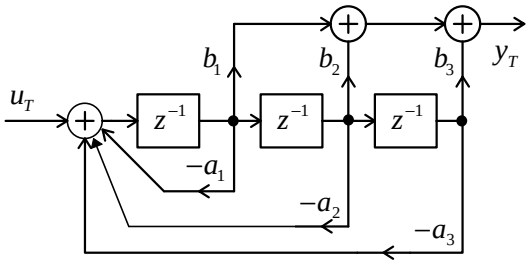


Figure 7 - Direct Form Digital Filter Topology

Figure 7 illustrates one of many possible digital filter topologies. The input and output of this filter are related by the difference equation (1) .

$$y_T = \sum_{i=1}^{3} b_i \cdot u_{T-i} - \sum_{i=1}^{3} a_i \cdot y_{T-i} \qquad (1)$$

In Equation 1 the y_T represents a sample from the output time history at time T. The z^{-1} boxes in Figure 7 represent unit sample delays such that if sample x_T is present on the input of the delay element, sample x_{T-1} is on its output. The u_{T-i} represents a sample from the input time history delayed by i samples.

During the identification process, the a_i and b_i coefficients are adjusted by the error minimization routine.

The order (or complexity) of the filter is defined by the number of unit delay elements (z^{-1}). In this example, we show a third order filter. Therefore, determining the filter order is also a part of the SID process. If a robust and numerically efficient SID algorithm is used, it will be shown that there is little penalty for specifying a model order that is in excess of the actual plant order.

The objective of the SID process is to produce filter coefficients minimize the mean square of the error sequence e_T in Figure 6.

The Initial Z-plane Model

The a_i & b_i in (1), along with the data acquisition sampling rate, is one version of a discrete time description of the plant dynamics. An alternative description is the discrete time transfer function $H(z)$. When working with discrete time systems, the Z transform is often employed:

$$Z\left(x_T\right) \equiv \sum_{T=\infty}^{\infty} x_T \bullet z^{-T} \qquad (2)$$

Start by defining $H(z)$ in terms of $U(z)$ and $Y(z)$ (the Z transforms of y_T and u_T).

$$Y(z) = Z(y_T)$$
$$U(z) = Z(u_T) \qquad (3,4,5)$$
$$H(z) \equiv \frac{Y(z)}{X(z)}$$

Equation 6 is simply (1) rewritten expressing an arbitrary model order with $a_0 = 1$.

$$\sum_{i=1}^{M} b_i \cdot u_{T-i} - \sum_{i=0}^{N} a_i \cdot y_{T-i} = 0 \qquad (6)$$

Taking the Z transform of (6) produces:

$$Z\left(\sum_{i=1}^{M} b_i \cdot u_{T-i} - \sum_{i=0}^{N} a_i \cdot y_{T-i}\right) = U(z)\sum_{i=1}^{M} b_i \cdot z^{-i} - Y(z)\sum_{i=0}^{N} a_i \cdot z^{-i} = 0 \qquad (7)$$

and solving (7) for $H(z)$ provides two equivalent forms. Equation 8 describes the discrete time system function $H(z)$, as the ratio of two polynomials.

$$H(z) = \frac{\sum_{i=1}^{M} b_i \cdot z^{-i}}{\sum_{i=0}^{N} a_i \cdot z^{-i}} \qquad (8)$$

Whereas (9) describes $H(z)$ in terms of the roots of these polynomials. The zero locations are the z_i (the magnitude of $H(z)$ goes to 0.0 when $z = z_i$) and the pole locations the p_k (the magnitude of $H(z)$ goes to infinity when $z = p_k$).

$$H(z) = K_{gain} \cdot \frac{\prod_{i=1}^{M-1}(z - z_i)}{\prod_{k=1}^{N}(z - p_k)} \qquad (9)$$

The poles and zeros define the magnitude and phase of the transfer function $H(z)$. The gain term, K_{gain}, is required in (9) since the absolute gain and sign is lost when $H(z)$ is written in terms of the roots of the numerator and denominator polynomials.

It should be noted that the objective of SID is to arrive at a description of the plant dynamics in terms of Equations 6, 8, or 9. There is little chance, however, that these results are exactly in the format you would want them. If a continuous time plant model is required, a transformation from discrete time to continuous time is in order. If a discrete time model is required, sampling rate change is often needed.

Changing the Z-plane Model Sampling Rate

The discrete time plant model produced by the SID process is valid at the sampling rate used for the data acquisition. Often the sampling rates of the end system are set well beyond those used during the data acquisition phase to ease the anti-alias filtering requirements of the end system. To make the discrete time model useful, it is important to be able to change the sampling rate. Fortunately, this is not too difficult.

The upper center plot in Figure 8 illustrates a discrete time model in terms of z-plane poles and zeros per (9). This model is valid for the frequency at which the data acquisition took place: Facq=5120 Hz. The gray regions in the plots represent the transition bands of the anti-alias filters in the data acquisition hardware. The transition bands, naturally, do not have the level of alias protection provided in the analysis band. Singularities in this region that do not lie on the negative real axis are, therefore, potentially suspect.

The pole and zero locations can be transformed using (10) in Figure 8. This equation defines how each singularity is relocated to accommodate the new sampling rate (Fnew). The resulting model for an 8000 Hz sampling rate is shown in the lower left z-plane plot (8). The new sampling rate should be greater than the rate used for acquisition to avoid what is tantamount to aliasing.

Mapping the Z- to S-plane

The z-plane model of the plant is most useful when it is used as part of a discrete time system.

On the other hand, a continuous time model is preferable if the plant is part of a continuous time system or if a better understanding of the parameters of the plant will be obtained with an s-plane description. Since the basic SID results are in the z-plane, a mapping must be used to transform the results to the s-plane.

An analogous set of equations to (3-9) exists for the continuous time case, where $\mathcal{L}$ is Laplace transform operator:

$$\mathcal{L}(x(t)) \equiv \int_{t=-\infty}^{\infty} x(t) \cdot e^{-st} dt = X(s)$$

$$U(s) = \mathcal{L}(u(t))$$

$$Y(s) = L(y(t)) \qquad (10, 11,12,13)$$

$$H(s) = \frac{Y(s)}{U(s)}$$

Solving the constant coefficient differential equation as in (7) yields the **system function** $H(s)$ as the ratio of two polynomials in s:

$$H(s) = \frac{\sum_{i=1}^{M} b_i \cdot s^i}{\sum_{i=0}^{N} a_i \cdot s^i} \qquad (14)$$

Equation 15 is a factored version of (14) in terms of the polynomial roots (s-plane pole/zero locations z_i, p_k).

$$H(s) = K_{gain} \cdot \frac{\prod_{i=1}^{M-1} (s - z_i)}{\prod_{k=1}^{N} (s - p_k)} \qquad (15)$$

There are numerous methods for transformation between the z- and the s-plane. One method is outlined in the upper right box in Figure 8.[3] Equation 16 defines how the z-plane singularities map to the s-plane. For this mapping, z-plane singularities on the negative real axis are dropped since they have no counterparts in a continuous time model. If there are poles and zeros outside the analysis bandwidth; naturally, they will have some effect on the data inside the analysis band. Therefore, a perfect match between the frequency response of the model and the frequency response from direct measurements of the plant may not be obtained. The resulting s-plane pole/zero frequencies are, however, accurate and unaffected by the out of band singularities.

Step 3: Model Validation

How Closely Do the Model Dynamics Match the Actual Plant Dynamics?

Above each of the models (z- or s-plane) in Figure 8, an overlay of the frequency response of the model computed from its pole-zero-gain description, and the frequency response computed from the data $\{u_T\}$ and $\{y_T\}$ per Figure 5f, is plotted. These plots appear to have only one line each, but this is due to the excellent match between the dynamics of the plant model and the plant.

Another popular way of examining the model accuracy is to plot the error signal e_T. If e_T is not explicitly available, it may be computed by feeding samples from the measured input sequence $\{u_T\}$ into the model described by (6). This will produce an output time history $\hat{y}_T$. Plotting sample-by-sample difference between the model output $\hat{y}_T$ and the measured output y_T is another indication of the accuracy of the model. In the ideal case, there would be no difference between $\hat{y}_T$ and y_T.

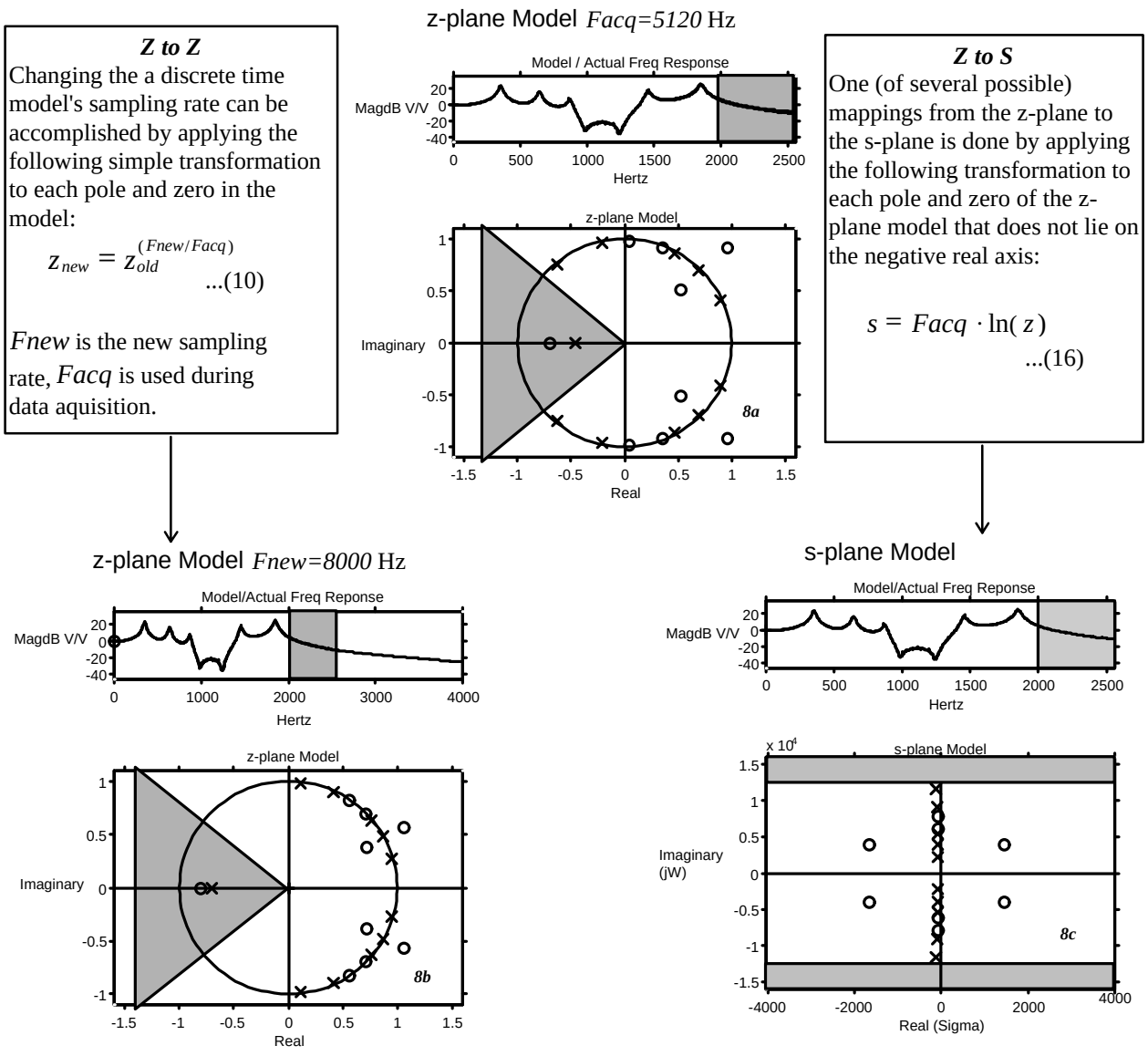


Figure 8 - Transformation of the original z-plane model (center) valid for a 5120 Hz sampling rate to a z-plane model with a sampling rate of 8000 Hz (lower left) or a s-plane continuous time model (lower right). In each case the frequency response of the model is computed and overlayed with the frequency response of the plant, which is computed per Figure 5f. The agreement between the model and the actual frequency response is excellent so there appears to be a single frequency response curve.

Some Real-world Examples

The following examples are intended to provide you with some real-world examples of SID in action. It is important to recognize that each of the examples involves an actual physical device. There was no mathematically contrived data whatsoever. Optimal measurement hardware (Figure 4) was used with the identification algorithm yet to be described.

Electro-Mechanical Actuator

The first example SID task consists of characterizing an electro-mechanical plant. In order to implement an optimal control design using state space techniques, the dynamic model of this plant is required.

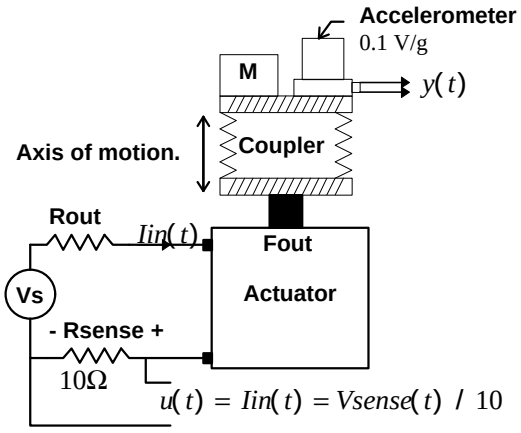


Figure 9 - Electro-Mechanical plant. Current Iin(t) produces acceleration measurement y(t).

The plant, illustrated in Figure 9, consists of four components: actuator, coupler, accelerometer, and mass. The electro-dynamic actuator produces a force proportional to input current $Iin(t)$. The mechanical coupler has dynamic characteristics (it cannot, however, be replaced by a non-deformable mass).

An accelerometer and a mass equal to the anticipated load (minus the mass of the accelerometer) are attached to the end of the coupler. The accelerometer output is used to measure the motion of the coupler output with mass loading.

A continuous time position control system will be constructed. The control loop bandwidth will be below 50 Hz, so a dc-200 Hz frequency band is selected for analysis. This will insure that any important resonances are included in the model.

The input to the system (Figure 9) is defined as current $Iin(t)$. If a current source is not available to test the system, a voltage source with a current sense resistor (Rsense) is also acceptable.

The output impedance of the voltage source (Rout) is not critical, but it must be low enough to provide adequate excitation to the system.

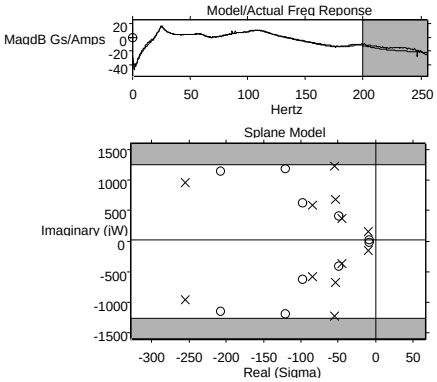


Figure 10 - s-plane Model of Electro-mechanical System Dynamics

Figure 10 illustrates the results of the SID process. Ideal mechanics would not have the in-band resonances. Detailed knowledge of these resonances in the form of an accurate s-plane model, will aid in the design of a stable control system.

Communication Channel

SID can be valuable in the study of any dynamic system. A communications engineer needs the s-plane model of a telephone transmission channel to assess the required complexity of an equalization scheme. A simple equalization scheme can

be implemented by inverting the system function of the communication channel. The system function can be estimated by using SID.

To accomplish this task, outbound and inbound telephone lines are used (at the same facility). The excitation $u(t)$ and response $y(t)$ are coupled via transformers to these lines.[4] A call is placed to connect line 1 to line 2 through the local telephone office.

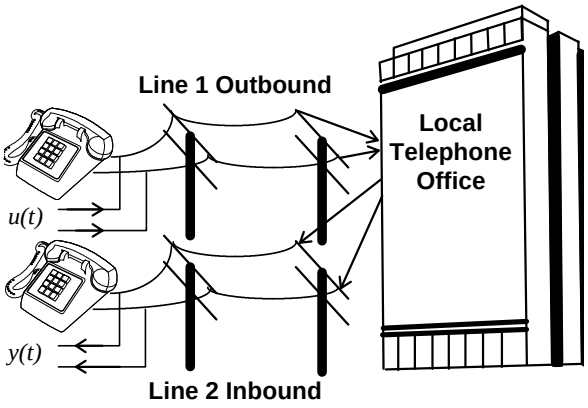


Figure 11 - Characterizing Telephone Transmission Dynamics

The three-step SID process is carried out on the telephony signals and the results are shown in Figure 12.

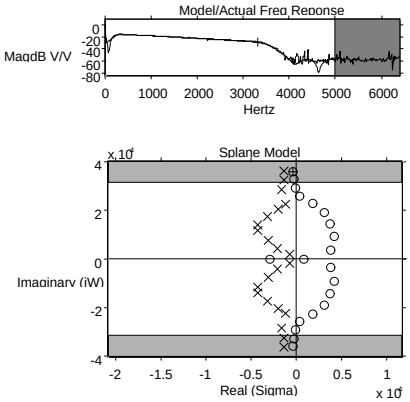


Figure 12 - s-plane Model of a Local Telephone Communication Path

A reasonably complex dynamic model (22nd order) is required to adequately model the transmission path. The frequency response resembles a passband filter since the response falls off rapidly at frequencies below 200 and above 3000 Hz. There is also approximately a 10 dB roll off in the passband. A first cut at an equalizer might be made by inverting the model e.g., $H(s)^{-1}$. The s-plane plot shows are many zeros in the right half s-plane due to the significant group delay of the channel. A simple inversion would therefore lead to an unstable equalizer design. However, the zeros in the right half of the s-plane can easily be reflected into the left half, thereby making the inverted system function $H(s)^{-1}$ stable. This will not affect the frequency response of the equalizer and is a simple way of obtaining a stable design.

Acoustics

An acoustical signal processing engineer is investigating noise-cancellation techniques to be used in automobiles. Typically, adaptive filtering algorithms are used to accomplish this task. Experiments indicate that the complexity (and therefore cost) of the adaptive filter can be reduced if a model of the dynamics between the loudspeaker and sensing microphone is available. The setup is shown in Figure 13.

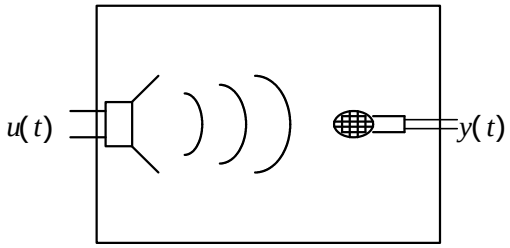


Figure 13 - Automobile Passenger Compartment

The objective is to determine the relationship of the microphone output $y(t)$ to the loudspeaker input $u(t)$. Although this might appear to be a simple physical system, it actually is not. The physical space between the speaker and microphone introduces a time delay, and the dynamics of the microphone and speaker are combined

with sound reflections from the interior of the passenger compartment.

To be of practical value, the model should be of a relatively low order to minimize the computational expense. Also, the dynamics will change when the passenger compartment is occupied, so modeling every detail is not required.

Therefore, the order was constrained to be less than that of the actual system. Since the noise cancellation hardware will be constructed using DSP, a discrete time model was desired. Figure 14 contains the results of the analysis.

Many resonance/anti-resonance (pole-zero) pairs can be seen in the magnitude plot of the system. The model order was constrained to be 18, which is sufficient to track the general shape of the transfer function (in both magnitude and phase) but not so high as to reproduce every detail. The SID algorithm has produced an acceptable reduced order model that can now be used in the design of the noise cancellation scheme.

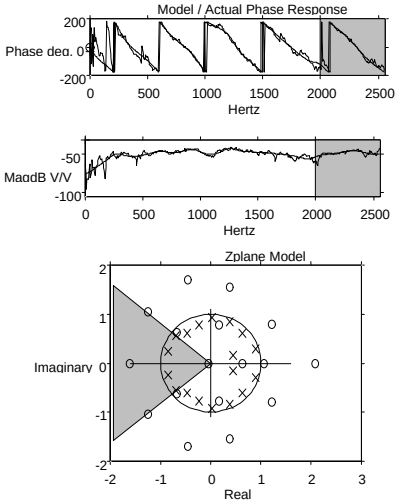


Figure 14 - z-plane Model of Automobile Acoustic System

Narrowband Filter

An electrical engineer has constructed a bandpass filter from a chip with four bi-quad stages.

The engineer used cad software to create a filter with the following characteristics:

Filter type:	elliptic
Filter order:	8
Passband:	16000-16200 Hz.
Ripple:	2 dB
Lower stop band:	dc-16400 Hz
Upper stop band	16600 Hz and up
Stopband attenuation	-50 dB

The filter was constructed and the transfer function measured. The upper left plot of Figure 15 shows a dc-20 kHz baseband frequency response measurement. The filter response looks like a spike and little detail of the pass and transition bands is available. Due to the narrowband nature of the system, zoom (or frequency translation) is essential to provide sufficient frequency resolution for the analysis of the filter. Using zoom analysis, the excitation and input subsystems are set to cover from 15600 to 16600 Hz. This focuses the excitation and analysis system resources over the passband and transition bands of the filter, providing detailed information in these regions.

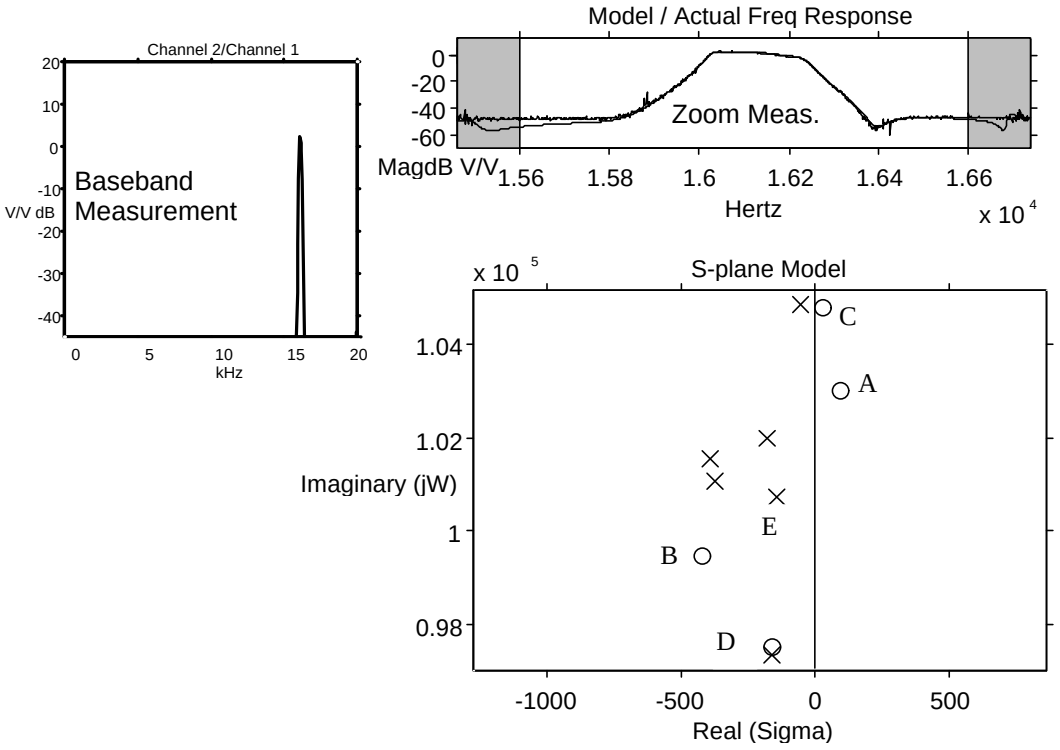


Figure 15 - Narrowband electrical filter. Upper left plot shows a baseband frequency response (dc-20 kHz.). Zoom analysis provides far greater detail of passband structure resolving the tightly clustered poles and zeros in the s-plane.

The transfer function measurement indicates that the filter does not address its design goals very well. First, the stopband attenuation is not as good as predicted by the software used to design the filter. Figure 15 shows that the placement of the zeros is not optimal These zeros have an important role in defining the stopband. Ideally, all the zeros would lie exactly on the *jw* axis. Two of the zeros are virtually canceled out by poles (D and C) rendering them ineffective.

The zero labeled A is in approximately the correct position but the zero at B is significantly removed from the axis.

Second, the passband is not flat. This is virtually impossible to see in the baseband measurement, but it is obvious in the zoomed measurement. The pole at location E appears to be too close to the *jw* axis.

It is clear that theory and practice are at odds with each other in this design. Armed with the pole-zero location information, the engineer will have a far easier time debugging the design.

Example Wrap-Up

It should be clear from the above examples that SID techniques can provide information above and beyond that provided by the transfer function estimate. Any dynamic system that is reasonably linear and time invariant can be analyzed with SID techniques to give a more comprehensive view of the system. This enhanced system modeling capability, although complicated from a signal processing point of view, can

be made accessible to the user by integrating the data acquisition with the SID algorithm in an easy to use GUI application.

The Identification Algorithm

Background

Our goal is to process the sampled input-output time histories $\{u_T, y_T\}$, and produce a set of model coefficients $\{a_i, b_i\}$ that describe the dynamics of the system in the form of Figure 7, 8, or their equivalent alternate representations. The process that produces these model coefficients is the SID algorithm.

This section will explain the underlying mathematics behind the identification algorithm. The original work on the underlying mathematics came from a group of researchers at Stanford University in the early 80s. A key paper, which put much of the existing work in a common national framework, was published by Benjamin Friedlander in August of 1982.[5]

This paper ("Lattice Filters for Adaptive Processing") covers the theory behind lattice filters and numerous applications of them. It also contains an extensive list of references for those who want to delve into the subject in more depth.

This article draws heavily from the Friedlander paper to explain the underlying mathematics of the SID algorithm.[6]

Overview

Often a solution to a problem can be modified or extended into what might initially appear to be an unrelated area. It will be shown that the SID algorithm can be obtained from the solution of a linear prediction problem. Linear prediction consists of attempting to predict the value of a future sample (usually the next value) based on a finite-weighted sum of previous samples.

The linear predictor can be constructed to handle a scalar time history or a vector time history (multiple channels). The trick used to turn the solution to the prediction problem into a solution for SID is called "embedding".

The input and output sample pairs (u_T, y_T) are processed by the lattice filter exactly the same as in a two channel linear prediction problem. As the time series is being processed, the lattice is generating a representation of the dynamics of the plant in terms of what are know as "reflection coefficients". For each new pair of input output samples, the algorithm updates its estimate of the plant dynamics. Typically, when the end of the burst portion of the excitation time history is reached, the lattice has converged to an accurate model of the plant dynamics. Little useful information is gained by processing samples beyond this point.

The reflection coefficients, as well as the internal states of the lattice, are then processed (by another lattice structure) to provide the set of model coefficients $\{\hat{a}_i, \hat{b}_i\}$, which is the goal of the SID algorithm.

The Embedding Technique

In order to understand why the linear prediction technique is used to implement the SID algorithm, the topic of embedding must be covered. Let's *assume* for a moment that we have a *solution* to the following problem:

Given a (vector or multichannel) time history $\{x_T\}$*, we can determine a set of coefficients* $\{\Lambda_i\}$*, from the data sequence*

$\{x_T\}$, *which minimizes the mean square value of the sequence* $\{\varepsilon_{N,T}\}$ *in (17).*

$$e_{N,T} = x_T + \sum_{i=1}^{N} \Lambda_i \cdot x_{T-i} \qquad (17)$$

Now, the question is, if such a tool existed, how can it be applied to the SID problem?

First, let:

$$x_T = \begin{bmatrix} y_T \\ u_T \end{bmatrix} \qquad (18)$$

where u_T and y_T are the sampled input and output signals from the plant at time T as before.

$$\text{Let } \Lambda_i = \begin{bmatrix} \hat{a}_i & -\hat{b}_i \\ \hat{c}_i & \hat{d}_i \end{bmatrix} \qquad (19)$$

and write $e_{N,T}$ as:

$$e_{N,T} = \begin{bmatrix} e^{y}_{N,T} \\ e^{u}_{N,T} \end{bmatrix} \qquad (20)$$

Then, rewrite (17) in terms of (18, 19, 20) as follows:

$$\begin{bmatrix} \varepsilon^{y}_{N,T} \\ \varepsilon^{u}_{N,T} \end{bmatrix} = \begin{bmatrix} y_T \\ u_T \end{bmatrix} + \sum_{i=1}^{N} \begin{bmatrix} \hat{a}_i & -\hat{b}_i \\ \hat{c}_i & \hat{d}_i \end{bmatrix} \cdot \begin{bmatrix} y_{T-i} \\ u_{T-i} \end{bmatrix} \qquad (21)$$

Expanding the top row of Equation 21 gives:

$$\varepsilon^{y}_{N,T} = y_T + \sum_{i=i}^{N} \hat{a}_i \cdot y_{T-i} - \sum_{i=i}^{N} \hat{b}_i \cdot u_{T-i} \qquad (22)$$

Compare this equation with (1). If the error term $\varepsilon^{y}_{N,T}$ were zero, they would be identical. This error can be interpreted as a modeling error. If the $\{y_T, u_T\}$ were related by (1), and if the error $\varepsilon^{y}_{N,T}$ were zero for all T, then it must follow that $\hat{a}_i = a_i$ and $\hat{b}_i = b_i$. Therefore, the system model would be in agreement with the system. We have tacitly assumed that the model order N is known, but for now it is safe to say that using a model order higher than the actual system order does not pose a significant problem.

This technique works since minimizing the mean square error of $\varepsilon_{N,T}$ is equivalent to minimizing mean square error of both $\varepsilon^{y}_{N,T}$ and $\varepsilon^{u}_{N,T}$. The $\{\hat{c}_i, \hat{d}_i\}$ coefficients do not add any useful information and are simply excess baggage stemming from using the solution to the 2 channel vector prediction problem for SID.

Therefore, one can adapt the solution to the linear prediction problem to implement the SID algorithm.

The Linear Prediction Solution

Now we will have a closer look at the details of this solution. First the linear prediction equation is:

$$\hat{x}_{T|T-1} = -\sum_{i=1}^{N} \Lambda_i \cdot x_{T-i} \qquad (23)$$

The quantity $\hat{x}_{T|T-1}$ is the prediction of x_T based on samples up to and including time $T-1$. The prediction is a weighted sum of previous samples of the measured time history.

Of course for real-world data we will never be able to exactly predict the next value of a time history by such a scheme (process noise in Figure 1, model order underestimation, non-linearities, the embedding process etc.), so there is always an error term $\varepsilon_{N,T}$ given by:

$$\varepsilon_{N,T} = x_T - \hat{x}_{T/T-1}$$

$$\varepsilon_{N,T} = x_T + \sum_{i=1}^{N} \Lambda_i \cdot x_{T-i} \qquad \text{(24a,b)}$$

which is simply the difference between reality (x_T) and the prediction $\hat{x}_{T/T-1}$ at time T. (Note (24b) is a repeat of (17).) Figure 16 is a signal flow diagram that directly implements (24b) (z^{-1} being a unit delay).

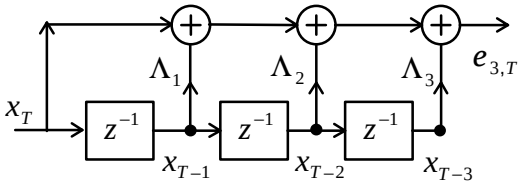


Figure 16 - Direct form (3rd Order) FIR Prediction Error Filter

Exactly the same filter relationship between x_T and $\varepsilon_{N,T}$ can be created using the lattice topology shown in Figure 17. In this case the structure consists of a cascade of sections that have the geometric form of a lattice; hence, the name of the filter. The characteristics of the filter are determined by the order (in this case 3) and the K_i coefficients (referred to as reflection coefficients).

The forward prediction errors ($\varepsilon_{p,T}$), as well as an auxiliary quantity called backward prediction errors ($r_{p,T}$), are the signals that propagate between stages of the lattice.

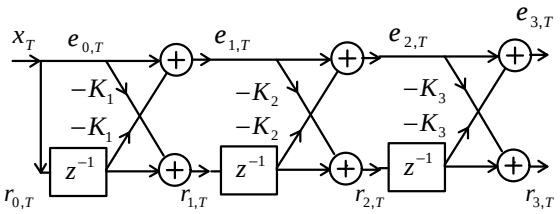


Figure 17 - Lattice (3rd Order) FIR Prediction Eror Flter

The challenge of linear prediction is determining the coefficients (either Λ_i or K_i) of the filter from the input data sequence such that the error ($\varepsilon_{N,T}$ in (24)) is minimized.

It will be shown below that the coefficients (K_i) are data dependent; therefore, the filter is actually an adaptive filter with an adaptation scheme designed to minimize the mean square value of the prediction error sequence.

It is clear that the filter in Figure 16 is a simpler solution (fewer multipliers and adders, same number of delays) than Figure 17. Since both implementations ostensibly provide the same functionality, why choose the more complex lattice implementation? The lattice has several advantages over the apparently simpler direct form realization.

First, predictors implemented with the lattice structure are far less sensitive to numerical errors in the reflection coefficients. This helps preserve accuracy when analyzing high-order systems.

Secondly, all lower-order predictors are contained in the filter. Figure 17 shows the lower-order prediction errors propagating in the lattice structure. This nesting property does not exist in the direct form realization. This property can be useful in extracting models of lower order then the initial order estimate.

Finally, the algorithm that updates the reflection coefficients and lattice state for each new input sample, requires computations and storage elements scale linearly with predictor order N. The comparable direct form realization typically requires on the order of N^2. There are so-called "fast" algorithms for the direct form but they do not have the good numerical and nesting advantages of the lattice structures.[7]

The lattice shown in Figure 17 may appear to be visually simple, but is actually computationally complex when the reflection coefficients are required to adaptively change to minimize the forward prediction error.

Among the alternatives, the lattice structure presented in Figure 18 is actually computationally simpler (and numerically better conditioned) than Figure 17 in spite of its apparent visual complexity. This structure normalizes the internal lattice variables to an absolute value of less than one. This scaling actually reduces the number of lattice variables that need to be updated each time step. This lattice structure is called the "Pre-Windowed, Variance Normalized Lattice." [8]

A number of multiplying coefficients (shown as triangles with an associated coefficient) have been introduced in the signal paths beyond those in the lattice structure shown in Figure 17. These coefficients are a part of the normalization process. Also, Figure 18 illustrates that the coefficients are time and data dependent. The input time history $\{x_T\}$ (remember, a two channel vector process due to embedding of y_T, u_T), drives the lattice. All other variables are a result of this input time history. A third order structure is shown in Figure 18, but orders up to 60 have been successfully used. The lattice's excellent numerical properties allow successful high-order system analysis.

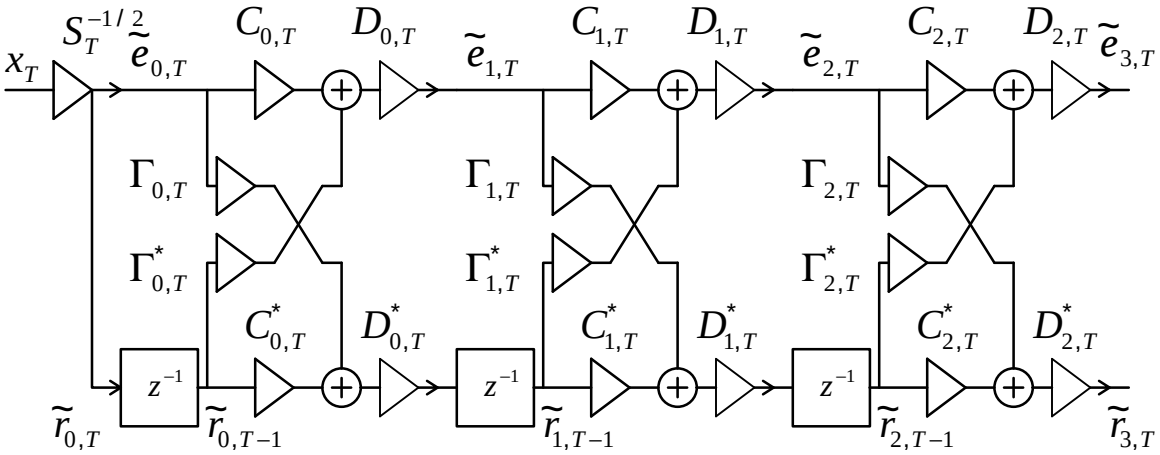
It is important to recognize that the objective in executing the above algorithm is to obtain a set of reflection coefficients modeling the system that produced the time history $\{x_T\}$. The pseudo code to implement this algorithm is shown in Listing I.

Listing I

```
for T=0:Tend
    if T==0
        K = S = r̃ = 0
    end;
    x_T = [ y_T ]
          [ u_T ]
    S_T = S_{T-1} + x_T · x_T'
    for p=0:min(N,T)-1
        K_{p+1,T} = R(K_{p+1,T-1}, r̃'_{p,T-1}, ẽ_{p,T})
        ẽ_{p+1,T} = F(ẽ_{p+1,T}, r̃_{p,T-1}, K_{p+1,T})
        r̃_{p+1,T} = F(r̃_{p,T-1}, ẽ_{p,T}, K'_{p+1,T})
    end;
end

function R(a,b,c)
  R=[ I − c · c' ]^{1/2} · a · [ I − b' · b ]^{T/2} + cb
end function R

function F(a,b,c)
  F=[ I − c · c' ]^{−1/2} · [ a − c · b ] · [ I − b' · b ]^{−T/2}
end function F
```



$$S_T = S_{T-1} + x_T \cdot x_T'$$

$$K_{p+1,T} = [\, I - \tilde{e}_{p,T} \cdot \tilde{e}_{p,T}' \,]^{1/2} \cdot K_{p+1,T-1} \cdot [\, I - \tilde{r}_{p,T-1} \cdot \tilde{r}_{p,T-1}' \,]^{T/2} + \tilde{e}_{p,T} \cdot \tilde{r}_{p,T-1}'$$

$$C_{p,T} = [\, I - K_{p+1,T} \cdot K_{p+1,T}' \,]^{-1/2}$$
$$C^*_{p,T} = [\, I - K_{p+1,T}' \cdot K_{p+1,T} \,]^{-1/2}$$

$$\Gamma_{p,T} = [\, I - K_{p+1,T} \cdot K_{p+1,T}' \,]^{-1/2} \cdot K_{p+1,T}$$
$$\Gamma^*_{p,T} = [\, I - K_{p+1,T}' \cdot K_{p+1,T} \,]^{-1/2} \cdot K_{p+1,T}'$$

$$D_{p,T} = [\, I - \tilde{r}_{p,T-1}' \cdot \tilde{r}_{p,T-1} \,]^{-T/2}$$
$$D^*_{p,T} = [\, I - \tilde{e}_{p,T}' \cdot \tilde{e}_{p,T} \,]^{-T/2}$$

Conventions:

- I is the identity matrix.
- Transpose of matrix M is M'
- A matrix square root is defined by: $M^{1/2} \cdot (M^{1/2})' = M$, a lower triangular square root works well.
- Transpose of a matrix square root is denoted by $(M^{1/2})' = M^{T/2}$
- Inverse of a superscripted matrix is $(M^s)^{-1} = M^{-s}$

Figure 18 - Variance Normalized Pre-windowed Lattice Algorithm

After executing the algorithm on the samples acquired during the data acquisition phase, a set of reflection coefficients $\{ K_{p+1,T} \}$, a set of backwards prediction errors $\{ \tilde{r}_{p,T} \}$, and the quantity $S_T^{-1/2}$ are available.

Recovery of the Direct form Model Coefficients from the Lattice Representation

Okay, so here we are with reflection coefficients and some other miscellaneous state information, now what? The desired

result is a set of coefficients:

$$\Lambda_i = \begin{bmatrix} \hat{a}_i & -\hat{b}_i \\ \hat{c}_i & \hat{d}_i \end{bmatrix}$$

where the $\{\hat{a}_i, -\hat{b}_i\}$ are the estimated model coefficients for the direct form realization of (1).

The mapping from reflection coefficients K_i to the direct form coefficients can be accomplished by executing yet another algorithm also has the topology of a lattice.[9] The structure (for a third order system) is shown in Figure 19. Each section of the filter has the topology shown in Figure 20. Both the reflection coefficients and backward prediction error states from the lattice in Figure 18 are used to obtain the direct form coefficient arrays Λ_i.

From these 2x2 arrays, the $\{\hat{a}_i, -\hat{b}_i\}$ are extracted. Listing II contains the pseudo code for the algorithm.[10]

Determination of Model Order

In order to execute the SID algorithm we must pick a value for the model order N. If the model order N is chosen to be greater than the order of the actual plant, N_{plant}, one might assume that the $\hat{a}_i$ coefficients would be 0 (or some negligible number) for $i > N_{plant}$. This is not the case. These higher-order coefficients are virtually always non-zero. Is then the model in error? No, not really. The algorithm has simply added identical pole/zero pairs, which have increased the order of the numerator and denominator polynomials up to N, and these excess roots (in the ideal noiseless data case) cancel out exactly since they are equal.

The point is that specifying a model order beyond that of the actual plant does not introduce significant error. We go through more computations, but since the lattice is numerically efficient and the work grows linearly with order, this is not a high price to pay. The excess roots that are added are easy to cull out. Canceling pole/zero pairs are within a specified radius of each other is a practical approach to removing excess terms.

Listing II

```
for i=0:N
    if i==0
        Λ̄0,0 = B̄0,0 = S_T^(-1/2)
        C̄0,0 = 0
    else
        Λ̄0,i = B̄0,i = C̄0,i = 0
    end;

    for p=0:N-1
        B̄*p,i = G(B̄p,i, C̄p,i, r̃p,T)
        C̄*p+1,i = G(C̄p,i, B̄p,i, r̃′p,T)
        Λ̄p+1,i = G(Λ̄p,i, B̄*p,i-1, Kp+1,T)
        Λ̄p+1,i = G(B̄*p,i-1, Λ̄p,i, K′p+1,T)
    end
end

for i=1:N
        Λ_N,i = Λ̄^(-1)_N,0 · Λ̄_N,i
end;


function G(a,b,c)
        G=[I − c · c′]^(-1/2) · [a − c · b]
end;
```

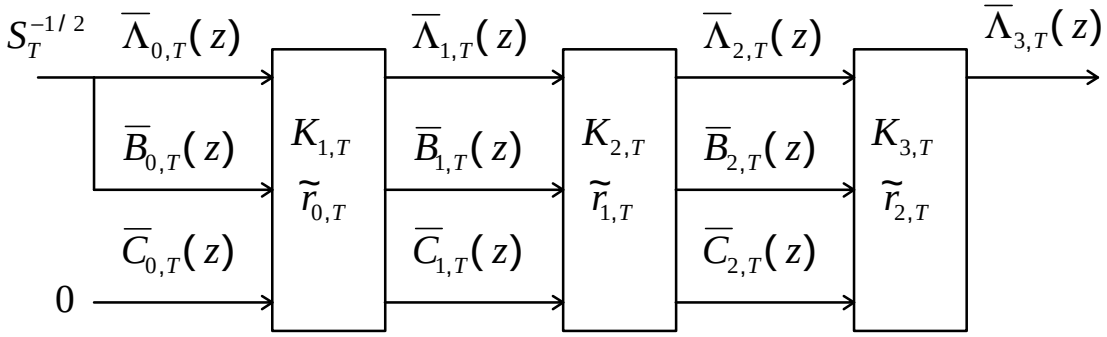


Figure 19 - Variance Normalized Pre-windowed Direct From Prediction Coefficient Recovery Filter

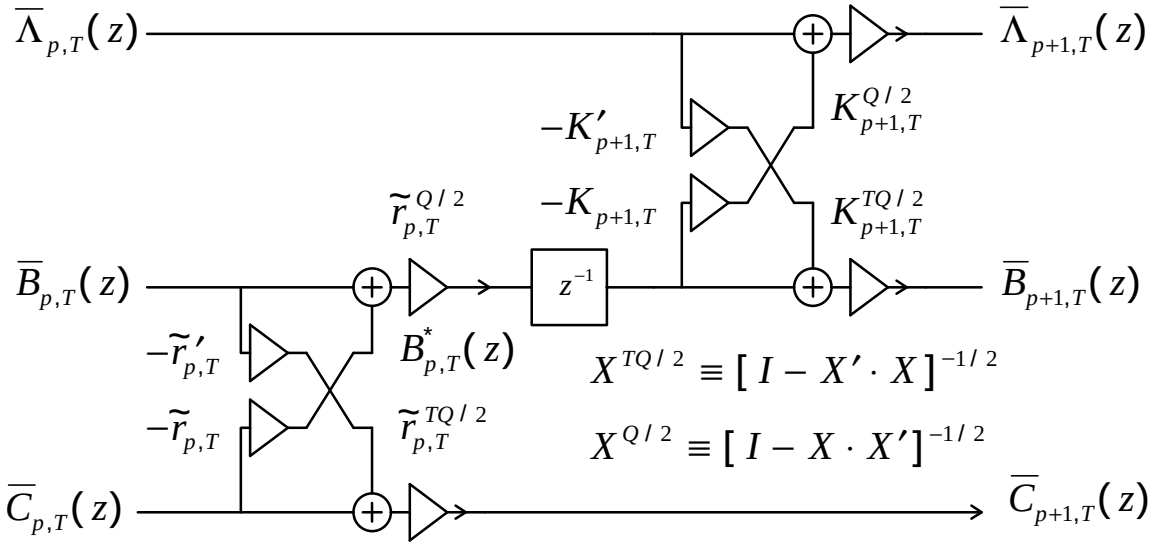


Figure 20 - Filter Section for Figure 19

Figure 21 - SigLab Model 20-22 Signal Analysis System

Creating an Integrated SID Application

As one can see from the discussion so far, the task of System Identification is somewhat complex and loaded with vector and matrix calculations. It should also be clear that the tasks of data acquisition and identification should be integrated to facilitate adjusting acquisition and identification parameters without swapping between applications or importing and exporting data from one environment to another (floppy net?).

The SigLab Model 20-22™ shown in Figure 21 was expressly designed to provide high-quality data acquisition and excitation for applications such as SID. It has all of the features listed in the "best" column of Table 1 and the architecture shown in Figure 4 is a subset of its actual architecture. It uses state-of-the-art sigma-delta conversion technology, coupled with two fixed-point DSPs and one floating-point DSP for real-time digital

filtering and general calculations and control. It is a compact unit the size of a notebook PC and is interfaced to a host computer via a SCSI bus. It has two analog input channels with high-quality anti-alias filters and two analog output channels with bandlimited burst random noise support. It also has full-triggering and time-averaging support to provide the best possible measurements of the input-output time histories required by the SID process.

The MATLAB (from The MathWorks Inc., Natick MA 01760) software environment is the ideal choice for creating an integrated acquisition and identification application. MATLAB v4.2 has the following important features: unmatched vector/matrix calculation capability.

- GUI tools to create the human interface
- Extensive graphics plotting capability
- The perfect environment for post-processing results and or control system design
- Interface support for SigLab 20-22

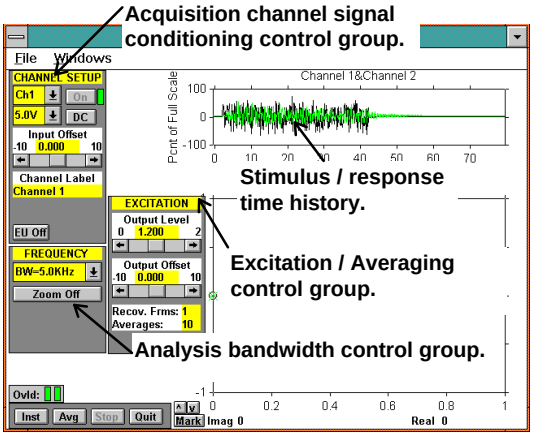


Figure 22 - System Identification Virtual Instrument in Data Acquisition Mode

As previously mentioned, there are three steps to the SID process. The first step is data acquisition. Figure 22 shows a screen of the MATLAB/SigLab-based integrated SID application called **vid**. The application is shown in the acquisition mode. Control over input channels, bandwidth, excitation, and averaging parameters is implemented with an easy-to-use graphical point-and-click interface. The input-output time histories are plotted as a percentage of full-scale to facilitate setting up input and output levels. Once the proper levels have been established you can click on the **Avg** button and the number of input-output records specified by the **Averages** will be acquired and averaged. Triggering is fully automatic since the stimulus and acquisition are internally linked in the SigLab hardware. When the averaging terminates, an FFT-based transfer function is calculated (per Figure 5f) and displayed as shown in Figure 23.

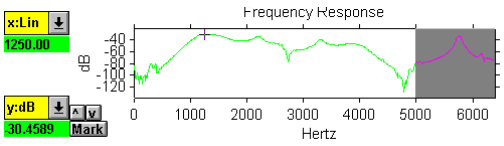


Figure 23 - FFT-based Transfer Function Estimate

Once satisfied with these results, the vid application is switched to analysis mode with a simple menu pick.

Figure 24 shows vid in the analysis mode. A z-plane model has been chosen. All the user had to do was set the **Model Order** slider to the desired maximum model order and press **Run**. Once the model has been produced the user can change the sampling rate if desired. The frequency response of the model is overlaid with the previously computed FFT-based estimate in the top graph area. This analysis takes less than 1 second on a 90 MHz Pentium-class machine.

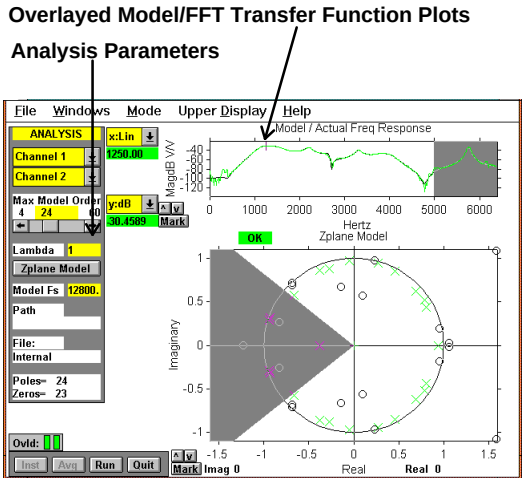


Figure 24 - vid in Analysis Mode (z-plane)

If an s-plane model is desired, the user simply pushes the button in the center of the Analysis controls group to change to the s-plane. Figure 25 shows an s-plane model.

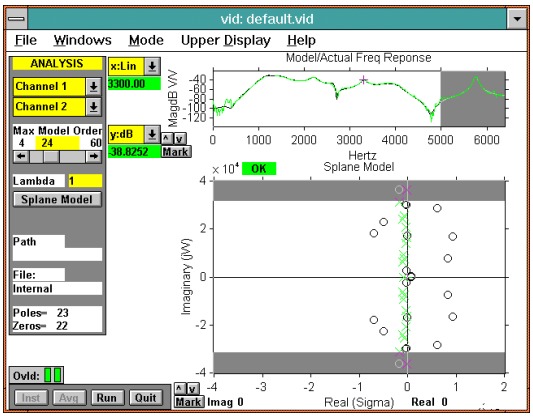


Figure 25 - vid Producing an s-plane Model

Once satisfied with the model, the results can be either saved to a file or transferred directly to another MATLAB application. The full state of the vid application can also be saved for future analysis of the same time history data or reference.

Conclusion

A detailed exposition on a practical SID method has been presented. This algorithm has been in use for over ten years and has provided consistently good results in modeling complex dynamic systems. A number of real-world examples have been given that are intended to give the reader a good idea of the various types of systems that are amenable to this form of analysis.

It should be remembered that the data acquisition phase of the process is critical to obtaining trustworthy results with any sophisticated signal processing algorithm.

Acknowledgment and Further Reading

This work relies heavily on the techniques published by Benjamin Friedlander. The notation used in this note is virtually identical to that used in the Friedlander paper. The lattice techniques evolved from pioneering work done by such luminaries as Martin Morf, Thomas Kailath, John Burg, David Messerschmitt, and a veritable host of others. For general background on adaptive filtering and time-domain system identification see references. [11],[12]

For more information contact:
Spectral Dynamics
1010 Timothy Drive
San Jose, CA 95133-1042
Phone: (408) 918-2577
Fax: (408) 918-2580
Email: siglabsupport@sd-star.com
www.spectraldynamics.com

---

[1] K. J Åström and P. Eykhoff, "System Identification-a Survey," *Automatica*, Vol. 7 (1971), 123-167.
[2] The MathWorks Frequency Domain Identification Toolbox uses these techniques.
[3] The MathWorks Control System Toolbox User's Guide (October 30, 1990) presents five methods for mapping between the Z and S plane, 2.42-2.44.
[4] DSP Technology Inc., "Interfacing Telephony Signals to DSPT SigLab™" Request AN2.0 from Fremont CA, 510-657-7555.
[5] Benjamin Friedlander, "Lattice Filters for Adaptive Processing," *Proceedings of the IEEE*, Vol. 70, No. 8 (Aug. 1982) 829-867.
[6] Ibid.
[7] L. Ljung, M. Morf, and D. Falconer,"Fast Calculations of Gain Matricies for Recursive Estimation Schemes," *International Journal of Control*, Vol. 27, No. 1 (1978), 1-19.
[8] Benjamin Friedlander, "Lattice Filters for Adaptive Processing," *Proceedings of the IEEE*, Vol. 70, No. 8 (Aug. 1982) p. 846.
[9] Ibid., p. 845, Fig. 10a, p. 847, Fig. 11, Table VI.
[10] Ibid., p. 848.
[11] Michael Honig and David Messerschmitt, *Adaptive Filters* (Hingham, MA, Kluwer Academic Publishers, 1984).
[12] Ljung, L. and Soderstrom, T., "Theory and Practice of Recursive Identification," *The MIT Press*, (Cambridge, MA., 1983).