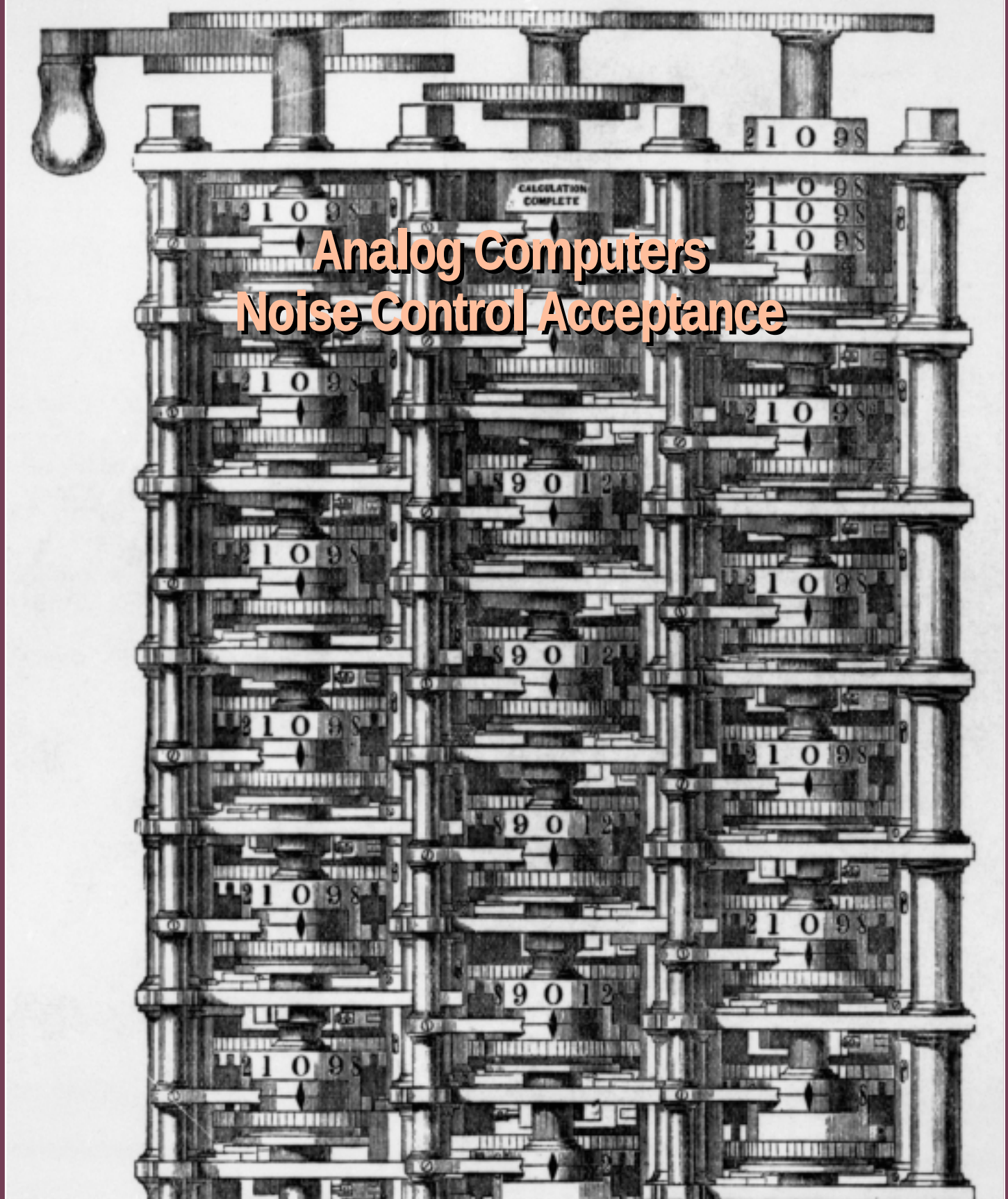
SOUND&VIBRATION
NOISE AND VIBRATION CONTROL AUGUST 2000



Analog Computers
Noise Control Acceptance

Analog was not a Computer Trademark!

Why Would Anyone Write About Analog Computers In Year 2000?

George Fox Lang, Data Physics Corporation, San Jose, California

The analog computer was once the leading-edge tool for dynamic simulation and vibration analysis. Now, it is an almost forgotten relic of technical history. Yet, this old war-horse still has important lessons to teach the new S&V practitioner! This article puts a new spin on an aging technology and provides an interesting "learn-by-doing" experience for young readers with a sense of intellectual curiosity and an experimental bent. It serves to narrow the communications gap between mechanical and electronic engineers and technicians by demonstrating the similarity of their respective efforts.

Data Physics Corporation is a sea of bright, industrious programmers; it is quite natural that I have made many friends among them. I recently shared a cup of coffee with one young fellow while he took a break from exercising the "wonders of Windows®" and I enjoyed the respite from designing a sales/training simulator. I casually asked him, "Have you ever worked with an analog computer?" I was not prepared for his response. "I've never heard of that brand!," he said. Although it would have been a great comedic one-liner, this young Computer Scientist was deadly serious . . . he had never heard of an analog computer.

I pondered this response during a long plane ride and came to realize that I was among the fortunate few engineers whose age placed his practice in overlap with the "golden age" of analog computation. Since the analog computer had such a profound and positive effect on my career both as a vibration analyst and as an instrumentation engineer, I recognized an obligation to reintroduce this important piece of our past to younger readers of *Sound and Vibration*. In essence, you bright folks were cheated by a fluke of time's passage and I would like to try to minimize your loss.

I was introduced to the analog computer during an undergraduate course in differential equations and had a chance to 'dabble' with one during a subsequent vibration lab/course. It was my very good fortune to study these topics with the former Chief Engineer of MB Electronics, Professor Thomas Warner, who conveyed them with sufficient enthusiasm and detail to set the course of my life's work.

My introduction to the "real world" of vibration analysis took place at Sikorsky Aircraft where I initially labored in Dr. Ray Carlson's Airframe Dynamics Group. I shared the harness with some truly skilled applied mathematicians (Drs. Art Jacobon, Cliff Astill, Bernie Traphan and Robert Johnston) who were kind enough to give "the kid" a solid introduction to mathematical modeling of structures and systems. Sikorsky also gave me an introduction to the *digital* computer, then still in its infancy. I "cut" a lot of punch cards to feed our twin Univac 1108s in that era. Many of these were inputs for *Mimic*, a Fortran-derived simulation language that tried to emulate an analog computer.

Several years later, I came to work for a real analog simulation laboratory at General Motors Proving Ground. I was mentored by a gifted engineer who really understood analog computation and was provided with all the "right stuff" to perform meaningful work. I was introduced to the "industrial strength" analog computer and later to the hybrid computer on which I built my first FFT system. It was a great learning experience.

In later years, I called upon these lessons to help resolve digital algorithm development problems. At Nicolet Scientific, Fox Technology and Liberty Technologies we built dedicated analog simulators to provide the "code cutters" with well understood problems against which to test their measurement algorithms. At all three companies, we also built sales-support simulators to allow our "sales troops" to do a more time-efficient and complete job of demonstrating the superiority of our high-technology products. In both applications, analog simulation provided exactly the right answer to a nagging and expensive problem.

What Is an Analog Computer and How Is It Used?

An analog computer (see Figure 1) is a collection of electronic components that may be interconnected in such manner as to produce a defined set of time-variant voltages, each analogous (and proportional) to a dependent variable in a group of equations to be solved simultaneously. These machines are naturally disposed toward the solution of differential equations where time is the *independent* variable. Thus, the simulation of structural vibration problems is a 'natural' for these machines.

Analog computers produce continuous signals; there is no concept of sample-rate involved in their basic operation. The computed variables are not quantized; there is no concept of "word-length" in an analog computer. All computations take place simultaneously, continuously and in real-time; there are no "dependent sequence" issues to be dealt with – all variables are always current and available. These characteristics are very desirable in a simulation or modeling tool.

However, the analog computer has some stringent restrictions on its computations. All computed variables must fall within the *full-scale voltage* span of the machine's amplifiers. This requires appropriate *voltage scaling* (subsequently discussed by example) to assure that each variable utilizes the available voltage span effectively. Since the computation voltage span is fixed, each variable has only a limited dynamic range. Further, all coefficients used in the equations must be *positive* and must also be scaled to fall within the range of zero-to-one. The *precision* of any coefficient is ultimately determined by the tolerance of components within the computer.

The computation is accomplished by component subsystems that are naturally governed by the mathematical operation they perform. At the risk of oversimplification, we will concentrate on the "big three" subsystems that dominate the solution of ordinary differential equations with constant coefficients. These are the (inverting) 'Summer,' the (inverting) 'Integrator' and the coefficient Potentiometer or 'Pot.' These components are represented schematically in Figure 2 and discussed in detail in the "How Summers, Integrator and Pots Work" sidebar.

Let's presume we want to use these elements to solve the forced-response of a single degree-of-freedom spring/mass/damper system. The equation to be solved is the familiar non-homogeneous, ordinary, linear differential equation with constant coefficients:

$$m\ddot{X} + c\dot{X} + kX = F \qquad (1)$$

where X is the displacement of mass m supported by a spring of stiffness k and a viscous damper of rate c and F is an external force applied to the mass. The "over-dots" represent differentiation with respect to time (in Newtonian notation). Since we are interested in the forced-response (the *complementary integral*), we may assume all of the initial conditions are equal to zero.

The method of solution (often called the "bootstrap method") is elegant in its simplicity. We implement a solution for the equation's highest derivative (d^2X/dt^2 in this case). This result is a summation of terms, each involving the dependent variable X and/or the lower order derivatives of X and constant coefficients. Multiple integrations (two in this case) of this result provide X and its remaining derivatives. These terms are

Figure 1. A typical analog computer of circa 1960 featured a removable program patch board, banks of ±100 Volt (vacuum tube!) amplifiers and hundreds of coefficient potentiometers plus other support circuits and accessories. The unit shown is an EAI model 231-R, one of five such 'workhorses' housed at the GMPG Noise and Vibration Laboratory and frequently driven in tandem! The human workhorse is far more unique; Tom Harris was my boss, mentor and best friend during my tenure with General Motors.

"fed back" to the initial computation (through constant coefficient potentiometers) as inputs. Where necessary, additional summers are used as signal inverters so that all voltage inputs have the required sign.

We start by solving (1) for the highest derivative of X (the acceleration). Specifically:

$$\ddot{X} = \frac{1}{m}F - \frac{c}{m}\dot{X} - \frac{k}{m}X \tag{2}$$

We recognize that the velocity can be found by integrating the result of (2). That is:

$$\dot{X} = \int \ddot{X}dt \tag{3}$$

We further recognize that integrating the velocity of (3) results in the displacement, or:

$$X = \int \dot{X}dt \tag{4}$$

Clearly, Equation (2) can be implemented by a Summer and three coefficient potentiometers. The excitation voltage F is an available external input. The dX/dt and X voltages are computed in accordance with Equations (3) and (4) by using Integrators. Proper (inverted) signal sense is obtained by using additional Summers as sign inverters. These motional terms and the fixed coefficients $1/m$, c/m and k/m are all that is required to satisfy the initial summation of (2) and we can thus solve the equation with the obvious 'loop' circuit shown in Figure 3.

We can also solve this problem with the more compact circuit of Figure 4. Here we have 'traded' the availability of the acceleration (and positive velocity) signals for a reduced hardware solution. This is clearly advantageous if the acceleration signal is not explicitly required. We have also gained a less obvious advantage – the circuit of Figure 4 is usable over a broader range of frequency than that of Figure 3, because the acceleration is not explicitly formed. (More about this, later.)

It is worth noting two common characteristics of Figures 3 and 4. First, the path from the F input to the X output is non-inverting in both cases because the signal path contains an even number of inverting amplifiers. That is, applying a positive force produces a positive displacement. Second, every closed loop within both models contains an odd number of inverting amplifiers in the signal path. That is, every feedback path introduces a stabilizing negative feedback. A retarding force is applied to the mass in reaction to its instantaneous displacement and velocity. In contrast, positive feedback loops (those with an even number of amplifiers) tend to destabilize the system leading it to saturate at "full-scale" when a small disturbance is applied. Since the system we are simulating is stable, its analog must also exhibit this property!

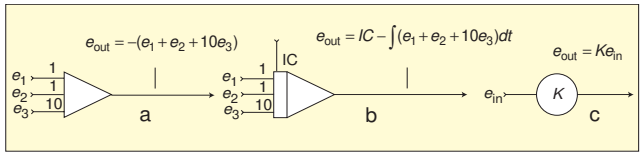


Figure 2. The big three linear computing elements: a) Summer accepts one or more inputs, each with an input gain (×1 or ×10 on most computers). The output voltage is minus the sum of the (gain weighted) input voltages. b) Integrator behaves in similar fashion, but the output voltage is minus the integral of the weighted input sum plus a specified Initial Condition (time = 0) voltage. c) Potentiometer multiplies an input voltage by a positive constant less than one ($0 \le K \le 1$).

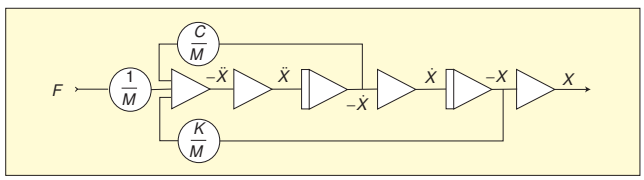


Figure 3. Obvious solution for forced response of spring/mass/damper.

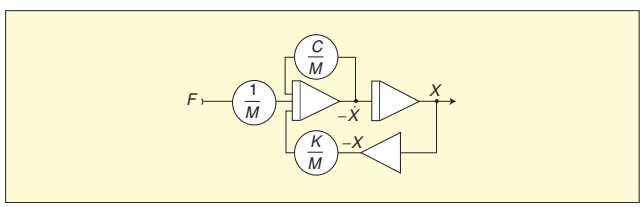


Figure 4. Compact analog solution for forced response of spring/mass/damper system.

The circuits of Figures 3 and 4 demonstrate the simplicity of the 'bootstrap' method. Since we are dealing with simultaneous voltages that are continuous signals, there are no "sequence details" to be attended to (as there must be in a time sampled digital implementation of Equations (2), (3) and (4)). Nor do we need to be concerned with how well the integration is approximated by a discrete summation; the integrations are *perfect* natural operations. However, our model is not yet complete. There are still several important matters of scaling to be attended to. We must scale the involved variables to fall within the voltage span of the computer amplifiers and we must scale the coefficients to fall within the zero-to-one restriction of the potentiometers. These matters are discussed in detail in the ensuing example.

Computer programming (of any type) is only efficiently implemented when it is approached systematically. Analog programming is no different from digital programming in this philosophical regard. The most important aspect of the activity is planning what to 'code' or 'patch' – the least important is the actual authoring of statements or the interconnection of components. Fail to plan the effort before embarking on implementation and you will, at best, be doomed to languish in an unnecessary resurrection period killing 'bugs' and making repairs. More likely, you will simply fail to accomplish your mission within a small multiple of the allotted time.

There are no 'shortcuts' to precise analog simulation. As with digital work, "the devil's in the details" and you won't understand those details without the proper preliminary work. In my opinion, there are eight steps through which you must pass before reaching for patch panel, bottle-plugs and cables. Try to shortcut this process and your simulation will surely be a poor mimic of the "real world."

Ask any "old-timer" about programming an analog computer and he will tell you that conceiving a simulation circuit is child's play, but scaling a model properly is the art that separates the men from boys. This task is not insurmountable; you have merely to follow all eight steps, in sequence, to succeed.

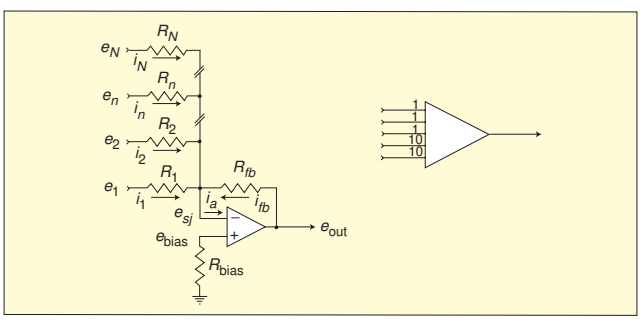
The "Down and Dirty" Details of Analog Programming

Figures 8 and 11 show an LDS model V-203 electrodynamic shaker. This small precise tool is a mainstay of our industry

How Summers, Integrator and Pots Work

The following figure shows the schematic of an inverting summation amplifier or Summer and its analog programming symbol. This circuit is built of precision resistors and an operational amplifier. 'Op-amps' now come 1, 2 or 4 to a 'chip' and are very inexpensive. A personal favorite is the NE5514, which provides 4 high performance op-amps in a 14-pin dual in-line pack (DIP).

The operational amplifier provides very high voltage gain, G, from DC to many kilohertz. It also exhibits extremely high input impedance *Z* and a near-zero output impedance. These devices have a differential input and provide a (single-ended) output voltage proportional to the difference of the voltages applied to the '+' and '–' input pins. Typical integrated circuit op-amps require +15 and a –15 volt power supplies and can provide very satisfactory analog computations within a ±10 Volt range from DC to 20 kHz. They provide no more than about 20 mA of output current, hence programming resistors should be in the 1 kΩ and higher range. Low-cost 1/4 watt resistors of 1% precision are readily available.



As many as *N* voltage inputs may be applied to a summing amplifier. As shown above, a network of resistors feeds the inverting input of the op-amp. This point in the circuit is termed the *summing junction* and it receives currents from the N input resistors and the output feedback resistor R_{fb}. This point in the circuit is at potential e_{sj} and the sum of currents entering and leaving this node must equal zero. Hence, we can express the current i_a entering the inverting input of the op-amp:

$$i_a = i_{fb} + \sum_{n=1}^{N} i_n = \frac{e_{\text{out}} - e_{sj}}{R_{fb}} + \sum_{n=1}^{N} \frac{e_n - e_{sj}}{R_n}$$

The amplifier's output is determined by its gain and the voltages at its differential inputs. The output voltage may be stated:

$$e_{\text{out}} = G(e_{\text{bias}} - e_{sj})$$

However, the amplifier's *non-inverting* input is held at (essentially) ground potential by the bias resistor R_{bias}. That is:

$$e_{\text{bias}} \cong 0$$

Because of its very high gain, the amplifier functions to maintain the *summing junction* at ground potential and we may state:

$$e_{sj} = -\frac{e_{\text{out}}}{\text{G}} \Rightarrow 0$$

Further, the very high input impedance of the op-amp means that virtually no current enters its *inverting input*. That is:

$$i_a = \frac{e_{sj}}{\text{Z}} \Rightarrow 0$$

Hence our summing junction current summation collapses to:

$$\frac{e_{\text{out}}}{\text{R}_{\text{fb}}} + \sum_{n=1}^{N} \frac{e_n}{R_n} = 0$$

The obvious result is that e_{out} must be minus the gain-weighted sum of the *N* applied input voltages e_{in}:

$$e_{\text{out}} = -R_{fb} \sum_{n=1}^{N} \frac{e_n}{R_n} = -\sum_{n=1}^{N} \frac{R_{fb} e_n}{R_n} = -\sum_{n=1}^{N} g_n e_n$$

In commercial computers, the g_n input gains were set to standard values (normally 1 and 10) by incorporating input resistors R_n with values of R_f and $R_f/10$, respectively. In "hand-patching" a summation amplifier, you are free to select each R_n for a desired coefficient value.

To minimize DC errors caused by differential currents entering the amplifier, R_{bias} should be equal to the parallel equivalent of all resistances connected to the summing junction. That is:

$$R_{\text{bias}} = \frac{1}{\frac{1}{R_{fb}} + \sum_{n=1}^{N} \frac{1}{R_n}}$$

The folowing figure illustrates the schematic and analog programming symbol for an inverting *Integrator*. The circuit is virtually identical with that of *Summer*, except that the feedback element is a low-leakage precision capacitor. (For "home-grown" integrators, 50-volt polyester capacitors are available at 2% precision in values up to 1 μF.)

and it offers a perfect target for analog simulation. Our intent is to build a model that accurately portrays the electrical and mechanical characteristics of this machine and its interaction with a device under test upon it.

The shaker has a compliant suspension of stiffness *K* and damping *C* suspending a load-table of mass *M*. The table is driven by a voice-coil of resistance *R* and inductance *L* which applies a force to the table in proportion to coil current *i* and the magnetic constant k_2. The table motion is resisted by the reaction force F_r of the test object. The moving coil also generates a back EMF in proportion to table velocity and the magnetic constant k_1. The shaker is driven by a voltage e_{in} from a power amplifier.

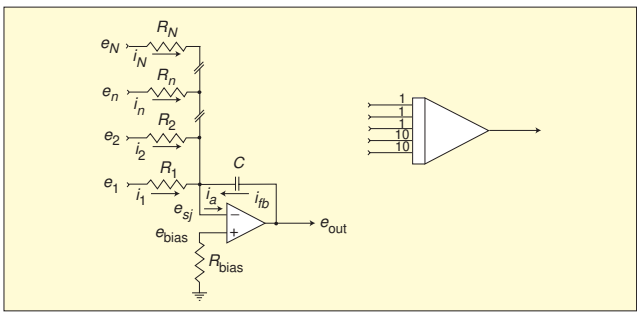
The pertinent equations are:

$$M\frac{dX^2}{dt^2} + C\frac{dX}{dt}KX = k_2 i + F_r \quad \text{(mechanical)} \qquad (5)$$

$$Ri + L\frac{di}{dt} = e_{\text{in}} - k_1\frac{dX}{dt} \quad \text{(electrical)} \qquad (6)$$

Since we understand the basic physics of our target and can write the governing differential equations, we are well on our way to building an analog simulation. However, we need to know all of the coefficients in the equations, as well as the expected range of every variable before proceeding. Fortunately, we know a little bit about this shaker, having analyzed it extensively for the Electrodynamic Shaker Fundamentals article that appeared in the April 1997 issue of S&V. In particular, we know:

$M = 178.04 \times 10^{-6}$ lb sec^2/in. ($M_g = 68.794 \times 10^{-3}$ lb)
$K = 16.54$ lb/in.
$C = 39.09 \times 10^{-3}$ lb sec/in.
$R = 1.6\ \Omega$
$L = 764\ \mu$H
$K_1 = 95.10 \times 10^{-3}$ Volt/IPS
$K_2 = 0.8416$ lb/A
Stroke = ±100 mil
Maximum Acceleration = ±135 g (use ±100 g full-scale for this model)
Maximum Force = ±4.4 lb (use ±5 lb full-scale for this model)



Following the same method of analysis, the current entering the op-amp's *non-inverting* input from the summing junction may be stated:

$$i_a = i_{fb} + \sum_{n=1}^{N} i_n$$

$$= C\frac{d(e_{\text{out}} - e_{sj})}{dt} + \sum_{n=1}^{N}\frac{e_n - e_{sj}}{R_n}$$

As in the *Summer* derivation, the summing junction voltage e_{sj} and the amplifier input current i_a are recognized as being vanishingly small. Thus, the current balance collapses to:

$$C\frac{d(e_{\text{out}})}{dt} + \sum_{n=1}^{N}\frac{e_n}{R_n} = 0$$

From which the output voltage may be solved, yielding:

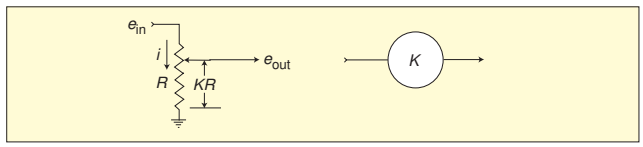
$$e_{\text{out}} = -\frac{1}{C}\int\sum_{n=1}^{N}\frac{e_n}{R_n}dt = -\sum_{n=1}^{N}\frac{\int e_n dt}{R_n C}$$

$$= -\beta\sum_{n=1}^{N} g_n \int e_n dt$$

In a commercial computer, the R_n values were set equal to R_{fb} and $R_{fb}/10$ to provide standard input gains g_n of 1 and 10. The capacitor was selected by the Time Scale selector to provide a $\beta = 1/R_{fb}\,C$ of 1/10, 1, 10 or 100. When fabricating your own integrator, you can choose from a broader repertoire of R_n values to achieve the desired integrator gain.

Again, to minimize DC errors caused by differential currents entering the amplifier, R_{bias} should be equal to the parallel equivalent of all resistances connected to the summing junction. In this instance:

$$R_{\text{bias}} = 1/\sum_{n=1}^{N}\frac{1}{R_n}$$



The figure above illustrates the schematic and programming symbol for a coefficient *potentiometer*. In commercial computers, expensive 10-turn (linear taper) variable resistors with calibrated geared dials were employed. For 'homegrown' circuits, inexpensive multi-turn Cermet® trimmers work well.

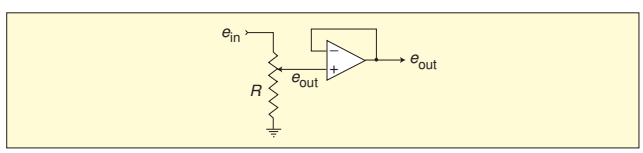
The current i passing through the potentiometer to ground is determined by the total resistance of the pot and the applied input voltage. That is:

$$i = \frac{e_{\text{in}}}{\text{R}}$$

The resistance between ground and the potentiometer's adjustable "arm" is determined by the arm's position. This is set to provide a resistance of KR, where K is the desired coefficient ($0 \le K \le 1.0$). Thus the output voltage is:

$$e_{\text{out}} = KRi = Ke_{in}$$

You will note that the assumption was made that no current flowed from the potentiometer's arm to the input of an amplifier. This is untrue; the pot always 'sees' a load of R_{fb} or less in a commercial machine or home-built circuit. To circumvent this problem, the commercial computers provided a *Pot Set* mode in which the input of the potentiometer was temporarily connected to a "full scale" reference voltage, while the arm's output (still attached to its operating load) was measured with a digital voltmeter. The coefficient was set by reading this meter, *not* the expensive multi-turn dial.



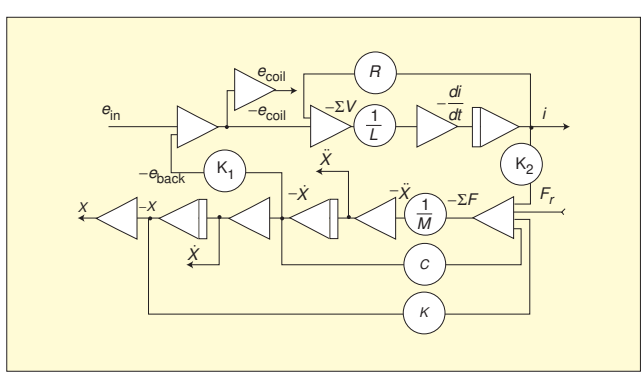
You can solve this "pot-loading" problem in another way. An operational amplifier can be used as a *voltage follower*, a unity gain non-inverting amplifier with extremely high input impedance and nearly zero output impedance. It is made by connecting the amplifier's output to its inverting input and applying the voltage to be 'followed' to its non-inverting input, as shown above. Since the amplifier draws only an infinitesimal current from the potentiometer's arm, the 'knob' of the potentiometer reads K exactly.

Maximum Current = ±2.5 Amp
Amplifier Voltage range = ±10 Volts

We lack a velocity full-scale. The maximum acceleration and stroke limits (135 g, 100 mil) 'cross' at 114.95 Hz, where the velocity would be 72.225 IPS. We will use a full-scale of 75 IPS for this model.

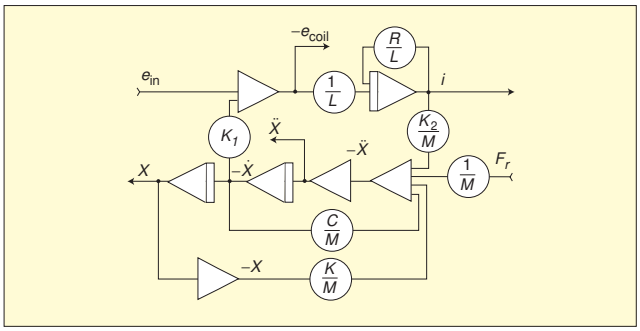
Step 1. Draw a "First-Pass" unscaled circuit diagram for the shaker simulation. This assures that all of the pertinent variables are present and can be computed. Be certain that any variable to be monitored is the output of an amplifier, not a pot. Verify the stability of the circuit by counting amplifiers within each closed loop; (in general) there should be an odd number, indicating a stabilizing "negative feedback" within that loop. Be very certain that any given variable is only computed in one place in the circuit; don't depend on two separate computations to arrive at the same conclusion!

You will note from the circuit below that the electrical drive circuit experiences a 'reflection' of the mechanical system being vibrated. At 'package' resonances the input impedance will rise due to the back EMF; this reduces the coil current



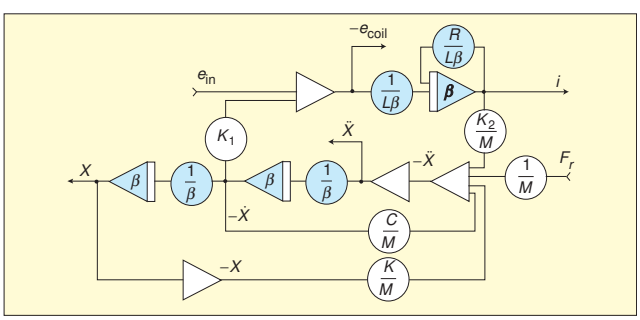
which, in turn, reduces the applied force just as in a real shaker application.

Step 2. Simplify the circuit by eliminating variables that are not explicitly needed for the study. In particular, minimize the *order* of differentiation applied to any one variable; this will conserve dynamic range. As an example, you will note that the



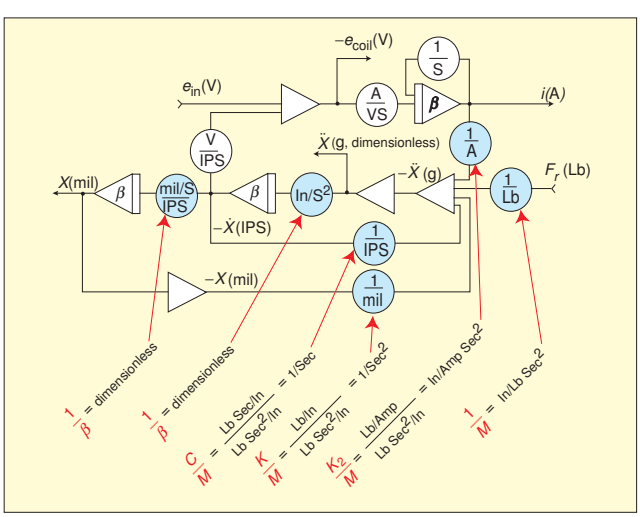
di/dt term has been eliminated by allowing an *integrator* to provide the necessary voltage summation. If the acceleration were not a desired output of the simulation, this same approach would have been applied to the d^2x/dt^2 term. Sometimes an amplifier can be saved if you are willing to make do with the negative of a desired input or output. We made this choice with the e_{coil} signal, but labored to assure that the e_{in}, i, F_r, x, and d^2x/dt^2 signals were all of positive sense. Make a concerted effort to minimize the amount of hardware employed at this stage . . . you will always find a need for those undeclared amplifiers and potentiometers when later 'milking' your model!

Step 3. Assign the arbitrary voltage gain β to all integrators. Divide the coefficient of any pot 'feeding' an integrator by this



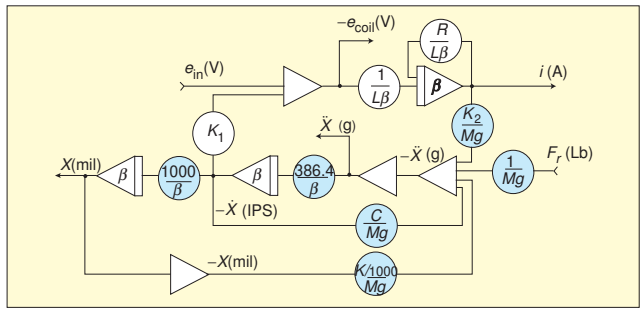
same constant. Wherever an integrator receives an input directly from another amplifier, insert a pot (as shown in two places) and assign a coefficient of $1/\beta$ to it. Ensuing steps will disclose why you should make these additions now.

Step 4. Assign the desired physical units to the output of each amplifier and perform a dimensional analysis to deter-



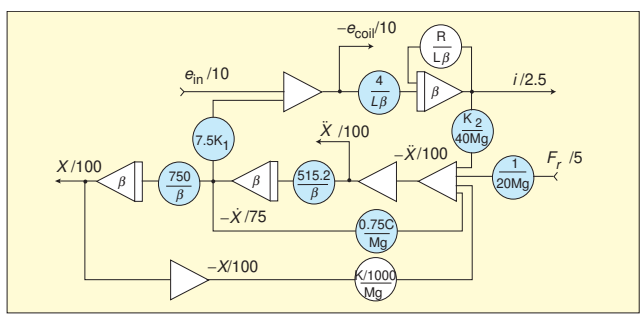
mine the required physical units of each coefficient. Treat the added gain factors β as dimensionless. Recall that the output of an integrator has the units of its input multiplied by seconds. Compare the coefficient units with those planned in step 3 and note all potentiometers with a unit disagreement (six found here). These coefficients will have to be modified to provide dimensional consistency with the selected output units.

Step 5. Correct the erroneous coefficient entries detected in step 4 (as shown here at six places). Note that such coefficient



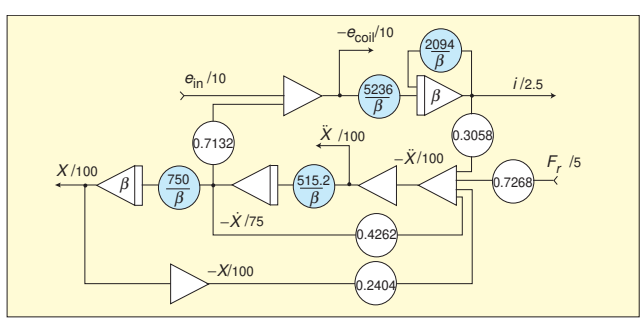
revision is rarely required when basic ISO physical units are consistently selected for each amplifier output. They are inevitable when *ye olde* English units are chosen (as they always were 30 years ago)!

Step 6. Divide each amplifier's output by its full scale value and amend coefficient values for those pots between amplifi-



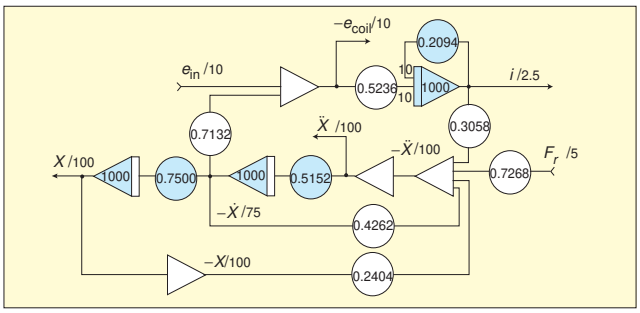
ers with two different full-scale numbers. The amended value is equal to the original coefficient multiplied by the numerical full-scale of the signal at the pot's input and divided by the numerical full-scale output of the amplifier 'fed' by the pot (as shown at seven highlighted locations).

Step 7. Apply the numeric values (or range of values) to every coefficient in the diagram. Pot settings must fall between



0 and 1. Larger coefficient values preceding inverters and summers require using amplifier input gain (i.e. ×10) so that the pot setting may be reduced to less than 1. Integrators (see step 8) offer another option. Pot settings less than 0.05 should be avoided as they lead to poor signal/noise ratio. If all coefficients cannot be satisfactorily rectified, return to step 6 and change the full-scale values involved. Note that our simulation is satisfactory with regard to all coefficients except the highlighted four which "feed" integrators.

Step 8. Select the required gain β of the integrators. Choose a value that allows all of the potentiometers 'feeding' integrators to be set to a value of 1 or less. In our example, a gain of β = 1000 satisfies this requirement for two of our three integrators. The third (upper-right) is also satisfied by this value when



the higher gain (×10) inputs of that integrator are used (as shown).

Analog computers provide a limited set of integrator β gains (normally 1/10, 1, 10 and 100). They are selected by the computer's Time Scale rotary switch. This selector changes the capacitor value used in *all* of the integrators simultaneously. This provides (×1 input) integrator gain of 1 Sec $/R_{x1}C$, numerically equal to the Time Scale Switch setting.

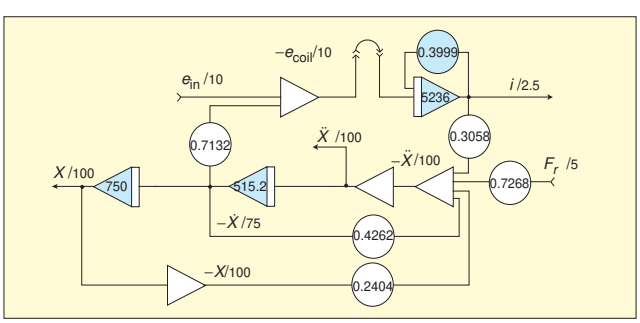
If your required β is equal to the Time Scale setting, your simulation will run in "real-time" behaving just like the physical hardware it models. If the required β is not available, the simulation can still be run but its responses will be Time Scale / β as fast as those of the modeled hardware. Analogs that run in this manner are said to be time-scaled.

In this example, a β of 1000 is required and the maximum available Time Scale is 100. Set the pots as though the required β gain was available and the model will run in 1/10 real-time. That is, the simulation will take 10 times as long to do something as the actual shaker does. Hence all model frequencies will be 1/10 of those in "real-life."

The requirement to run this model more slowly than real life is the result of including the desired dx^2/dt^2 term in concert with the selected full-scale values. (Recall $dx^2/dt^2 = (2\pi f)^2 x$ for simple sinusoidal motion at frequency f.) This combination required a broader voltage dynamic range than could be accommodated in real-time given the limited choices of capacitors available, the restriction that only *one* capacitor choice must be made globally for all integrators and that the simulation must run within a fixed voltage range. By reducing the frequency, computational voltage range was recovered through time scaling.

Time scaling is also useful in its own right. We often "slowed down" simulated transient events like automotive barrier-crashes to allow more detailed analysis of the events that transpired than our available "real-time" instruments could follow. We also "time compressed" some analyses to bring low frequency terms into the range of human hearing. Problems involving partial differential equations were studied using two analog computers running at vastly different time scales; the spatial derivative computations were run rapidly (and repetitively) and used as a "lookup table" for the slower running temporal simulation.

Step 8a. Elect to construct the analog using IC operational amplifiers and discrete resistors and capacitors. In this instance, each integrator may have a unique gain β. Coefficients that will not be varied during the model study do not require



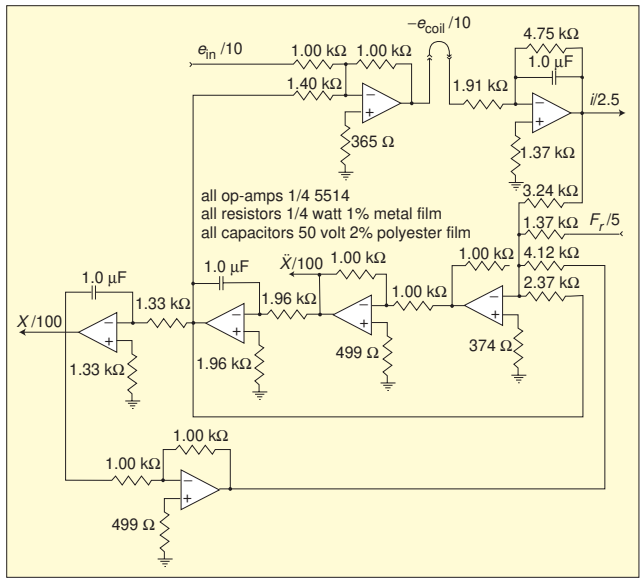


Figure 5. Schematic diagram for the shaker simulation.

Figure 6. The circuit of Figure 5 was constructed from two integrated circuits (with one amplifier left over!) on a shop-built prototyping rig. The rig contains precision power supplies and a convenient way to interface BNC connectors with the circuit. Commercial development boards, such as that shown, can also be used to build small circuits. Planning the component placement before patching is always a good idea . . . note the pictorial layout.

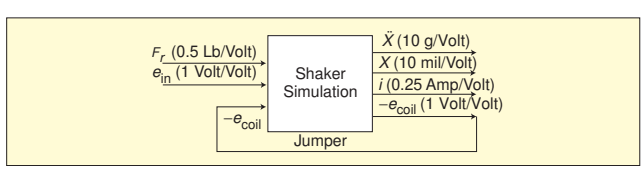


Figure 7. Black box representation of the shaker simulation circuit showing scaled inputs and outputs. This circuit may now be used as a hardware 'subroutine' and incorporated in larger simulations.

a potentiometer. Instead, the required input gain is set by fixed resistor selection. This figure shows three different integrator gains, all of which can easily be achieved so that the analog will run in real-time. As an example, the gain of 5236 can be achieved using a 0.1 μF capacitor and a 1.91 $k\Omega$ resistor while the gain of 515.2 calls for 1.0 μF and 1.96 $k\Omega$. Note the change in the 'feedback' coefficient around the upper-right integrator; this reflects the selected integrator gain. A 'jumper' has been planned for to allow the shaker to be driven by the e_{in} input or examined with its coil unterminated in a 'pluck' test. Figure 5 illustrates a schematic generated from this planning diagram and Figure 6 illustrates the physical arrangement of its parts.

The circuit of Figures 5 and 6 may be thought of as a hardware subroutine, a component of a larger model. Figure 7 represents the shaker simulation subroutine as a "black box." Fig-

Figure 8. 'Pluck' testing an LDS model V-203 shaker. The finger depresses the table against its limit stop, then releases it abruptly. The voltage generated by the shaker coil is measured with a high impedance instrument. Comparative spectral measurements made between a bare table and one loaded with a known mass permit identifying the suspension parameters.

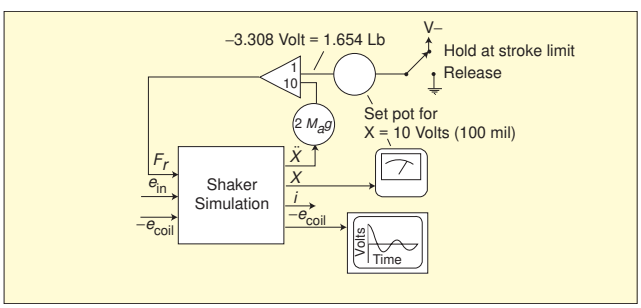


Figure 9. Setup for a simulated pluck test of the shaker model. The e_{coil} jumper is left open, so that no current flows through the coil. The $-e_{coil}$ output allows the coil's back-emf to be monitored while the shaker table is mechanically excited. With the switch in the "hold at stroke limit" position, the potentiometer is adjusted until the X output reaches 10 volts (corresponding to 100 mils), indicating the shaker suspension is at its extreme limit. (This should require a value of 3.308 Volts corresponding to 1.654 lb.) When the switch is moved to the 'release' position, this simulated force is removed from the shaker table, which is then free to vibrate. The 2 M_a g pot is used to simulate attachment of a mass to the shaker table. Set this pot to 0 for a "bare-table" test; use a value of 0.1376 to simulate the 31.27 gram accelerometer which I used in the hardware test of the September 1997 article.

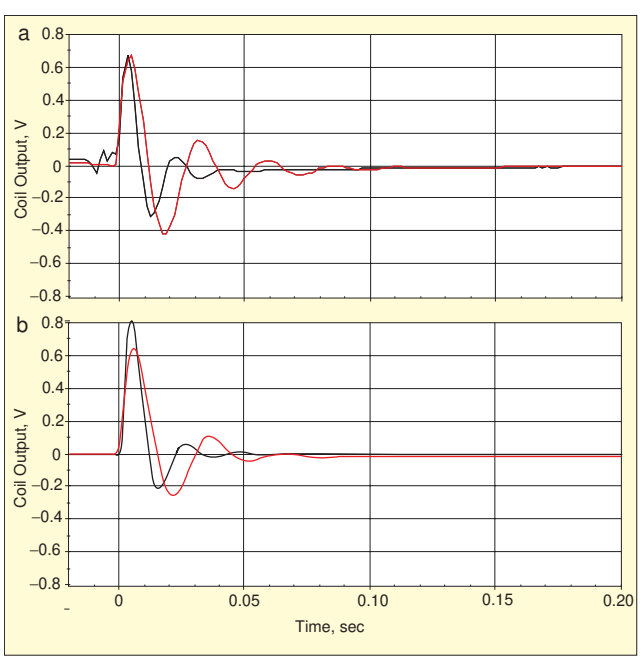


Figure 10. Pluck test measurements made with bare (black) and mass-loaded (red) table. a) Real hardware in time domain. b) Analog model in time domain.

Figure 11. Mechanical configurations for traditional coil impedance tests. The unit on the left is unrestrained, the shaker on the right has its table motion blocked by a bracket.

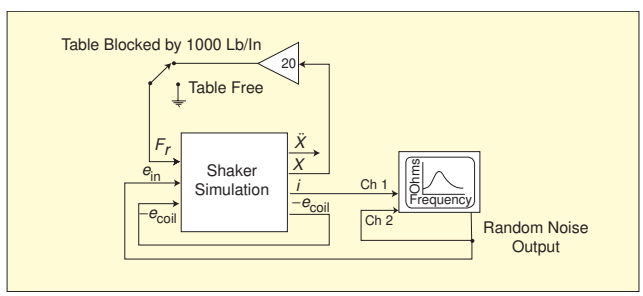


Figure 12. Simulation setup for a coil impedance measurement in "blocked table" and "bare table" conditions. The e_{coil} jumper is closed so that current, i, flows though the coil and drives the shaker. The e_{in} input is driven with white noise and the impedance ratio e_{in}/I is measured as a transfer function. When the switch is in the "table blocked" position, a force proportional to the negative of displacement X is fed back as the restraining force F_r. The gain shown simulates the restraining bracket as a 1000 lb/in spring.

ures 8 through 14 illustrate various experiments performed upon this analog element and compare the measured results against those of "real hardware." Please note the simplicity with which additional modeled elements are "boot-strapped" to the shaker circuit.

Better Ain't Spelled Digital!

The experimental expansion of Figures 9, 12 and 14 present no difficulty for the analog practitioner. In contrast, digital simulations are always plagued by issues of sample rate and equation sequencing. To bring home these points, you might try solving Equations (5) and (6) in a spreadsheet such as Excel®, using the scaled diagram of Step 8a as a guide.

Create a column for time t and each of the seven dependent variables. Enter t=0 and the initial condition for i, dX/dt and X in the first row. Solve for the remaining dependent variables in this row in terms of the initial conditions so that a complete set of "time=0" variables is available. (Start by assuming F_r and e_{in} equal zero.) In the second row, set t equal to its first row value plus a constant increment dt. Implement a simple (rectangular) integration for i, dX/dt and X in terms of dt and the variables from the preceding row. Copy (actually *Fill ⇒ Down*) the equations for e_{in}, e_{coil}, F_r and d^2X/dt^2. Select the entire row and use *Fill ⇒ Down* to provide about 500 sample points.

Run a simple test using an initial condition of 1 for X (actually for X/100 mil) and 0 for i and dX/dt. Start with a dt value of 100 µsec (.0001). Make a simple *scatter plot* of all dependent variables against time. If your coding is correct, you will be rewarded by smooth exponentially decaying sinusoids for all variables, with the activity ceasing in about 1 1/2 cycles in 50 msec. Now make the dt steps courser (0.0002, 0.0003, etc.). Things will be fine until you reach dt=0.001, at which point

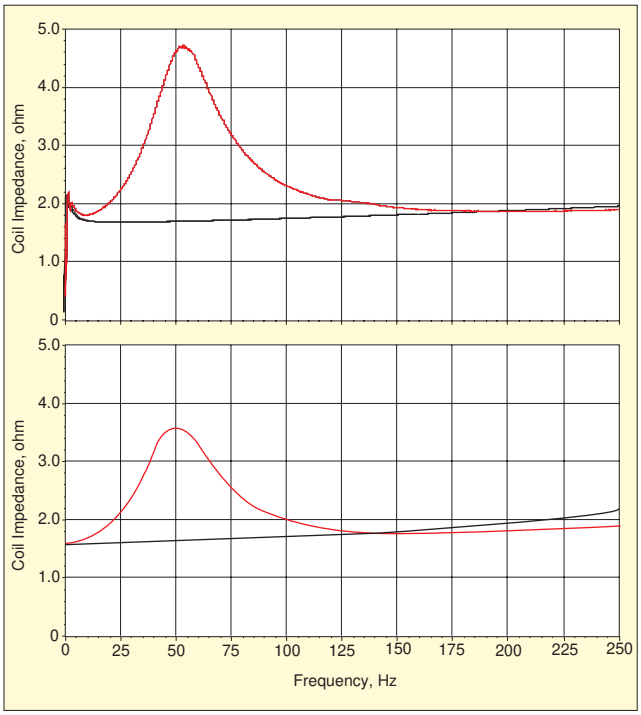


Figure 13. Coil impedance test measurements showing table 'blocked' (black) and free conditions (red). Top figure illustrates actual shaker, bottom figure shows measurements from the analog simulation.

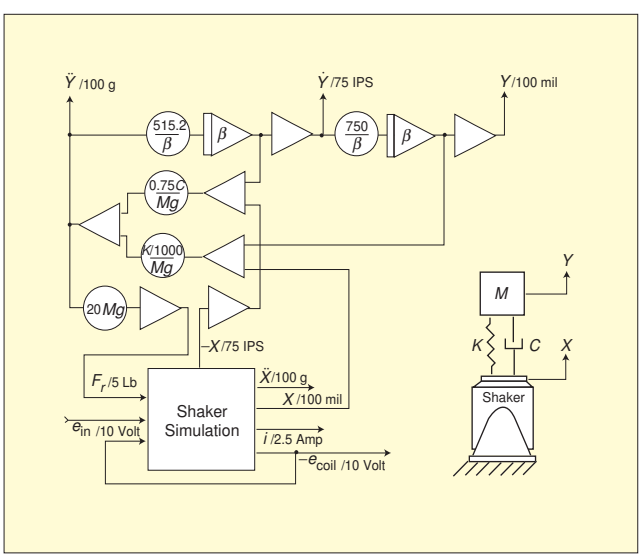


Figure 14. Food for thought and personal study! This scaled circuit simulates the "base-shake" of a single degree-of-freedom system attached to the shaker. Y and its derivatives have the same scale factors as those of the shaker simulation. Enter Mg in pound, K in pound per inch and C in pound-second per inch. By adding two more amplifiers to this circuit, you can create another hardware 'subroutine' suitable to stack a chain of spring/mass/damper systems on the shaker.

the simulation will 'blowup' due to a sample-rate induced instability of the current equation! Analog computers never introduced such artifacts.

Return *dt* to 0.0001 and try "boot-strapping" additional equations for the appended components of Figures 9, 12 and 14. You should have no problem with the experiment of Figures 12 and 14, but that of Figure 9 will defy you! Excel will complain about a "Circular Reference" and offer to help you resolve this issue by editing or performing an iterative computation to a standard you cannot control. (Thirty years ago *Mimic* complained of an "algebraic loop" and offered the same help!) You're far better off to reject the offered iterative solution and recast the equation (using your Control Systems text as a guide). You have just been foisted upon the horns of an "equation sequencing" problem, never encountered in continuous signal analog systems.

Thoughts in Closure

This article has merely scratched the surface of an important **body of knowledge in severe danger of being lost**. We have only discussed the most basic analog 'coding' of ordinary, linear differential equations with constant coefficients. While the analog computer handled such studies with elan, it really made heavy contributions in the understanding of nonlinear and time-variant systems, as well as those described by partial differential equations.

This example has not discussed the powerful *mode control* that allowed precise initial conditions to be specified for every integrator (and, therefore, for every variable in the simulation). Mode control also provided a precise means of verifying every potentiometer setting (pot set) and to hold the simulation, 'freezing' all variables at a specific instant in time. Mode control was extended in later computers to include *repetitive operation*. In 'Rep-Op' a precise time sequence consisting of a brief interval where initial conditions were applied followed by a specified duration of simulation operation was executed repetitively. This allowed various signals to be monitored on an oscilloscope while initial conditions and coefficients were varied.

We have not discussed any of the standard *nonlinear* analog components such as diodes, amplitude comparators, relays, track/store amplifiers, the "1/4-square" signal multiplier (which also provided square and square-root functions) or the diode function generator which provided graphically programmable nonlinear input/output amplitude relationships. Nor have we touched on the accompanying *parallel synchronous logic* circuits that augmented later machines.

Learning to use an analog computer forced the practitioner to cross the (unnecessary) boundaries between engineering disciplines; this may be its greatest virtue. More than a little dose of 'EE' was infused in this Mechanical Engineer by contact with the analog computer and this has stood me in good stead throughout my career. Creating a real physical entity (a circuit) governed by a particular set of equations and interacting with it provides unique insight into those mathematical statements. There is no better way to develop a "gut feel" for the interplay between physics and mathematics than to experience such an interaction. The analog computer was a powerful interdisciplinary teaching tool; its obsolescence is mourned by many educators in a variety of fields.

The continuous-signal analog computer interfaced naturally with measurements from existing mechanical hardware. At General Motors Proving Grounds we routinely used shakers and sensors to integrate existing automotive chassis with computer-modeled components. A frequent "rite-of-passage" found engine mounts being optimized by 'tweaking' coefficient pots in a (rigid body) analog simulation of the engine on its elastic mounts. This real-time simulation was driven by mount-point acceleration and force measurements and provided the drive signals to multiple shakers at the engine attachment points. The chassis had no 'real' engine, only that inferred by the computer. A "calibrated posterior" rode the car on a motored dynamometer with simulated road inputs and conferred with the 'tweaker' by radio. In this manner, many old engines found comfortable homes in new chassis and vice versa.

The analog machines had a short courtship with the digital computer giving rise to the very powerful but short-lived *hybrid computer*, perhaps the best real-time simulation tool ever devised by man. You can learn a lot more about the history and application of these old marvels at *Doug Coward's Analog Computer Museum* (www.best.com/~dcoward/analog/index.html). Doug includes a time-line of related inventions from 87 BC until the year of my graduation. He also provides a very complete reference bibliography. Most of the pertinent reference texts he lists are now out of print. I recently purchased several of these as used tomes in good condition at reasonable prices from *Powell's Books* (www.powells.com).

While these grand old machines are now passé, the lessons they offer can still be experienced today. One has merely to buy

power supplies, a wire-patchable prototyping board, a handful of "op-amp chips" and some precision resistors and capacitors (see Figure 6). These are but a few keystrokes away at www.digikey.com or www.mouser.com, for example. The author fervently hopes the technically curious and foresighted will recognize that this special area of our history has much to teach in ensuing years and that the best way to learn is by doing. Try a little of this "old stuff," I promise you'll learn something new!

The author can be contacted at: lang@data physics.com.

The Data Physics Mobilyzer®

SignalCalc® **Mobilyzer** is a modern-architecture Dynamic Signal Analyzer that distributes the processing load over multiple DSPs and CPUs to achieve a cost-effective signal analysis of the highest quality. A Mobilyzer chassis interfaces to an ordinary PC running Windows® 95, 98, 2000 or NT via a standard Ethernet network cable. The lightly-loaded computer becomes a viewport into the activity and content of dynamic signals.

Mobilyzer systems are offered in three chassis sizes: 4 channel, 16 channel and custom high-count systems. Within each size, many input/output configurations are possible. Each Mobilyzer chassis contains a Pentium CPU processor controlling multiple Digital Signal Processor (DSP) modules. Inputs may be differential or single-ended with AC, DC or ICP coupling. Signal generators provide sine, random, burst-random, pseudorandom, chirp and other waveforms of up to ±10 Volts. Multiple tachometer channels serve machinery analysis.

The instrument provides FFT (with zoom), octave and 1/3 octave analysis, order-tracking, MIMO, shock response spectra, correlation, synchronous averaging, histograms, waterfall and spectrogram presentations and disk-recording and playback. It shares the highly intuitive Windows interface of other SignalCalc analyzers and is fully ActiveX compliant.

The experimental measurements illustrated and utilized for this article were made with the four-channel SignalCalc Mobilyzer shown above. More information is available at www.dataphysics.com.

COVER

Our cover shows an 1889 woodcut of a portion of Charles Babbage's first Difference Engine, an amazing machine built to automate the computation of mathematical and navigational tables. This milestone in the history of computing machines was the direct outgrowth of a frustrated exchange between Babbage (1791-1871) and John Herschel, a noted astronomer of the time. The two friends were members of the Astronomical Society of London and labored together during the summer of 1821, checking the hand calculations of two independent "human computers" who had prepared sight-reduction tables describing the true positions of all of the Greenwich stars. These were fundamental navigational aides, allowing latitude to be deduced from an angle measured with a sextant. The calculations were ponderous and each discrepancy between table-pairs had to be rectified by arduous recomputation. Many errors were found, leading Herschel to exclaim in exasperation, "I wish to God these tables had been calculated by steam!" Babbage replied that he thought they could.

Babbage made good on his prognostication over the ensuing 37 years. While his invention was driven by a hand crank rather than by steam, he produced a machine that could calculate precise table values for a broad range of problems. The complex mechanism was based on the method of Finite Differences with its roots in differential calculus. It was programmable by component arrangement and incorporated mechanical memory elements retaining sub-step calculations. Parallel aspects of the calculation took place simultaneously, paced by a mechanical operating 'clock' input. Each machine cycle required four alternate-direction half rotations of the hand crank. Our woodcut shows about 1/7 of the total machine, that part responsible for the basic computation. This element was finished in 1832 and still functions today in the Science Museum in London.

As with John Harrison, whose chronometer solved the longitude enigma, Charles Babbage was treated poorly by the British Crown. He was never granted a patent for his efforts nor awarded full measure of the government support monies promised over the years. His saga provides further evidence that the most difficult problems to solve are not technical, they are political. *(Illustration © Bettman/CORBIS)*