Some techniques for accuracy improvement in analog computation

by ROBERT M. HOWE
University of Michigan
Ann Arbor, Michigan

ROBERT M. HOWE was born in Oberlin, Ohio in 1925. He received his BS in Electrical Engineering from California Institute of Technology in 1945, his AB in Physics from Oberlin College in '47, the MS in Physics from the University of Michigan in '48, and his PhD from Massachusetts Institute of Technology in '50.

Since 1950 he has been at the University of Michigan, where he is Professor of Aerospace and Astronautical Engineering and serves as chairman of the Information and Control Engineering Program. His technical areas of teaching and research include design and application of analog computers, automatic control, the flight simulation. He has numerous publications in these areas, including a book on analog computers.

He was one of the founders of Applied Dynamics, Inc., and currently serves as board chairman. He was also a "founder" and first chairman of the Midwestern Simulation Council, one of our largest and most active regional Councils.

Dr. Howe is a member of Tau Beta Pi, Phi Beta Kappa, Sigma Xi, and Simulation Councils Inc.

1. INTRODUCTION

Although analog computers have many attributes which make them well suited to the solution of differential equations, it is clear that one of the disadvantages of analog computers is their limited computing accuracy, particularly in the nonlinear area. For many types of problems this accuracy limitation presents no difficulty, particularly if the computer mechanization is suitably scaled. There are certain critical problems, however, where the accuracy limitation does make the application of analog computers marginal, if acceptable at all. The purpose of this paper is to illustrate some of the techniques markedly increasing the computing accuracy of the nonlinear analog components, as well as an example technique of rescaling for improved computational accuracy.

2. METHODS OF IMPROVING NONLINEAR COMPUTATIONAL ACCURACY

2.1 *Accuracy improvement of squarers*

Computation of an output voltage proportional to the square of an input voltage x is of interest in analog computation not only because we are often required to generate quadratic functions in solving problems, but also because two such square-law devices can be combined using the quarter-square principle to produce the product of two voltages. Square-law function generation is implemented in most state-of-the-art analog computers by approximating the parabolic function with a series of straight-line approximations using biased diode circuits. Normally these circuits are combined into a single circuit element with input voltages $\pm x$ which, when terminated into the summing junction of an operational amplifier as shown in figure 2.1, produces an output voltage proportional to $-x^2$. One can show that ideally the fractional error ε_{x^2} in the approximation is given by

$$\varepsilon_{x^2} = \frac{1}{8_n{}^2}, -1 \leq x \leq 1 \qquad (2.1.1)$$

where n is the number of segments used in the approximation for $0 \leq x \leq 1$. In practice the rounding of the segment corners caused by the diode characteristics can reduce the actual error obtained to a value considerably lower than the formula shows. On the other hand, in practice the error is increased by inability to manufacture or adjust the square-law circuit perfectly.

Typical static accuracies of such square-law function generators range between 0.01% and 1% of computer reference voltage. There may be times when the accuracy of a square-law function generator in a given computer is not sufficient for satisfactory problem-solution accuracy. Let us now consider a circuit which allows us to extend this accuracy considerably. First of all, the circuit is based on a highly accurate mechanization for taking the absolute value of x using amplifiers 1 and 2 in figure 2.2. In this well-known circuit diode D_1 conducts when $x > 0$, since the output of amplifier 1 is negative. Thus e_1 represents the output of a unity gain inverter for $x > 0$. On the other hand, for $x < 0$ the amplifier output goes positive, diode D_2 conducts, and diode D_1 is back biased. Therefore $e_1 = 0$ (assuming an ideal amplifier with zero summing junction offset). The result is a highly-accurate half-wave rectification, i.e.,

$$\begin{aligned} e_1 &= -x, & x > 0 \\ e_1 &= 0, & x < 0 \end{aligned} \qquad (2.1.2)$$

In amplifier 2 the voltage $2e_1$ is summed with the voltage x, producing an output voltage given by $|x|$. Assuming very high open-loop gain for the amplifiers, negligible amplifier offsets, and very large diode back-resistance (all three excellent assumptions for precision analog computers), the circuit accuracy depends on the computing-resistor accuracy and can easily exceed 0.01% of full-scale reference. The only potential problem is excessive output noise when $x = 0$ volts.

Having obtained a precision absolute value function, we can proceed to the implementation of the high-accuracy squaring circuit shown in the remainder of figure 2.2. Amplifiers 3 and 4 are used to compute the voltages $\pm(2|x|-1)$, which in turn are applied as inputs to the squaring circuit terminated in amplifier 5. Also summed into amplifier 5 are the reference voltage -1 (we assume that full-scale reference voltage is considered unity for scaling purposes) along with the voltage $|x|$. It is then easy to show that the output of amplifier 5 is, ideally, equal to $-x^2$. In fact, we are implementing the following equality:

$$x^2 = \frac{1}{4}(2|x|-1)^2 - \frac{1}{4} + |x|. \qquad (2.1.3)$$

Note that the net gain in the squaring circuit is 0.25, *i.e.,* any errors ε_{x^2} in that circuit are reduced by a factor of 4

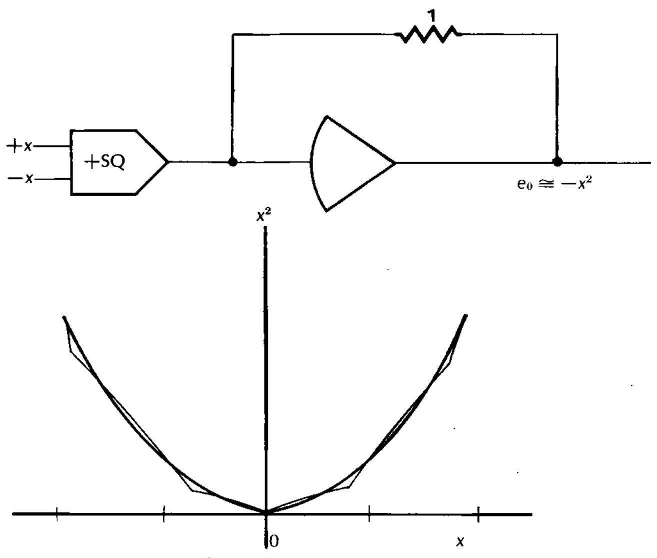


Figure 2.1 – Mechanization of quadratic function generation using a segmented straight-line approximation

in the final output of amplifier 5. Figure 2.2 shows graphically that to synthesize x^2, we are adding the absolute value of a quadratic function of x, a linear function of x, and a constant. The net result is an overall error in computing x^2 which is a factor of 4 less than the square-law function generator provides if used separately. Note, however, that this accuracy is realized only if the linear component inaccuracies can be neglected. In practice calibration procedures can be used to insure this.

It is interesting to observe that the square-law function generator in figure 2.2, which presumably would consist of an n-segment diode function generator, can be itself replaced by another circuit identical with figure 2.2. The net result would be an ideal error reduction by a factor of 16 over the n-segment diode function generator by itself. In fact, the process can be repeated indefinitely, in principle, each time picking up an ideal accuracy improvement of a factor of 4. Careful calibration techniques for the linear elements have allowed the writer to implement analog x^2 function generation to 10 parts per million starting with a 100 ppm 37-segment square-law diode function generator and applying the circuit of figure 2.2 twice. The resulting x^2 function has the equivalent of 148 segments with a peak ripple error of less than 5 ppm (0.0005%) due to segmentation.

This general technique for square-law accuracy improvement has been known for some time[1] and is even the basis for physical implementation for at least one commercial quarter-square multiplier design. In the context of this paper its main attractiveness lies in being able to generate highly-accurate square-law functions using a device of more modest accuracy on a standard computer. Although at least 5 operational amplifiers are required instead of the usual 2, the added complexity may be warranted if a particular problem needs a very accurate quadratic function. Actually, the writer was motivated to utilize this circuit in order to have a precision square-law reference function generator for the purpose of calibrating quarter-square multipliers.

2.2 *Accuracy improvement of multipliers*

Two of the circuits described in the previous section can be combined to mechanize an improved-accuracy multiplier. The usual quarter-square multiplier is based on the identity

$$\left|\frac{x+y}{2}\right|^2 - \left|\frac{x-y}{2}\right|^2 = xy \qquad (2.2.1)$$

The circuit shown in figure 2.3 is based on the following identity:

$$\frac{1}{4}(1-|x+y|)^2 - \frac{1}{4}(1-|x-y|)^2 + \frac{|x+y|}{2} - \frac{|x-y|}{2} = xy \qquad (2.2.2)$$

Again note that the gain of each of the two squaring units terminated in amplifier 9 is 0.25 so that the errors which these function generators would normally provide in implementing a product xy are reduced by a factor of 4. Thus the output product, $-xy$, is 4 times more accurate, assuming that linear computing errors can be calibrated out. As described in the previous section, one can substitute the circuit of figure 2.2 for each of the squaring units in figure 2.3 to obtain, ideally, an additional factor of 4 reduction in error. Starting with individual square-law diode function generators accurate to, say, ±150 ppm, with careful calibration of the linear components it is possible to construct a multiplier with a static accuracy of about 20 ppm or 0.002%. Not only could this be useful for providing a single critical product in certain problems, but also it could be extremely useful as a reference multiplier against which any given multiplier could be compared in order to produce an error map for all inputs x and y.

2.3 *Accuracy improvement of dividers*

It is well known that the analog operation of division can be performed using a multiplier in the feedback loop of a high-gain amplifier, as shown in figure 2.4. It is also easy to show that if ε_m is the maximum multiplier error, then the error ε_Q in the quotient is given by

$$\varepsilon_Q = \frac{\varepsilon_m}{y} \qquad (2.3.1)$$

where $0 < y \leq 1$ (reference voltage equals unity). Thus the smaller the divisor, the larger will be the error.

One can immediately provide an ideal accuracy increase of a factor of two by dividing by a new variable $\bar{y}$ given by

$$\bar{y} = 2y - 1 \qquad (2.3.2)$$

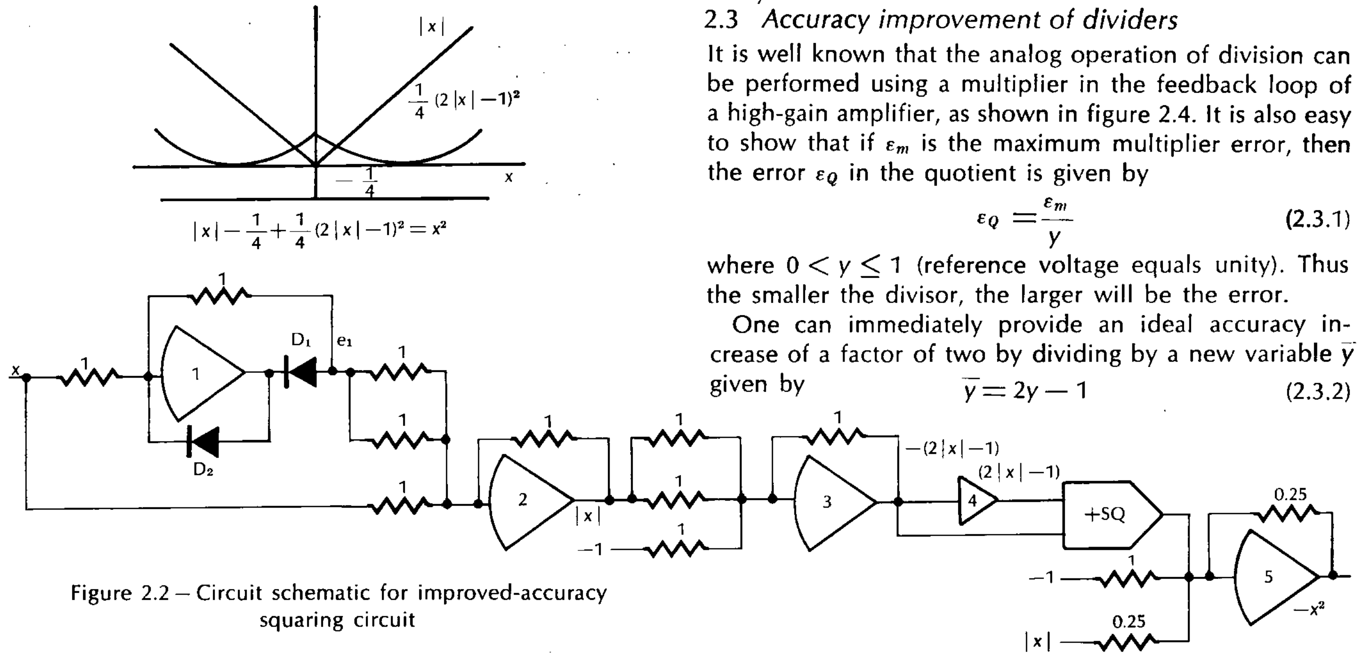


Figure 2.2 — Circuit schematic for improved-accuracy squaring circuit

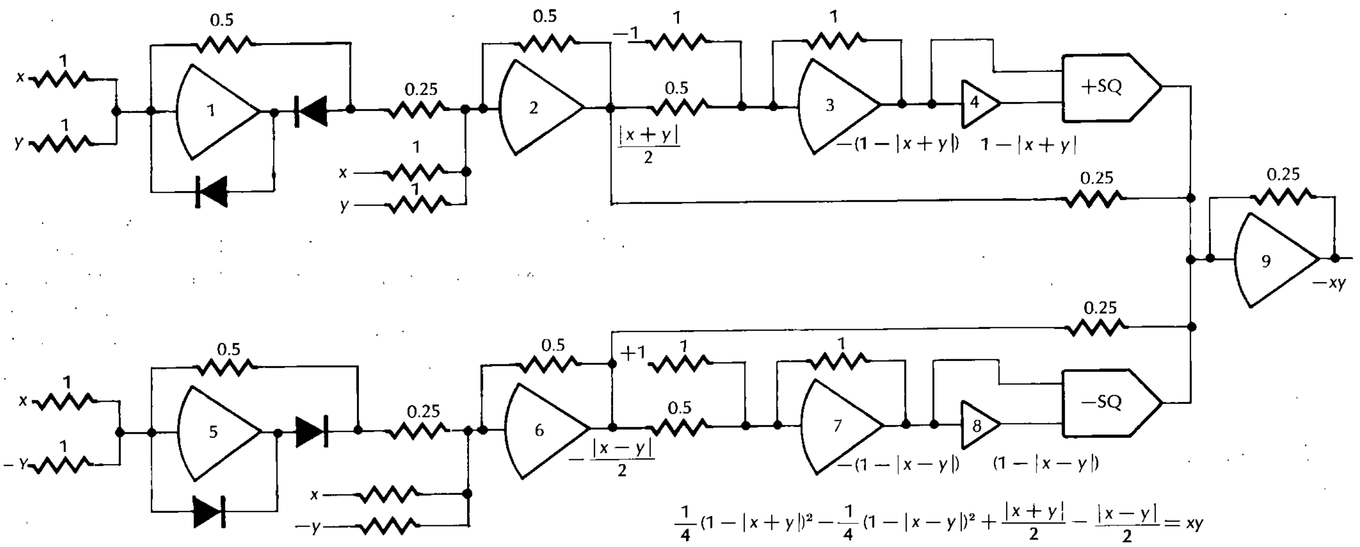


Figure 2.3 — Circuit schematic for improving multiplier accuracy

Clearly for $0 < y \leq 1$, $-1 < \bar{y} \leq 1$, *i.e.*, $\bar{y}$ ranges over full $\pm$ reference rather than just $+$ reference. Computing the quotient $Q = x/y$ can now be written as

$$Q = \frac{x}{y} = \frac{2x}{1+\bar{y}}, \text{ or } Q = 2x - Q\bar{y} \qquad (2.3.3)$$

Circuit implementation is shown in figure 2.5. Note that amplifier 1 has a unity feedback resistor and that the multiplier is used over all four quadrants for $-1 \leq x \leq 1$. It is easy to show that for a maximum multiplier error ε_m, the error ε_Q in the quotient is equal to $\varepsilon_m/2y$, *i.e.*, half the error of the previous circuit. Actually, the circuit is easy to understand when one realizes that for $y = 1$, $\bar{y} = 2y - 1 = 1$, and the multiplier in the feedback loop acts as a unity feedback resistor. This, in parallel with the already existing unity feedback, yields an effective feedback resistor of 0.5, or overall unity gain for amplifier 1. This is indeed correct for $y = 1$ in the quotient $Q = x/y$. For $y = 0.5$, $2y - 1 = 0$, the multiplier provides no feedback around amplifier 1, which therefore has a net operational gain of 2, as required for $y = 0.5$. Finally, for $y = 0$, the multiplier acts like a feedback resistor of -1 around amplifier 1. The net result is infinite operational gain, as required for $y = 0$.

It is apparent that two additional operational amplifiers, numbers 3 and 4 in figure 2.5, are required to compute $\pm(2y - 1)$ from y. In many cases $2y - 1$ can be computed directly, *e.g.*, by integrating $\dot{y}$ to produce $2y - 1$ rather than y. In this case it may be possible to save the two extra amplifiers.

In many problems the divisor does not range down to zero. For example, assume that we wish to compute the quotient $Q = x/y$ where $(1 - \alpha) \leq y \leq (1 + \alpha)$ and where $\Delta y = y - 1$. Good scaling practice indicates that we should compute $\Delta y/\alpha$ rather than y, so that $-1 \leq \Delta y/\alpha \leq 1$. Assuming this is the case, the circuit in figure 2.6 shows the mechanization of the dividing operation. It is easy to show that for a given multiplier accuracy this circuit produces an error which is 0.5α times the error of the conventional divide circuit in figure 2.4.

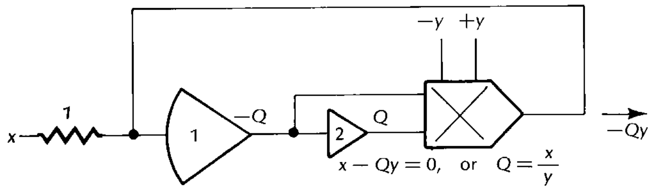


Figure 2.4 – Conventional divide circuit for $y > 0$

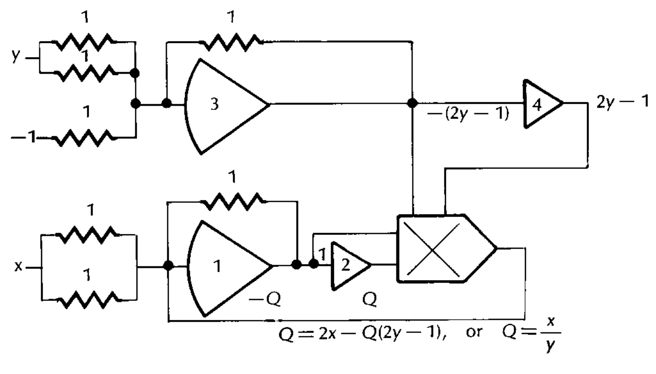
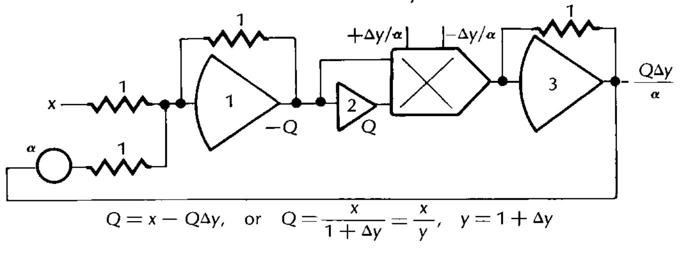


Figure 2.5 – Divide circuit to provide a factor-of-two increase in accuracy



1
$+\Delta y/\alpha$
$-\Delta y/\alpha$
1
x
1
1
$-Q$
2
Q
3
$\frac{Q\Delta y}{\alpha}$
α
1
$Q = x - Q\Delta y$, or $Q = \frac{x}{1+\Delta y} = \frac{x}{y}$, $y = 1 + \Delta y$

Figure 2.6 – General divide circuit for accuracy improvement

2.4. *Mechanization of a high-accuracy sine function*

The generation of precision sine and cosine functions in general-purpose analog computers is usually accomplished with special-purpose fixed-diode function generators. The technique described in this section requires only squaring and multiplying circuits. It is useful when no sine-cosine diode function generators are available or when a high-precision master sine-cosine generator is needed for calibration purposes. The first approximation to the sine function is obtained using a Taylor series expansion. Thus

$$f(x) = f(x_0) + f'(x_0)\Delta x + \frac{f''(x_0)}{2!}(\Delta x)^2 + \ldots \qquad (2.4.1)$$

where $\Delta x = x - x_0$. For $f(x) = \sin x$ and $x_0 = 0$, we have

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots \qquad (2.4.2)$$

Assume we wish to generate $\sin x$ over the range $-\pi/2 \leq x \leq \pi/2$, since by the use of zig-zag circuits this can be used to generate both $\sin x$ and $\cos x$ over any range. If we let a new variable $\bar{x}$ be given by

$$\bar{x} = \frac{2}{\pi} x \qquad (2.4.3)$$

then x ranges through ± 1 corresponding to $\pm\pi/2$ radians (± 90 degrees). In terms of $\bar{x}$ equation (2.4.2) becomes

$$\sin\left(\frac{\pi}{2}\bar{x}\right) = 1.570796\bar{x} - 0.645964\bar{x}^3 + 0.079693\bar{x}^5 - 0.004682\bar{x}^7 + 0.000160\bar{x}^9 + \ldots \qquad (2.4.4)$$

If we take only the first four terms in the series, then for $\bar{x} = \pm 1(\pm 90$ degrees) the error will be a maximum and will equal 0.016 percent. However, we can represent the $\bar{x}^9$ term which has been neglected by using a Chebyshev polynomial $T_9(x)$ given by[2]

$$T_9(\bar{x}) = 256\bar{x}^9 - 576\bar{x}^7 + 432\bar{x}^5 - 120\bar{x}^3 + 9\bar{x} \qquad (2.4.5)$$

Chebyshev polynomials of this type have the characteristic that over the interval $-1 \leq \bar{x} \leq +1$ they oscillate back and forth across $T_n = 0$ a maximum number of times for the order of the polynomial, and with a maximum value of ± 1 at each oscillation peak. Thus if we solve equation (2.4.5) for x^9 to obtain

$$\bar{x}^9 = \frac{1}{256}[576\bar{x}^7 - 432\bar{x}^5 + 120\bar{x}^3 - 9\bar{x} + T_9(\bar{x})] \qquad (2.4.6)$$

and neglect $T_9(\bar{x})$, the maximum error will be 1/256. Hence replacing $\bar{x}^9$ in equation (2.4.4) by equation (2.4.6) with $T_9 = 0$ will result in a representation for the sine with terms only up to order $\bar{x}^7$ but with a maximum error of $160 \times 10^{-6}/256$, or 0.6 ppm, plus the error due to the neglect of the $\bar{x}^{11}$ term. In this case the series representation for the sine function becomes

$$\sin\left(\frac{\pi}{2}\bar{x}\right) \cong 1.570790\bar{x} - 0.645889\bar{x}^3 + 0.079423\bar{x}^5 - 0.004322\bar{x}^7 \qquad (2.4.7)$$

where the maximum error over ± 90 degrees is approximately 2 ppm (0.0002 per cent).

By applying the same method to equation (2.4.7) we can eliminate the $\bar{x}^7$ term. Since the Chebyshev polynomial $T_7(x) = 64x^7 - 112x^5 + 56x^3 - 7x$, we can approximate $\bar{x}^7$ with

$$\bar{x}^7 \cong \frac{1}{64}\left[112\bar{x}^5 - 56\bar{x}^3 + 7\bar{x}\right] \qquad (2.4.8)$$

with a maximum error of 1/64 over the interval $-1 \leq \bar{x} \leq 1$. Substituting equation (2.4.8) into equation (2.4.7), we obtain

$$\sin\left(\frac{\pi}{2}\bar{x}\right) \cong 1.570317\bar{x} - 0.642107\bar{x}^3 + 0.071859\bar{x}^5 \qquad (2.4.9)$$

with a maximum error of approximately $4322/64 = 68$ ppm. Figure 2.7 shows a plot of the actual error in equation (2.4.9) in representing the true sine function over ± 90 degrees.

In mechanizing equation (2.4.9) with analog circuitry it should be noted that any errors in computing $\bar{x}^3$ are multiplied by 0.642. Thus a squaring error of 0.02 percent in computing $\bar{x}^2$ along with an additional multiplier error of 0.03 percent in computing $\bar{x}^3$ leads to a total error of 0.05 percent in addition to the approximation error of 0.007 percent in using equation (2.4.9). By using the techniques of sections 2.1 and 2.2 the component errors can be reduced arbitrarily except for linear calibration errors.

Next consider a Taylor series expansion of $\sin x$ about $x = \pi/4$ (*i.e.*, 45 degrees). Application of equation (2.4.1) gives

$$\sin x = \frac{1}{\sqrt{2}}\left[1 + \Delta x - \frac{(\Delta x)^2}{2} - \frac{(\Delta x)^3}{6} + \frac{(\Delta x)^4}{24} + \dots\right] \qquad (2.4.10)$$

If we let a new variable $\bar{x}$ be given by

$$\bar{x} = \frac{4}{\pi}\Delta x = \frac{4}{\pi}\left(x - \frac{\pi}{4}\right) \qquad (2.4.11)$$

then as x ranges over $0 \leq x \leq \pi/2$, $\bar{x}$ ranges over $-1 \leq \bar{x} \leq 1$. In terms of $\bar{x}$, equation (2.4.10) becomes

$$\sin x = 0.7071068\,[1 + 0.7853982\bar{x} - 0.3084252\bar{x}^2 - 0.0807457\bar{x}^3 + 0.0158543\bar{x}^4 + 0.002490\bar{x}^5 - 0.0003260\bar{x}^6 - 0.0000366\bar{x}^7 - \dots] \qquad (2.4.12)$$

Let us use equation (2.4.8) to replace the $\bar{x}^7$ term and the following equation based on the Chebyshev polynomial T_6 to replace $\bar{x}^6$ in equation (2.4.12).

$$\bar{x}_6 \cong \frac{1}{32}\left|1 - 18\bar{x}^2 + 48\bar{x}^4\right| \qquad (2.4.13)$$

Equation (2.4.12) then becomes

$$\sin x \cong 0.7071068\,[0.9999898 + 0.7853942\bar{x} - 0.3082419\bar{x}^2 + 0.0153653\bar{x}^4 - 0.0807137\bar{x}(\bar{x}^2 - 0.0300616\bar{x}^4)] \qquad (2.4.14)$$

If $\pm\bar{x}$ is available as an input voltage, mechanization of equation (2.4.14) takes 7 operational amplifiers, one dual squarer, and one multiplier. Figure 2.8 shows a plot of the actual error in the sine approximation in equation (2.4.14). Note that the maximum error is approximately 8 ppm. To this must be added the analog error in implementing the circuit. The predominant nonlinear component error contribution lies in the $\bar{x}^2$ term, which has a net coefficient of

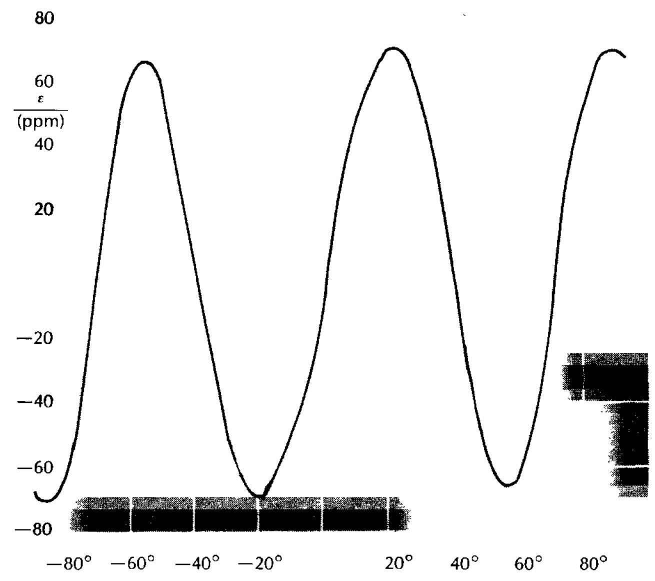


Figure 2.7 – Error versus angle in sine approximation of equation (2.4.9)

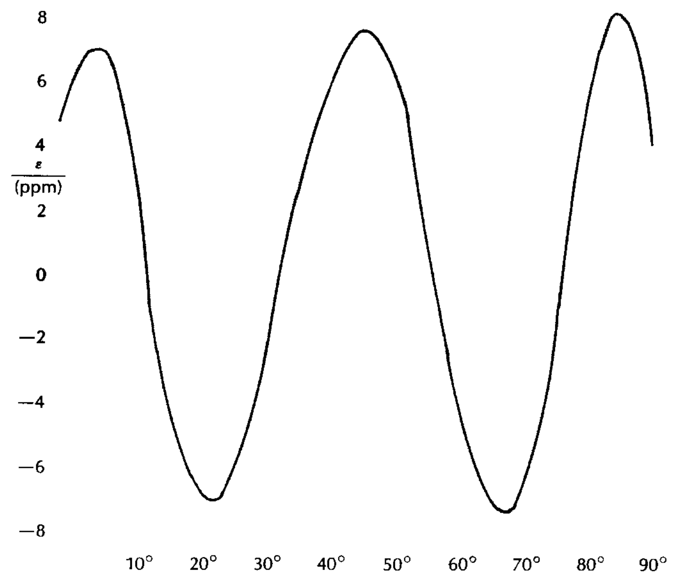


Figure 2.8 – Error versus angle in sine approximation of equation (2.4.14)

0.218. Thus any errors in generating $\bar{x}^2$ are reduced by a factor of 4.6 in the sin x computation. Using this mechanization the author has computed precision sin x voltages to the order of 20 ppm accuracy (0.002%). By appropriate biasing and sign reversals the same circuit can be used to generate both sin x and cos x over all four quadrants with 90° range in each quadrant. Such a mechanization has proven extremely useful in calibrating and checking general-purpose sine-cosine diode function generators. Because of the circuit complexity it is of questionable value as a means for computing sin x in problem solutions.

2.5 *Improvement of function-generator accuracy*

In the previous section we saw how the power series expansion method could be used to compute an accurate approximation to a sine function. Had it been worthwhile we could have used a general purpose diode function generator to compute the remaining error function $\epsilon(\bar{x})$. This forms the basis for the technique proposed in this section, namely, to approximate a desired function $f(x)$ using the first several terms of a power series $\hat{f}$ and using the diode function generator to compute the remainder of the function. For example, consider the function shown in figure 2.9. This function is broken into a constant f_0, a linear term f_1x, a quadratic term f_2x^2, and a remainder $\Delta f(x)$. Only the small remainder function need be set up on the variable diode function generator, and the scaling is such that perhaps $20\,\Delta f(x)$ could be set on the diode function generator and then multiplied by 0.05 when summed with $\hat{f}(x)$. The constant and linear terms in $\hat{f}$ can be computed with high accuracy and stability, and even the x^2 term can be computed with a precision square-law fixed diode function generator (typically one-half of a quarter-square multiplier) with far more segments and much more drift stability than a variable diode function generator would have.

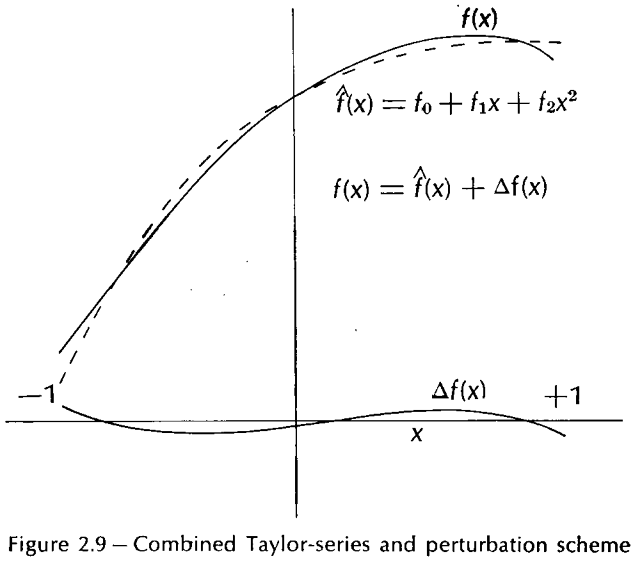


Figure 2.9 — Combined Taylor-series and perturbation scheme for accurate function generation

Although this technique requires more components than a direct setup of $\hat{f}(x)$ on a diode function generator, in critical applications the increased number of segments affected by the quadratic term and the greatly-increased stability might very well make the technique attractive.

Obviously any number of additional terms could be added to the power series $\hat{f}(x)$. If only two terms are present, f_0 and f_1x, these correspond essentially to the parallax and central-slope term which are already a part of any variable diode function generator. Whatever the range of the input variable for the function, accuracy can always be gained by transforming it to a variable x ranging over ± 1, as shown in figure 2.9.

It should be noted that if the function $f(x)$ has higher derivatives which are quite large, *i.e.*, is very badly behaved, the power series approximation may gain little in comparison with the added equipment requirements.

3. RESCALING FOR IMPROVED COMPUTATIONAL ACCURACY

We have seen in Section 2.3 how the multiplier used in the conventional analog divide circuit is, at best, utilized in only two of its four quadrants. By rescaling the divisor voltage, as shown in figure 2.5 or 2.6, the multiplier can be used over all four quadrants with a corresponding improvement in accuracy.

This same technique can be extended to nonlinear operations in general. It is reasonable to assume that proper analog scaling techniques have resulted in voltage excursions for each problem variable which extend either to full positive or full negative reference voltage. What we are proposing here is to bias each variable as necessary to have its voltage excursion extend over *both* the plus and minus reference range. Although this may add some complexity, the accuracy improvement in the case of nonlinear computations may make it worthwhile.

For example, consider the solution of a set of flight vehicle equations where one of the problem variables is the total vehicle velocity V_p, which is computed by integrating $\dot{V}_p$. Suppose that V_p varies from 0 to some V_{max}. Then we can define a dimensionless velocity Δv_p given by

$$\Delta v_p = \frac{2V_p}{V_{max}} - 1 \qquad (3.1)$$

where the range in v_p is $-1 \leq v_p \leq 1$. This effectively doubles the voltage range for the velocity (the fact that we chose a dimensionless variable merely simplifies the scaling). By appropriately scaling the input to the $\dot{V}_p$ integrator and starting with the proper initial condition, one can compute Δv_p directly without requiring an additional summing amplifier.

Having obtained Δv_p in this way, one can use the circuit of figure 2.5 to implement division by V_p. The computation of V_p^2, as needed, for example, in the calculation of dynamic pressure, can be implemented using for formula:

$$v_p^2 = \frac{V_p^2}{V_{max}^2} = (1 + \Delta v_p)^2 = 1 + 2\Delta v_p + \Delta v_p^2 \qquad (3.2)$$

The equation is mechanized using the single operational amplifier shown in figure 3.1.

If the velocity squared, as given in equation (3.2), is to be multiplied by a second variable, such as density ρ, then additional improvement in scaling may be obtained by defining a dimensionless density variation $\Delta\rho$ given by

$$\Delta\rho = \frac{2\rho}{\rho_{max}} - 1 \qquad (3.3)$$

where $-1 \leq \Delta\rho \leq 1$. The dynamic pressure q is now given by the formula

$$q = \frac{1}{2}\left[\frac{\rho_{\max}}{2}(1 + \Delta\rho)V_{\max}^2 v_p^2\right] \qquad (3.4)$$

where the circuit of figure 3.1 can be used to compute v_p^2. But this must be multiplied by $\Delta\rho$ and, since $0 < v_p^2 \leq 1$, we are only using two quadrants of the multiplier. The answer, of course, is to compute $2v_p^2 - 1$ directly from Δv_p^2 using the formula

$$2v_p^2 - 1 = 1 + 4\Delta v_p + 2\Delta v_p^2 \qquad (3.5)$$

From equations (3.4) and (3.5) the equation for q becomes

$$q = \frac{\rho_{\max} V_{\max}^2}{8}\left[(2v_p^2 - 1) + \Delta\rho + \Delta\rho(2v_p^2 - 1)\right] \qquad (3.6)$$

The overall mechanization of equation (3.6), starting with V_p and assuming ρ is a function of altitude h, is shown in figure 3.2b. Note that it requires no more amplifiers than the straightforward mechanization shown in figure 3.2a, yet the effect of multiplier errors is reduced by a factor of four and of square-law diode function generator errors by a factor of two.

If the velocity V_p or density ρ range from full scale positive to a lower limit which is above zero, then even further scaling gains can be made, similar to the method illustrated in figure 2.6. Whether in fact the overall computing accuracy is really increased by a factor of four in figure 3.2b compared with figure 3.2a depends on whether or not the multiplier errors predominate and also whether nonlinear computation errors tend to be as large with low-level inputs as with high-level inputs. In typical general-purpose analog systems, however, techniques of the type illustrated here and in section 2.3 can offer marked accuracy improvement. This is particularly true when the variables involved in nonlinear operations have a range which is restricted to a small fraction of their maximum value.

It should be emphasized that the above methods are really not perturbation methods in the sense that approximations are made in the equations by dropping high-order terms. In each case the equations are exact in every way.

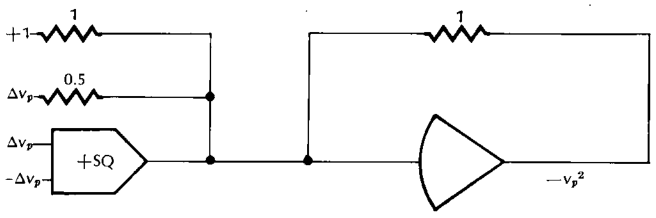
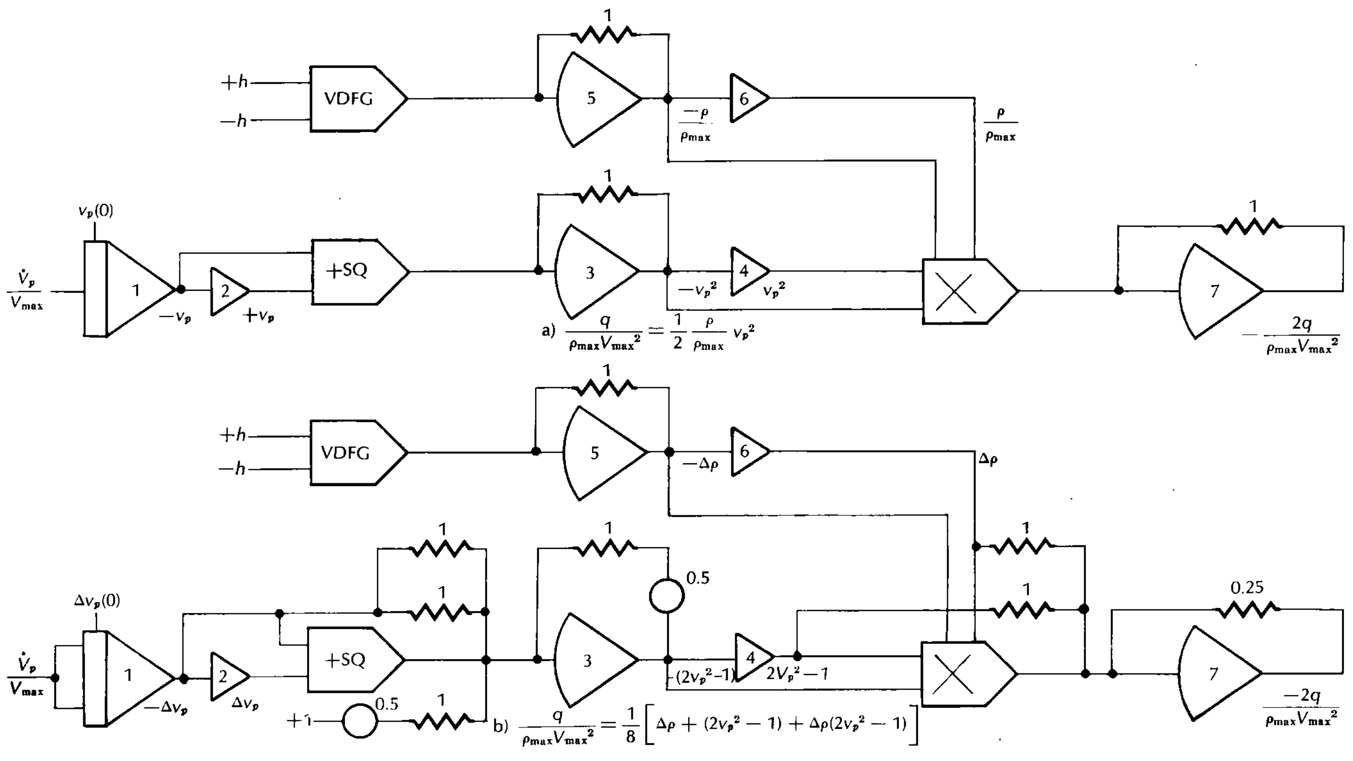


Figure 3.1 — Circuit for computing v_p^2 from Δv_p, where $\Delta v_p = v_p - 1$

REFERENCES

1 A NATHAN
Interpolating multipliers and related interpolators
IEEE Proceedings vol 51 no 11 pp 1549-1554 November 1963

2 M ABRAMOWITZ I A STEGUN (editors)
Handbook of mathematical functions
National Bureau of Standards Applied Mathematics series 55 p 795 June 1964



+h
−h
VDFG
5
6
$-\frac{\rho}{\rho_{\max}}$
$\frac{\rho}{\rho_{\max}}$
$v_p(0)$
$\frac{\dot V_p}{V_{\max}}$
1
2
$-v_p$
$+v_p$
+SQ
3
4
$-v_p^2$
v_p^2
7
$-\frac{2q}{\rho_{\max}V_{\max}^2}$
a) $\frac{q}{\rho_{\max}V_{\max}^2} = \frac{1}{2}\frac{\rho}{\rho_{\max}}v_p^2$
$-\Delta\rho$
$\Delta\rho$
$\Delta v_p(0)$
$-\Delta v_p$
Δv_p
0.5
+1
0.25
$(2v_p^2-1)$
$2V_p^2-1$
$-\frac{2q}{\rho_{\max}V_{\max}^2}$
b) $\frac{q}{\rho_{\max}V_{\max}^2} = \frac{1}{8}\left[\Delta\rho + (2v_p^2-1) + \Delta\rho(2v_p^2-1)\right]$

Figure 3.2 — Two alternative circuits for computing dynamic pressure

DANIEL TEICHROEW is Professor and Head of the Division of Organizational Sciences at Case Institute of Technology. He was formerly Professor of Management in the Graduate School of Business at Stanford University.

A native of Canada, Dr. Teichroew received his BA and MA degrees in Mathematics from the University of Toronto and his PhD degree in Statistics from the Institute of Statistics, North Carolina, in 1953. He served as an instructor and research associate at the Institute of Statistics, a statistician with the Institute of Numerical Analysis, National Bureau of Standards, and as Head of a Business Systems Analysis Group for the National Cash Register Company.

Dr. Teichroew is author of *Introduction to management science,* published in 1964 by John Wiley, the co-author (with James Howell) of *Mathematical analysis for business decisions,* published in 1963 by Richard D. Irwin, co-author (with Michael Connors) of a forthcoming book entitled *Optimal control of dynamic operations research models,* and the author of numerous papers in professional journals.

He is a Fellow of the American Statistical Association, a Vice-President of the Institute of Management Science, and the Editor of the Business Applications Section of the *Communications of the ACM.* He is also a member of the Operations Research Society of America, Operational Research Society, Ltd., the Association for Computing Machinery, the American Mathematical Society, and the International Institute for Statistics in the Physical Sciences.

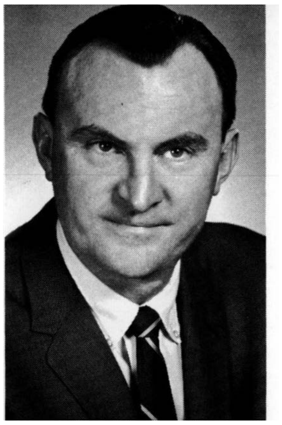
JOHN FRANCIS LUBIN is Director for Computing Activities at the University of Pennsylvania (since 1964) and Associate Professor of Industry in the Wharton School of Finance and Commerce. He obtained a Bachelor of Electrical Engineering degree from the College of Engineering of New York University in 1947, his MS from the Massachusetts Institute of Technology in 1949, and his PhD at the Graduate School of Arts and Science of the University of Pennsylvania in 1956. He is currently Editor-In-Chief of *Computing Reviews,* the review journal of the Association for Computing Machinery.

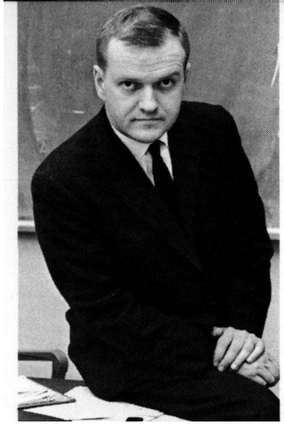
THOMAS D. TRUITT'S background includes a mix of practical and academic interests: airborne electronics equipment, electronic flight simulators; teaching of college mathematics; BS in math; analog computer programming; MSE from Princeton University; teaching electrical engineering subjects at the University of Pennsylvania; digital computer programming; hybrid computer design; and co-authoring of two books on computer programming.

Mr. Truitt joined EAI in 1956. Since then he has performed and directed studies of digital and analog computer organization and programming techniques including automatic programming, parallel digital computer structures, incremental computing techniques, special-purpose digital computer organization, and combined analog-digital computer systems. He is responsible for initiating the development of the EAI HYDAC analog/digital computers, as well as for the software systems for EAI hybrid computers.

As Director of the Advanced Development and Programming Department of the EAI Digital Computer Division, at Princeton, New Jersey, Mr. Truitt is responsible for Systems-Programming for EAI computers and for development of special software and hardware systems.

He is a member of ACM, IEEE, and SCi.


Teichroew


Truitt

Editor's note

The following article is based on a paper by Teichroew and Lubin which was published in the October 1966 issue of the *Communications of the ACM.* While it deals mainly with the comparison of discrete-change simulation languages, it does define the difference between continuous-change and discrete-change languages and lists a number of continuous-change languages as references. Its coverage of discrete-change languages is excellent, although parts of the article suffer from having been written in 1964.

Thomas Truitt of EAI pointed out in a letter to John McLeod that the Teichroew-Lubin comparisons would be more complete if they included the specifications for HSL (HYTRAN Simulation Language), the EAI 8400 implementation of CSSL (Continuous System Simulation Language), which was developed by the Simulation Software Committee of Simulation Councils, Inc. Our feeling is that, in addition to rounding out the comparison tables, the inclusion of HSL draws parallels between discrete-change and continuous-change simulation languages that illuminate their differences and similarities much more strongly than has been done before.

This article is composed of pertinent sections of the Teichroew-Lubin paper with Truitt's HSL specifications added to the original comparison tables. Interested readers should refer to the full ACM paper for much elaboration which we have omitted.

If, as Truitt states in his letter, the basic difference between discrete-change and continuous-change simulation languages lies in the recognition by the former of individual items flowing through a system and the treatment by the latter of material as aggregates of homogeneous items (neglecting, of course, differences that are mainly of taste and not substance), the two simulation schools are not as far apart in their techniques as some seem to think.

Three excellent discrete-change simulation language comparisons have been made so far—one by Krasnow and Merikallio in 1963,[2] one by Tocher in 1965[3] and the quoted one by Teichroew and Lubin in 1966. Two continuous-change language surveys have been published, one by Brennan and Linebarger in 1964[4] and Clancy and Fineberg in 1965.[5] We reproduce the following survey because of its currency and the Thomas Truitt contributions, which allow comparison of discrete-change and continuous-change language concepts.

PJK