Teaching simulation with a ‘‘digital’’ analog computer

Gary E. J. Bold and S. M. Tan

Citation: American Journal of Physics **53**, 437 (1985); doi: 10.1119/1.14195
View online: https://doi.org/10.1119/1.14195
View Table of Contents: https://aapt.scitation.org/toc/ajp/53/5
Published by the American Association of Physics Teachers

Teaching simulation with a "digital" analog computer

Gary E. J. Bold and S. M. Tan
Department of Physics, University of Auckland, Auckland, New Zealand

(Received 7 November 1983; accepted for publication 29 May 1984)

A hybrid approach to system simulation is proposed in which a digital computer is used to simulate an analog computer, which is then programmed by the user. This retains the simplicity of the analog computer approach but removes the limitations imposed by analog electronics. It is shown how a "virtual" analog computer can be simulated in BASIC using Runge–Kutta integration procedures. Quite complicated systems involving coupled, nonlinear differential equations, which would be intractable on a "real" analog computer, can be readily simulated. Several such simulations are illustrated and discussed.

I. INTRODUCTION

Digital system simulation has become a valuable tool in virtually every branch of physics and engineering, and indeed the demand for faster and cheaper simulation hardware is one of the driving forces behind the development of future mainframes and array processors (Refs. 1 and 2). It is clearly valuable to expose students to the principles involved as early as possible. But digital simulation poses philosophical problems for students, as one cannot exactly represent a differential equation by a corresponding difference equation, and a variety of different numerical algorithms which appear to be quite dissimilar may be used to approximately simulate the same continuous system.

Hence analog computers are often used in teaching laboratories to demonstrate the principles of simulation. It is hypothesized that students will gain a clearer picture of the relationship between physical systems and the differential equations describing them if they can set up the simulations themselves, see the resulting system states evolve as a function of time, and investigate the effect of varying the system parameters.

However although most students rapidly grasp the principle of an analog computer (i.e., we just construct a "working model" of a system), they often become exasperated coping with the electronic idiosyncracies of the real hardware they are expected to use. They find that considerable time is necessary to correctly program a given simulation, calculate scaling parameters, adjust drifts, and set initial conditions. Even then the simulation may not work entirely as expected because of practical limitations (nonideal operational amplifiers, etc.) and students may be left with essentially negative feelings, and a conviction that system simulation is somehow very "difficult," and at best tedious and uninteresting.

The authors feel that this is because the "electronics" has come between the student and the "physics" of the simulation. Additionally, many students and faculty are aware that virtually all realistic physical simulations are now performed digitally, and that despite the essential simplicity of analog computers, they are no longer of any practical relevance.

In this paper we advocate a hybrid approach in which we simulate not the system under study, but a "virtual" analog computer, which is then programmed by the student. This is done in software by defining relations between system variables. The system approximation problem need not even be considered by the student (as this is the "task" of the virtual machine) and it is unnecessary to learn a dedicated simulation language. Programming the simulator becomes as simple as programming an analog computer, except that limitations due to analog hardware are entirely removed. Also, since addition, multiplication, and division of constants and variables become simple arithmetic statements in high-level computer language, much more complex and physically interesting simulations are possible than would normally be considered appropriate for "teaching grade" analog computers.

The central problem in implementing the virtual machine is the provision of suitable integrating procedures. The next section describes how this can be done, and shows how the simulation can be structured in a logical manner.

II. INTEGRATOR SIMULATION

We have used Runge–Kutta algorithms to simulate the virtual integrators because these algorithms are easy to program, have nice "self-starting" properties, and are stable for a wide range of physically interesting problems. The second-order algorithm is usually acceptably accurate for teaching purposes, but algorithms of higher order (Refs. 3 and 4) can be substituted if required without changing the procedure calls. These algorithms compute solutions to differential equations at discrete time intervals. An nth-order algorithm generates a solution, which, after the time interval specified, is identical to the Taylor series expansion of the exact solution to order n. For systems of first-order differential equations which can be written in the vector form

$$\frac{d\mathbf{X}}{dt} = \mathbf{f}(t,\mathbf{X}),$$

the second-order algorithm is

$$\begin{aligned}\mathbf{X}_{m+1} &= \mathbf{X}_m + \tfrac{1}{2}(\mathbf{D}_1 + \mathbf{D}_2)dt,\\ \mathbf{D}_1 &= \mathbf{f}(t,\mathbf{X}),\\ \mathbf{D}_2 &= f(t + dt, \mathbf{X}_m + \mathbf{D}_1 dt),\end{aligned}$$

where t and dt denote time and the time increment desired between solutions. The use of a vector function $\mathbf{f}$ of a vector variable $\mathbf{X}$ enables the derivatives of any one component of $\mathbf{X}$ to be expressed in terms of any or all of the other components of $\mathbf{X}$, plus arbitrary functions of time. Since any nth-order differential equation can be written as a system of n first-order equations, we can cope with differential equations of arbitrary complexity by repeated integration.

The software provides an array of virtual integrators which use a specified Runge–Kutta algorithm. These are incorporated into the simulation system as shown in Fig. 1. For teaching purposes we implement the algorithm in BA-

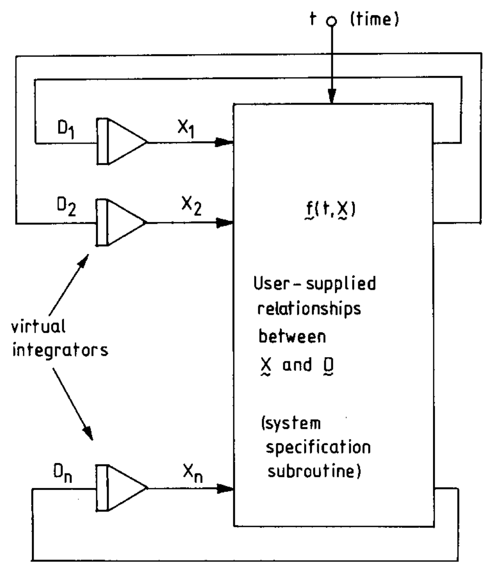
t (time)
D1
X1
D2
X2
f(t,X)
User-supplied relationships between X and D
(system specification subroutine)
virtual integrators
Dn
Xn

Fig. 1. Model of the virtual analog computer simulated by software.

SIC because of the ease with which program statements, and hence system parameters, can be changed between simulation runs. The second-order algorithm can be coded as follows:

```
500  FOR I = 1 TO NI:XO(I) = X(I):NEXT I
510  GOSUB 1000
520  FOR I = 1 TO NI:E(I) = D(I)*DT:X(I)
     = XO(I) + E(I):NEXT I
530  T = T + DT:GOSUB 1000
540  FOR I = 1 TO NI:X(I) = XO(I)
     + .5*(E(I) + D(I)*DT):NEXT I
550  RETURN.
```

Code for the fourth-order algorithm is available from the authors.

The variables used are

NI	the number of integrators required by the simulation
T,DT	time and time increment between solutions.

Four NI dimensioned arrays are used:

D(I),X(I)	the input and output, respectively, of the *i*th integrator. The system specification subroutine, at line 1000, defines each D(I) in terms of X(1) to X(NI), and time. X(I) must be loaded with the initial value of the *i*th integrator output before the simulation starts. After a step, the X(I)'s contain the new system variable states.
XO(I),E(I)	are only required for internal calculations, and can be ignored by the user. XO(I) stores the initial value of X(I) at the start of a step. E(I) is used to sum the intermediate values used to compute the final value of X(I).

The complete simulation procedure is

(1) Define the time increment and the number of virtual integrators required. Set the initial values of system variables in array X.

(2) Call the integration subroutine. This interrogates the system specification subroutine, advances the system state by one time increment, and places the resulting new values of the system variables into X.

(3) Output the new system state, as defined by the components of array X.

(4) Go to step 2.

For example, assume we wish to simulate the damped SHM equation:

$$\frac{d^2X}{dt^2} + K\frac{dX}{dt} + W{\cdot}X = 0.$$

Two integrators are required. We write

$$\frac{dX}{dt} = V,$$

$$\frac{dV}{dt} = -K{\cdot}V - W{\cdot}X.$$

The system may be specified by the flow chart of Fig. 2. This is identical to the corresponding analog computer flow chart, except that noninverting integrators are used. The system specification subroutine at line 1000 is

```
1000 D(1) = - K*X(1) - W*X(2)
1010 D(2) = X(1):RETURN.
```

X(1) and X(2) are the velocity and displacement, respectively. K, W, DT and the initial values of X(1) and X(2) must be defined prior to the first integration call.

If X(2) is plotted against T, a graph of system amplitude as a function of time results. In this simple case we can compare the simulation results with the exact analytic solution. Choosing K negative, so that a decaying oscillatory solution results, initial amplitude X(2) = 1, and initial velocity X(1) = 0, we find that typical simulations give the errors shown in the table below. Nr is the order of the Runge–Kutta algorithm used, Nc is the number of simulation steps per oscillatory cycle, and the error (the difference between exact and simulated solutions) is expressed as a percentage of the initial value of the amplitude.

Nr	Nc	Percentage error
2	40	1.8
2	80	0.47
2	2000	0.03
4	40	0.002

For this simulation, typical microcomputers running interpreted BASIC compute four to six solutions per second using the second-order algorithm, and between two to three solutions per second using a fourth-order algorithm. If a display unit having a vertical resolution of 400 units is used (typical of medium resolution microcomputer systems), then choosing 80 time steps per cycle and the second-order algorithm results in a plot which is identical with that of the exact solution. Five cycles of the waveform

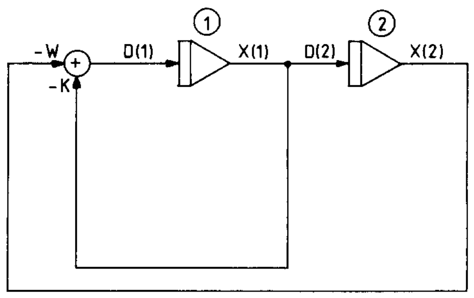

Fig. 2. Flow diagram for simulation of damped simple harmonic oscillator.

can typically be computed and displayed in about 1.5 min. Choosing 40 time steps per cycle and using a fourth-order algorithm takes about the same time, but gives substantially better accuracy. For a "quick look" at a solution, a 40 step per cycle, second-order simulation is acceptably accurate, and twice as fast.

The advantages of using the digitally simulated analog computer rather than a real one are:

(1) We can program the simulation using the same pictorial representation as used for the analog computer, with the additional simplification of noninverting integrators.

(2) No scaling of variables is necessary.

(3) The operation of constructing a confusing network of wires is replaced by writing a few simple BASIC statements. This is much faster, and more likely to be correct.

(4) If reasonable graphics are supported by the host computer, versatile and physically realistic system state displays are possible.

(5) A given simulation is exactly repeatable, and devoid of any analog electronic peculiarities.

(6) If reasonable step sizes are chosen the accuracy of the simulation is much better than the corresponding simulation performed on a real analog computer.

We emphasize that it is not initially necessary for the student to understand *how* the integration routine works, just as he or she need not understand what the transistors inside the operational amplifiers do. However, unlike the operational amplifier electronics, which is inside a black box, the code for the integrators is readily accessible for inspection.

When using an analog computer, multiplication and division of variables and the introduction of nonlinear functions of variables require nonlinear analog devices and further scaling problems. Here, however, they become trivial operations in BASIC. For example, if the system above is to experience damping proportional to velocity squared, and is to be driven by an exponentially decaying sinusoidal forcing function, the first statement in the user specification subroutine just becomes

```
D(1) = - K*X(I) ∧ 2 - W*X(2)
    - A*EXP( - T/TD)*SIN(WO*T).
```

As an example of simulating systems controlled by coupled differential equations, consider the motion of two coupled pendula having displacements x and y. The forces on each pendulum are

$$F_x = -kx + k_1(y-x),$$
$$F_y = ky + k_1(x-y),$$

where k describes the restoring force and k_1 is a coupling constant. The motion of either pendulum is a combination of motions at the symmetric and antisymmetric mode angular frequencies given by

$$\omega_s = k^{1/2},$$
$$\omega_A \doteq \omega_s\left(1 + \frac{2k_1}{k}\right)^{1/2}.$$

The system specification subroutine must describe the equations

$$\frac{d^2x}{dt^2} = Ax + By,$$
$$\frac{d^2y}{dt^2} = Ay + Bx,$$

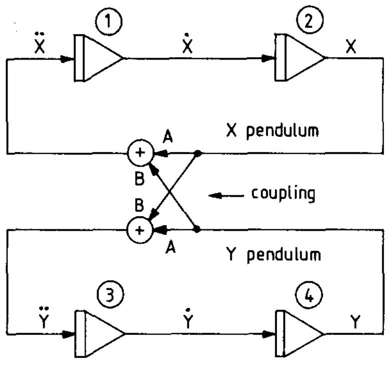

Fig. 3. Flow diagram for simulation of coupled pendula.

where

$a = -(k+k_1)/m,$

$B = k_1/m,$

m = mass of both pendula (assumed equal).

The flow chart of the simulation is shown in Fig. 3. Virtual integrators 1 and 2 have been used for the x pendulum, 3 and 4 for the y pendulum. The system specification subroutine statements are

```
D(1) = A*X(2) + B*X(4)
D(2) = X(1)
D(3) = A*X(4) + B*X(2)
D(4) = X(3).
```

The second-order integrator is adequate if DT is about 0.05 times the uncoupled natural pendulum period.

The following examples illustrate simulations that would not normally be attempted on a "teaching grade" analog computer because of their nonlinearity and complexity.

III. ORBIT SIMULATIONS

The force between a primary body and an orbiting satellite of negligible mass is given by

$$F = GMm/r^2.$$

For a circular orbit the satellite orbital velocity and period are

$$v = (GM/r)^{1/2}, \quad T = 2\pi(r^3/GM)^{1/2},$$

where G = universal gravitational constant, M and m are the masses of primary and satellite, and r = orbit radius. Four integrators are required—two each for the x and y satellite positional coordinates. Using second-order integration with about 100 steps per orbit, and starting the satellite with different x and y velocities, circular, elliptical, and hyperbolic orbits are readily demonstrated. If a two-dimensional plan view of the system is displayed, Kepler's laws may be experimentally verified by "measuring angles and counting steps." It is easily demonstrated that any force law other than inverse square results in unstable orbits.

An interesting extension is to assume that the primary has an atmosphere which exerts a retarding force on the satellite, causing it to continuously lose energy. Simple considerations suggest that the force should be proportional to satellite velocity squared times the atmospheric density, and should act in a direction opposite to the instantan-

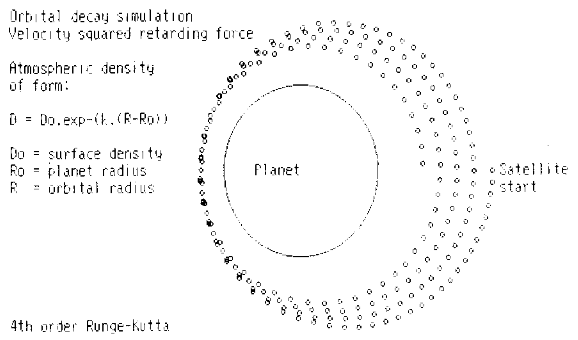

Fig. 4. Result of simulating the orbital decay of a satellite orbiting a planet having an atmosphere.

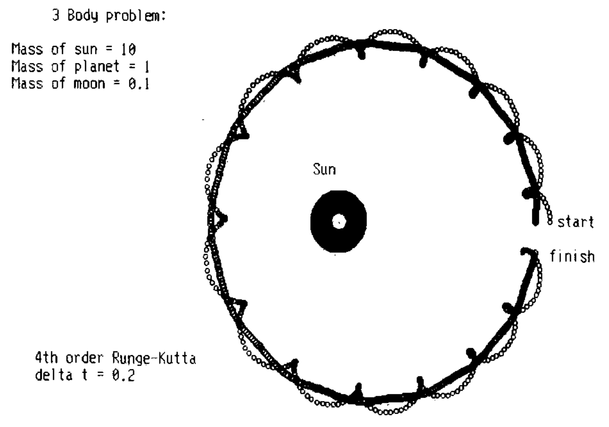

Fig. 5. Result of a three-body simulation involving a sun, a planet, and a moon orbiting the planet.

eous direction of motion. If the atmospheric density is assumed to decrease exponentially with distance from the primary, typical user subroutine statements defining the system are

```
R     = SQR(X(2) ∧ 2 + X(4) ∧ 2)
F     = − K/(R*R):V = X(1):W = X(3)
D(1)  = − F*X(2)/R
        − V*ABS(V)*100*EXP( − 2*(R − 5))
D(3)  = − F*X(4)/R
        − W*ABS(W)*100*EXP( − 2*(R − 5))
D(2)  = X(1):D(4) = X(3):RETURN.
```

These specifications resulted in the simulation shown in Fig. 4. The satellite was started moving radially with a velocity of 0.83 of that required for a circular orbit. The first four rotations are shown. Three further phenomena are correctly predicted: The orbit becomes progressively more circular, the orbital period continually decreases, and the height of the perigee remains almost constant. Eventually the orbit becomes completely circular, and the satellite rapidly crashes into the primary.

Finally, it is instructive to simulate a simple form of the three-body problem. If the number of bodies interacting gravitationally is greater than two, no analytic solution for their resulting motion exists. The corresponding simulation may however be readily performed. The system is described by the three coupled nonlinear equations:

$$\frac{d^2\mathbf{r}_1}{dt^2} = -G\left(\frac{m_2}{r_{12}^2} + \frac{m_3}{r_{13}^2}\right),$$

$$\frac{d^2\mathbf{r}_2}{dt^2} = -G\left(\frac{m_1}{r_{12}^2} + \frac{m_3}{r_{23}^2}\right),$$

$$\frac{d^2\mathbf{r}_3}{dt^2} = -G\left(\frac{m_1}{r_{13}^2} + \frac{m_2}{r_{23}^2}\right),$$

where

r_{ij} = distance between ith and jth bodies,

m_i = mass of ith body,

G = gravitational constant.

Twelve integrators are necessary. Figure 5 shows the results of a simulation where bodies 1, 2, and 3 are the sun, a planet, and a moon, having relative masses 100, 10, and 1, respectively. All three bodies were initially aligned on the right-hand side of the plot. The planet and moon were positioned 10 and 11 (arbitrary) units distance from the sun. The planet was set in motion at the apogee of a slightly elliptical orbit, with the moon describing a nominally circular motion about it. The sun's motion appears as an annulus, the superposition of the consecutive circles used to display it, as the simulation is plotted in the center of mass reference frame.

IV. UNDERWATER ACOUSTIC RAY TRACING

The velocity of sound in the sea is about 1500 m/s. Slight variations are caused by fluctuations of ambient salinity, temperature, and pressure. Environmental effects cause the velocity in the top hundred meters or so to be highly variable, but below this depth the velocity, at first, decreases regularly as the water temperature decreases. A minimum velocity is reached at a depth of about 1 km in mid-latitudes, after which it increases monotonically again under the influence of increasing pressure. Hence a duct is formed, called the SOFAR channel, which can guide acoustic energy over long distances without interaction with either the surface or the bottom.

It is well known that this channel exhibits focusing properties. Rays from a point source on or near the channel axis (velocity minimum) which do not encounter the surface or bottom will all repeatedly cross the axis again at nearly the same horizontal distance from the source. This can be readily demonstrated by performing a two-dimensional ray trace for several different rays.

The differential equations describing rays can be derived from the eikonal equation (Ref. 5). For a medium having no variations of velocity in the z direction, these can be written

$$\frac{d}{ds}\left(\mu\frac{dx}{ds}\right) = \frac{\partial\mu}{\partial x},$$

$$\frac{d}{ds}\left(\mu\frac{dy}{ds}\right) = \frac{\partial\mu}{\partial y},$$

where

s = distance along a ray, and μ = refractive index.

Since it is more convenient to integrate over equal intervals of time than ray path distance we use the transformation

$$\frac{d}{ds} = \frac{n}{c_0}\frac{d}{dt},$$

where

c_0 = ray velocity at the SOFAR channel axis,

n = normalized refractive index

= 1 on channel axis at depth y_0.

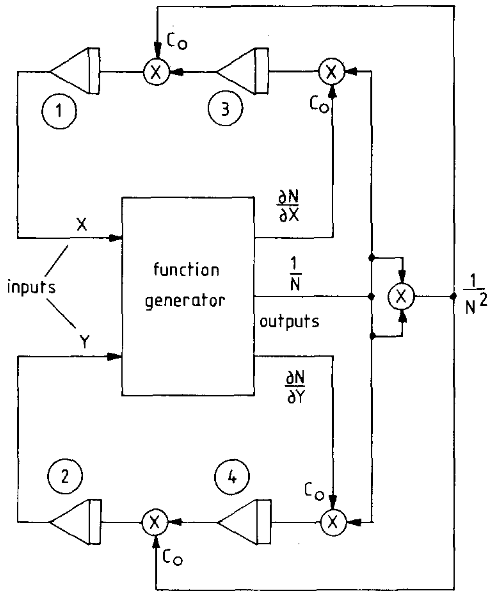

Fig. 6. Flow diagram for simulation of two-dimensional ray trace.

The ray equations then become

$$\frac{d}{dt}\left(\frac{n^2}{c_0}\frac{dx}{dt}\right)=\frac{c_0}{n}\frac{\partial n}{\partial x},$$

$$\frac{d}{dt}\left(\frac{n^2}{c_0}\frac{dy}{dt}\right)=\frac{c_0}{n}\frac{\partial n}{\partial y}.$$

Let

$$A=\frac{n^2}{c_0}\frac{dx}{dt},\quad B=\frac{n^2}{c_0}\frac{dy}{dt}.$$

Substituting, we find the rays are described by the four coupled, first-order, nonlinear differential equations

$$\frac{dA}{dt}=\frac{c_0}{n}\frac{\partial n}{\partial x},\quad \frac{dB}{dt}=\frac{c_0}{n}\frac{\partial n}{\partial y},$$

$$\frac{dx}{dt}=\frac{c_0}{n^2}A,\quad \frac{dy}{dt}=\frac{c_0}{n^2}B.$$

If we let the inputs to virtual integrators 1, 2, 3, and 4 be the time differentials of x, A, y, and B, respectively, the required simulation configuration is shown in Fig. 6.

For an analog computer simulation we would require a function generator module which accepted x and y as inputs (horizontal distance and depth, respectively) and provided outputs of $1/n$, and the derivatives of n with respect to both x and y. Five multipliers would also be required. This would be a formidable simulation if set as a teaching exercise.

If n is analytic in n and y, so that the partial derivatives are also analytic, the system is readily simulated digitally. Initial conditions are

$x(0)=0$	(integrator 1),
$y(0)=$ initial depth	(integrator 2),
$A(0)=n_0\cos\theta_0$	(integrator 3),
$B(0)=n_0\sin\theta_0$	(integrator 4),

where

n_0 = normalized refractive index at ray start point,

θ_0 = initial angle between ray direction and horizontal.

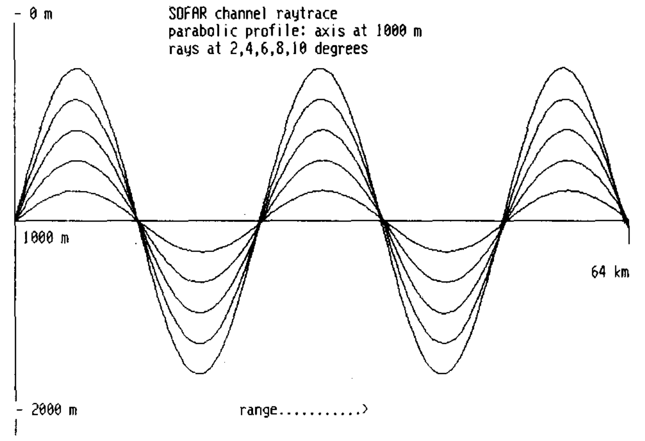

Fig. 7. Result of a two-dimensional ray trace simulating ray paths trapped by an invariant SOFAR channel.

The system specification statements are

```
T1 = CO/FNN(X(1),X(2))
T2 = T1/FNN(X(1),X(2))
D(1) = T2*X(3)
D(2) = T2*X(4)
D(3) = T1*FNX(X(1),X(2))
D(4) = T1*FNY(X(1),X(2)),
```

where

FNN(X,Y)	= normalized refractive index at (X,Y),
FNX(X,Y)	= partial derivative at (X,Y) with respect to X.
FNY(X,Y)	= partial derivative at (X,Y) with respect to Y.

These functions must also be specified by the user.

Consider initially the simplest situation where the SOFAR channel axis is invariant with x, and has the parabolic profile

$$n=1-k_c(y-y_0)^2,$$

then

$$\frac{\partial n}{\partial y}=-2k_c(y-y_0),$$

where k_c is a constant determining the "strength" of the focusing channel.

Figure 7 shows the results of such a ray trace where $y_0=1000$ m, $k_c=7\times10^{-8}$, and the time increment is 0.3 s. Rays are shown leaving the axis at angles of 2, 4, 6, 8, and 10 degrees from the horizontal. The focusing distance is about 12.3 km. If the scale of the x axis is expanded and the time increment reduced, it can be demonstrated that the focusing distance is slightly smaller for rays departing at larger angles, and that such rays arrive back at the axis slightly earlier. Hence impulsive sounds which have traveled via the SOFAR channel will be received as dispersed pulses having a slow rise time and an abrupt cutoff, as is well known.

Figure 8 shows the result of a simulation where the SOFAR channel strength decays exponentially with horizontal distance, but the channel axis remains at a constant depth. n is given by

$$n=1-K_c(y-y_0)^2\exp(-x/x_0)$$

and both partial derivatives of n exist. It is seen that ray

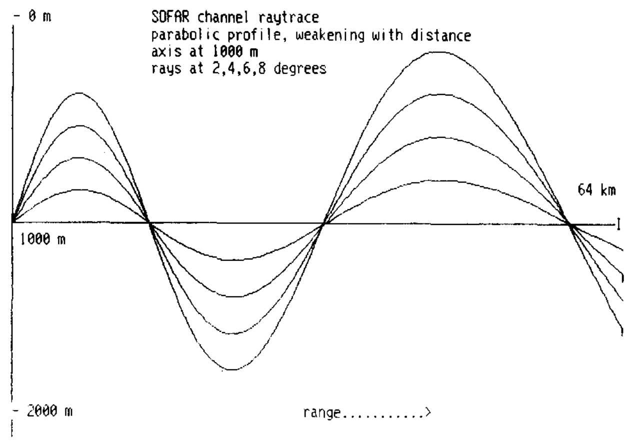

Fig. 8. Result of a two-dimensional ray trace simulating ray paths trapped in a SOFAR channel which is at constant depth, but decays in strength with propagation distance.

excursions become larger and that the focusing distance increases as x increases, as intuitively expected.

Although developed in the context of underwater acoustics, this ray tracing procedure could equally well demonstrate ionospheric bending of HF radio waves, seismic rays, etc. Boundary reflections can be included if desired.

V. CONCLUSION

Replacing a "real" analog computer with a digitally simulated "virtual" one retains the conceptually easy analog computer approach to simulation while bypassing the limitations of "real" analog computers. Our procedure avoids utilizing specialized simulation languages which introduce additional software "black boxes" to perplex the student—at least in the initial stages. The simplest of microcomputer systems can be used (although some form of graphical output is desirable) and only a BASIC interpreter is necessary for software support. The simulations shown in this paper were performed on a Panasonic JB-3000 computer running BASICA. Graphical output was obtained from a screen dump.

We have found that simulations can be programmed faster, and that results are much better than with analog computers. Students grasp the procedure readily, and are intrigued to see solutions evolving while they watch. Relatively complex nonlinear simulations are straightforward. We have for example also simulated phase-locked loop acquisition behavior, oscillatory systems obeying Van der Pohl's equation, harmonic oscillator problems involving Schrödinger's equation, and the response of active electrical networks to arbitrary excitations.

[1]"Doing Physics with Computers," Phys. Today **36** (5) (May 1983) (special issue on computers in physics).

[2]"Applications for Array Processors," IEEE Computer **16** (6) (June 1983) (special issue).

[3]M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions* (U. S. Natl. Bur. Stand., Washington, DC, 1964).

[4]I. Babuska, M. Prager, and E. Vitasek, *Numerical Processes in Differential Equations* (Wiley, New York, 1966).

[5]L. E. Kinsler, A. R. Frey, B. R. Coppens, and J. V. Sanders, *Fundamentals of Acoustics* (Wiley, New York, 1982), 3rd ed.

Experimental analysis of the work done by a variable force

Joel F. Sherman
Arts and Science Division, Brevard Community College, Melbourne, Florida 32935

(Received 14 May 1984; accepted for publication 24 July 1984)

The work done in moving a mass at the end of a string through an arc, under the influence of a variable horizontal force, is measured by pre-engineering students in a general physics laboratory. The experiment is designed to illustrate the concept of work done by a force as the area under a force–distance curve and to demonstrate the relation between the work done by forces on a system and resulting changes in the system's mechanical energy. The student finds that an experimental approximation to the integration process may yield reasonably good results using finite intervals over which the work is calculated.

I. INTRODUCTION

The general physics student often has difficulty making the connection between concepts introduced in the calculus course and their applications to problems in physics. One area which should provide this link is the introduction to the line integral definition of work done by a force. The student is carefully led through the general definition of work, with examples of work done by forces varying both in magnitude and direction as a function of position, and shown the treatment of constant forces as a special limiting case. When it comes to the laboratory, however, all of the commonly performed experiments deal with work due to constant forces, or, utilize an energy approach and bypass the need to calculate the work due to variable forces. These include the ballistic pendulum experiment and a variety of