DOCUMENT RESUME

ED 029 516 EM 007 253

By-Bork, Alfred M.; And Others

Introductory Computer-Based Mechanics: A One Week Sample Course.

Pub Date Nov 68

Note-87p.

Available from-Editor, Commission on College Physics, University of Maryland, 4321 Hartwick Road, College Park, Md. 20740 (upon request)

EDRS Price MF-$0.50 HC-$4.45

Descriptors-*Acceleration, *Algorithms, Calculus, *Computer Assisted Instruction, Instructional Innovation, Manuals, Physics Curriculum, *Physics Instruction, Programing Languages

Identifiers-BASIC, FORTAN, JOSS, PL-1

Very little material exists for utilizing the computer in the physics classroom, and even that little is not widely known. It is hoped that this monograph will provide some stimulus both to innovation and to discussion of the role of the computer in physics education. The paper describes how this might be achieved with a detailed account of one week of instruction in the physics of the harmonic oscillator, without calculus, for either physics or non-physics majors. The course is organized into three lectures, Days One to Three, and a Laboratory Session. Day One develops the basic first-order numerical integration scheme for computing velocity and position from a knowledge of acceleration and initial conditions. Day Two discusses the nature and languages of computers, and the construction of algorithms for computation, and Day Three is available in four different versions, one for each of the well-known computer languages-- BASIC, FORTRAN, JOSS, and PL-1. A Student Manual and a Teacher's Guide are included in this paper. (GO)

INTRODUCTORY COMPUTER-BASED MECHANICS

A One Week Sample Course

by

ALFRED M. BORK
University of California, Irvine

ARTHUR LUEHRMANN
Dartmouth College

JOHN W. ROBSON
University of Arizona

RONALD BLUM, Editor
Commission on College Physics

November 1968

PREFACE

This monograph was written to illustrate how one might use the computer to advantage in an introductory physics course, and also to indicate, more generally, how the computer can be intimately interwoven into the teaching of physics. Although computers have been very much with us for a decade, there has been little attempt to involve them in the education of physicists. However, one may recognize three possible modes of computer usage in the physics curriculum: (1) calculator; (2) simulator; (3) tutor. That is, the computer may be used to perform calculations, or as a pseudo-analog device to simulate physical phenomena (for example, radioactive decay by the use of a random number generator), or for the actual interactive presentation of material and evaluation of responses more commonly known as computer-assisted instruction (CAI). In this paper, the authors utilize the computer as a calculator, preferring a time-shared teletype terminal or a small readily accessible computer which will be immediately available to students during the laboratory session and provide them with the experience of direct interaction with the computer. This type of arrangement enables them to freely change the nature of the forces or initial conditions and to observe the effects of these changes at once, without loss of continuity or interest.

The segment of curriculum presented here is intended to comprise one week of instruction in the physics of the harmonic oscillator, without calculus, for either physics or non-physics majors. The philosophy motivating this presentation is that knowledge of the unique conceptual advantages and problems of the computer should be acquired early in the physics curriculum if the computer is to become a fundamental part of the physicist's problem-solving repertory. Ultimately, we may see the methods of numerical analysis and the calculus of finite differences fully integrated into physics curricula in anticipation of their relevance to the utilization of the computer. This is not to supplant the standard mathematical analysis which has traditionally accompanied the physics curriculum, but rather to complement the mathematics by enabling the student to explore a broader range of more meaningful problems. Thus, both student and teacher are no longer restricted to the classical setpiece problems of physics by the students' lack of mathematical sophistication, but are free to go as far and as fast as physical understanding can carry them.

The course is organized into three lectures, Days One to Three, and a Laboratory Session. A Student Manual and a Teacher's Guide are available; both are bound together in this monograph. Day One develops the basic first-order numerical integration scheme for computing velocity and position from a knowledge of acceleration and initial conditions. Day Two goes on to discuss the nature of computers and their languages, and the construction of algorithms for computation. The case in point is harmonic oscillation under the Hooke's Law linear restoring force; the problem is appropriately scaled and a flow chart constructed for the basic computational loop. Day Three is available in four different versions, one for each of the well-known computer languages: BASIC, FORTRAN, JOSS, and PL/1. The versions are interchangeable, and all four are included in this monograph. In Day Three the structure of the programs is explained in detail and computations performed. The emphasis is not on the language *per se*, but on the analysis; the language is discussed only insofar as needed for the analysis. The work is extended to cover the damped harmonic oscillator problem, a subject customarily not treated before the second year of physics *cum* calculus.

Very little material exists for utilizing the computer in the physics classroom, and even that material is not widely known.* It is hoped that this paper will provide some stimulus both to innovation and to discussion of the role of the computer in physics education. The authors, having bravely ventured into these uncharted waters, will warmly welcome any comments and suggestions from their readers. Of particular relevance would be remarks from educators who use this material in classroom situations. Communications or inquiries concerning extra copies of the Student Manual should be addressed to the Editor at the Commission on College Physics, University of Maryland, 4321 Hartwick Road, College Park, Maryland 20740.

In closing, the Editor would like to acknowledge the capable and devoted assistance of Miss Kathryn E. Mervine and Mrs. Faye von Limbach of the Commission staff in the preparation of this manuscript.

Ronald Blum
Commission on College Physics
University of Maryland
4321 Hartwick Road
College Park, Maryland 20740

*See *American Journal of Physics* 35, 273 (1967).

TABLE OF CONTENTS

STUDENT MANUAL

INTRODUCTORY COMPUTER-BASED MECHANICS

A One Week Sample Course

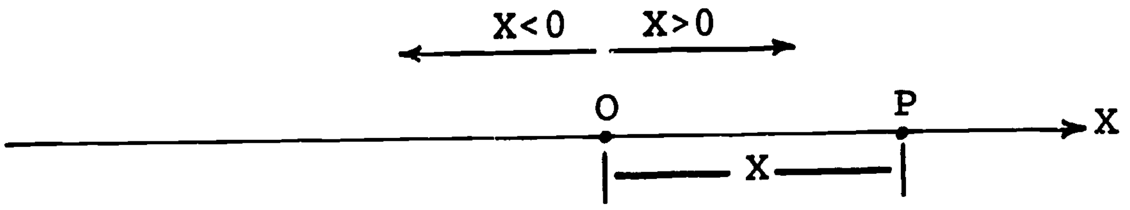
INTRODUCTION

The area of physics treated here is mechanics, the study of the motions of bodies; the ideas and concepts used here are related to those from your previous work in your physics course. What is new is the use of the computer as an integral part of your studies. That is, we shall attempt to indicate how the particular capabilities of the computer earn for it a place in the conceptual framework of the physics curriculum beyond its mere usefulness as a calculating device.

The following material is intended to introduce you to the use of computers within physics, to acquaint you with some important mathematical concepts, and to apply these to the solution of a fundamental physical problem, the harmonic oscillator. We shall discuss the relationships between the displacement, or position, of a body, its velocity and its acceleration; and we will see how the computer can be used to perform the requisite calculations so that, at this stage of your development, you can understand the physical principles and apply them without a prior knowledge of calculus. We then show how these principles can be used to construct and solve the equations of motion of a harmonic oscillator with the aid of the computer. Although we shall treat the case of a mass on a spring, the idea of harmonic oscillatory phenomena is of great and fundamental importance throughout the whole of physical science.

DAY ONE

The physical situation you are invited to consider is that of a point particle P moving along a straight line, as shown below:



An origin has been selected on the line, and distances are measured from this origin, positive in one direction and negative in the other. First, let us look at a case with which you are already familiar, motion at constant velocity. We want to find where the particle is going to be, knowing its position at an earlier time, and the constant value of the velocity.

We know that the particle has moved a distance equal to the product of its speed and the elapsed time. The new position is simply obtained by adding this distance to the earlier position. Expressing this symbolically, in a notation that will probably not be too different from that used in your text, we can write

$$X_{new} = X_{old} + V \cdot D \tag{1a}$$

where X_{new} and X_{old} are the new and old positions along the line, V is the constant velocity, and D is the time that has elapsed as the particle moves from X_{old} to X_{new}. In other words, $D = T_{new} - T_{old}$. If you have been using different notation, you should "translate" this expression into the more familiar notation, so that you are convinced it is nothing new. The information contained in this equation is nothing more than the common sense information you already know from your everyday experience with automobiles; that is, if the automobile at X is traveling at a constant velocity of 30 miles per hour, it travels 90 miles further in three hours.

Suppose we consider one numerical example, to see how to use this relation. If the particle is initially at 3.4 cm., and the velocity was 7.2 cm./second, to find its position two seconds later we would proceed as follows:

$$\begin{aligned} X_{new} &= X_{old} + V \cdot D \\ &= 3.4 + (7.2) \cdot (2) \\ &= 3.4 + 14.4 \\ &= 17.8 \end{aligned}$$

This is rather trivial calculation, and should not cause any difficulty.

However, one can, as we will see, use simple relations like this to tell us much more about the motion of bodies than you might suspect. Consider the following table of velocities at different times.

TIME (sec.)	VELOCITY (cm./sec.)
.0	6.34
.1	6.46
.2	6.74
.3	6.81
.4	6.77
.5	6.41
.6	6.10
.7	5.70
.8	5.54
.9	5.60

The velocity is not constant here, as it was in the previous example. For the moment we will ignore that difficulty; later we will show how to treat the general case of non-constant velocity.

The problem we are to solve is this:

> Given the velocities of the particle at particular times, we want to find its position at the same times, using the relationship that we have developed above and adding any new information needed.

Suppose we want to calculate the position at .1 sec, knowing the velocity at time 0. Here we are immediately confronted with a problem: we need more information. Our equation allows us to compute a new position when we already know an older position. We must have some initial information about where the particle is at the beginning of our calculation. We shall find that the need for such initial values is an important and characteristic feature in all calculations of the type we are considering here, even in areas quite different from mechanics.

Let's then choose an initial value for the position of the particle at the time 0; suppose X = 4.3 cm. at T = 0.

Now the calculation can proceed on a line-by-line basis, using the information we have on hand and the relation for computing the new position. The new position just calculated becomes the old position in the next calculation. Thus, to find the position at the time .1 sec., we make the following calculations:

$$X_{new} = 4.30 + (6.34)(.1) = 4.93 \text{ cm.}$$

To find the position at .2 sec., we use this newly calculated position as the "old" position and proceed as follows:

$$X_{new} = 4.93 + (6.46)(.1) = 5.58 \text{ cm.}$$

Here is the result of carying out some of these computations.

TIME (sec.)	VELOCITY (cm./sec.)	POSITION (cm.)
0	6.34	4.30
.10	6.46	4.93
.20	6.74	5.58
.30	6.81	6.25
.40	6.77	6.93
.50	6.41	7.61
.60	6.10	
.70	5.70	
.80	5.54	
.90	5.60	
1.00	-	

Exercise 1: Check to see that these values are correct and complete the table.

You should make certain that you understand how we are making these computations before you proceed further, as they are the basis for what we will be doing later.

Acceleration

Exactly the same type of relation can be used for obtaining new velocities, knowing accelerations and previous velocities. The relation here is as follows:

$$V_{new} = V_{old} + A \cdot D \qquad (1b)$$

The new quantity, A, is acceleration, while the subscripts as before indicate the new and old values of the velocity, at the beginning of the time interval, and then a time D later. Just as before, this relation is not strictly true unless the acceleration has been constant over the time interval, and so we are still faced with the same problem we had before. We will return to this problem later. Again, you should relate this to expressions used in your own text, and read the discussions there.

We could use Equation (1b) exactly as we have used Equation (1a); that is, if you were given a table of values of acceleration against time, you could, with just this relation, compute the velocities at each of the times, assuming we ignore the fact that acceleration is actually changing rather than constant.

However, this is not our main interest. Rather, what we would like to do, and what we have been working toward, is to use <u>both of these relations together</u>.

Suppose we start with the following values of the acceleration at different times, as given in the table:

TIME (sec.)	ACCELERATION (cm./sec.^{2})
0	1.3
.10	1.1
.20	.9
.30	.8
.40	.9
.50	1.0
.60	1.3
.70	1.5
.80	1.6
.90	1.4
1.00	1.2

We can use Equation (1b) to compute the <u>velocity</u> at T = .1 sec.; then by the use of Equation (1a) we can <u>compute</u> the position at T = .1 sec. Note that in this case we keep using the two equations alternately. We use first the velocity equation (1b) to give us the new velocity, and then we use this value in the position equation (1a) to find the new position. We could combine the two equations into a single one to calculate the new position; however, for reasons of simplicity we will avoid it.

<u>Exercise 2</u>: Make the calculations, finding what the positions will be from knowledge of the acceleration at a number of times. As-

(continued) sume that the initial position is 3.80 cm. and the initial velocity -1.30 cm./sec.

The information you are calculating is already nontrivial, in spite of the fact that the two relations are both very simple.

We could stop here, because we have all that is necessary for proceeding to the next stage of work. However, some additional insight can be gained, not only for our calculations here but for your future work in physics and mathematics, by some further mathematical considerations of the two basic equations we have used. Suppose you take the position equation, $X_{new} = X_{old} + V \cdot D$. We can rewrite this in the following form:

$$V = \frac{X_{new} - X_{old}}{D} \tag{1c}$$

You probably have used a similar relation for defining the average velocity during a time interval, so this expression should not be too unfamiliar. You will also, in all likelihood, have considered the case of the time interval D becoming smaller and smaller. Under these conditions the difference between the two positions also becomes smaller, but in most physical situations the <u>ratio</u>, V, tends to stay almost the same. This, you may recognize, is an intuitive physical approach to the concept of limit; one can allow D to get arbitrarily smaller and speak of the value of the ratio <u>in the limit of D approaching zero</u>.

No attempt will be made to give a full discussion of the concept of limit here, but if you have a calculus text available you can pursue these details further. The idea that the ratio of two quantities, each separately becoming smaller and smaller, can in the limit approach a finite value leads to a concept you may have met: the <u>derivative</u>. Velocity is often defined, in courses using calculus, in the following way: $V = dX/dT$, where dX/dT can be read as "the derivative of X with respect to T." What this means is that we have taken Equation (1c) and gone to the limit, using smaller and smaller D's. The mathematical details are not all here, and many mathematicians would approach the problem somewhat differently.

Exactly the same comments can be made about the acceleration, for which our second algebraic result can be written as a limit in the following way: $A = dV/dT$.

DAY TWO

We have been using two simple relations, one for computing the new position, $X_{new} = X_{old} + V \cdot D$, and one for computing the new velocity, $V_{new} = V_{old} + A \cdot D$, for a particle moving along a straight line. We have seen also how these two equations can be used together so that starting from information about the acceleration we can find information about where the particle will be at different times. We have also seen that to make such a calculation it is necessary to have information about initial position and velocity.

However, during our previous work we mentioned, but ignored, a problem which we now need to treat. Equation (1a) holds exactly only if the velocity is constant during the time-interval. If, for example, the velocity is increasing during the interval we would expect the new position calculated to be too small, because we are using the velocity at the beginning of the time interval. Exactly the same problem arises in the velocity calculation, Equation (1b), if the acceleration is changing.

Clearly, if the *velocity* is changing, the calculation for position will not be accurate. The critical question is, just *how much* error is introduced. The physicist is already well acquainted with error, because it is always present in any experimental situation. Hence he knows that his data, while they may look precise, are not. Numerical calculations are another possible source of error. Numerical errors generally arise from two sources: *roundoff* errors due to rounding numbers off to significant figures; and *truncation* errors due to using approximate formulas for certain mathematical functions. The former type will mainly concern us here, although the latter becomes more important as one deals with more complicated physical problems where such approximations are essential to obtain workable solutions.

Whether error is due to the measurement process or to computational simplifications the question is how much can we tolerate? If we can control the error, make it small enough, then it becomes tolerable. The criteria for "small enough" must ultimately reside in the judgment of the physicist concerning the nature of the problem, the purposes of his study, and the facilities available to him.

Consider our particle moving along a straight line. If the time interval is long, the particle's velocity during that interval may have changed radically. Hence, the calculated position at the end of that interval may bear little resemblance to its actual position. On the other hand, if the time interval is quite small, the situation is different. You know that most physical objects do not make *extremely* rapid changes in their velocity; velocity tends to change smoothly over appreciable time intervals. Hence, for small changes the error in calculation of the new position will be small even if you are using the velocity at the beginning of the time interval, because the velocity has not changed too greatly in this interval. This still does not tell us whether the error will be tolerable for our purposes, but at least suggests that with a small enough time interval the calculations may be acceptable. We will return to this issue later, when computer facilities are available and we can do some experimental work to determine how small is "small enough".

Exercise 3: In the first exercise you calculated each successive position by using the velocity at the beginning of each time interval. What would happen if you used the velocity at the end instead? Make a quick guess at how much difference you would expect in the positions, and then carry out the computations. Can you think of a better choice for the velocity than either of the above?

So far all of our calculations have been hand calculations. If you calculated the positions from the accelerations, as suggested, you will already have discovered that such calculations are not only tedious but also repetitive. Once the basic procedure of the calculation is established, it only remains to grind out the answers routinely. This work can be done by anyone who can follow the calculational procedures. Calculations of this kind can be performed by machines that have the same capability, digital computers. Most of the discussion that follows is concerned with using computers for problems which are amenable to such calculations. We use the word "amenable" quite deliberately; most problems can be approached in a variety of ways. The computer is one such way; sometimes the only way, sometimes simply more practical than other methods. However, in any case the computer must be considered as an essential part of the physicist's repertory of conceptual tools for solving problems, since its use requires special analytical schemes which take advantage of its unique ability to perform many routine procedures repetitively at high speed. The typical situation in which the computer is useful to the scientist is one where there is a large amount of calculation (or other symbolic manipulation) required. In our problem, so far, we have used only crude approximations; we still have not seen the complete situations, where even more calculation is necessary and the use of a computer not only reasonable, but, as calculations get longer and longer, essential.

How can we rewrite our expressions for position and velocity so that they can be used directly in a computer calculation?

Modern computer languages are problem-oriented, and allow a usage similar to that found in ordinary algebra. We will not be able to exhaust the enormous variety of computer languages in this brief treatment. In this section we will write expressions suitable, with very slight modifications, for a wide class of languages. In the next chapter we will consider detailed programs, using material from this chapter, for one of the four computer languages: FORTRAN, JOSS, BASIC, and PL/1.

Here is how we will translate our two algebraic statements:

$$X_{new} = X_{old} + V \cdot D \quad \text{becomes} \quad X = X + V*D \tag{2a}$$

$$V_{new} = V_{old} + A \cdot D \quad \text{becomes} \quad V = V + A*D \tag{2b}$$

As algebraic statements, (2a) and (2b) appear incorrect; however, the (=) statement in a computer language does not mean exact equality. Instead, it says compute the number on the right and, after it is computed, store it in that memory cell of the computer which has been given the name of the variable which appears on the left. Thus, expression (2a) says: retrieve the number stored in memory cell V, multiply it by the number stored in memory cell D, and add to that the number stored in memory cell X. The result is then stored in memory cell X, replacing the old value of X. Professional computer scientists will refer to (=) not as equality, but as an "assignment" statement, since it assigns certain values to be stored in certain memory cells. (To emphasize the distinction, they may even replace (=) by (←) or (:=).) If we had wished to save the old value of X we should have had to designate a separate memory cell for the purpose and fill it before we executed Equation (2a). The values of V and D however, are unchanged, since they only appear on the right side of Equation (2a). In Equation (2b) V is changed. It should be clear from this that the left side of an assignment statement can only have one variable. Note that the (*) is being used as a multiplication sign. This is not universal with all computer languages--some still use the familiar dot. It is necessary in some of the computer languages, including some we will consider, to put some introductory material in front of the statement. We will consider these details later.

In performing a routine computation many times with the computer, the procedure for doing so is called a "loop". In addition to calculating the new position and the new velocity, the basic calculational loop should also keep track of how far we have gone by finding the new time, T, at each stage in the calculation. You should be able to see from the previous discussion that the "new" time can be found from

$$T = T + D \tag{3}$$

Returning to physics, we come to the most important relation in dynamics. The critical factor in determining how particles move is Newton's Second Law of motion, the law which relates force, mass, and acceleration by the relation

$$F = M \cdot A \tag{4a}$$

Thus, if we know the force acting on the particle, Newton's law tells us how to find the acceleration by computing

$$A = F/M \tag{4b}$$

You can see why the Second Law is important: if we understand the nature of the physical forces at work we can determine the acceleration and, from the procedures discussed above, the velocity and position at each time.

Since the mass does not present any great problem we will defer its discussion for a moment. However, the force can change in complicated ways during the particle motion. In some situations the force can be measured directly, but most common forces in physics fortunately depend on position, on velocity, and on time, in relatively simple ways. If we know the force as a function of position, for example, and we know the position at a certain time, we can calculate the acceleration at that time, using the Second Law. With this information we can calculate the position and velocity at a later time. Thus we have the ability to perform much more powerful calculations than we did earlier. We can now undertake the fundamental problem of dynamics: to predict the motion of a body.

The details may be clearer if we focus on a specific example. Consider a force that you are probably already familiar with, the force on a mass on the end of a spring. This force is the familiar Hooke's Law force, proportional to the distance from the equilibrium position. The force is always directed toward the equilibrium position. Your text has probably already introduced a simple analytic expression for this force:

$$F = -K \cdot X \tag{5}$$

K is a factor of proportionality (the spring constant) which measures the stiffness of the spring. The coördinate system has been chosen so that the force is zero at the origin. The minus sign reflects the fact that the force is toward the origin.

We can combine Newton's Second Law, the Hooke's Law force, and the equation for the calculation for the new velocity. If we put these together we can write the statement for calculating velocity in our computer language as

$$V = V - (K/M)*X*D \tag{6a}$$

where the slash indicates division, as usual. Our three basic computer-language statements are now as follows:

$$X = X + V*D \tag{2a}$$

$$V = V - (K/M)*X*D \tag{6a}$$

$$T = T + D \tag{3}$$

You should notice that the strength of the spring and the mass of the particle enter the problem only in the combination K/M. We could solve for the motion starting with many different values of K and of M, but it is obvious that the motion only depends on the ratio K/M. In fact, there is a way to change the variables of the problem so that even this ratio disappears. A simple dimensional argument gives us a hint as to how to do this.

Exercise 4: Express K in terms of the fundamental units, mass, length and time. Show that the ratio K/M has dimensions $1/(\text{time})^2$.

It follows from the above result that M/K is measured in units of time. It is, as we shall see, a "natural" unit of time for this problem.

What happens if we measure all times in units of $\sqrt{M/K}$? In other words, let us replace T by $\sqrt{M/K}$ *T' and D, since it is just a time interval, by $\sqrt{M/K}$ *D'. (Note that T' and D' are now dimensionless variables.) Since velocity is measured in units of length divided by time, let us replace V by $\sqrt{K/M}$ *V'. With these changes the three steps in our calculation become

$$X = X + V'*D' \tag{7a}$$

$$V' = V' - X*D' \tag{7b}$$

$$T' = T' + D' \tag{7c}$$

where we have eliminated common factors from all terms in each statement.

Exercise 5: Convince yourself that this cancellation is legitimate by rewriting Equations (2a) and (6a) as algebraic equations and making the substitutes for V and T described above.

As a final step in this simplification let us drop all the primes in the previous computer statements. We must remember that when we want to compare the motion of different masses, for example, it is only necessary to multiply the calculated times by $\sqrt{M/K}$ and the calculated velocities by $\sqrt{K/M}$.

You should notice that the above changes have not only simplified the calculation but also have yielded a deeper understanding of the physical situation. Without having solved for the motion we have seen that changes in M/K only affect the "time scale" of the motion but not its fundamental character. Increasing M, for example, will increase the time scale and, in effect, do nothing more than slow down the motion.

Exercise 6: What will be the effect on the motion if K is increased by a factor of four?

Let us return now to the computational problem of solving for the motion by using the simplified statements:

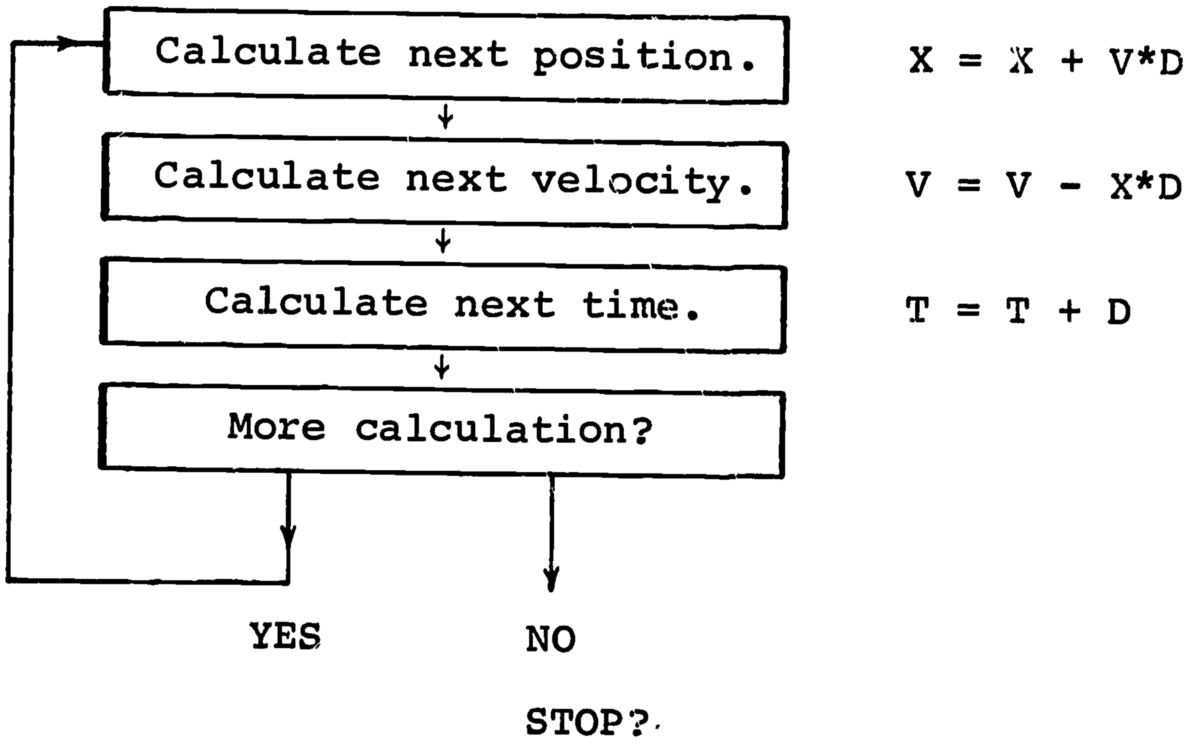
$$X = X + V*D \qquad (2a)$$

$$V = V - X*D \qquad (6b)$$

$$T = T + D \qquad (3)$$

(If you have difficulty in following the above line of reasoning, you may think of what follows as merely the special case in which M/K = 1.)

We now have on hand the basic mechanism for finding how bodies move, or at least how one particle, the mass on the end of a spring, moves. The method will later be generalizable to other situations in ways that you can probably already envision. We can illustrate the calculation by a diagram. The computer language statements are placed to the right of the blocks in which they are used.



Just as before, initial values of position and velocity will be needed to get the calculation started.

Exercise 7: Assume that at T = 0 sec. the position is X = 1 and the velocity is V = 0. This is equivalent to displacing the spring through unit distance, holding it at rest, then letting it go at the moment which we start the clock. Use the calculational loop presented above to find the positions, velocities, and times at intervals of .1 sec. in time, tabulating the data so obtained.

You may also want to describe this calculation to someone who knows no physics, but who knows how to follow directions and carry out arithemetical operations. Once the basic physics has been accounted for the problem becomes a computational task, rather than a physics problem.

You can note from the order in which the computations are presented that V_{new} is calculated using X_{new} on the right side of Equation (6b). It might seem more symmetrical to use X_{old} ; however, the accuracy is about the same in either case, and the method we have chosen is easier to program.

DAY THREE

We have outlined a method for finding how a particle moves, assuming motion along a straight line. The method requires a description of the force acting on the particle and uses Newton's Second Law to find the resulting acceleration of the particle (Equation (4b)). Assuming initial values of position and velocity, it is possible to calculate later values of position and velocity. We completed the last section by showing how all of this information could be brought together in a computational procedure to determine the position of the particle at successive times.

In the present chapter we shall examine a complete computer program for the calculation of successive values of the particle position and will explain how it contrives to accomplish this end. The program is written in BASIC, a commonly available language for use in systems having remote terminals connected to a computer. BASIC exists in forms with other names, so you may be using a "dialect" with a slightly different name in your class: XBASIC, SUPER BASIC, or HPBASIC.

Here is the BASIC program for the harmonic oscillator problem (your instructor will show you how to input this information to the computer):

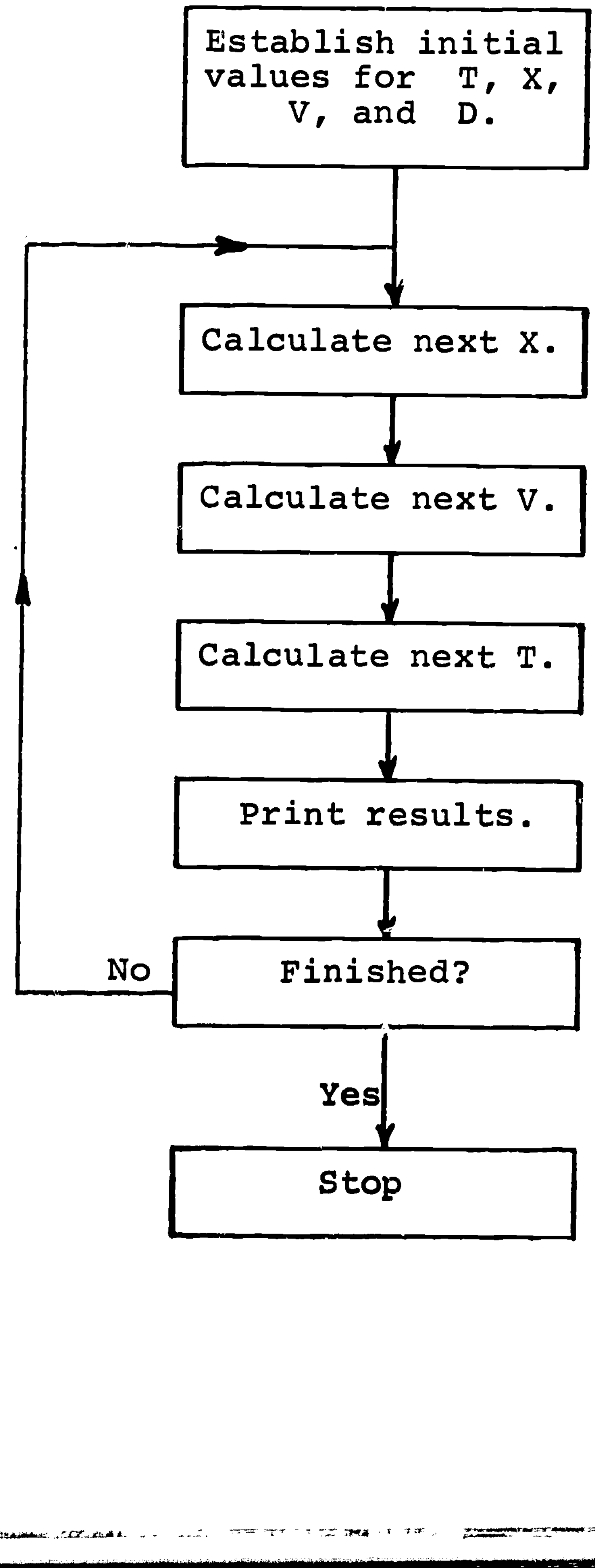
```
110 LET T=0
120 LET X=1
130 LET V=0
140 LET D=.1
150 LET X=X+V*D
160 LET V=V-X*D
170 LET T=T+D
180 PRINT T;X
190 IF T<3 THEN 150
200 END
```

This is a complete computer program, with all the instructions the computer needs to carry out the calculation.

You should have no trouble recognizing the basic features of the calculation. The first four lines assign initial values to the time, T, position, X, velocity, V, and time interval, D. New values of the position, velocity, and time are calculated in the next three lines. You can recognize the two steps which calculate the new position and the new velocity, because the relations, except for the statement-numbers at the beginning and the prefix "LET", are those that you have already seen. The PRINT statement is a rather obvious instruction to print the time T and position X on the terminal typewriter.

Statement 190 is a "branching" statement which alters the flow of the calculation. Normally, in most computer languages, the program statements are done in the sequential order in which they appear. In a BASIC program this order is the same as the numerical order of the numbers which precede the statements. However, if time T is less than 3, statement 190 transfers to 150, which is the next calculation of the new position. After T gets sufficiently large the branch is not made, and the program proceeds to the next statement after the IF-THEN statement. Since the next statement in the program is an END statement, the calculation is terminated when T > 3. (Statement numbers in BASIC must lie between 0 and 99999. The highest statement number used in a given program must be an END statement.)

The flow of computations can be represented in the following schematic flow chart:



```
110 LET T=0
120 LET X=1
130 LET V=0
140 LET D=.1

150 LET X=X+V*D

160 LET V=V-X*D

170 LET T=T+D

180 PRINT T;X

190 IF T<3 THEN 150

200 END
```

You can recognize the previous flow chart (page 13) of the basic computational loop as part of this chart.

When executed by computer, this program produces the following output:

```
.1      1
.2      .99
.3      .9701
.4      .940499
.5      .901493
.6      .853472
.7      .796916
.8      .732392
.9      .660543
1.      .582089
1.1     .497814
1.2     .408561
1.3     .315222
1.4     .218731
1.5     .120052
1.6     2.01736 E-2
1.7    -7.99069 E-2
1.8    -.179188
1.9    -.276678
2.     -.371401
2.1    -.46241
2.2    -.548794
2.3    -.629691
2.4    -.704291
2.5    -.771848
2.6    -.831687
2.7    -.883208
2.8    -.925898
2.9    -.959328
3.     -.983166
3.1    -.997171
```

It may puzzle you to see output for $T = 3.1$, since our IF test was to terminate for $T \geq 3$. The number 3.1 appears because internal computations are usually performed in binary, or base-two, arithmetic, and the conversion from binary to decimal (base-ten) numbers is often not quite exact. Thus, the value of T actually stored may have been something like 2.99999 instead of 3.0 when the IF test was performed.

This is still a crude program, although it is sufficient for the basic calculation. It types every value of time and position after it is calculated. In the present case this will only produce about thirty sets of values, so there is no great difficulty. We have already seen that our basic approximation only works when the time step D is small enough, but we do not know how small it would be. Therefore, we may want to change the time step, to .01

or even to .001, to explore the effect of smaller choices of D. However, our program becomes impractical if we change the time step to .001, because it would yield one hundred times as many lines of output as the present version.

One way to solve this problem is to generate printout only at specified "print-times", P, which may occur at intervals greater than D. We can do this by introducing an IF statement which bypasses PRINT T;X whenever T ≤ P. Furthermore, after each printing P is immediately increased by some multiple of D so that the statement is executed again only after a decent interval has elapsed.

For example, we may wish to use a time step only 1/10th as large as the previous one, but print out results only after every tenth computation, so the output will be similar to our previous program. We can provide for this by initializing D and P:

```
140 LET D=.01
141 LET P=.1-D
```

comparing T to P with

```
171 IF T<=P THEN 150
```

and if T > P executing the PRINT T;X statement and increasing P

```
181 P=P+.1
```

immediately after printing.

As before, the output of this program is a column of times and a parallel column of corresponding positions. But the columns are not labeled. Furthermore, a program needs to have its own self-contained documentation; the output should identify the problem and state what parameters were used in the calculation. This is not just window dressing, but a useful part of programming. The print statement can be used both to type a heading and to type initial values, as follows:

```
100 PRINT "HARMONIC OSCILLATOR"
101 PRINT
143 PRINT "TIME STEP =";D
144 PRINT "INITIAL X =";X
145 PRINT "INITIAL V =";V
146 PRINT
147 PRINT " T           X"
148 PRINT "---------------"
```

When the program is run, line 100 will type a heading before any output begins. Line 101 will merely produce a line with no printing on it. Lines 143-145 will each print the characters contained between the quotation marks, followed immediately by the appropriate number. (You should note in line 144 that the X between quotation marks is typed as an X, but the one outside causes the numerical value of X to be typed.) Line 146 inserts another blank line in the output. Line 147 prints column headings for the rest of the output, and line 148 is inserted merely to improve appearance of the output.

Note that lines 143-145 must appear later than lines 120, 130, and 140. If, instead, they were numbered 103-105, they would not type out the correct initial values of T, X, and V. The computer, remember, executes the statements of a program in sequence, and T, X, and V are not all defined until line 140 is reached.

The modified program now appears as follows:

```
100 PRINT "HARMONIC OSCILLATOR"
101 PRINT
110 LET T=0
120 LET X=1
130 LET V=0
140 LET D=.01
141 LET P=.1-D
143 PRINT "TIME STEP =";D
144 PRINT "INITIAL X =";X
145 PRINT "INITIAL V =";V
146 PRINT
147 PRINT " T           X"
148 PRINT "----------------"
150 LET X=X+V*D
160 LET V=V-X*D
170 LET T=T+D
171 IF T<=P THEN 150
180 PRINT T;X
181 LET P=P+.1
190 IF T<3 THEN 150
200 END
```

and execution by a computer produced the results given on the following page.

Exercise 8: Go through this calculation by hand far enough to confirm that output will be printed only when T = .1, .2, .3, etc. (Note that the transfer back to line 150 occurs when T is less than or equal to P.)

What about other force laws? So far we have a program only for the harmonic oscillator force, but it is not difficult to see where the dependence on force enters the calculation. We used the harmonic oscillator force to replace the acceleration in the expression for computing the new velocity (line 160 in our program). By simply changing this one line, we can use the same program to compute the motion for a different one-dimensional force.

We would also like to change the initial conditions, because

```
HARMONIC OSCILLATOR

TIME STEP = .01
INITIAL X = 1
INITIAL V = 0

 T           X

-----------------

.1          .995503
.2          .98106
.3          .956814
.4          .923007
.5          .879979
.6          .828157
.7          .768061
.8          .700291
.9          .625524
1.          .544506
1.1         .458048
1.2         .367013
1.3         .272311
1.4         .174889
1.5         7.57184 E-2
1.6        -2.42083 E-2
1.7        -.123893
1.8        -.22234
1.9        -.318566
2.         -.411608
2.1        -.500538
2.2        -.584466
2.3        -.662555
2.4        -.734023
2.5        -.798158
2.6        -.854317
2.7        -.90194
2.8        -.940551
2,9        -.969765
3.         -.989289
3.1        -.998928
```

different initial conditions might lead to very different motions. Again, in a BASIC system, it is easy to retype the lines assigning initial values to position and velocity (lines 120 and 130). Sometimes, additional statements will also be needed for specifying other constants, particularly when we want to explore a family of forces with different constants.

This is illustrated in the more realistic case of the damped oscillator. If you generate the harmonic oscillator solution for large values of time by making the necessary alteration in line 190, you will find that, according to the calculation, the oscillator always reaches the upper and lower limits, plus 1 and minus 1, during its cycle. But an actual mass on the end of a spring will eventually go less and less far, finally coming to rest in the equilibrium position. This is due to the frictional force of the air against the mass, which has been ignored in the computations so far. To a good approximation this force is proportional to the velocity but always has the opposite sign, because frictional forces always oppose the motion. Thus, the total force for such a system would be the sum of the Hooke's Law force and the frictional damping force:

$$F = -K*X - B*V \qquad (8)$$

where B, the damping constant, is a positive quantity.

It is not difficult to modify our program to solve for the motion of a damped oscillator by changing the velocity calculation. With an argument similar to that used before (see Exercise 5), we can eliminate K and M by a change of units; if we replace B by C to indicate this change the velocity equation becomes V = V-(X+C*V)*D, and the program for the damped oscillator then takes the following form:

```
100 PRINT "DAMPED HARMONIC OSCILLATOR"
101 PRINT
110 LET T=0
120 LET X=1
130 LET V=0
132 LET C=.5
140 LET D=.01
141 LET P=.1-D
142 PRINT "DAMPING CONST. =";C
143 PRINT "TIME STEP =";D
144 PRINT "INITIAL X =";X
145 PRINT "INITIAL V =";V
146 PRINT
147 PRINT " T           X"
148 PRINT "----------------"
150 LET X=X+V*D
160 LET V=V-(X+C*V)*D
170 LET T=T+D
171 IF T<=P THEN 150
180 PRINT T;X
181 LET P=P+.1
190 IF T<3 THEN 150
200 END
```

The new expression for the force has been used to calculate the velocity in line 160. Furthermore, note these additional lines in the program: 132 defines the damping constant C = .5, and 142 types its value. Presumably it would be interesting to run this problem with many choices of damping constant, so that one can get a physical picture of how damping effects the motion. This may be done by merely retyping line 132 with a different value of C before each run.

It should be apparent that the computer can be a powerful tool for solving problems in mechanics. Generalizations of many kinds are possible. We have used two force laws; it is not difficult to envision further extensions in that direction. Furthermore, the extension of these methods to two- and three-dimensional problems is straightforward. Because two coordinates are necessary for the specification of position and velocity, the solution of two-dimensional problems will require two equations, rather than the one equation used here. Details about such developments can be obtained in the references at the end of this material.

Thus, we have on hand the rudiments of a tool which will enable us to find out how any particle moves, and with the assistance of this tool, we have gone far toward understanding how dynamical systems behave. However, we should not let the computer obscure the importance of the mathematics. The kinds of equations that we are solving by our basic numerical procedures are ones which are not simply ordinary algebraic equations. We can see this if we go back and write the two equations that we already have as definitions of velocity and acceleration, using the derivative notation: V = dX/dT and A = dV/dT. These were the general equations. If we replace A by its value for a Hooke's Law force, now keeping the M and K, these two equations become

$$\frac{dX}{dT} = V \qquad (9a)$$

$$\frac{dV}{dT} = -\left(\frac{K}{M}\right)X \qquad (9b)$$

These equations contain derivatives, so they are called differential equations. What we have seen is one way of solving differential equations, a way particularly suitable for use with computers as it requires the manipulation of numbers.

If you proceed further in your study of physics, mathematics or other sciences, you will find that the differential equation, handled often by methods quite different from those used here, is one of the most powerful mathematical tools of modern science.

REFERENCES

Bork, Alfred M., Fortran for Physics, Addison-Wesley, Cambridge, 1965.

Feynman, Richard P., Feynman Lectures on Physics, Addison-Wesley, Cambridge, 1963, Chapter 9, Volume I.

Sawyer, Walter W., What is Calculus About?, Random House, New York, 1961, Chapters 2 and 3. (Also in Harvard Project Physics Reader 1.)

Sherwin, Chalmers W., Basic Concepts of Physics, Holt, Rinehart & Winston, New York, 1961.

DAY THREE

We have outlined a method for finding how a particle moves, assuming motion along a straight line. The method requires a description of the force acting on the particle and uses Newton's Second Law to find the resulting acceleration of the particle (Equation (4b)). Assuming initial values of position and velocity, it is possible to calculate later values of position and velocity. We completed the last section by showing how all of this information could be brought together in a computational procedure to determine the position of the particle at successive times.

In the present chapter we shall examine a complete computer program for the calculation of successive values of the particle position and will explain how it contrives to accomplish this end. The program is written in FORTRAN, perhaps the most commonly used computer language.

Here is the FORTRAN program for the harmonic oscillator problem (your instructor will show you how to input this information to the computer):

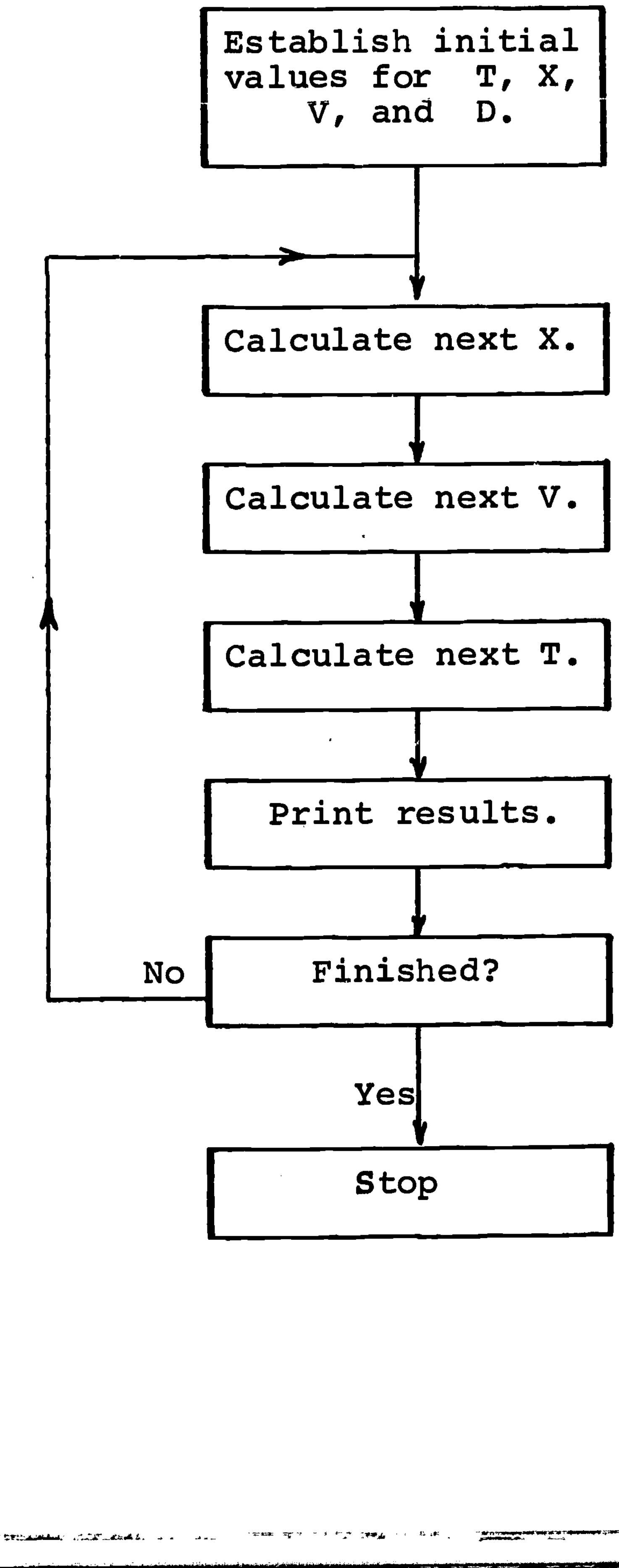
```
   T=0.
   X=1.
   V=0.
   D=.1
10 X=X+V*D
   V=V-X*D
   T=T+D
   WRITE(6,70)T,X
   IF(T-3.)10,10,14
14 STOP
70 FORMAT(F10.2,F12.4)
   END
```

This is a complete computer program, with all the instructions the computer needs to carry out the calculation.

You should have no trouble recognizing the basic features of the calculation. The first four lines assign initial values to the time, T, position, X, velocity, V, and time interval, D. New values of the position, velocity, and time are calculated in the next three lines. The WRITE statement results in the printing of new values of time and position. The numeral 6, following "WRITE", selects the output device on which to print (in this case, the "line printer"). The numeral 70, following "WRITE", tells us to look for a line beginning with the label "70" and to find there instructions for details of the printing operation.

The line which begins with "IF", appropriately called an IF statement, is a "branching" instruction which can alter the normal flow of the calculation. The normal flow is simply the order in which the statements are presented. In the case of the IF statement, the sequence of operations can be altered depending on the value of the expression appearing in parentheses after the IF. If the expression is negative, the next statement to be executed will be the statement labeled by the first numeral following the parentheses; if the expression is positive, the third numeral following the parentheses labels the next statement to be executed. Finally, if the expression is zero, the middle numeral labels the next statement to be executed. Thus, the calculation of the next value of the position by the statement which is labeled with the numeral 10 will be performed as long as T is less than or equal to 3.; only when T is greater than 3. will the computer realize that it has done enough computation.

The flow of computations can be represented in the following schematic flow chart:



```
         T=0.
         X=1.
         V=0.
         D=.1

      10 X=X+V*D

         V=V-X*D

         T=T+D

         WRITE(6,70)T,X

         IF(T-3.)10,10,14

      14 STOP
```

(The FORMAT statement does not appear here; we must consider it, not as an executable part of our program, but rather as a separate piece of information which we give the computer so that it will know how to print out our results. Hence, it can occur anywhere in the program, even after the STOP command.) You can recognize the previous flow chart (page 13) of the basic computational loop as part of this chart.

When executed by computer, this program produces the following output:

```
0.10      1.0000
0.20      0.9900
0.30      0.9701
0.40      0.9405
0.50      0.9015
0.60      0.8535
0.70      0.7969
0.80      0.7324
0.90      0.6605
1.00      0.5821
1.10      0.4978
1.20      0.4086
1.30      0.3152
1.40      0.2187
1.50      0.1201
1.60      0.0202
1.70     -0.0799
1.80     -0.1792
1.90     -0.2767
2.00     -0.3714
2.10     -0.4624
2.20     -0.5488
2.30     -0.6297
2.40     -0.7043
2.50     -0.7718
2.60     -0.8317
2.70     -0.8832
2.80     -0.9259
2.90     -0.9593
3.00     -0.9832
3.10     -0.9972
```

Note the use of numerals as labels for some statements in the program. If the normal sequence of performing operations (i.e., in the order in which they are written) were never to be altered there would be no need for statement labels. On the other hand, programs would become impossibly long if it were not possible to alter the normal sequence; our present simple program would be about 30 times as long if we could not branch from the normal sequence and use the basic computational sequence over and over again.

This is still a crude program, although it is sufficient for the basic calculation. It prints every value of time and position after it is calculated. In the present case this will only produce about thirty sets of values, so there is no great difficulty. We have already seen that our basic approximation only works when the time step D is small enough, but we do not know how small it should be. Hence we may want to change the time step, to .01 or even to .001, to explore the effect of smaller choices of D. However, our program becomes impractical if we change the time step to .001, because it would yield one hundred times as many lines of output as the present version.

One way to solve this problem is to generate printed output only at specified "print-times", P, which may occur at intervals greater than D. We can do this by introducing an IF statement which writes T, X only when T > P. Furthermore, after each writing, P is immediately increased by some multiple of D so that the next WRITE statement is executed only after a decent interval has elapsed.

For example, we may wish to use a time step only 1/10th as large as the previous one, but print out results only after every tenth computation, so the output will be similar to our previous program. We can accomplish this by initializing D and P:

```
D=.01
P=.1-D
```

comparing T to P with

```
IF(T-P)10,10,12
```

branching to the WRITE statement, now labeled 12, when T > P

```
12 WRITE(6,70)T,X
```

and increasing P by

```
P=P+.1
```

immediately after writing.

As before, the output of this program is a column of times and a parallel column of corresponding positions. But the columns are not labeled. Furthermore, a program needs to have its own self-contained documentation; the output should identify the problem and state what parameters were used in the calculation. This is not just window dressing, but a useful part of programming. Thus, we further modify the program by adding the lines

```
   WRITE(6,30)
```

and

```
30 FORMAT(24H1    HARMONIC OSCILLATOR,//)
```

to produce the top line of the heading on the output, and the lines

```
      WRITE(6,40)X,V,D,P
```

and

```
   40 FORMAT(5H0  X=,F4.1,4H  V=,F4.1,4H  D=,F5.2,4H  P=,F5.2,//)
```

to command the computer to print some of the initial values. In statement 30, "24H" is a signal to the computer to treat the next 24 characters, including blank spaces, as a text or "character string". This text can be printed by a WRITE statement with the exception that the first character on a line is interpreted as a "carriage control" signal to the line printer and is not printed; the 1 in this statement is the signal to start on the top of a new page. In statement 40, the zero immediately following "5H" is a carriage control signal to skip a line before printing. Note that the "X" in the WRITE statement is an instruction to print the current value of X while the "X" in the FORMAT statement is an instruction to print the character X. The two slashes result in one blank line following this printed line. Finally, the column headings are produced by

```
      WRITE(6,60)
```

and

```
   60 FORMAT(22H          T          X)
```

All FORMAT statements are placed after the STOP command for the sake of neatness, but must precede the END statement. On some computers the STOP command may be optional after the last executable statement, but the END statement is always the last statement of every FORTRAN program.

The modified program now appears as follows:

```
      WRITE(6,30)
      T=0.
      X=1.
      V=0.
      D=.01
      P=.1-D
      WRITE(6,40)X,V,D
      WRITE(6,60)
   10 X=X+V*D
      V=V-X*D
      T=T+D
      IF(T-P)10,10,12
   12 WRITE(6,70)T,X
      P=P+.1
      IF(T-3.)10,10,16
   16 STOP
   30 FORMAT(24H1    HARMONIC OSCILLATOR,//)
   40 FORMAT(5H0  X=,F4.1,4H  V=,F4.1,4H  D=,F5.2,//)
   60 FORMAT(22H          T          X)
   70 FORMAT(F10.2,F12.4)
      END
```

Execution of this program by a computer produced these results:

```
          HARMONIC OSCILLATOR

        X= 1.0  V= 0.   D= 0.01

              T          X
           0.10       0.9955
           0.20       0.9811
           0.30       0.9568
           0.40       0.9230
           0.50       0.8800
           0.60       0.8282
           0.70       0.7681
           0.80       0.7003
           0.90       0.6255
           1.00       0.5445
           1.10       0.4580
           1.20       0.3670
           1.30       0.2723
           1.40       0.1749
           1.50       0.0757
           1.60      -0.0242
           1.70      -0.1239
           1.80      -0.2223
           1.90      -0.3186
           2.00      -0.4116
           2.10      -0.5005
           2.20      -0.5845
           2.30      -0.6626
           2.40      -0.7340
           2.50      -0.7982
           2.60      -0.8543
           2.70      -0.9019
           2.80      -0.9406
           2.90      -0.9698
           3.00      -0.9893
```

Exercise 8: Go through the calculation by hand far enough to convince yourself that output will only be printed when T = .1, .2, .3, etc. (Note that the branch back to the beginning of the computational loop occurs when T is less than or equal to P.)

What about other force laws? So far we have a program only for the harmonic oscillator force, but it is not difficult to see where the dependence on force enters the calculation. We used the harmonic oscillator force to replace the acceleration in the expression for computing the new velocity. By simply changing this one line, we can use the same program to compute the motion for a different one-dimensional force.

We would also like to change the initial conditions, because different initial conditions might lead to very different motions. Sometimes additional statements will be needed for specifying other constants, particularly when we want to explore a family of forces with different constants.

This is illustrated in the more realistic case of the damped oscillator. If you generate the harmonic oscillator solution for large values of time by testing against a larger time in the second IF statement, you will find that, according to the calculation, the oscillator always reaches the upper and lower limits, plus 1 and minus 1, during its cycle. But an actual mass on the end of a spring will eventually go less and less far, finally coming to rest in the equilibrium position. This is due to the frictional force of the air against the mass, which has been ignored in the computations so far. To a good approximation, this force is proportional to the velocity but always has the opposite sign because frictional forces always oppose the motion. Thus, the total force for such a system would be the sum of the Hooke's Law force and the frictional damping force:

$$F = -K*X - B*V \tag{8}$$

where B, the damping constant, is a positive quantity.

It is not difficult to modify our program to solve for the motion of a damped oscillator by changing the velocity calculation. With an argument similar to that used before (see Exercise 5) we can eliminate K and M by a change of units; if we replace B by C to indicate this change, the velocity equation becomes V = V - (X + C*V)*D, and the program for the damped oscillator then takes the form shown on the following page.

The expression for the new force has been used to calculate the velocity. Furthermore, note the additional line in the program defining the damping constant C = .5 and the heading which prints this value. Presumably it would be interesting to run this problem with many choices of damping constant, so that one can get a physical picture of how damping effects the motion. There are other ways to do this, but the easiest would be to change the statement defining the damping constant and to resubmit the program to the computer.

It should be apparent that the computer can be a powerful tool for solving problems in mechanics. Generalizations of many kinds are possible. We have used two force laws; it is not difficult to envision further extensions in that direction. Furthermore, the extension of these methods to two- and three-dimensional problems is straightforward. Because two coordinates are necessary for the specification of position and velocity, the solution of two-dimensional problems will require two equations, rather than the one equation used here. Details about such developments can be obtained in the references at the end of this material.

```
   WRITE(6,20)
   WRITE(6,30)
   T=0.
   X=1.
   V=0.
   D=.01
   P=.1-D
   C=.5
   WRITE(6,40)X,V,C
   WRITE(6,50)D,P
   WRITE(6,60)
10 X=X+V*D
   V=V-(X+C*V)*D
   T=T+D
   IF(T-P)10,10,12
12 WRITE(6,70)T,X
   P=P+.1
   IF(T-3.)10,10,16
16 STOP
20 FORMAT(17H1          DAMPED)
30 FORMAT(24H     HARMONIC OSCILLATOR,//)
40 FORMAT(5H   X=,F4.1,4H  V=,F4.1,4H  C=,F5.3)
50 FORMAT(8H      D=,F5.2,4H  P=,F5.2,//)
60 FORMAT(22H         T           X)
70 FORMAT(F10.2,F12.4)
   END
```

Thus, we have on hand the rudiments of a tool which will enable us to find out how any particle moves, and with the assistance of this tool we have gone far toward understanding how dynamical systems behave. However, we should not let the computer obscure the importance of the mathematics. The kinds of equations that we are solving by our basic numerical procedures are ones which are not simply ordinary algebraic equations. We can see this if we go back and write the two equations that we already have as definitions of velocity and acceleration, using the derivative notation: $V = dX/dT$ and $A = dV/dT$. These were the general equations. If we replace A by its value for a Hooke's Law force, now keeping the M and K, these two equations become

$$\frac{dX}{dT} = V \tag{9a}$$

$$\frac{dV}{dT} = -\left(\frac{K}{M}\right)X \tag{9b}$$

These equations contain derivatives, so they are called differential equations. What we have seen is one way of solving differential equations, a way particularly suitable for use with computers as it requires the manipulation of many numbers.

If you proceed further in your study of physics, mathematics or other sciences, you will find that the differential equation, handled often by methods quite different from those used here, is one of the most powerful mathematical tools of modern science.

REFERENCES

Bork, Alfred M., Fortran for Physics, Addison-Wesley, Cambridge, 1965.

Feynman, Richard P., Feynman Lectures on Physics, Addison-Wesley Cambridge, 1963, Chapter 9, Volume I.

Sawyer, Walter W., What is Calculus About?, Random House, New York, 1961, Chapters 2 and 3. (Also in Harvard Project Physics Reader 1.)

Sherwin, Chalmers W., Basic Concepts of Physics, Holt, Rinehart & Winston, New York, 1961.

DAY THREE

We have outlined a method for finding how a particle moves, assuming motion along a straight line. The method requires a description of the force acting on the particle and uses Newton's Second Law to find the resulting acceleration of the particle (Equation (4b)). Assuming initial values of position and velocity, it is possible to calculate later values of position and velocity. We completed the last section by showing how all of this information could be brought together in a computational procedure to determine the position of the particle at successive times.

In the present chapter, we shall examine a complete computer program for the calculation of successive values of the particle position and will explain how it contrives to accomplish this end. The program is written in JOSS, a commonly available language for use in systems having terminals to a computer. JOSS exists under other names--AID, ISIS, CAL, PIL, BRUIN, FOCAL, TELCOMP--and you may be using a dialect of JOSS in your class.

Here is the JOSS program for the harmonic oscillator problem (your instructor will show you how to input this information to the computer):

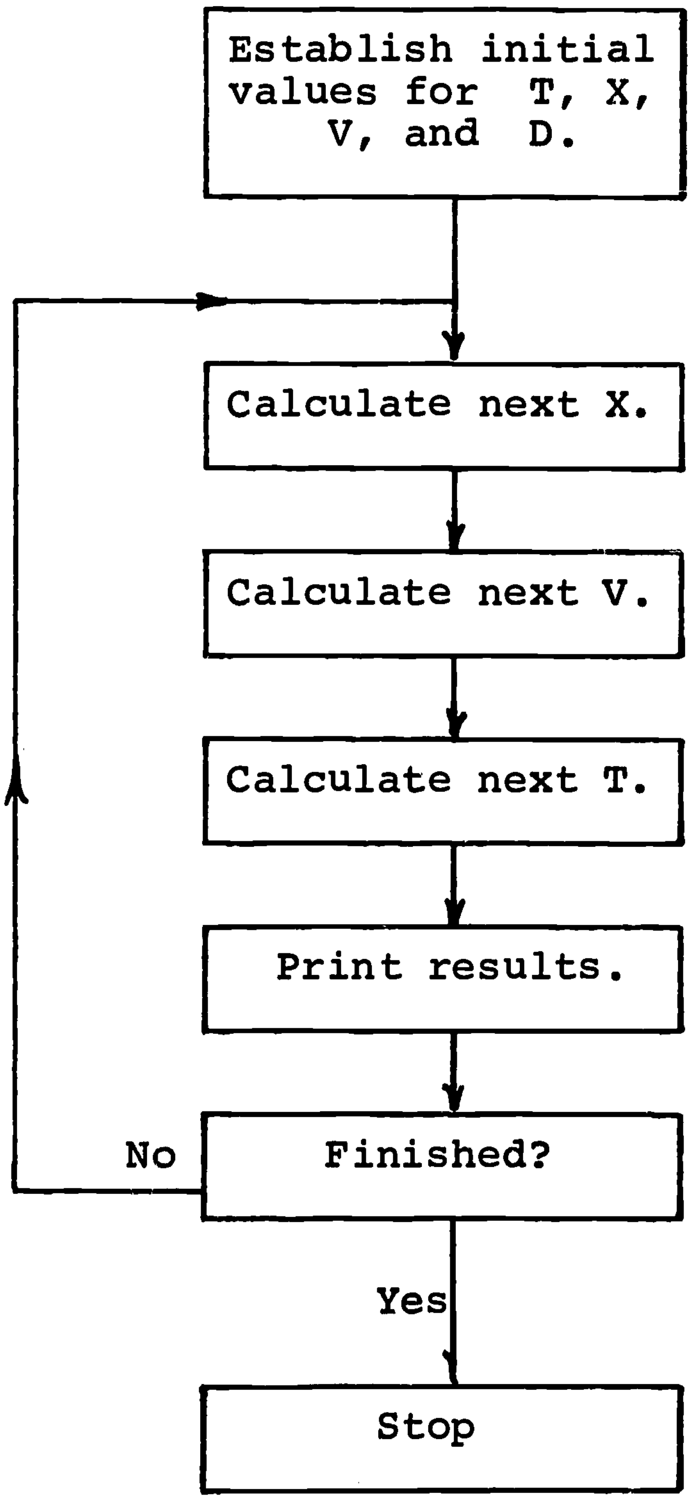
```
1.1    T = 0
1.2    X = 1
1.3    V = 0
1.4    D = .1
1.5    X = X + V*D
1.6    V = V - X*D
1.7    T = T + D
1.8    TYPE T, X
1.9    IF T<3, TO STEP 1.5
```

This is a complete computer program, with all the instructions the computer needs to carry out the calculation.

You should have no trouble recognizing the basic features of the calculation. The first four lines assign initial values to the time, T, position, X, velocity, V, and time interval, D. New values of the position, velocity, and time are calculated in the next three lines. You can recognize the two steps which calculate the new position and the new velocity, because the relations, except for the step numbers at the beginning, are those that you have already seen. The TYPE statement is a rather obvious instruction to type the time T and position X on the terminal typewriter.

Normally, in most computer languages, the program statements are done in the sequential order in which they appear. In a JOSS program, this order is the same as the numerical order of the step numbers which precede the statements. Note that each step number contains a decimal point. This whole part of the program is called part 1, because of the 1 occuring to the left of the decimal point in each statement. (More complex programs can have several parts.) The numbers after the decimal point designate the individual steps in part 1. Statement 1.9 is a branching statement which alters the flow of the calculation. Step 1.9 returns to 1.5, the next calculation of the new position, if time T is less than 3. However, after T gets sufficiently large, the branch is not made. Since there is no statement after this in the program, the calculation is terminated when T > 3.

The flow of computations can be represented in the following schematic flow chart:



```
1.1    T = 0
1.2    X = 1
1.3    V = 0
1.4    D = .1

1.5    X = X + V*D

1.6    V = V - X*D

1.7    T = T + D

1.8    TYPE T, X

1.9    IF T<3, TO STEP 1.5

       No more statements
```

You can recognize the previous flow chart (page 13) of the basic computational loop as part of this chart.

The following output is a sample of what this program produces when it is executed by the computer.

```
X = -0.6296916
T =  2.4
X = -0.7042915
T =  2.5
X = -0.7718485
T =  2.6
X = -0.831687
T =  2.7
X = -0.8832086
T =  2.8
X = -0.9258982
T =  2.9
X = -0.9593288
T =  3.0
X = -0.9831661
```

This is still a crude program, although it is sufficient for the basic calculation. It prints every value of time and position after it is calculated. In the present case this will only produce about thirty sets of values, so there is no great difficulty. We have already seen that our basic approximation only works when the time step D is small enough, but we do not know how small it should be. Therefore, we may want to change the time step, to .01 or even to .001, to explore the effect of smaller choices of D. However, our program becomes impractical if we change the time step to .001, because it would give us a hundred times as many lines of output as the present version.

One way to solve this problem is to generate printout only at specified "print-times", P, which may occur at intervals greater than D. We can do this by introducing an IF statement which bypasses the TYPE T, X command whenever T ≤ P. Furthermore, after each printing, P is immediately increased by some multiple of D, so that the next TYPE statement is executed only after a decent interval has elapsed.

For example, we may wish to use a time step only 1/10th as large as the previous one, but print out results only after every tenth computation, so the output will be similar to our previous program. We can provide for this by initializing D and P:

```
1.4    D = .01
1.41   P = .1 - D
```

comparing T to P with

```
1.71   IF T≤P, TO STEP 1.5
```

and, if T > P, executing the TYPE T, X statement and increasing P

```
1.81   P = P + .1
```

immediately after printing. Note that the step numbers for the new statements are intermediate between those already in the program; you only need type them at the terminal as the computer will insert them in the proper sequence before the modified program is executed.

This program still produces two columns of equations for output. Usually it is more convenient to obtain large amounts of output in tabular form. Furthermore, a program needs to have its own self-contained documentation; the output should identify the problem and state what parameters were used in the calculation. This is not just window dressing, but a useful part of programming. The TYPE statement can be used both to type headings and to type initial values, as follows:

```
1.02   TYPE "HARMONIC OSCILLATOR"
1.42   TYPE D, X, V, ""
1.44   TYPE "TIME    POSITION"
1.45   TYPE "----------------"
```

The TYPE statements containing quotes (") produce the headings; the extra pair of quotes in step 1.42 provides a blank line. Note that step 1.42 must follow the steps assigning values to these variables; one cannot print variables until they are defined. For tabular output, we need to add a FORM statement and a reference to this form in the TYPE statement:

```
1.8    TYPE IN FORM 1, T, X
FORM 1.

-.--    --.----
```

The form is a pictorial way of describing the output. It specifies both the location of X and T on the page and the number of decimal places which will be in each.

The modified program is given on the following page. Execution of this program, by a computer, will produce the results given on page 20J.

<u>Exercise 8</u>: Go through the calculation by hand far enough to convince yourself that output will only be printed when T = .1, .2, .3, etc. (Note that the branch back to the beginning of the computational loop occurs when T is less than <u>or</u> equal to P.)

```
1.02   TYPE "HARMONIC OSCILLATOR"
1.1    T = 0
1.2    X = 1
1.3    V = 0
1.4    D = .01
1.41   P = .1 - D
1.42   TYPE D, X, V, ""
1.44   TYPE "TIME     POSITION"
1.45   TYPE "----------------"
1.5    X = X + V*D
1.6    V = V - X*D
1.7    T = T + D
1.71   IF T≤P, TO STEP 1.5
1.8    TYPE IN FORM 1, T, X
1.81   P = P + .1
1.9    IF T<3, TO STEP 1.5

FORM 1.

-.--     --.----
```

```
HARMONIC OSCILLATOR
D =  0.01
X =  1.0
V =  0.0

TIME      POSITION
-----------------
0 10        0.9955
0.20        0.9810
0.30        0.9568
0.40        0.9230
0.50        0.8799
0.60        0.8281
0.70        0.7680
0.80        0.7002
0.90        0.6255
1.00        0.5445
1.10        0.4580
1.20        0.3670
1.30        0.2723
1.40        0.1748
1.50        0.0757
1.60       -0.0242
1.70       -0.1238
1.80       -0.2223
1.90       -0.3185
2.00       -0.4116
2.10       -0.5005
2.20       -0.5844
2.30       -0.6625
2.40       -0.7340
2.50       -0.7981
2.60       -0.8543
2.70       -0.9019
2.80       -0.9405
2.90       -0.9697
3.00       -0.9892
```

What about other force laws? So far we have a program only for the harmonic oscillator force, but it is not difficult to see where the dependence on force enters the calculation. We used the harmonic oscillator force to replace the acceleration in the expression for computing the new velocity (line 1.60 in our program). By simply changing this one line, we can use the same program to compute the motion for a different one-dimensional force.

We would also like to change the initial conditions, because different initial conditions might lead to very different motions. Sometimes additional statements will be needed for specifying other constants, particularly when we want to explore a family of forces with different constants.

This is illustrated in the more realistic case of the damped oscillator. If you generate the harmonic oscillator solution for large values of time by testing against a larger time in the second IF statement, you will find that, according to the calculation, the oscillator always reaches the upper and lower limits, plus 1 and minus 1, during its cycle. But an actual mass on the end of a spring will eventually go less and less far, finally coming to rest in the equilibrium position. This is due to the frictional force of the air against the mass, which has been ignored in the computations so far. To a good approximation, this force is proportional to the velocity, but always has the opposite sign because frictional forces always oppose the motion. Thus, the total force for such a system would be the sum of the Hooke's Law force and the frictional damping force:

$$F = -K*X - B*V \tag{8}$$

where B, the damping constant, is a positive quantity.

It is not difficult to modify our program to solve for the motion of a damped oscillator by changing the velocity calculation. With an argument similar to that used before (Exercise 5) we can eliminate K and M by a change of units; if we replace B by C to indicate this change, the velocity equation becomes V=V-(X+C*V)*D, and the program for the damped oscillator then takes the form shown on the next page.

The expression for the new force has been used to calculate the velocity. Furthermore, note the additional line in the program defining the damping constant C = .5. Presumably it would be interesting to run this problem with many choices of damping constant, so that one can get a physical picture of how damping effects the motion. This may be done by inserting a different value of C for each run.

It should be apparent that the computer can be a powerful tool for solving problems in mechanics. Generalizations of many kinds are possible. We have used two force laws; it is not difficult to envision further extensions in that direction. Furthermore, the

```
1.02   TYPE "DAMPED OSCILLATOR"
1.1    T = 0
1.2    X = 1
1.3    V = 0
1.32   C = .5
1.4    D = .01
1.41   P = .1 - D
1.42   TYPE D, C, X, V, ""
1.44   TYPE "TIME     POSITION"
1.45   TYPE "-----------------"
1.5    X = X + V*D
1.6    V = V - (X + C*V)*D
1.7    T = T + D
1.71   IF T≤P, TO STEP 1.5
1.8    TYPE IN FORM 1, T, X
1.81   P = P + .1
1.9    IT T<3, TO STEP 1.5

FORM 1.

-.--      --.----
```

extension of these methods to two- and three-dimensional problems is straightforward. Because two coordinates are necessary for the specification of position and velocity, the solution of two-dimensional problems will require two equations, rather than the one equation used here. Details about such developments can be obtained from the references at the end of this material.

Thus, we have on hand the rudiments of a tool which will enable us to find out how any particle moves, and with the assistance of this tool we have gone far toward understanding how dynamical systems behave. However, we should not let the computer obscure the importance of the mathematics. The kinds of equations that we are solving by our basic numerical procedures are ones which are not simply ordinary algebraic equations. We can see this if we go back and write the two equations that we already have as definitions of velocity and acceleration, using the derivative notation: $V = dX/dT$ and $A = dV/dT$. These were the general equations. If we replace A by its value for a Hooke's Law force, now keeping the M and K, these two equations become

$$\frac{dX}{dT} = V \tag{9a}$$

$$\frac{dV}{dT} = -\left(\frac{K}{M}\right)X \tag{9b}$$

These equations contain derivatives, so they are called differential equations. What we have seen is one way of solving differential equations, a way particularly suitable for use with computers as it requires the manipulation of many numbers.

If you proceed further in your study of physics, mathematics or other sciences, you will find that the differential equation, handled often by methods quite different from those used here, is one of the most powerful mathematical tools of modern science.

REFERENCES

Bork, Alfred M., Fortran for Physics, Addison-Wesley, Cambridge, 1965.

Feynman, Richard P., Feynman Lectures on Physics, Addison-Wesley, Cambridge, 1963, Chapter 9, Volume I.

Sawyer, Walter W., What is Calculus About?, Random House, New York, 1961, Chapters 2 and 3. (Also in Harvard Project Physics Reader 1.)

Sherwin, Chalmers W., Basic Concepts of Physics, Holt, Rinehart & Winston, New York, 1961.

DAY THREE

We have outlined a method for finding how a particle moves, assuming motion along a straight line. The method requires a description of the force acting on the particle and uses Newton's Second Law to find the resulting acceleration of the particle (Equation (4b)). Assuming initial values of position and velocity, it is possible to calculate later values of position and velocity. We completed the last section by showing how all of this information could be brought together in a computational procedure to determine the position of the particle at successive times.

In the present chapter, we shall examine a complete computer program for the calculation of successive values of the particle position and will explain how it contrives to accomplish this end. The program is written in PL/1, a commonly used computer language available in several forms.

Here is the PL/1 program for the harmonic oscillator problem (your instructor will show you how to input this information to the computer):

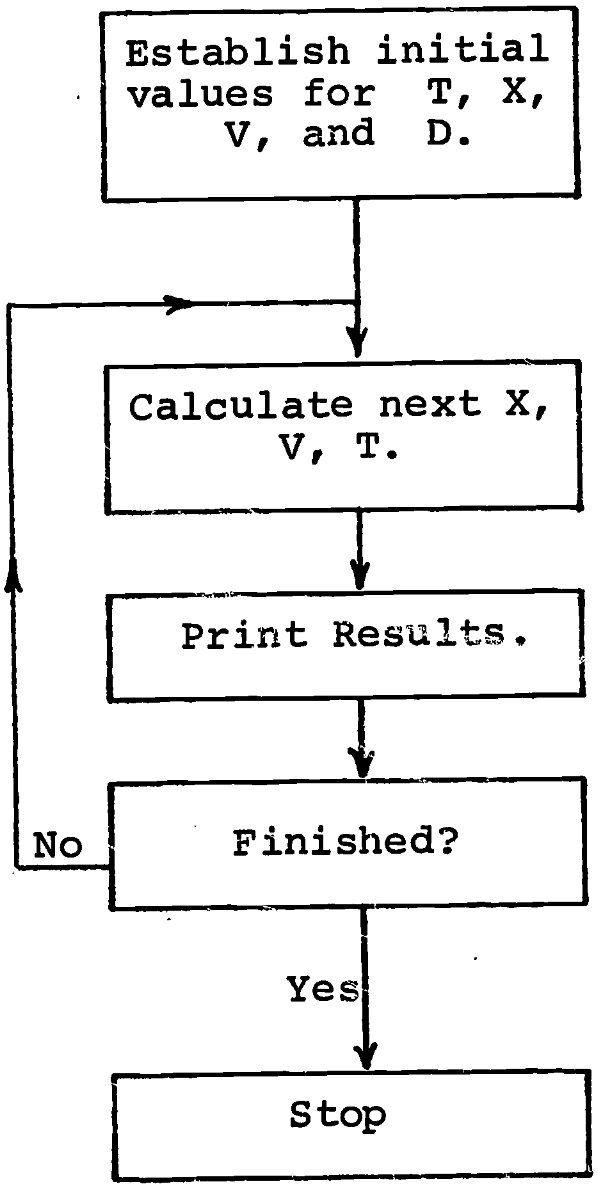
```
OSCILLATOR:   PROCEDURE OPTIONS (MAIN);
              T = 0; X = 1; V = 0; D = .1;
     CALCULATE:  X = X + V*D; V = V - X*D; T = T + D;
                 PUT SKIP DATA (T, X);
              IF T<3 THEN GO TO CALCULATE;
 END OSCILLATOR;
```

This is a complete computer program, with all the instructions the computer needs to carry out the calculation.

You should have no trouble recognizing the basic features of the calculation. The first line identifies the program as a "procedure", named OSCILLATOR, the name which identifies the program when stored in the computer. The next line assigns initial values to the time, T, position, X, velocity, V, and time interval, D. New values of the position, velocity, and time are calculated on the line labeled CALCULATE. The PUT statement instructs the computer to "skip" to a new line and print the time T and position X.

The line which begins with "IF", appropriately called an IF statement, is a "branching" instruction which can alter the normal flow of the calculation. The normal flow is simply the order in which the statements are presented. In the case of the IF statement, the sequence of operations can be altered depending on the truth or falsity of the expression appearing after the IF. If the expression if true, the next statement to be executed will be the statement designated by the identifier CALCULATE; if the expression is false, the computer executes the next statement which, in this case, ends the program. Thus, successive computations of the loop beginning with CALCULATE will be performed as long as $T \leq 3$; when T becomes larger than 3, the program is terminated. Note that all PL/1 statements end with semicolons, and that labels are followed by colons.

The flow of computations can be represented in the following schematic flow chart:



```
T = 0; X = 1; V = 0; D = .1;

CALCULATE:
 X = X + V*D; V = V - X*D; T = T + D;

PUT SKIP DATA (T, X);

IF T<3 THEN GO TO CALCULATE;

END OSCILLATOR;
```

You can recognize the previous flow chart (page 13) of the basic computational loop as part of this chart.

When executed by computer, this program produces the output given on the following page.

Note that the results are presented as small equations. This is due to the use of the word "DATA" in the PUT statement; later, we will see another type of output. The "E" in the numbers is a common computer way of saying "10 to the power"; thus, 2.3E+04 would mean $2.3 \times 10^4 = 23{,}000$.

17P

```
T= 9.99999E-02            X= 1.00000E+00;
T= 1.99999E-01            X= 9.90000E-01;
T= 2.99999E-01            X= 9.70099E-01;
T= 3.99999E-01            X= 9.40499E-01;
T= 4.99999E-01            X= 9.01493E-01;
T= 5.99999E-01            X= 8.53472E-01;
T= 6.99999E-01            X= 7.96916E-01;
T= 7.99999E-01            X= 7.32391E-01;
T= 8.99999E-01            X= 6.60543E-01;
T= 9.99999E-01            X= 5.82089E-01;
T= 1.09999E+00            X= 4.97814E-01;
T= 1.19999E+00            X= 4.08561E-01;
T= 1.29999E+00            X= 3.15222E-01;
T= 1.39999E+00            X= 2.18731E-01;
T= 1.49999E+00            X= 1.20053E-01;
T= 1.59999E+00            X= 2.01745E-01;
T= 1.69999E+00            X=-7.99059E-01;
T= 1.79999E+00            X=-1.79187E-01;
T= 1.89999E+00            X=-2.76676E-01;
T= 1.99999E+00            X=-3.71399E-01;
T= 2.09999E+00            X=-4.62408E-01;
T= 2.19999E+00            X=-5.48792E-01;
T= 2.29999E+00            X=-6.29689E-01;
T= 2.39999E+00            X=-7.04289E-01;
T= 2.49999E+00            X=-7.71846E-01;
T= 2.59999E+00            X=-8.31684E-01;
T= 2.69999E+00            X=-8.83206E-01;
T= 2.79998E+00            X=-9.25896E-01;
T= 2.89998E+00            X=-9.59326E-01;
T= 2.99998E+00            X=-9.83164E-01;
T= 3.09998E+00            X=-9.97169E-01;
```

One surprising aspect is that the times are not just .1, .2, .3, etc., as you might expect from the program. This is due to the fact that internal computations are usually performed in binary, or base-two, arithmetic and the conversion from binary to decimal, base-ten, numbers is often not quite exact. While PL/1 does not automatically round numbers, it is possible, in some PL/1 implementations, to avoid problems arising from binary-to-decimal conversion by providing for rounding in the program.

This is still a crude program, although it is sufficient for the basic calculation. It prints every value of time and position after it is calculated. In the present case, this will only produce about thirty sets of values, so there is no great difficulty. We have already seen that our basic approximation only works when the time step D is small enough, but we do not know how small it should be. Therefore, we may want to change the time step, to .01 or even .001, to explore the effect of smaller choices of D. However, our program becomes impractical if we change the time step to .001, because this would yield one hundred times as many lines of output as the present version.

One way to solve this problem is to generate printout only at specified "print-times", P, which may occur at intervals greater than D. We can do this by introducing an IF statement which bypasses printing (T, X) except when T > P. Furthermore, after each printing P is immediately increased by some multiple of D so that the PUT statement is executed again only after a decent interval has elapsed.

For example, we may wish to use a time step only 1/10th as large as the previous one, but print out results only after every tenth computation, so the output will be similar to our previous program. We can provide for this by initializing D and P:

```
T = 0; X = 1; V = 0; D = .01;

P = .1 - D;
```

comparing T to P with

```
IF T<=P THEN GO TO CALCULATE;
```

and, if the expression is false and T > P, executing the PUT statement and increasing P

```
P = P + .1;
```

immediately after printing.

This program still produces two columns of equations for output. Usually, it is more convenient to obtain large amounts of output in tabular form. Furthermore, a program needs to have its own self-contained documentation; the output should identify the problem and state what parameters were used in the calculation. This is not just window dressing, but a useful part of programming. The PUT statement can be used both to type a heading and to type initial values, as follows:

```
PUT PAGE LIST (' HARMONIC OSCILLATOR ');
PUT SKIP DATA (D, X, V);
PUT SKIP;
PUT SKIP LIST ('TIME     POSITION');
PUT SKIP LIST ( (20) '-' );
```

In PL/1, "SKIP" indicates that a new line is to be started and "PAGE" denotes the beginning of a new page. Note that to write 20 dashes we need only use a repetition factor. For tabular output, however, we need to alter the PUT statement:

```
PUT EDIT (T, X) (SKIP, F(5,2), X(5), F(10,6) );
```

This command instructs the computer to skip to a new line, allow five spaces for the value of T, and then to print the value of T with two places to the right of the decimal. The X(5) specification requests that the computer leave five blanks and then print X to six decimal places in a ten-space field.

Adding these statements gives the final form of the harmonic oscillator program which appears as follows:

```
OSCILLATOR:    PROCEDURE OPTIONS (MAIN);
               T = 0; X = 1; V = 0; D = .01;
               P = .1 - D;
               PUT PAGE LIST (' HARMONIC OSCILLATOR ');
               PUT SKIP DATA (D, X, V);
               PUT SKIP;
               PUT SKIP LIST ('TIME     POSITION');
               PUT SKIP LIST ( (20) '-' );
     CALCULATE:  X = X + V*D; V = V - X*D; T = T + D;
                 IF T<=P THEN GO TO CALCULATE;
                 P = P + .1;
                 PUT EDIT (T, X) (SKIP, F(5,2), X(5), F(10,6) );
               IF T<3 THEN GO TO CALCULATE;
 END OSCILLATOR;
```

Execution by a computer produced the results shown on the following page, which are rounded by the EDIT command before printing.

<u>Exercise 8</u>: Go through the calculation by hand far enough to convince yourself that output will only be printed when T = .1, .2, .3, etc. (Note that the branch back to the beginning of the computational loop CALCULATE occurs when T is less than, *or* equal to, P.)

20P

```
 HARMONIC OSCILLATOR
D= 9.99999E-03              X= 1.00000E+00              V= 0.00000E+00;

TIME      POSITION
--------------------
 0.10         0.995503
 0.20         0.981059
 0.30         0.956813
 0.40         0.923007
 0.50         0.879978
 0.60         0.828157
 0.70         0.768061
 0.80         0.700290
 0.90         0.625523
 1.00         0.544506
 1.10         0.458048
 1.20         0.367013
 1.30         0.272312
 1.40         0.174889
 1.50         0.075719
 1.60        -0.024207
 1.70        -0.123891
 1.80        -0.222337
 1.90        -0.318562
 2.00        -0.411603
 2.10        -0.500532
 2.20        -0.584460
 2.30        -0.662548
 2.40        -0.734016
 2.50        -0.798149
 2.60        -0.854308
 2.70        -0.901931
 2.80        -0.940541
 2.90        -0.969754
 3.00        -0.989278
 3.10        -0.998917
```

What about other force laws? So far we have a program only for the harmonic oscillator force, but it is not difficult to see where the dependence on force enters the calculation. We used the harmonic oscillator force to replace the acceleration in the expression for computing the new velocity. By simply changing this one line, we can use the same program to compute the motion for a different one-dimensional force.

We would also like to change the initial conditions, because different initial conditions might lead to very different motions. Sometimes additional statements will also be needed for specifying other constants, particularly when we want to explore a family of forces with different constants.

This is illustrated in the more realistic case of the damped oscillator. If you generate the harmonic oscillator solution for large values of time by testing against a larger time in the second IF statement, you will find that, according to the calculation, the oscillator always reaches the upper and lower limits, plus 1 and minus 1, during its cycle. But an actual mass on the end of a spring will eventually go less and less far, finally coming to rest in the equilibrium position. This is due to the frictional force of the air against the mass, which has been ignored in the computations so far. To a good approximation, this force is proportional to the velocity, but always has the opposite sign because frictional forces always oppose the motion. Thus, the total force for such a system would be the sum of the Hooke's Law force and the frictional damping force:

$$F = -K*X - B*V \qquad (8)$$

where B, the damping constant, is a positive quantity.

It is not difficult to modify our program to solve for the motion of the damped oscillator by changing the velocity calculation. With an argument similar to that used before (see Exercise 5) we can eliminate K and M by a change of units; if we replace B by C to indicate this change, the velocity equation becomes V = V - (X + C*V)*D, and the program for the damped oscillator takes the form shown on the following page.

The expression for the new force has been used to calculate the velocity.and the additional line in the program defines the damping constant C = .5. It might be interesting to run this problem with many choices of damping constant, so that one can get a physical picture of how damping effects the motion. This may be done by inserting a different value of C before each run.

It should be apparent that the computer can be a powerful tool for solving problems in mechanics. Generalizations of many kinds are possible. We have used two force laws; it is not difficult to envision further extensions in that direction. Furthermore, the extension of these methods to two- and three-dimensional problems is

```
OSCILLATOR:   PROCEDURE OPTIONS (MAIN);
              T = 0; X = 1; V = 0; D = .01;
              P = .1 - D;
              C = .5;
              PUT PAGE LIST (' DAMPED OSCILLATOR ');
              PUT SKIP DATA (C, D, X, V);
              PUT SKIP;
              PUT SKIP LIST ('TIME     POSITION');
              PUT SKIP LIST ( (20) '-' );
     CALCULATE:  X = X + V*D; V = V -  (X + C*V )*D; T = T + D;
                 IF T<=P THEN GO TO CALCULATE;
                 P = P + .1;
                 PUT EDIT (T, X) (SKIP, F(5,2), X(5), F(10,6) );
              IF T<3 THEN GO TO CALCULATE;
 END OSCILLATOR;
```

straightforward. Because two coordinates are necessary for the specification of position and velocity, the solution of two-dimensional problems will require two equations, rather than the one equation used here. Details about such developments can be obtained in the references at the end of this material.

Thus, we have on hand, the rudiments of a tool which will enable us to find out how any particle moves, and with the assistance of this tool, we have gone far toward understanding how dynamical systems behave. However, we should not let the computer obscure the importance of the mathematics. The kinds of equations that we are solving by our basic numerical procedures are ones which are not simply ordinary algebraic equations. We can see this if we go back and write the two equations that we already have as definitions of velocity and acceleration, using the derivative notation: V = dX/dT and A = dV/dT. These were the general equations. If we replace A by its value for a Hooke's Law force, now keeping the M and K, these two equations become

$$\frac{dX}{dT} = V \qquad (9a)$$

$$\frac{dV}{dT} = -\left(\frac{K}{M}\right)X \qquad (9b)$$

These equations contain derivatives, so they are called differential equations. What we have seen is one way of solving differential equations a way particularly suitable for use with computers, as it requires the manipulation of many numbers.

If you proceed further in your study of physics, mathematics or other sciences, you will find that the differential equation, handled often by methods quite different from those used here, is one of the most powerful mathematical tools of modern science.

REFERENCES

Bork, Alfred M., Fortran for Physics, Addison-Wesley, Cambridge, 1965.

Feynman, Richard P., Feynman Lectures on Physics, Addison-Wesley, Cambridge, 1963, Chapter 9, Volume I.

Sawyer, Walter W., What is Calculus About?, Random House, New York, 1961, Chapters 2 and 3. (Also in Harvard Project Physics Reader 1.)

Sherwin, Chalmers W., Basic Concepts of Physics, Holt, Rinehart & Winston, New York, 1961.

ADDITIONAL PROBLEMS

The following problems go beyond the material you have had here, suggesting further directions for the student who wishes to continue to use these techniques.

Problem 1:

Recall that the position calculation is based on knowing the velocity and that the velocity calculation is based on knowing the acceleration. You may have wondered why we stopped at the velocity calculation in our computer program. Why didn't we compute the acceleration based on knowing another quantity?

a) Write such an expression, calling the new quantity Q.

b) Suppose Q is known as a function of X. To be specific, let $Q = -X^3$. Write a computer program to calculate X at successive times. What variables must be assigned initial values to start the calculation?

c) Show that in the limit of small time intervals $Q = \frac{dA}{dT}$, the rate of change of acceleration.

This problem may suggest that the mechanical method developed here can be generalized to solve more complicated differential equations. We stopped with the calculation of velocity because Newtonian mechanics assumes that it is sufficient to know the acceleration of a body in order to be able to predict its motion.

Problem 2:

Let us add a <u>third</u> force to the damped harmonic oscillator problem, independent of the Hooke's Law force and the frictional damping force. It is a function of time only and varies sinusoidally. The net force on the particle becomes

$$F = -KX - BV + G\sin(HT)$$

a) Show that the change in velocity can be represented by

$$\Delta V = (-(K/M)X - (B/M)V + (G/M)\sin(HT))D$$

and that, for a particular change in the time unit, this equation becomes

$$\Delta V' = (-X - CV' + F\sin(ET'))D'$$

Find the relations between the "new" variables V' and T' and the "old" variables V and T. Find the relations between the "new" constants C, F, E, and D' and the "old" constants M, K, B, G, H, and D.

b) Modify the "DAMPED HARMONIC OSCILLATOR" program so that it will describe the motion of an object acted on by these three forces. How many values must you specify in the beginning of your program? Explore the nature of the motion of the object for several choices of the initial values.

Problem 3:

In order to limit the printing of output for the harmonic oscillator program we defined a variable P and set it equal to .1-D. We printed values of T and X only when T exceeded P and then we incremented P by .1.

a) Show that logically we could accomplish the same purpose if we originally set P = .1 and printed values of T and X when T was either equal to P or exceeded P.

In the same way that one-third is a repeating decimal fraction, one-hundredth is a repeating fraction in the binary number system used by most computers. Obviously, a repeating fraction cannot be represented exactly if one is restricted to a finite number of significant figures. Thus when we write .01 the computer usually approximates the number by truncating the infinite binary fraction beyond some position characteristic of each particular computer. If this happens, the number in the computer is slightly less than .01. How will this situation affect the results of the logically equivalent procedures described above?

Problem 4:

Suppose that a particle is attracted to a fixed point by a Hooke's Law force, and that it is now free to move in a plane containing this point. Its position can be specified by coordinates along two perpendicular axes. Call these position coordinates X1 and X2, and call the corresponding components of the velocity and acceleration V1 and V2 and A1 and A2. Each pair of components can be thought of as a two-dimensional vector.

a) Set up algebraic equations analogous to Equations (1) and (2) for calculating the components of the new position vector (i.e., X1 and X2) from a knowledge of the velocity vector and the components of the new velocity vector from the acceleration vector.

b) The force must now also be treated as a vector with two components. Write equations for the two components F1 and F2, of the Hooke's Law force in terms of X1 and X2.

c) Make changes in the "HARMONIC OSCILLATOR" program so that it will compute the motion in this two-dimensional problem.

d) Investigate the effect of various choices of initial values.

Problem 5:

Another interesting problem is the motion of a planet about the sun. The force of gravity is always directed toward the sun and its magnitude is inversely proportional to the square of the separation distance. It can be shown that the motion always lies in a plane; thus we have another two-dimensional problem.

a) Write equations for the two components of the force on the planet in terms of its position components, X1 and X2. (Check to make certain that your equations satisfy the above conditions.)

b) Make changes in the program of Problem 4(c) so that it will solve the gravitational problem.

Problem 6:

You have probably already encountered the definitions of work and energy in your course. The work done by a force, in a one-dimensional problem, is just the product of the force and the distance through which the body moves. If the force varies during the motion, the total work done may be calculated by subdividing the motion into steps sufficiently small that the force does not change much during a step. Then the work done in a given step can be calculated by multiplying the current value of the force (-K*X in the harmonic oscillator problem) by the distance moved during that step (V*D). The result should be added to the work accumulated up to the beginning of the new step.

a) Write an expression for the work done by the Hooke's Law force in the harmonic oscillator problem. Perform the same transformation of coordinates which was discussed in DAY TWO and used in Problem 2. You should find that you cannot eliminate both K and M from the expression for the work; in fact, you should find a multiplicative factor of K remaining. Divide by K to obtain what could be called "the work per unit K".

b) Insert into the "HARMONIC OSCILLATOR" program the steps needed for the calculations of the work per unit K and to print it each time T and X are printed.

c) If you run this program, try to discover the relationship between W and X. (Hint: compare W with various powers of X, or look at logarithms of W and X.)

Problem 7:

The kinetic energy of a particle is defined to be equal to $\frac{1}{2}MV^2$. By the same change of units described earlier we may set $M = 1$ and represent the scaled kinetic energy by $\frac{1}{2}V^2$.

a) Alter the program of Problem 6 so that it prints the value of the kinetic energy each time T, X, and W are printed.

b) If you run this program, compare the work done by the force with the kinetic energy of the particle. You should try to relate your results to the corresponding discussion in your textbook.

* * * *

We would appreciate any comments that you might have for improving this material. Such comments should be sent to the authors, at their home institutions, or directed to the:

Commission on College Physics
Department of Physics & Astronomy
University of Maryland
4321 Hartwick Road
College Park, Maryland 20740

TEACHER'S GUIDE

INTRODUCTORY COMPUTER-BASED MECHANICS

A One Week Sample Course

INTRODUCTION

"Introductory Computer-Based Mechanics" is designed to be used in any introductory physics course either for majors or non-majors. This guide will suggest various ways to use the material with students; other possibilities will occur to you. In order to be as specific and detailed as possible, we have designed these notes and the student material with a particular course organization in mind: we assume that three lectures (more precisely, three reading assignments) and one laboratory period will be available. Whether you discuss this material in lectures will depend upon the background of your students and your general teaching strategy. For best results your students should have the use of a time-sharing system; a batch processing system with short turnaround time (5 to 20 minutes) is also effective. Batch processing with longer turnaround time will pose the most problems to you and your students. The materials herein could be used without any computer if necessary, and this might be advisable if your local computer has a long turnaround time. You will have to decide when to insert this material into your course, but in most instances it will probably fit best during or immediately after the study of dynamics.

The students should be familiar with velocity and acceleration and with Newton's Second Law of motion. Since the basic system studied is the harmonic oscillator, some familiarity with the Hooke's Law force will be desirable, although thorough understanding of oscillatory motion is in no way prerequisite. No previous exposure to the ideas of calculus is necessary although some understanding of simple differentiation will allow some interesting extensions of this work.

The first two lectures, Day One and Day Two, are independent of any computer or programming language. Day Three is available in any one of four common programming languages: PL/1, JOSS, BASIC, and FORTRAN. If your school has BASIC or JOSS available, it is likely that you will be working through teletype terminals on a time sharing system. However, most FORTRAN and PL/1 systems currently in use in colleges and universities are batch processing systems. A later section of the Teacher's Guide discusses these details further.

Our aim is to teach as much mechanics as possible to students who have no knowledge of calculus and differential equations. In particular we would like to expose them to important material which is generally omitted from elementary classical mechanics, because the students lack analytic or computational capability (note the distinction--the aim of this paper is to show how the latter can be used in place of, or complementary to, the former). We have found it very difficult to introduce differential equations into the initial mechanics course without using numerical and/or computer techniques. In fact, we aver that this material simply cannot be taught to most students using the conventional approach. Yet the rewards of early and gainful use of differential equations in the study of mechanics are so great that we feel warranted in introducing the computer at this stage in the conviction that it will make a viable contribution to the teaching of physics.

It is our intent to complement rather than supplant the use of analytic methods in physics. Undoubtedly fundamental physics will still be heavily dependent on analytic procedures, and even applied physics will best be done by a combination of analytic and numerical methods. Both approaches will be needed and increasingly more problems may be resolvable only by a combination of them, to be thought of as complementary methods for the solution of physical problems. For the future scientist, as well as for students in other areas, direct contact with computers in the context of his studies is very valuable, and we feel that this should begin at the earliest possible moment in his education. If computers are to be used widely, students must grow up with a sophisticated attitude toward them.

We wish to stress that a numerical method, far from being a gimmick, is as respectable an approach to a problem as an analytic method. A procedure, an algorithm, is a solution to a physical problem, just as an analytic method is a solution to the problem; it simply uses different techniques. From the research point of view, no scientist can afford to neglect any available technique, and those used here are of major importance today.

There is yet another strong reason for using computers within physics courses. Most students find computers exciting, particularly when they themselves are actively involved in constructing and running programs. This pedagogical incentive cannot be ignored, especially during a time of declining physics enrollments.

We do not expect the teacher to instruct the student in the computer language in which his programs are written. The languages, as used here, are quite straightforward; the entire thrust of this treatment is to demonstrate that the details of the language are subordinate to the physical problem and will be learned in the process of solving it.

OUTLINE OF STUDENT MATERIAL

The student material is intended to be self-contained and should provide the student with all the information he will need. The following outline of the three days' work may be useful to the teacher.

Day One - Outline

Motion at constant velocity--brief discussion.

Review of relation for finding new position.

$$X_{new} = X_{old} + V \cdot D \qquad (1a)$$

Numerical example.

Recursive use of this relation for step-by-step iteration calculation of X_{new}, V constant.

Numerical integration, for the case of non-uniform velocity.

Exercise 1: Check values of position.

Numerical integration of acceleration to obtain velocities.

$$V_{new} = V_{old} + A \cdot D \qquad (1b)$$

Beginning with a table of acceleration vs. time, combine the above equations to compute velocity and position as a function of time.

Exercise 2: Perform the indicated computations.

The concept of limit and taking derivatives.

Day Two - Outline

Brief review of Day One.

Discussion of interval size, D. Velocity and acceleration may be changing during the interval. Heuristic argument for reducing error by reducing D.

(Optional: Discuss possible ways to improve the situation.

1) Feynman's trick of the initial half-step calculation of velocity. (See Feynman or Bork in reference.)

2) Average velocities.

3) More accurate approximations to derivatives.)

Exercise 3: Re-do Exercise 1, using V at the <u>end</u> of each interval in Equation (1a).

Translation of recursive relations for X and V (Equations (1a) and (1b)) into computer language; discussion of meaning of the resultant (2a) and (2b) in terms of calculation and storage of values of X, V.

$$X_{new} = X_{old} + V \cdot D \quad \text{becomes} \quad X = X + V*D \qquad (2a)$$

$$V_{new} = V_{old} + A \cdot D \quad \text{becomes} \quad V = V + A*D \qquad (2b)$$

Calculation of elapsed time: $T = T + D$ (3)

Newton's Second Law of motion gives us acceleration:

$$F = M \cdot A \qquad (4a)$$

$$A = F/M \qquad (4b)$$

Example: Hooke's Law force (simple harmonic oscillator)

$$F = -K \cdot X \qquad (5)$$

Harmonic oscillator version of Equation (2b) using Equation (4a):

$$V = V - (K/M)*X*D \qquad (6a)$$

Exercise 4: Find dimensions of K, K/M.

Exercise 5: Eliminate mass and spring constant, K, by proper choice of units.

Exercise 6: Effect of increase in K on the motion.

Flow chart presentation of computation of harmonic oscillator motion, using dimensionless Equations (2a), (3), and (6b).

$$V = V - X*D \qquad (6b)$$

Exercise 7: Hand calculation for harmonic oscillator.

Day Three - Outline
(Computer language dependent--four versions.)

Brief review of Day Two.

Full computer program for the harmonic oscillator. (No print control as yet.) D and initial conditions fixed in the program.

Line-by-line explanation of what is happening in the program flow chart.

Computer language details introduced and explained as needed.

Expanded program with print control.

(Optional: Half-step calculation.)

Exercise 8: Verify that program only prints results in time intervals of one-tenth seconds.

Changing the time-step.

What about other forces?

Damped ocillator--complete program, allowing damping constant and initial conditions to be altered.

Discussion of differential equations of oscillator; generalization to two- and three-dimensional problems.

(Optional: Pursue one or more such problems.)

(Optional: Class speculation, based on knowledge of computer solution, as to the analytic solution. Verfication, if possible.)

Brief discussion of laboratory session.

LABORATORY SESSION

No student notes are provided for the laboratory material, because these will vary according to local conditions. It is assumed the instructor or a capable assistant will be present during the laboratory sessions.

In the laboratory the student is to run the harmonic oscillator and the damped oscillator programs, with differing values of the parameters C, D. initial X and/or initial V, and to write a lab report based on the results. In particular, the harmonic oscillator program should be run for various choices of the time step D. Since the student notes give no method for determining error, only "trial-and-error" techniques can be applied. It should prove interesting to note that for the first order numerical approximations used here (see Equations (1a) and (1b)), the errors in X and V are proportional to D^2, until "roundoff" errors begin to accumulate for very small D. (It is unwise to let students waste computer time by taking D too small; place an appropriate limit on the size of D beforehand, for example, $D > 10^{-4}$.)

When the harmonic oscillator program is run for many different time steps, and the results are compared with the exact solution, the cosine function, one discovers that there is an optimal value of the time step (dependent on the computer and other factors) which minimizes the error in the computation. This discovery may come as a surprise to the student, because the discussion in the text stresses only that the time step must be sufficiently small for the approximation to be useful. If the only error involved were the "truncation" error of the approximation of the differential equation by a difference equation, smaller time steps would always yield better answers. However, there is another source of error in all computational work, whether done by computer or by hand. Only a certain number of significant figures can be stored in a computer or carried along in a practical hand computation. The part of the number that is thrown away in each calculation leads to roundoff errors which accumulate with the number of calculations. Making the time step smaller and smaller increases the number of calculations needed to arrive at the final value of the time. Eventually, for sufficiently small time steps, roundoff becomes the major source of errors.

Many computer systems use both standard and extended precision; that is, the user can choose the number of significant figures to be retained in each computation. If you have this choice with your system, the student can get a better view of the interaction of the two kinds of errors discussed above by also using extended precision. The roundoff error of a single arithmetic operation is roughly on the order of one-half in the last significant digit.

The student should be assigned the problem of determining how the results and errors depend on the choice of the time step. His lab report should contain this analysis.

In running the damped oscillator problem on the computer, the student should be directed toward the following physical problem: how does the choice of damping constant affect the motion? The student should be able to discern, by himself, two types of motion, underdamped and overdamped; and he should be able to determine the range of values of the damping constant C for each type. The student may also be able to discover how the choice of the damping constant interacts with other aspects of the motion, such as the frequency.

How much freedom the student can be allowed in running the program for different values of the damping constant may depend on the computer facilities available. Ideally one would like the student to run enough cases to enable him to understand the effects of damping on the oscillatory motion. However, if computer time is dear or scarce the teacher may have to indicate the range of values to use for C in order to display all the significant physical effects. In our case a range of $0 \leq C \leq 3$ is appropriate.

If the students have written any further programs, following the suggestions in the notes, these should be run during the laboratory session.

Language Variation

Although the programs are given in four languages, FORTRAN, PL/1, JOSS and BASIC, language implementations are not completely computer-independent, and each system generally has its own quirks, so that minor adjustments may be necessary in these programs. This variation is particularly noticeable with FORTRAN. If you are not familiar with the operation of your local system, you should ask someone in your computation center to look over the programs before you attempt to run them.

Mechanism of the Laboratory Session - Batch Processing

Many university computing centers, particularly those in small schools and those oriented strongly toward research uses, will have only batch processing computer systems, where jobs are entered (usually from punched cards) at a central computer facility. The details vary considerably. If your computer is a small computer it will probably be possible, and desirable, to bring the students into the computer room for the laboratory session, particularly if key punches, etc., are available nearby and the students can work directly with the operator in submitting programs to the computer. If you can reserve the computer room for this period, so that no other jobs are running, this student-computer exchange becomes a very interactive one, generating much excitement on the part of the students. Whether you can negotiate such an arrangement will depend on your individual computer center. It has been done!

With larger batch processing computers the chances are slight that you will be able to take the students into the computer room for long periods of time, although with a very cooperative computer center director this may still be manageable. In such a center there will be a place to put in programs and a place for output. The details of how the laboratory can be conducted will depend heavily on the turnaround time, the time between submission and return of programs. This time can vary anywhere from a few minutes to days.

If your school has a fast batch system or a fast remote job entry system the turnaround time may only be a few minutes, and students can conveniently work in a room near the input-output point. Using keypunches the student can modify and resubmit programs immediately on the basis of the output received. If, on the other hand, the turnaround time is longer than the laboratory period, students will be unable to make modifications during the lab period. Under these circumstances the "laboratory session" could be spread over a couple of weeks, perhaps using the first twenty minutes of subsequent conventional laboratories to enable the student to retrieve the programs he submitted previously and to make necessary changes and resubmit them before beginning the regular laboratory session. This scheme is somewhat more clumsy than the others discussed here, but is still viable.

Another possibility would be for the student to start the work in the laboratory, and then to submit programs as needed; homework could be reduced during the period to give time for this work.

Mechanisms of the Laboratory Session - Time Shared System

The laboratory session will be run quite differently if your school has a teletype terminal system, or has access to someone else's time sharing system. Here again there are several possibilities. First, many terminals may be available, perhaps in a common room. This room then can hopefully be reserved for the laboratory time. Students can work at the terminals either individually or in small groups. Opinions differ as to which is better. When he works individually, the student is actively involved. On the other hand, student interaction in small groups can be a valuable teaching device. It is not convenient for more than three or four students to use one terminal. Under these group circumstances you should try to see that everyone gets to use the system.

If only a single terminal is available the situation becomes similar to batch processing. A teletype terminal which punches and reads paper tape can be used off-line (not connected to the computer) to punch the program. Then students can run their programs at the terminal from the paper tape. If each student must type in on-line to the computer, it may be necessary to split the lab as suggested above.

One advantage of the terminal, even a single terminal, is that one usually has an on-line editing system for correcting program errors. This would be particularly useful if any student-written programs are run. The details differ from one system to another.

If you are not a user of the computer system, it might be desirable to have someone familiar with it on hand during the laboratory session. Perhaps one of your departmental members would be able to offer such assistance.

Connections with Other Laboratory Work

Students can be prepared for the numerical analysis of the harmonic oscillator by first taking measurements of a mass on a spring in an earlier laboratory, and then comparing the data with the results of our analysis. One of several ways in which this can be done is as follows: a small blinky (relaxation oscillator), flashing at about 30 or 40 times a second, is hung on a spring. A Polaroid camera photographs this system. As the blink goes from the lowest to the highest point; it is photographed using a Polaroid camera with the shutter open. The result is a series of dots on the film, recording the position of the blinky at equal time intervals. Stroboscopic lights will produce similar results. Then the Polaroid film can be used as a slide by making small pin holes at each dot and projecting the picture onto graph paper. The student can then obtain fairly reliable X-T data and can plot these experimental points on an X-T graph.

After the harmonic oscillator problem is run on the computer, the experimental points can be compared with those obtained by measurement. When the measured and calculated points are plotted together, with proper scaling, the results are impressive.

Blinkies can be obtained from Holt, Rinehart and Winston, Damon Industries, and Ealing Corporation, or can be constructed locally.

FURTHER WORK AND ADDITIONAL MATERIAL

Further Extensions

For students who may want to go further, there are some obvious extensions. First, students can be encouraged to study other one-dimensional force laws. Furthermore, there are many interesting two-dimensional problems. Treatments exist in the literature for the one-particle gravitational problem, both analytically and as a computer problem. Interesting insight can be gained by students in finding when open and closed orbits occur. Feynman considers only a closed orbit case. A student running on the computer with different initial conditions may discover that the orbits are not always closed. This discovery can be effective pedagogically. But how does one distinguish between the open and closed orbits? The program can be constructed so as to lead the student toward the answer; it can print the total energy for each set of initial conditions. Students can relate open and closed orbits, and positive and negative values of the energy. Energy considerations can also be raised in other problems; with the oscillator, energy conservation can be shown dynamically.

Other two-dimensional problems are possible; for example, a particle moving in a constant magnetic field. The numerical solutions may, by displaying symmetries or recognizable mathematical properties, suggest how to approach the analytic solution of the differential equations, particularly if the student has some familiarity with the differentiation of sine and cosine functions.

These extensions of the present material are concerned with mechanics. Similar tactics can be used in other areas of beginning or intermediate physics courses. Generally, any area in which ordinary differential equations are important is a candidate for effective use of the computer, particularly in cases where the student does not have the analytic ability to handle differential equations. Thus, Sherwin (see References) discusses quantum mechanics numerically. He does not directly refer to computers, but several of us have had satisfactory experience using computers with the type of approach he outlines. Although most students at this stage do not have the analytical abilities required to handle differential equations, the basic ideas can be mastered by even relatively unsophisticated students. What is needed is the capability to play with solutions of the Schrödinger equation, capability available through computers. (For an example, see A. Luehrmann, The Square Well in Quantum Mechanics, Am. J. Phys. 35, 275 (1967).)

Other Materials

You may find computer-produced films useful in connection with this material. The most widely available is Frank Sinden's

16mm film, "Force, Mass, and Motion", available on loan through Bell Laboratories, Education Development Center, and commercial sources. It deals with mechanics, mostly two-body mechanics, including the effects of the inverse square force and others. Motions are shown on the screen, with the moving particles tracing their paths in an effective and convincing fashion. Students can relate the techniques involved in producing the film to those that they have been seeing. (Running time: 15 minutes.)

The 8mm computer-animated film loops (in cartridges) produced by Harvard's Project Physics and available from the Ealing Corporation, are useful. "Program Orbit I", "Program Orbit II", "Kepler's Laws", "Central Forces", and "Unusual Orbits" were all produced by computer programs similar to those shown here. The tie-in between the step-by-step calculations and the geometrical orbits composed of straight-line segments shown in the film can be very helpful to visually-oriented students. (Running time: 3 minutes in Technicolor cartridge projector.) The 16mm, color animated film, "Newton's Equal Areas" (produced by Bruce and Katherine Cornwell), also displays many of these line segment orbits, explaining how Newton used the two laws of motion to construct such orbits. Again the relation between geometry and analysis is significant; several sequences are based on the output of computer programs much like the present ones. The film is available from International Film Bureau, in Chicago. (Running time: 8 minutes.)

While it is not strictly relevant to the present approach, the instructor who wishes to give his class some additional insight into computer usage might consider showing the 16mm Charles Eames film entitled "The Information Machine", available on loan from IBM offices. (Running time: 10 minutes.)

Short Films for Physics Teaching, a recent catalog issued by the Commission on College Physics, includes an appendix listing computer-animated films; the Commission's Newsletter is also listing subsequent productions.

A Project Physics laboratory, described in the Unit 2 Student Handbook, directs the student to plot an orbit for the gravitational problem, using methods like those mentioned in the above films. Project Physics material is available from Holt, Rinehart and Winston.

You might find it advantageous to use overhead transparencies of the flow charts and programs in this material. All the material included is in the public domain and may be reproduced.

The Commission on College Physics operates a consulting service which allows physics teachers to request the assistance of qualified physicists in curriculum development projects. Consultants with experience in the use of computers in physics courses can be provided, and the school can request such advice.

Feedback from users will be essential to the authors and to the Commission on College Physics in determining future needs. We would appreciate receiving from you an informal letter discussing how you used this material, the problems you encountered or suggestions you may have for improvements.

References

Several references (also listed at the end of the student material) may be useful to you or your students, particularly those who wish to explore this approach beyond these notes.

Sawyer's book, What is Calculus About?, contains two chapters on speed based on a numerical treatment similar to that used here. This material is elementary, and might be used by weak students as an introduction to the present notes. These chapters are also reproduced in the Harvard Project Physics Reader 1.

Possibly the first text to use numerical analysis for teaching physics was Chalmers Sherwin's Basic Concepts of Physics. The introductory material discusses numerical solutions of the differential equations of classical mechanics, using methods like the ones used here. A later chapter indicates how one could use this numerical approach in quantum mechanics to get serious quantum mechanics across to beginning students.

In the Feynman Lectures on Physics, Chapter 9 of Volume I is a treatment of classical mechanics based on numerical methods. Feynman considers both the harmonic oscillator and the one-body two-dimensional gravitational problem. He uses a trick, not used in the present notes, which improves the accuracy of the calculation to second-order; that is, errors are proportional to D^3. He first calculates an initial half-step in the velocity so that in the calculational loop the velocity calculations are intermediate between the position calculations.

You can have students run the program with and without Feynman's initial half-step calculation. To insert the half-step calculation requires only the addition of a new velocity-calculating line, before the main calculation. In place of D, D/2 is used. The improvement made by this simple change is remarkable, and is likely to be a considerable surprise to the student.

Feynman makes no reference to computers until the last section of the chapter, in which he considers the n-body gravitational problem. The Feynman chapter is written in an exciting way, and would be valuable for most classes. It does, however, start with the differential equations, so one might have difficulty with it in some courses. The chapter has been reprinted in Fortran for Physics and in the Harvard Project Physics Reader 1.

In Fortran for Physics (Alfred M. Bork) the Feynman chapter is reprinted. Programs are presented for both harmonic oscillator and one-particle gravitational problems, following the methods used in Feynman. For further study it also contains student problems, and a more advanced physical problem, that of three gravitationally interacting bodies. The physics and the introduction of further details of FORTRAN are interspersed.

SOLUTIONS TO THE ADDITIONAL PROBLEMS

Problem 1:

(a) The equation for the acceleration will be

$$A_{new} = A_{old} + Q \cdot D$$

The equivalent computer statement would be

```
A = A + Q*D
```

(b) For $Q = -BX^3$, the expression for the acceleration becomes

```
A = A - B*X*X*X*D
```

The computational loop which is performed for each new time will then be

```
X = X + V*D
V = V + A*D
A = A - B*X*X*X*D
T = T + D
```

At the beginning of the computation we would have to specify values for A, V, X and also for the constant, B. The program, written in FORTRAN, is shown in Figure 1. It produced the output shown in Table 1.

The most striking feature of the computation is that the maximum positive and negative values of the position steadily increase. Qualitatively, this can be explained in the following way: when X is positive Q is negative and the change in acceleration with time will be negative. Thus, whether the acceleration is positive or negative, it will decrease with time and will eventually become negative. When X is negative Q and the rate of change of the acceleration are positive. Now the acceleration will increase and eventually become positive. We have what might be called a "delayed" restoring force. Due to the "delay", however, oscillations will build up. Because the change in all of the variables during the time interval of the numerical computation becomes very large the computation is probably not very accurate. The basic behavior, however, is fairly accurately represented by the output shown in Table 1. Students might wish to experiment with variations in the value of D in this problem. An interesting result is that varying D may change the "amplitude" of the oscillations but has very little effect on the "period".

```
   PROGRAM QUARTIC(OUTPUT,TAPE6=OUTPUT)
   WRITE(6,30)
   X=1.
   V=0.
   A=0.
   B=1.
   T=0.
   D=.01
   P=.1-D
   WRITE(6,40)X,V,A
   WRITE(6,50)B
   WRITE(6,60)
10 A=A-B*X*X*X*D
   V=V+A*D
   X=X+V*D
   T=T+D
   IF(T-P)10,10,12
12 WRITE(6,70)T,X,V,A
   P=P+.1
   IF(T-3.)10,10,16
16 STOP
30 FORMAT(22H1    THE QUARTIC FORCE)
40 FORMAT(7H0    X=,F4.1,4H  V=,F4.1,4H  A=,F4.1)
50 FORMAT(30H     THE COEFFICIENT OF FORCE=,F7.4,//)
60 FORMAT(9X,1HT,11X,1HX,11X,1HV,11X,1HA)
70 FORMAT(F10.2,3F12.4)
   END
```

Figure 1

```
THE QUARTIC FORCE

X= 1.0  V= 0.0  A= 0.0
THE COEFFICIENT OF FORCE= 1.0000


     T          X          V          A
   .10      .9998     -.0055     -.1000
   .20      .9985     -.0210     -.1998
   .30      .9950     -.0464     -.2989
   .40      .9885     -.0817     -.3967
   .50      .9780     -.1266     -.4920
   .60      .9624     -.1809     -.5837
   .70      .9409     -.2441     -.6703
   .80      .9126     -.3156     -.7504
   .90      .8768     -.3947     -.8227
  1.00      .8326     -.4805     -.8858
  1.10      .7796     -.5721     -.9389
  1.20      .7171     -.6684     -.9815
  1.30      .6448     -.7684    -1.0138
  1.40      .5623     -.8711    -1.0365
  1.50      .4695     -.9756    -1.0508
  1.60      .3661    -1.0812    -1.0585
  1.70      .2522    -1.1872    -1.0617
  1.80      .1276    -1.2935    -1.0626
  1.90     -.0076    -1.3997    -1.0626
  2.00     -.1534    -1.5060    -1.0626
  2.10     -.3098    -1.6122    -1.0613
  2.20     -.4769    -1.7181    -1.0554
  2.30     -.6545    -1.8228    -1.0378
  2.40     -.8424    -1.9245     -.9969
  2.50    -1.0402    -2.0202     -.9154
  2.60    -1.2469    -2.1043     -.7690
  2.70    -1.4611    -2.1688     -.5253
  2.80    -1.6801    -2.2017     -.1443
  2.90    -1.8999    -2.1867      .4210
  3.00    -2.1147    -2.1028     1.2201
  3.10    -2.3160    -1.9240     2.2970
```

Table 1

Problem 2:

(a) The equation for updating the velocity will be

$$V_{new} = V_{old} + (-(K/M)\cdot X - B/M)\cdot V + (G/M)\cdot \sin(H\cdot T))\cdot D$$

If we make the substitution

$$V = (K/M)^{1/2}\cdot V'$$

and

$$T = (M/K)^{1/2}\cdot T'$$

into the preceding equation and simplify by eliminating as many common factors as possible we will have the following equation:

$$V'_{new} = V'_{old} + (-X - C\cdot V' + F\cdot \sin(E\cdot T'))\cdot D'$$

where

$$F = G/K$$

$$C = B/(MK)^{1/2}$$

and

$$E = H\cdot (M/K)^{1/2}$$

Finally we can drop the primes from velocities and times to obtain the following computer expression:

```
V = V - (X + C*V - F*SIN(E*T))*D
```

(b) A program, written in FORTRAN, incorporating this additional force is shown in Figure 2. This program's output is shown in Table 2.

Problem 3:

If we insert the suggested test for $T \geq P$ note that a rounded value of, for example, $T = 0.5$ is actually slightly less than 0.5, so that the equality $T = P = 0.5$ is not satisfied, and the values of X,V for $T = 0.5$ will actually not print until the next time through the computational loop when they will have been calculated.

```
      PROGRAM FORCE(OUTPUT,TAPE6=OUTPUT)
      WRITE(6,30)
      X=0.
      V=0.
      C=1.
      F=1.
      E=1.
      T=0.
      D=.01
      P=.1-D
      WRITE(6,40)X,V
      WRITE(6,50)C,F,E
      WRITE(6,60)
   10 V=V-(X+C*V-F*SIN(E*T))*D
      X=X+V*D
      T=T+D
      IF(T-P)10,10,12
   12 WRITE(6,70)T,X,V
      P=P+.1
      IF(T-4.4)10,10,16
   16 STOP
   30 FORMAT(39H1    FORCED, DAMPED HARMONIC OSCILLATOR  )
   40 FORMAT(7H0     X=,F4.1,4H  V=,F4.1)
   50 FORMAT(7H      C=,F6.3,4H  F=,F6.3,4H  E=,F6.3,//)
   60 FORMAT(9X,1HT,11X,1HX,11X,1HV)
   70 FORMAT(F10.2,2F12.4)
      END
```

Figure 2

```
FORCED, DAMPED HARMONIC OSCILLATOR

X= 0.0  V= 0.0
C= 1.000  F= 1.000  E= 1.000


   T          X          V
  .10       .0002      .0044
  .20       .0013      .0178
  .30       .0042      .0391
  .40       .0096      .0672
  .50       .0181      .1007
  .60       .0302      .1385
  .70       .0463      .1792
  .80       .0665      .2216
  .90       .0910      .2646
 1.00       .1198      .3069
 1.10       .1528      .3474
 1.20       .1896      .3850
 1.30       .2300      .4189
 1.40       .2735      .4482
 1.50       .3197      .4720
 1.60       .3679      .4897
 1.70       .4176      .5009
 1.80       .4680      .5050
 1.90       .5183      .5017
 2.00       .5680      .4909
 2.10       .6161      .4725
 2.20       .6620      .4466
 2.30       .7049      .4133
 2.40       .7441      .3729
 2.50       .7788      .3258
 2.60       .8085      .2725
 2.70       .8326      .2135
 2.80       .8504      .1495
 2.90       .8617      .0812
 3.00       .8659      .0094
 3.10       .8627     -.0651
 3.20       .8520     -.1413
 3.30       .8337     -.2185
 3.40       .8076     -.2956
 3.50       .7738     -.3718
 3.60       .7325     -.4461
 3.70       .6839     -.5176
 3.80       .6284     -.5854
 3.90       .5664     -.6487
 4.00       .4982     -.7068
 4.10       .4247     -.7588
 4.20       .3462     -.8041
 4.30       .2637     -.8421
 4.40       .1777     -.8722
 4.50       .0892     -.8942
```

Table 2

Problem 4:

If we make the same substitution for velocities and times which we have made several times previously the computational equations become

(a)

$$V1 = V1 - X1*D$$

$$V2 = V2 - X2*D$$

$$X1 = X1 + V1*D$$

and

$$X2 = X2 + V2*D$$

(b) A program written in FORTRAN, incorporating these changes, but with print times, P, occurring at intervals of .2, is shown below in Figure 3. A typical run of this program produces the output shown in Table 3.

```
   PROGRAM HOOK(OUTPUT,TAPE6=OUTPUT)
   X1=1.
   X2=0.
   V1=0.
   V2=1.3
   T=0.
   D=.01
   P=.2-D
   WRITE(6,30)
   WRITE(6,40)X1,X2,V1,V2
   WRITE(6,60)
10 V1=V1-X1*D
   V2=V2-X2*D
   X1=X1+V1*D
   X2=X2+V2*D
   T=T+D
   IF(T-P)10,10,12
12 WRITE(6,70)T,X1,X2
   P=P+.2
   IF(T-9.)10,10,16
16 STOP
30 FORMAT(34H1    HOOKE'S LAW IN TWO DIMENSIONS  )
40 FORMAT(8H0    X1=,F6.3,5H  X2=,F6.3,5H  V1=,F6.3,5H  V2=,F6.3,///
60 FORMAT(9X,1HT,10X,2HX1,10X,2HX2)
70 FORMAT(F10,2,2F12.4)
   END
```

Figure 3

Problem 5:

(a) It can be seen in Figure 4A that the Cartesian components of the force are given by

$$F1 = -F \cos\theta \ ; \ F2 = -F \sin\theta$$

where the magnitude of the gravitational force $F = 1/R^2$ and $\tan\theta = X2/X1$.

(b) The program to solve this problem is shown in Figure 4B and the output produced by this program is given in Table 4. It is interesting to compare the Hooke's Law force and the inverse square force. Choosing initial conditions which do not correspond to circular orbits will shed some light on this comparison.

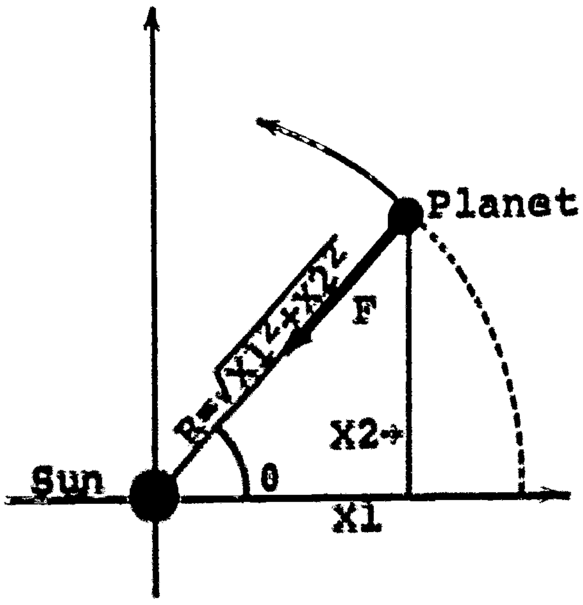


Figure 4A

```
      PROGRAM PLANET(OUTPUT,TAPE6=OUTPUT)
      X1=1.
      X2=0.
      V1=0.
      V2=1.
      T=0.
      D=.01
      P=.1-D
      WRITE(6,30)
      WRITE(6,40)X1,X2,V1,V2
      WRITE(6,60)
   10 RSQR=X1*X1+X2*X2
      ROOT=SQRT(RSQR)
      RCUB=RSQR*ROOT
      A1=-X1/RCUB
      A2=-X2/RCUB
      V1=V1+A1*D
      V2=V2+A2*D
      X1=X1+V1*D
      X2=X2+V2*D
      T=T+D
      IF(T-P)10,10,11
   11 WRITE(6,70)T,X1,X2,ROOT
      P=P+ 1
      IF(T-3 -D)10,10,16
   16 STOP
   30 FORMAT([illegible]     INVERSE SQUARE FORCE  )
   40 FORMAT([illegible]     X1=,F6.3,5H  X2=,F6.3,5H  V1=,F6.3,5H  V2=,F6.3,//)
   60 FORMAT(8X,1HT ,10X,2HX1,10X,2HX2,11X 1HR)
   70 FORMAT(F10 3,3F12 3)
      END
```

Figure 4B

INVERSE SQUARE FORCE

X1= 1.000 X2= 0.000 V1= 0.000 V2= 1.000

T	X1	X2	R
.10	.9945	.0998	.9996
.20	.9791	.1987	.9991
.30	.9538	.2955	.9986
.40	.9190	.3894	.9981
.50	.8749	.4794	.9977
.60	.8220	.5645	.9972
.70	.7609	.6440	.9968
.80	.6920	.7169	.9965
.90	.6162	.7826	.9961
1.00	.5342	.8404	.9958
1.10	.4467	.8897	.9956
1.20	.3547	.9300	.9954
1.30	.2592	.9609	.9952
1.40	.1610	.9820	.9951
1.50	.0611	.9932	.9951
1.60	-.0393	.9943	.9951
1.70	-.1394	.9853	.9951
1.80	-.2380	.9663	.9952
1.90	-.3342	.9375	.9953
2.00	-.4271	.8993	.9955
2.10	-.5156	.8519	.9958
2.20	-.5989	.7959	.9960
2.30	-.6761	.7319	.9963
2.40	-.7466	.6604	.9967
2.50	-.8094	.5823	.9971
2.60	-.8642	.4984	.9975
2.70	-.9102	.4094	.9980
2.80	-.9471	.3163	.9984
2.90	-.9744	.2200	.9989
3.00	-.9920	.1215	.9994
3.10	-.9997	.0218	.9999
3.20	-.9974	-.0781	1.0004
3.30	-.9851	-.1772	1.0009
3.40	-.9630	-.2746	1.0014
3.50	-.9314	-.3692	1.0018
3.60	-.8905	-.4602	1.0023
3.70	-.8407	-.5466	1.0027
3.80	-.7826	-.6276	1.0031
3.90	-.7168	-.7024	1.0035
4.00	-.6439	-.7702	1.0039
4.10	-.5646	-.8304	1.0042
4.20	-.4797	-.8825	1.0044
4.30	-.3902	-.9258	1.0046
4.40	-.2967	-.9600	1.0048
4.50	-.2004	-.9848	1.0049

Table 4

Problem 6:

(a) Work done by a force F in a displacement ΔX is given by

$$\Delta W = F\Delta X = -KX\Delta X$$

for a Hooke's Law force. Since time is not present in this equation the transformation leaves it unaffected. Hence, if we omit K from the calculation we must remember that we are computing "work per unit K".

(b) Since $\Delta X = V \cdot D$ the FORTRAN program is that which is given below in Figure 5, with its output given in Table 5 on the following page.

```
      PROGRAM WORK(OUTPUT,TAPE6=OUTPUT)
      WRITE(6,30)
      T=0.
      X=1.
      V=0.
      W=0.
      D=.01
      P=.1-D
      WRITE(6,40)X,V,D
      WRITE(6,60)
   10 X=X+V*D
      V=V-X*D
      W=W-X*V*D
      T=T+D
      IF(T-P)10,10,12
   12 WRITE(6,70)T,X,V,W
      P=P+.1
      IF(T-4.4)10,10,16
   16 STOP
   30 FORMAT(36H1    WORK IN SIMPLE HARMONIC MOTION    )
   40 FORMAT(5H0  X=,F4.1,4H  V=,F4.1,4H  D=,F5.2)
   60 FORMAT(1H0,8X,1HT,11X,1HX,11X,1HV,11X,1HW)
   70 FORMAT(F10.2,3F12.4)
      END
```

Figure 5

WORK IN SIMPLE HARMONIC MOTION

X= 1.0   V= 0.0   D=   .01

T	X	V	W
.10	.9955	-.0998	.0055
.20	.9811	-.1987	.0207
.30	.9568	-.2955	.0451
.40	.9230	-.3894	.0777
.50	.8800	-.4794	.1172
.60	.8282	-.5647	.1621
.70	.7681	-.6442	.2105
.80	.7003	-.7174	.2606
.90	.6255	-.7833	.3103
1.00	.5445	-.8415	.3577
1.10	.4580	-.8912	.4009
1.20	.3670	-.9321	.4382
1.30	.2723	-.9636	.4681
1.40	.1749	-.9855	.4895
1.50	.0757	-.9975	.5014
1.60	-.0242	-.9996	.5035
1.70	-.1239	-.9917	.4956
1.80	-.2223	-.9739	.4781
1.90	-.3186	-.9463	.4517
2.00	-.4116	-.9093	.4175
2.10	-.5005	-.8632	.3767
2.20	-.5845	-.8085	.3311
2.30	-.6626	-.7457	.2825
2.40	-.7340	-.6755	.2329
2.50	-.7982	-.5985	.1841
2.60	-.8543	-.5155	.1383
2.70	-.9019	-.4274	.0971
2.80	-.9406	-.3350	.0623
2.90	-.9698	-.2392	.0353
3.00	-.9893	-.1411	.0171
3.10	-.9989	-.0416	.0085
3.20	-.9986	.0584	.0099
3.30	-.9883	.1578	.0211
3.40	-.9681	.2556	.0418
3.50	-.9382	.3508	.0711
3.60	-.8990	.4425	.1079
3.70	-.8507	.5299	.1507
3.80	-.7940	.6119	.1979
3.90	-.7294	.6878	.2475
4.00	-.6574	.7568	.2976
4.10	-.5789	.8183	.3462
4.20	-.4946	.8716	.3914
4.30	-.4054	.9162	.4314
4.40	-.3121	.9516	.4645
4.50	-.2157	.9775	.4896

Table 5

Problem 7:

(a) A FORTRAN program to calculate the kinetic energy is shown in Figure 6 (below) and its output is given in Table 6. The expected agreement between W and E is fairly good. It can be improved a great deal by choosing D = .001.

```
      PROGRAM ENERGY(OUTPUT,TAPE6=OUTPUT)
      WRITE(6,30)
      T=0.
      X=1.
      V=0.
      W=0.
      D=.01
      P=.1-D
      WRITE(6,40)X,V,D
      WRITE(6,60)
   10 X=X+V*D
      V=V-X*D
      W=W-X*V*D
      E=V*V/2.
      T=T+D
      IF(T-P)10,10,12
   12 WRITE(6,70)T,X,V,W,E
      P=P+.1
      IF(T-4.4)10,10,16
   16 STOP
   30 FORMAT(28H1    WORK AND KINETIC ENERGY  )
   40 FORMAT(5H0  X=,F4.1,4H  V=,F4.1,4H  D=,F5.2)
   60 FORMAT(1H0,8X,1HT,11X,1HX,11X,1HV,11X,1HW,11X,1HE)
   70 FORMAT(F10.2,4F12.4)
      END
```

Figure 6

WORK AND KINETIC ENERGY

X= 1.0 V= 0.0 D= .01

T	X	V	W	E
.10	.9955	-.0998	.0055	.0050
.20	.9811	-.1987	.0207	.0197
.30	.9568	-.2955	.0451	.0437
.40	.9230	-.3894	.0777	.0758
.50	.8800	-.4794	.1172	.1149
.60	.8282	-.5647	.1621	.1594
.70	.7681	-.6442	.2105	.2075
.80	.7003	-.7174	.2606	.2573
.90	.6255	-.7833	.3103	.3068
1.00	.5445	-.8415	.3577	.3540
1.10	.4580	-.8912	.4009	.3971
1.20	.3670	-.9321	.4382	.4344
1.30	.2723	-.9636	.4681	.4642
1.40	.1749	-.9855	.4895	.4856
1.50	.0757	-.9975	.5014	.4975
1.60	-.0242	-.9996	.5035	.4996
1.70	-.1239	-.9917	.4956	.4917
1.80	-.2223	-.9739	.4781	.4742
1.90	-.3186	-.9463	.4517	.4478
2.00	-.4116	-.9093	.4175	.4134
2.10	-.5005	-.8632	.3767	.3726
2.20	-.5845	-.8085	.3311	.3268
2.30	-.6626	-.7457	.2825	.2780
2.40	-.7340	-.6755	.2329	.2281
2.50	-.7982	-.5985	.1841	.1791
2.60	-.8543	-.5155	.1383	.1329
2.70	-.9019	-.4274	.0971	.0913
2.80	-.9406	-.3350	.0623	.0561
2.90	-.9698	-.2392	.0353	.0286
3.00	-.9893	-.1411	.0171	.0100
3.10	-.9989	-.0416	.0085	.0009
3.20	-.9986	.0584	.0099	.0017
3.30	-.9883	.1578	.0211	.0124
3.40	-.9681	.2556	.0418	.0327
3.50	-.9382	.3508	.0711	.0615
3.60	-.8990	.4425	.1079	.0979
3.70	-.8507	.5299	.1507	.1404
3.80	-.7940	.6119	.1979	.1872
3.90	-.7294	.6878	.2475	.2365
4.00	-.6574	.7568	.2976	.2864
4.10	-.5789	.8183	.3462	.3348
4.20	-.4946	.8716	.3914	.3798
4.30	-.4054	.9162	.4314	.4197
4.40	-.3121	.9516	.4645	.4528
4.50	-.2157	.9775	.4896	.4778

Table 6