

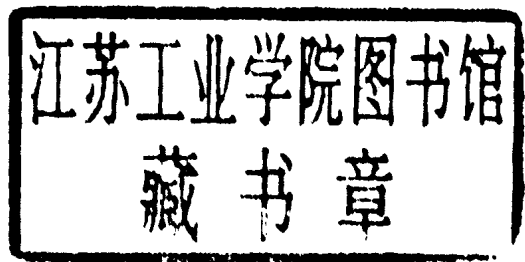
INTERACTIVE DYNAMIC SYSTEM SIMULATION



GRANINO A. KORN

Interactive Dynamic System Simulation

Granino A. Korn



McGraw-Hill Book Company

New York St. Louis San Francisco Auckland
Bogotá Hamburg London Madrid Mexico
Milan Montreal New Delhi Panama
Paris São Paulo Singapore
Sydney Tokyo Toronto

Library of Congress Cataloging-in-Publication Data

Korn, Granino Arthur, date
Interactive dynamic system simulation.

Bibliography: p.

Includes index. 1. Computer simulation. 2. Dynamics. 3. Micro-computers. 4. Interactive computer systems. I. Title.

QA76.9.C65K68 1988 001.4'34 88-786

ISBN 0-07-852262-5

ISBN 0-07-852261-7 (book and disk)

ISBN 0-07-852263-3 (disk)

Copyright © 1989 by McGraw-Hill, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

1234567890 DOC/DOC 89321098

ISBN 0-07-852262-5 {BK}

The editors for this book were Theron Shreve and Ingeborg M. Stochmal, the designer was Naomi Auerbach, and the production supervisor was Dianne Walber. It was set in Century Schoolbook by The William Byrd Press, Inc.

Printed and bound by R. R. Donnelley & Sons Company.

For more information about other McGraw-Hill materials, call 1-800-2-MCGRAW in the United States. In other countries, call your nearest McGraw-Hill office.

Information contained in this work has been obtained by McGraw-Hill, Inc., from sources believed to be reliable. However, neither McGraw-Hill nor its authors guarantee the accuracy or completeness of any information published herein and neither McGraw-Hill nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that McGraw-Hill and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.

Preface

This book introduces scientists and engineers to interactive, direct-executing dynamic-system simulation with ordinary personal computers. We demonstrate an entirely new environment for practical, interactive experimentation with models of aerospace vehicles, control systems, chemical processes, and biological systems.

The importance of simulation for design, research, and education needs little discussion. But it is less generally known that inexpensive personal computers can do very substantial scientific computation if their software produces good code for the 8087, 80287, or 80387 math coprocessor. This is doubly true for the newer models with fast clocks or accelerator boards. 10-MHz AT clones readily solve hundreds of differential equations at one-third VAX 11/780 speed; inexpensive 16-MHz 32-bit PCs outrun the timeshared VAX.

Our emphasis is on *interactive* computation. Interpreted BASIC is interactive but far too slow. We really need much faster programs which still *execute directly*, that is, without noticeable delay for compilation and linking. This book describes new direct-executing simulation systems for IBM-compatible 16-bit and 32-bit personal computers. These systems are source-code-compatible with similar software for the DEC VAX and MICROVAX.

The convenience and speed of a direct-executing simulation system—which does not interrupt an experimenter’s train of thought with repeated compilation delays—must be experienced to be believed. DESIRE (Direct-Executing SIMulation in REal time) systems for simple PC clones solve 100 differential equations on a typed “run” command, immediately display bright color graphs, and can perform multirun statistical and optimization studies. DESIRE/287/387 for AT clones and 32-bit PCs solves 400 differential equations with a variety of double-precision fixed- and variable-step Runge-Kutta methods and includes variable-order, variable-step Adams and Gear (stiff) rules for 50th-order systems.

In addition, the new direct-executing software also performs general-purpose scientific computation. DESIRE handles complex variables, vectors, matrices, and fast Fourier transforms, includes a screen editor, and can use operating-system commands or command procedures as program lines. A modern trace facility simplifies debugging.

Chapter 1 explains the nature of the *state-equation models* used for dynamic-system simulation and outlines *hardware and software requirements for interactive simulation*. This chapter introduces the program flow of traditional simulation languages as well as ENHANCED DESIRE and concludes with our first short program for a direct-executing simulation.

In the remainder of this book we do not merely discuss simulation: *we attempt to demonstrate actual practical simulation procedures, step by step, with complete, runnable personal-computer programs.* Tutorial versions of DESIRE with color or monochrome graphics, complete except for a size restriction to five ordinary differential equations and 900 REAL array or FFT elements, are either included with the book or may be obtained from the publisher. The diskette contains the programs for many examples in the book.

Chapter 2 surveys *classical applications of dynamic-system simulation in physics, aerospace engineering, physiology, and population dynamics.* Fifteen complete personal-computer programs with display photos and screen prints serve as illustrations. Interestingly, the simulation-language programs can be read much like ordinary mathematics, before language details are discussed.

Chapters 3 and 4 present a *tutorial on personal-computer simulation programming.* This material has already been used both at the universities of Arizona and Nebraska and in industrial refresher courses. Each direct-executing simulation program naturally divides into *a compiled model definition and an interpreted experiment protocol.* We discuss modeling, displays, and multirun simulation studies, but also practical screen editing, file manipulation, and hard-copy output.

Chapter 5 includes much new material on interactive simulation programming and modeling techniques. Included are new implementations of *submodels and user-defined functions; program combinations; special transfer characteristics and signal generators; recursion relations for hysteresis, backlash, and track/hold circuits; function storage in arrays; time delays; and automatic display scaling.* The chapter concludes with an exhibit of several more advanced simulation programs, including blood-circulation simulation and ecosystem simulation.

Chapter 6 describes *nonsimulation applications of the direct-executing software.* The DESIRE interpreter permits *complex-number calculations,* including conformal mapping on the CRT display; *linear vector transformations and other matrix operations;* and calculation of *fast Fourier transforms (FFT's) and convolutions.* The compiled program segment is especially useful for its automatic graph-plotting feature and permits fast operations on arrays, solution of difference equations, and statistical averaging.

Interesting applications include digital signal processing and computation of statistics, such as averages and correlation functions. For students and teachers, DESIRE provides *user-written help screens and menus.* DESIRE's *debugging facilities* (trace and dump) and the *automatic notebook file* are also discussed.

Chapter 7 completes the volume with a *tutorial on control-system simulation*, probably the most important and fruitful application area. Complete programs illustrate interactive and programmed parameter-sensitivity studies for a simulated servomechanism and a simple example of automatic parameter optimization. A fairly sophisticated satellite autopilot simulation demonstrates the use of submodels, nonlinear transfer characteristics, and true time delay. We continue with special simulation techniques for sampled-data control systems, simulate an analog plant with a digital controller, and conclude with a discussion of frequency-response and matrix calculations.

An appendix presents a brief discussion of *integration methods*, *perturbation techniques*, and other mathematical topics. We have tried to provide useful reference tables and a very comprehensive index.

Acknowledgments

The writer is grateful to the Society for Computer Simulation for permitting the use of Figs. 2.5, 7.6, and 7.10 from his articles in *Simulation*. Parts of Secs. 5.7 to 5.11 and Fig. 5.15 are based on Ref. 3, Chap. 5. The examples of Figs. 7.5, 7.8, 7.9, and 7.10 are modified versions of the writer's examples in Refs. 2 and 4, Chap. 7. IBM PC and PC/AT are registered trademarks of the IBM Corporation; DEC, VAX, PDP-11, and VMS are registered trademarks of the Digital Equipment Corporation.

Granino A. Korn

About the Author

Granino A. Korn has pioneered interactive simulation on minicomputers for over 20 years. Author or coauthor of nearly a dozen books on computers, mathematics, and simulation techniques, he currently heads G.A. and T. M. Korn Industrial Consultants, a firm that develops software and systems for computer simulation and laboratory automation.

Contents

Preface ix

Chapter 1. Introduction to Dynamic-System Simulation	1
Overview	1
Introduction	2
1.1 Models, Time Histories, and State Equations	2
1.2 Differential-Equation Models	4
1.3 Applications—Practical Choice of State Variables	4
1.4 Models with Defined Variables	7
Computer Simulation of Dynamic Systems	8
1.5 Dynamic-System Simulation	8
1.6 Computer Requirements for Interactive Simulation	8
Simulation Programs and Simulation Languages	9
1.7 Anatomy of a Simulation Program—Model Definition, Experiment Protocol, and Simulation Runs	9
1.8 The Integration Routine	10
1.9 The Simulation-Run Routine—Time-History Output and Integration	11
1.10 Simulation Subroutine Packages and Simulation Language Systems	14
1.11 Block-Diagram Languages	14
1.12 CSSL-Committee-Type Languages	15
True Interactive Simulation Means Direct Execution	17
1.13 Software for Interactive Simulation—Separating Model and Experiment	17
1.14 A Complete Direct-Executing Simulation Program	19
References	20
 Chapter 2. Simulation: The Standard Applications	 23
Overview	23
Simple Examples from Physics	24
2.1 Damped and Coupled Oscillators	24
2.2 Simulation of a Rigid Pendulum; Phase-Plane Plots for Nonlinear Oscillators	24
2.3 Van der Pol's Differential Equation	25
2.4 Nuclear-Reactor Simulation	25
2.5 Delay-Line Circuit Simulation—100 Differential Equations	25
Aerospace and Related Applications	30
2.6 Ballistic Trajectories	30
2.7 Simple Flight Simulation	33

2.8	A Simplified Autopilot	36
2.9	A Torpedo Trajectory	38
2.10	Translunar Satellite Orbit	42
	Models from Physiology and Population Dynamics	44
2.11	Simulation of a Glucose-Tolerance Test	44
2.12	Simulation of Epidemic Propagation	44
2.13	Simulation of Ecological Systems—A Predator/Prey Model	46
	References	48
Chapter 3. DESIRE Programs and Program Files		49
	Overview	49
	We Get Started	49
3.1	Before You Begin	49
3.2	Loading and Leaving the System	51
	We Run a Complete Program—Execution of Commands	53
3.3	An Example	53
3.4	Command Mode	53
3.5	Operating-System Commands	54
	We Enter a Simple Program and List It on the Screen	55
3.6	A Simple Program	55
3.7	Program Listings	56
3.8	Auto-Line-Number Mode	56
	Editing ENHANCED DESIRE Programs	57
3.9	Simple Line Editing	57
3.10	One-Line Screen Editing	57
3.11	The ENHANCED DESIRE Screen Editor	58
	Program Loading, Saving, Recovering, and Printing	58
3.12	“old” and “save” Statements	58
3.13	The Problem Identification Code (PIC)	60
3.14	Hard-Copy Listings and Data Transmission	61
	Source Program Manipulation	62
3.15	Saving ASCII Source Files	62
3.16	Loading ASCII Source Files	62
3.17	“new” Clears the Memory	62
3.18	Deleting and Renaming Files	63
Chapter 4. Programming Interactive Experiments		65
	Overview	65
	Interpreter Programs—The Bare Minimum	66
4.1	Expressions and Functions	66
4.2	Output to a Display, Devices, and Files	67
4.3	Conditional Branching—“if” Statements	70
4.4	Labels and Spaghetti Code	74
	Structured Interpreter Programs	74
4.5	Preview of Improved Interpreter Programs	74
4.6	“for”, “while”, and “repeat” Loops	75
4.7	Subscripted Variables and Arrays	77
4.8	Data Lists and “read” Assignments	78
	The Dynamic Program Segment Defines a Simulation Model	79
4.9	Programming a Simulation Run	79
4.10	Writing the DYNAMIC Segment—Defined Variables and State Equations	80

4.11	Sorting Expression Assignments	81
4.12	Library Functions and Table-Lookup/Interpolation Function Generation	81
4.13	Run-Time Displays and Listings	83
4.14	Saving and Recovering Time-History Files	85
4.15	The Run-Termination Operator	86
	Interpreter Programs and Commands Define Interactive Experiments	87
4.16	How to Program a New Simulation	87
4.17	Resetting Initial Conditions for Repeated Runs	88
4.18	Interactive Experiments	88
4.19	Programmed Multirun Experiments—Display and Storage Control	89
4.20	Stopping and Continuing a Simulation Run	93
4.21	“Continuing” Runs with Program Changes— “Mode-Switched” Integration and Bouncing-Ball Simulation	93
Chapter 5.	More Advanced Programming Techniques	95
	Overview	95
	User-Defined Language Extensions	96
5.1	User-Defined Functions	96
5.2	Procedures and Procedure Arguments	97
5.3	Submodel Declaration and Invocation	99
	Combining Program Segments and Programs	100
5.4	Combining Program Segments and Library Files	100
5.5	Multiple DYNAMIC Subsegments and Program Chaining	101
5.6	Operating-System Commands and Command Procedures as Program Lines	103
	Special Functions and Transfer Characteristics	103
5.7	Common Nonlinear Library Functions—Limiters	103
5.8	Function Switches and Comparators	106
5.9	Useful Relations between Functions—Maxima and Minima	106
5.10	Recursion Relations, Hysteresis, and Signal Generators	107
5.11	Hysteresis, Difference Equations, and Track/Hold Circuits	108
	Function Storage and Time Delays	111
5.12	Function Storage and Recovery	111
5.13	Fixed and Variable Time Delays	114
	Automatic Display Scaling	117
5.14	Automatic Rescaling on Display-Scale Overloads	117
5.15	Automatic Scaling Using min/max Functions	118
	Some More Advanced Simulation Programs	119
5.16	A Blood-Circulation Model	119
5.17	System Dynamics and World Simulation	124
5.18	Pilot-Ejection Study and Cross Plot	125
5.19	Solution Envelopes for Multiple Runs	127
	References	129
Chapter 6.	General-Purpose Scientific Computation	131
	Overview	131
	Compiled Programs Need Not Solve Differential Equations	131
6.1	The Dummy Integration Rule	131
6.2	Semiautomatic Graph Plotting and File Output	133
6.3	Fast Array Manipulation	134
6.4	Recursion and Difference Equations	135

6.5 Use of Euler Integration (irule 2) for Difference Equations and Accurate Addition—Statistical Averaging	136
Vector/Matrix Operations and Fast Fourier Transforms	138
6.6 Linear Vector Transformations and Dot Products	138
6.7 Matrix Multiplication, Transposition, and Inversion—Solution of Linear Equations	138
6.8 Fast Fourier Transforms and Convolutions	140
COMPLEX Variables and Other Topics	141
6.9 INTEGER and COMPLEX Numbers and Arrays	141
6.10 Interpreter Graphics—Conformal Mapping and Frequency-Response Plots	144
The Interactive Workstation Environment	147
6.11 The Automatic Notebook File	147
6.12 Debugging Facilities—Program Stepping and Tracing	149
6.13 The Dump Facility	150
6.14 “time” and “size” Statements	150
6.15 The Help Facility—ASCII Screen Displays and Pulldown Menus	150
Chapter 7. Control-System Simulation	153
Overview	153
Interactive Control-System Simulation	154
7.1 Simulation of a Servomechanism	154
7.2 Recording the Experiment	157
7.3 Parameter-Sensitivity Studies	158
7.4 Making Model Changes	158
7.5 Performance Measures, Cross Plots, and Automatic Parameter Optimization	159
7.6 Special Nonlinear Transfer Characteristics	161
7.7 Use of Submodels—A Satellite-Attitude-Control Simulation	162
Simulation of Sampled-Data Control Systems	162
7.8 Programming Sampled-Data Operations and Quantization	162
7.9 An Analog Plant with a Digital PID Controller	163
Linear Systems and Frequency-Response Computations	168
7.10 State-Equation Models for Linear Time-Invariant Systems	168
7.11 Frequency-Response Plots—Use of Fast Fourier Transforms	168
Matrix Techniques	172
7.12 Matrix Operations in the Interpreter Program	172
7.13 Matrix Differential Equations	172
References	173
Appendix	175
A.1 Simulation Accuracy and Choice of Integration Rules	175
A.2 Integration of Discontinuous Functions	178
A.3 Choice of the Integration Step Size—Stiff Systems	179
A.4 Perturbation Methods for Improved Accuracy	180
A.5 Avoiding Division	180
A.6 Implicit Computation of Inverse Functions	181
A.7 Reference Tables	182
References	186

Introduction to Dynamic-System Simulation

Overview

Simulation is experimentation with models, typically models set up on a digital computer. **Dynamic-system simulation**, in particular, employs state-variable models described by differential equations or difference equations. Simulation for research and design, education, training, and partial system tests accounts for a substantial fraction of engineering computation. With valid models, simulation is often dramatically more cost-effective than are real experiments, which can be expensive, dangerous, or, in fact, impossible because a new system is not yet available.

Realistic models of aerospace vehicles, chemical or nuclear reactors, biological systems, or social systems can be complicated, involving many differential equations and nonlinear functions. Simulation experiments typically require multiple simulation runs, producing time histories of model variables. Models or model parameters are changed manually or automatically between runs, frequently as a result of earlier observations. A good simulation system lets you program models and experiments *interactively*, displays and records results quickly, and then lets you modify model or experiment for another try. Good system design also attempts to free simulation users from details

of computation and programming, so that they can concentrate on their experiments.

This chapter introduces the reader to state-variable models of dynamic systems and to the hardware and software requirements for effective simulation. We describe the flow of a simulation program, discuss conventional simulation languages, and introduce the newer direct-executing simulation environments.

Introduction

1.1 Models, time histories, and state equations

Scientists and engineers deal with the complexity of the real world in terms of simplified **models**. **Model relations** between **model objects** “abstract” useful or interesting properties of corresponding real-world relations and objects. Well-defined relations between model objects are by definition mathematical relations, numerical or not. Model objects and their properties in different states are normally specified by numerical variables related to real measurements. Experiments can then check measured states predicted from model relationships, and the model may be amended as needed. This is the basis of the scientific method and of rational engineering design.¹

Models may, for instance, predict relations between measurement-related variables such as electric currents and voltages, or gas volumes and pressures; or between drug dosages and heart rate, or commodity supply and demand. Model-based predictions have been immensely useful and are intellectually satisfying. But even successful and familiar models (such as electrons) are merely simplified and idealized abstractions, *never* identical with real-world experiences.

Many model descriptions require **time histories of numerical model variables**, say,

$$x = x(t), y = y(t), \dots$$

where the **independent variable** t is the time measured by an agreed-on clock mechanism. Thus, $x(t)$ might predict the distance traveled, the current speed, or the fuel remaining for an automobile or aircraft; the chemical composition of a reactor charge; or the current price of wheat. The time t could be specified as a continuous variable, or as a set of discrete observation times t_0, t_1, t_2, \dots

Simple models predict *output* time histories directly as mathematical functions of known *input* time histories, for example,

Tire pressure = $A * \text{absolute temperature}$

Current = $\text{voltage}/R$

Such relations imply instantaneous responses of the output variable to each input change and, more often than not, agree with experiments only for very slow changes (*static* models).

State-transition models (state-variable models) account much more realistically for the observed behavior of real-world *dynamic* systems in that:

1. Effects of input changes on output variables are delayed.
2. The entire output time history depends on the initial values of the output variables as well as on the input time history.

Given the value $x(t)$ of a **state variable** x at the time t , a state-transition model predicts the value of x at some future time $t + \Delta t$ by a **state equation**,

$$x(t + \Delta t) = S[x(t), t, \Delta t] \quad (1.1a)$$

that is, *the future state $x(t + \Delta t)$ is a given function of the current state $x(t)$, of the current time t , and of the time increment Δt* . The time dependence of the function S includes effects of time-variable system inputs, if any.

The state equation, Eq. (1a), is a **difference equation**, relating current and future values of $x(t)$. If a state-transition model specifies the function S from empirical or theoretical considerations for even a small range of time increments Δt , then *the state equation permits recursive computation of the state variable $x(t)$ for future values of t once an initial value $x(t_0)$ is known*,

$$\begin{aligned} x(t_0 + \Delta t) &= S[x(t_0), t_0, \Delta t] \\ x(t_0 + 2\Delta t) &= S[x(t_0 + \Delta t), t_0 + \Delta t, \Delta t] \\ x(t_0 + 3\Delta t) &= S[x(t_0 + 2\Delta t), t_0 + 2\Delta t, \Delta t] \\ &\dots\dots\dots \end{aligned}$$

The simple state-transition model is readily generalized to model states and defined by n state variables x_1, x_2, \dots, x_n . We now write n state equations,

$$x_i(t + \Delta t) = S_i[x_1(t), x_2(t), \dots; t; \Delta t] \quad i = 1, 2, \dots \quad (1.1b)$$

which can be solved recursively with the aid of n **initial conditions**. Quite often one knows the **initial values** $x_i(t_0)$ of the n state variables.

We can also consider the original equation (1a) as a **matrix (vector) equation** relating current and future values of an n -element column matrix, the **state vector** $x = [x_1, x_2, \dots, x_n]$.

Important examples of state-equation models include particle and rigid-body dynamics, electric-circuit theory, and population dynamics. Interestingly, economic theory has passed from *static* to *dynamic* models much more recently than physics. In some applications, the time increment Δt is a constant, and time histories are specified at uniformly spaced sampling times t_0, t_1, t_2, \dots , separated by Δt (**time series**). This type of description is used in the social sciences and for digital control systems.

1.2 Differential-equation models

Many state-transition models, mainly in physics, assume continuous variables t and x (*continuous dynamic systems*) and express the state equations (1) for small changes Δt of t in the *incremental* form

$$\begin{aligned} x(t + \Delta t) &= x(t) + \Delta x(t) \\ &\approx x(t) + G[x(t), t] \Delta t \end{aligned} \quad (1.2)$$

In the limiting case $\Delta t \rightarrow 0$, the state equations then become **first-order ordinary differential equations**,

$$dx/dt = G[x(t), t] \quad (1.3a)$$

or

$$dx_i/dt = G_i[x_1(t), x_2(t), \dots; t] \quad i = 1, 2, \dots, n \quad (1.3b)$$

These differential equations must be satisfied by the **solution** $x(t)$ or $x_1(t), x_2(t), \dots, x_n(t)$ together with n initial-value conditions.

1.3 Applications—practical choice of state variables

Dynamic-system models based on differential equations are widely used in the areas of mechanics, electricity, and chemical reactions. More recently, such models have served to describe plant growth, physiological systems, population dynamics, and economic systems.²⁻⁷

Quite often a model is already formulated in terms of differential equations. A *differential equation of order n*

$$\frac{dy^n}{dt^n} = F\left(y, \frac{dy}{dt}, \frac{dy^2}{dt^2}, \dots, \frac{dy^{n-1}}{dt^{n-1}}; t\right) \quad (1.4)$$

reduces to n first-order state equations if one introduces y and its first $n - 1$ derivatives as the n state variables x_1, x_2, \dots, x_n ,

$$y = x_1, \frac{dx_1}{dt} = x_2, \frac{dx_2}{dt} = x_3, \dots, \frac{dx_{[n-1]}}{dt} = x_n \quad (1.5)$$

$$\frac{dx_n}{dt} = F(x_1, x_2, x_3, \dots, x_n; t)$$

State variables obtained in this way are sometimes called **phase variables**. Special approximation techniques are needed if it is impossible to solve Eq. (4) explicitly for the highest derivative of y .

In physics, state variables often describe energy storage, as is the case for capacitor voltages and inductor currents in electric-circuit problems. But the best known of many convenient and general “natural laws” formulated in state-equation form are **Newton/La-grange equations of motion** in mechanics.⁴ Note that, for each type of mechanical system, the same relatively simple differential equations hold for a wide variety of initial conditions (Fig. 1.1). Hence Newton’s “laws” describe a truly huge class of phenomena.

The following simple examples of state-equation models are typical of those used in computer simulations (see also Chap. 2).

Motion of a falling body. The motion of a body in free fall (no air resistance) with constant acceleration $g = 32.2 \text{ ft/s}^2$ is described by the second-order equation of motion

$$d^2y/dt^2 = -g$$

We use the altitude y and its derivative y_{dot} (vertical velocity) as state variables. The state equations are

$$d/dt y = y_{\text{dot}} \quad d/dt y_{\text{dot}} = -g$$

with given initial values $y(0)$ and $y_{\text{dot}}(0)$.

This simple problem could be solved analytically without any computer. But a simulation program makes it far easier to display or plot the solution time histories. The problem becomes a little more interesting if we refine the model so that the acceleration of gravity depends on the altitude y and also add a drag force **DRAG** dependent on altitude (air density) and velocity,

$$d/dt y = y_{\text{dot}} \quad d/dt y_{\text{dot}} = -g(y) - \text{DRAG}(y, y_{\text{dot}})/\text{mass}$$

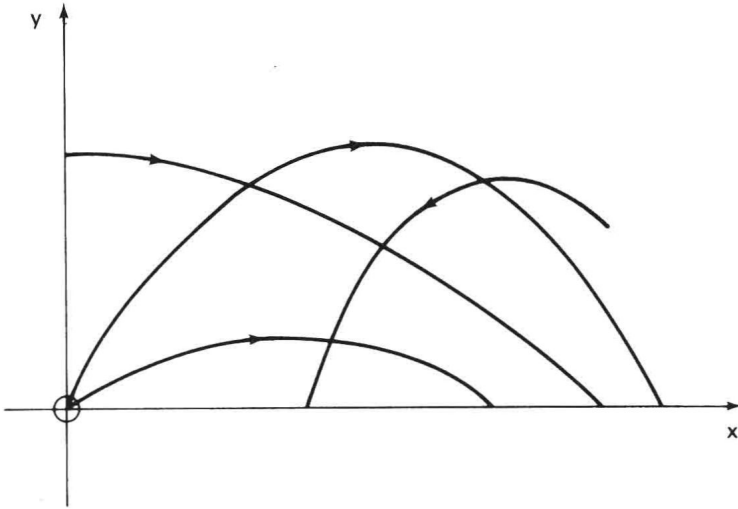


Figure 1.1 Many different projectile trajectories are derived from the same state-equation model

$$\begin{aligned} \frac{d}{dt} x &= \dot{x} & \frac{d}{dt} \dot{x} &= 0 \\ \frac{d}{dt} y &= \dot{y} & \frac{d}{dt} \dot{y} &= -g \end{aligned}$$

with different initial values for x , y , \dot{x} , and \dot{y} . Simple differential-equation models like Newton's laws can, thus, deal with a very wide variety of practical problems.

A simulation program can easily accept functions $g(y)$ and $\text{DRAG}(y, \dot{y})$ as mathematical expressions and/or defined by tables of experimental data; but an analytical solution is now impossible.

Electric-circuit problem. The simple electric circuit of Fig. 1.2 satisfies two first-order differential equations,

$$\frac{d}{dt} i = (E - V)/L \quad \frac{d}{dt} V = i/C - V/RC$$

which express Kirchhoff's laws for currents and voltages together with the definitions of inductance L , capacitance C , and resistance R . We

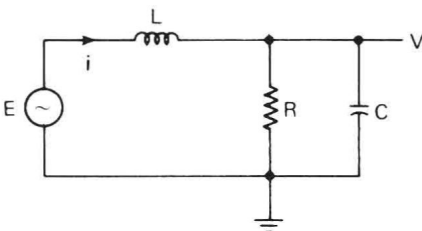


Figure 1.2 Simple electric circuit.