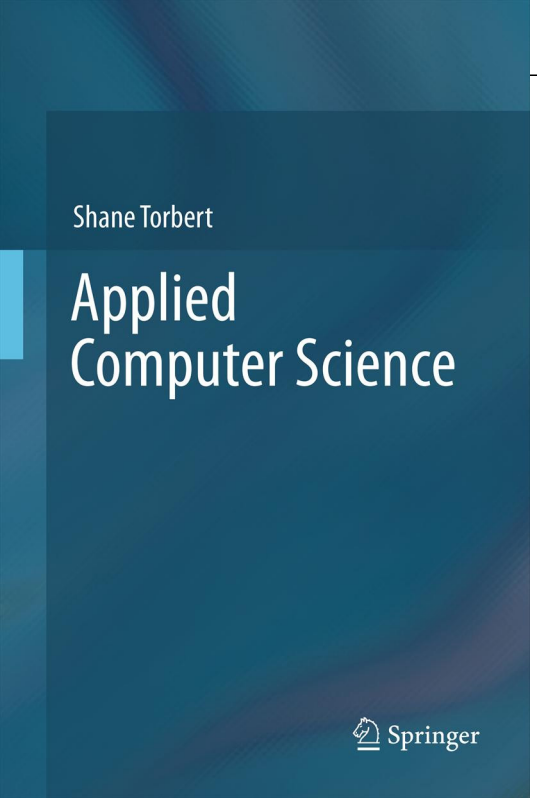
Shane Torbert
Applied
Computer Science
Springer

Applied Computer Science

Shane Torbert

Applied Computer Science

Springer

Shane Torbert
Center for Computational Fluid Dynamics
George Mason University
Fairfax, Virginia, USA

ISBN 978-1-4614-1887-0 e-ISBN 978-1-4614-1888-7
DOI 10.1007/978-1-4614-1888-7
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2011941876

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

For my wife

I don't believe that talent belongs to a certain place... it's spread equally. The problem is, in very few places we create conditions to help this talent to be discovered and nurtured.

–Garry Kasparov

Contents

Chapter 1
Simulation

Experiments are often limited by a high level of danger, and because they are too expensive or simply impossible to arrange, such as in developing new medical treatments, vehicles for space flight, and also studying geologic events. In these cases we may benefit from the use of computer simulation to refine our understanding and narrow our investigation prior to an "official" observation. Since the promise of this technique is to accelerate information gathering for a relatively low total cost, interest has been gaining momentum *everywhere*.

Examples in this chapter include discrete, continuous, and interactive systems with particular importance given to the long-term (asymptotic) trend as the scale of a problem grows toward infinity. Our approach favors clarity and context over trivia or theory with the hope that this better enables early success, but of course in learning there are never any real guarantees. Efficiency matters but is not paramount as we prefer to implement more widely accessible solutions while perhaps merely suggesting a state-of-the-art method.

For all code listings we use Python 2 with Tk and PIL, plus gnuplot, but other options are available (e.g., Matlab®, *Mathematica*, Python 3, Java, C/C++, Fortran, Scheme, Pascal) and our belief is these choices are sufficiently intuitive to serve as instructive pseudocode, that also happens to run!, for any environment you use. Regardless, we feel the central issues up-front are quality problems, a thoughtful sequencing of topics, and rapid feedback.

1.1 Random Walk

Imagine yourself outside on a pleasant day looking for a nice place to sit and read. By chance you are standing exactly halfway between your two favorite spots but are unable to decide which one to take. Out of curiosity you engage in a rather obscure process to settle the matter: you flip a coin and take one step in the indicated direction, then flip again followed by another step, and again and again, moving back-and-forth as the coin dictates, possibly coming very close to one spot or the

S. Torbert, *Applied Computer Science*, DOI 10.1007/978-1-4614-1888-7_1,

Table 1.1: Three examples of a size $n = 5$ random walk.

```
-----X-----
----X|-----
-----X-----
----X|-----
---X-|-----
--X--|-----
-X---|-----
--X--|-----
-X---|-----
--X--|-----
---X-|-----
--X--|-----
-X---|-----
X----|-----
-----|-----
Steps:  14
```

```
-----X-----
-----|X----
-----|-X---
-----|X----
-----X-----
----X|-----
---X-|-----
----X|-----
-----X-----
----X|-----
---X-|-----
--X--|-----
-X---|-----
--X--|-----
-X---|-----
X----|-----
-X---|-----
X----|-----
-----|-----
Steps:  18
```

```
-----X-----
-----|X----
-----|-X---
-----|X----
-----X-----
-----|X----
-----|-X---
-----|--X--
-----|-X---
-----|--X--
-----|-X---
-----|X----
-----|-X---
-----|--X--
-----|---X-
-----|----X
-----|-----
Steps:  16
```

other but ending only when a final step brings you all the way there. We wish to simulate this random drifting process with computer code. Each lab is numbered so that the first digit indicates chapter (1-7), the second a particular problem (1-3) in the chapter, and the third digit your specific assignment (1-5) related to the problem.

Lab111: Trace of a Single Run

We begin with output as shown in Table 1.1 where variables specify the size of our walk and also track current position. If n is the distance from the halfway point to the edge just next to either destination then $m = 2n+1$ is the total distance available for drifting.

Code Listing 1.1: Initializing variables.

```
#
n=5
m=2*n+1
j=  n+1
#
```

The values of n and m remain constant here but our current position j will start at the halfway point $j = n+1$ and then change as we drift until either $j = 0$ or $j = m+1$ and we have arrived at a reading spot. Whenever $j = 1$ or $j = m$ we are at one of the two edges, needing but a single step more in that direction to complete our walk. Commands to initialize and update these variables "over time " are shown in Code Listings 1.1 and 1.2, respectively.

Code Listing 1.2: A complete walk's loop.

```
while 1<=j<=m:
   #
   if random()<0.5: # coin flip
      j+=1
   else:
      j-=1
   #
```

Of course we want to *see* the walk, too. Once we know everything is working properly this "trace" will be less important but we should first verify that our code is behaving as expected. We could just print the value of j at each step but it will be better if we draw a "picture" instead. Helper variable k loops over the entire drifting area in Code Listing 1.3 to display a single row from any of Table 1.1's examples.

Code Listing 1.3: A loop to display the entire drifting area.

```
k=1
while k<=m:
   if k==j:
      print 'X',     # current position
   elif k==n+1:
      print '|',     # halfway point
   else:
      print '-',
   k+=1
print
```

Your first assignment is to put all this code together into a 1-D random walk simulation, including counting-up the total number of steps.

Questions to consider:

- Do our examples represent a typical size $n = 5$ random walk?
- What happens to the number of steps, on average, as n increases?

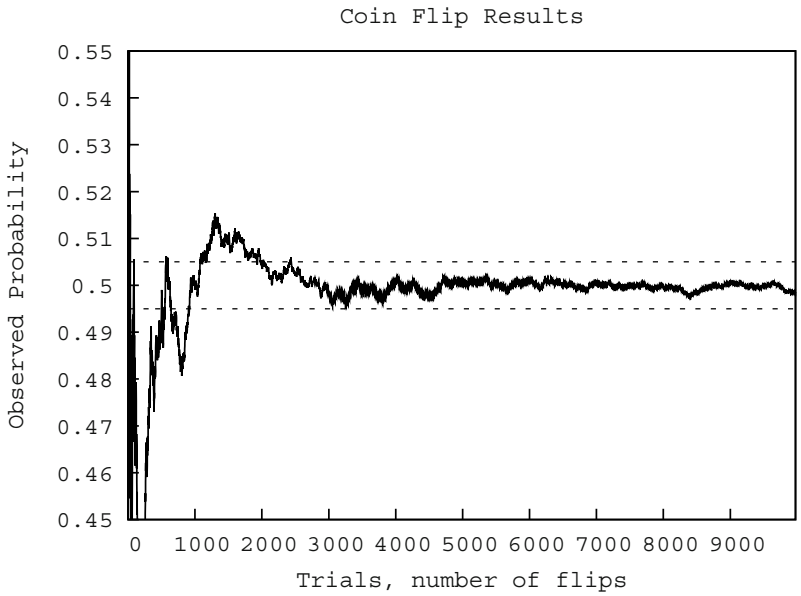

Fig. 1.1: Law of large numbers applied to coin flips.

Lab112: Law of Large Numbers

Before proceeding we ask an essential question: After how many coin flips are we reasonably sure the heads-tails ratio is "close" to even? Your assignment is to write a small program that only flips coins, over and over and over again, to calculate the percentage of heads, or tails, obtained. Code Listing 1.4 is a sample gnuplot script for a plot similar to the one shown in Figure 1.1, where we assume the program's output (total observed probability after each trial) has been stored in a plain text file.

Code Listing 1.4: Sample gnuplot script.

```
set terminal png
set output "lab112.png"
set title  "Coin Flip Results"
set ylabel "Observed Probability"
set yrange[0.45:0.55]
set ytics 0.01
plot "lab112.txt" with lines notitle,0.505 w l notitle
```

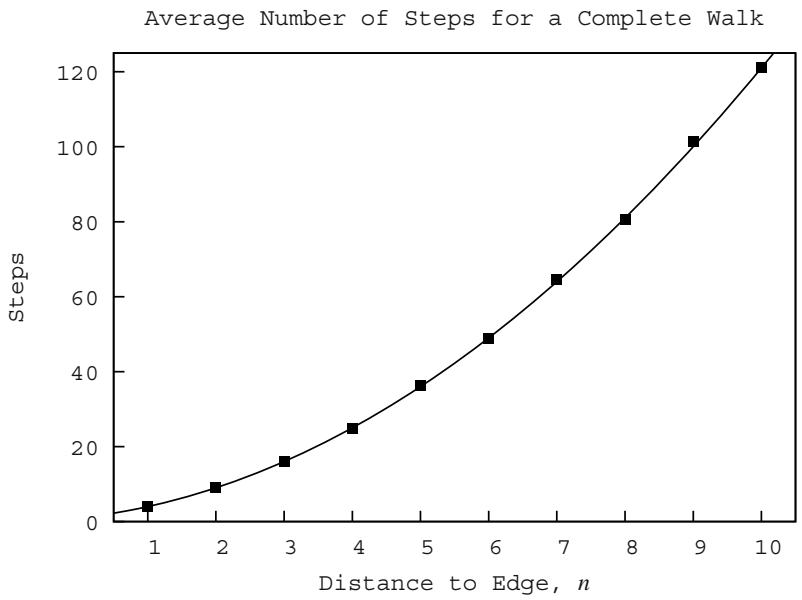

Fig. 1.2: Quadratic growth in the average number of steps to complete a walk.

Lab113: Scaling the Problem Size

We show the average number of steps for walks of size $n \leq 10$ in Figure 1.2 overlayed with the curve $f(n) = (n+1)^2$ since it takes $n+1$ total steps in a particular direction to actually end the walk. A trial now is an entire random walk, not just a single coin flip, and we use Algorithm 1.1.1 and $10,000$ trials for accurate results.

Algorithm 1.1.1 Average number of steps for various n.

```
1:  while n ≤ n_max do
2:     steps = 0
3:     while t = 1 → trials do
4:        while random_walk do
5:           ...
6:           steps = steps + 1
7:        end while
8:     end while
9:     print n, steps/trials
10: end while
```

Fig. 1.3: A large plume of ash during an eruption of the Mt. Cleveland volcano, Alaska, as seen from the International Space Station on 23 May 2006. Image courtesy of the Image Science and Analysis Laboratory, NASA Johnson Space Center.

Transport and Diffusion

Observation of the physical world often reveals behavior that may be explained at least partially by drifting. For instance, the plume of ash shown in Figure 1.3 moves in two fundamental ways. First, wind blows the ash in some direction, a topic we will consider in more detail with the next problem. Second, the ash spreads out and eventually the plume will break-up in a process called *diffusion* that can be modeled by the random motion of individual ash particles. This means that our random walk, while an absurd method for picking a spot to read, does relate accurately to the diffusive aspect of a plume's movement over time.

However, volcanic eruptions may have a worldwide impact over many years, scales far too large for 3-D particle-by-particle calculations even for state of the art algorithms running night and day on the fastest computers in the world. Simulations of such cases require a team of experts (we need you!) to build different models, better algorithms, and bigger computers.

Fig. 1.4: Blue Mountain supercomputer, Los Alamos National Laboratory, New Mexico, one of the most powerful computing systems in the world at the end of the twentieth century. Image courtesy of Los Alamos National Security, LLC.

Parallel Processing

One popular technique for large-scale simulations is the use of multiple systems connected together to process sub-units of the overall code in parallel. For instance, individual trials in the previous lab are independent of each other and therefore may be run simultaneously on separate machines in order to speed up calculation.

The parallel computing system shown in Figure 1.4 was one of the world's most powerful supercomputers. It has since been decommissioned and not even a decade later the same level of capability became commercially available in graphics cards, also massively parallel but no longer the exclusive domain of a premier national lab the use of whose image requires:

> Unless otherwise indicated, this information has been authored by an employee or employees of the Los Alamos National Security, LLC (LANS), operator of the Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396 with the U.S. Department of Energy. The U.S. Government has rights to use, reproduce, and distribute this information. The public may copy and use this information without charge, provided that this Notice and any statement of authorship are reproduced on all copies. Neither the Government nor LANS makes any warranty, express or implied, or assumes any liability or responsibility for the use of this information.

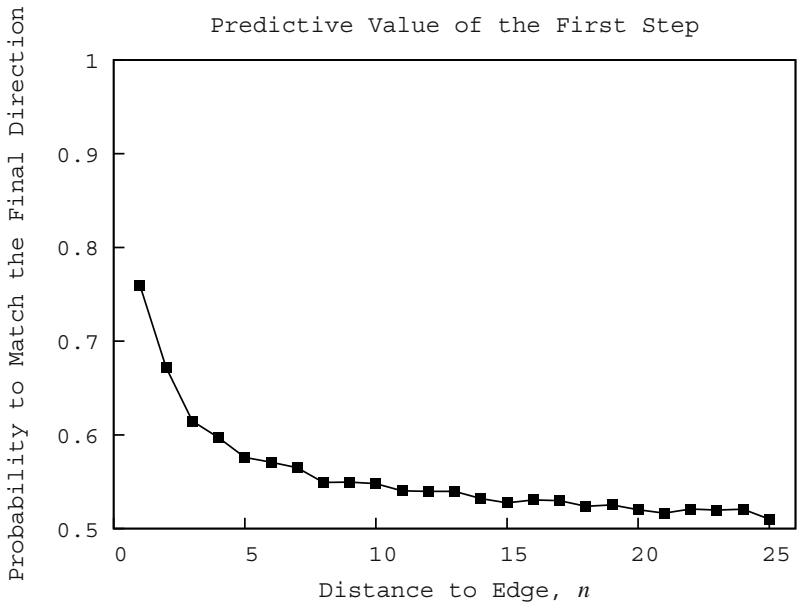

Fig. 1.5: As the size of the walk increases the importance of the first step decreases.

Lab114: Predictive Value of the First Step

We want to know how important the first step is in determining the final direction our drifting leads. Code Listing 1.5 shows how the first step can be "remembered" for later comparison after the entire walk has finished. Your job is to count the number of times this very first step matches the final direction. Results for $n \leq 25$ shown in Figure 1.5 reflect again the use of $10,000$ trials for each size.

Code Listing 1.5: Remembering the first step for later comparison.

```
j=n+1
if random()<0.5:
   j+=1
else:
   j-=1
first_step=(j-(n+1)) # either 1 or -1
#
while 1<=j<=m:
   #
   ...
```

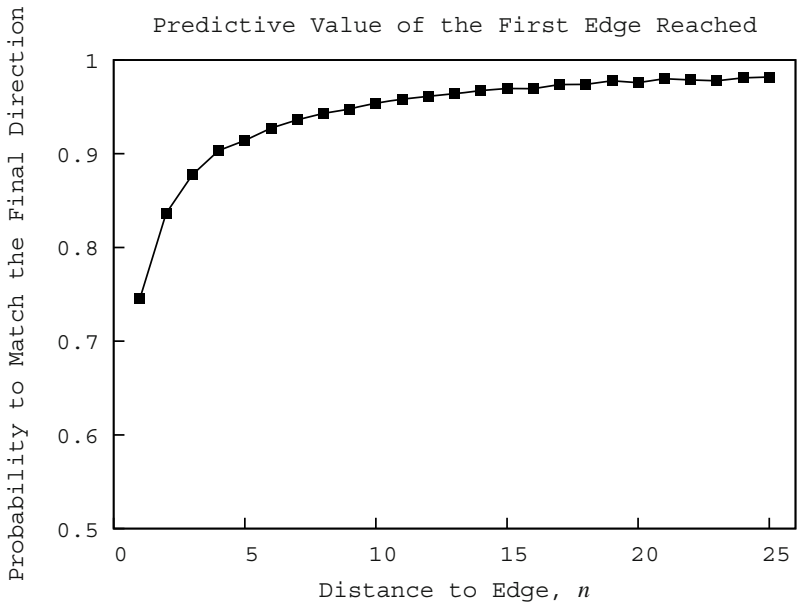

Fig. 1.6: The importance of the first edge reached increases with the size of the walk.

Lab115: Predictive Value of the First Edge Reached

Asking a similar question about the first edge allows us to use the symmetry of our problem to speed up runtime. It is not necessary to start in the middle and wait for the drifting to eventually reach an edge; instead, just start on an edge at step one! Results for $n \leq 25$ shown in Figure 1.6 are consistent regardless of which edge we start on and again reflect the use of $10,000$ trials for each size.

Code Listing 1.6: Using symmetry to speed up a first edge calculation.

```
j=m
#
while 1<=j<=m: # walk starts at edge
    #
    ...
    #
#
if j==m+1:     # was there a match ?
    #
    ...
```

1.2 Air Resistance

Imagine some friends in a sunny field kicking the soccer ball around. If the ball leaves your foot at 60 miles per hour and forming a 30 degree angle with the ground then how far does it go? What does its trajectory look like? Such questions are often asked in math and science classrooms* but our investigation will approach this problem from the standpoint of computer simulation.

Lab121: Deconstructed Parabola

We begin by neglecting air resistance but it will be clear that, using our approach, including air resistance in the model is straightforward. In all cases we restrict ourselves to 2-D and some behavior, such as lift and spin, will not be considered.

The soccer ball's velocity and position are initialized in Code Listing 1.7 where we assume $v_0 = 26.82$ meters per second and $\theta = \pi/6$ radians have already been converted from mph and degrees, and `cos` and `sin` are imported from `math`.

Our main loop shown in Code Listing 1.8 updates the position (x,y) based on the velocity (v_x, v_y) with v_x constant (for now) and v_y itself updated by the "pull" of gravity, $g = -9.81$ m/s^2 near the surface of the Earth.

Code Listing 1.7: Initializing variables.

```
#
vx=v0*cos(theta)
vy=v0*sin(theta)
#
x=0.0
y=0.0
```

Code Listing 1.8: A complete trajectory's loop.

```
#
while y>=0.0:
    #
    x+=(vx*dt)
    y+=(vy*dt)
    #
    vy+=(g*dt)
    #
```

* See, for instance: R. Larson et al., *Algebra 2*. McDougal Littell, 2004.

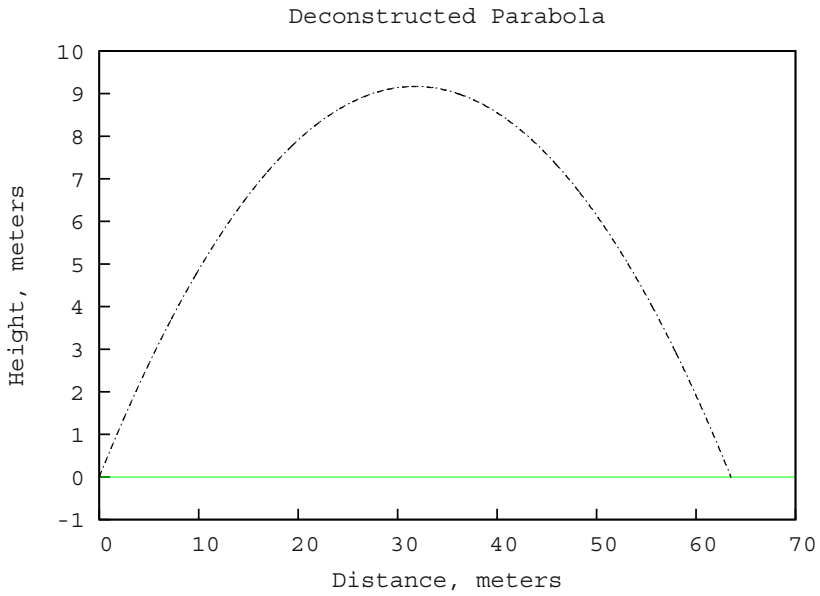

Fig. 1.7: Trajectory plot without air resistance.

Obviously v_y changes since what goes up must come down. We can think of v_x as a non-diffusive "transport" similar to the previously unexamined aspect of the ash plume's motion. We use a timestep $dt = 0.001$ seconds and total time t can also be tracked. If our loop outputs (t,x,y) at each step then Code Listing 1.9 plots the overall (x,y) trajectory as shown in Figure 1.7.

Code Listing 1.9: Trajectory gnuplot script.

```
set terminal png
set output "lab121.png"
set title  "Deconstructed Parabola"
set xtics nomirror
plot "lab121.txt" using 2:3 w l notitle,0 w l notitle
```

Two observations about a soccer ball's motion make this simulation possible:

- Horizontal and vertical movement can be treated separately.
- Small enough timesteps allow for straightforward modeling.

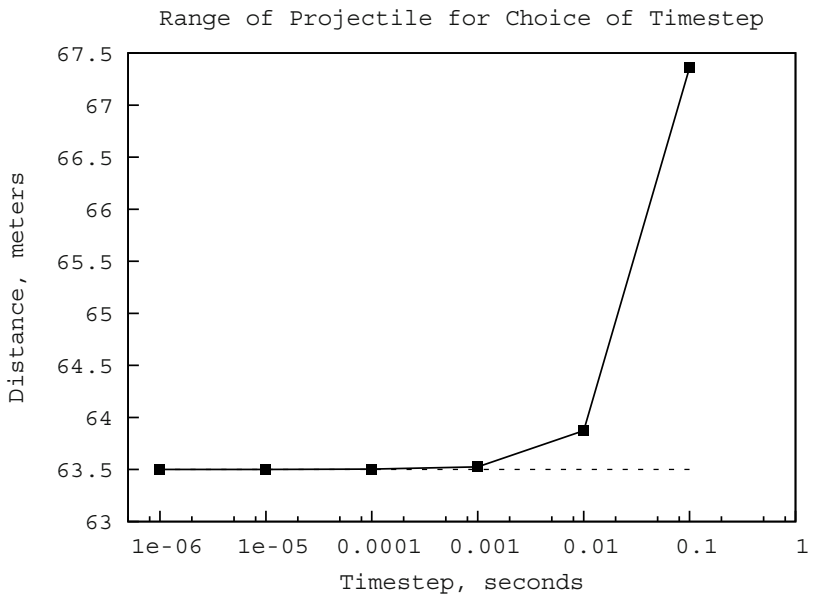

Fig. 1.8: Logarithmic scale for x with `set logscale x` in gnuplot.

Lab122: Timestep Study

The range of a projectile is horizontal distance traveled before impact; a test of timesteps for the soccer ball range is shown in Figure 1.8 with $dt = 10^{-N}$ and $N \leq 6$. Our results indicate $dt = 0.001$ is a good compromise (if $dt = 0.0001$ then our main loop would require 10 times the number of steps). Some anticipated objections:

- Use the exact formula $y = gt^2/2 + v_0 \sin(\theta)t + y_0$ instead of simulation.
  - First, with air resistance the exact formula will be more complicated to derive.
  - Second, our Figure 1.7 had the simulation and "exact" values in two different dash-patterns and yet they overlayed so well as to seem like only one, not two.
- Our choice of $dt = 0.001$ seconds gives an incorrect range.
  - It is hard to say precisely what is the correct range. Did the ball leave your foot at 60 mph or was it really 61 mph? Was the angle 30 degrees or 29 degrees?

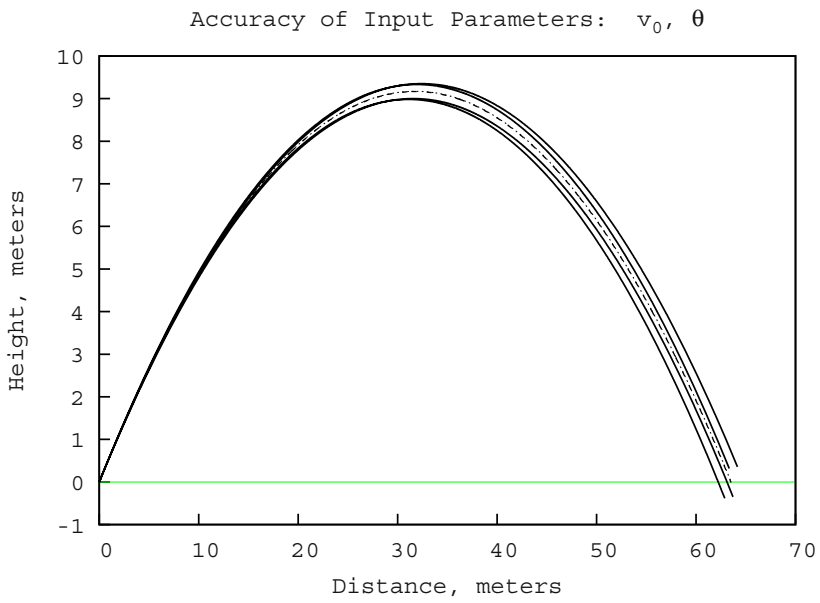

Fig. 1.9: Variation in trajectory for a 1% change in each input parameter.

To elaborate on this, Figure 1.9 shows the variation in trajectory corresponding to a 1% change in each input parameter, using the exact formula. Results are closer when only the angle changes but still the two dashed curves clearly coincide best. Even comparing the range to its exact value $x = v_0^2 \sin(2\theta)/|g|$, the horizontal line from Figure 1.8, our simulation with $dt = 0.001$ seconds is accurate to within one-tenth of one percent.

Lab123: Physical Model

Since the simulation is now working we wish to augment our model $a_x = 0$, $a_y = g$ with terms that reflect air resistance. Code Listing 1.10 shows how this might look.

Code Listing 1.10: Acceleration that varies with velocity.

```
#
ax=( -c1*vx)
ay=(g-c1*vy)
#
```