Modeling Projectile Trajectories

New Mexico High School
Adventures in Supercomputing Challenge
Final Report

April 5, 2006
11:59:56 P.M.

Team Number 045
"*The Penguins International*"
Las Cruces High School, Las Cruces, NM

TEAM MEMBERS:

ABRAHAM SANOGO, Freshman
JEFFREY WANG, Freshman
NATHAN SMITH, Freshman
TOM BURNHAM, Sophomore
CALEB ROTH, Sophomore

TEACHER: WE SHAN'T SPARE HIS EMBARASSMENT:

GREG MAREZ

E.0 Executive Summary

The main purpose of this study is to investigate the factors and variables which influence the trajectory of a projectile, find the significance of said influence on a projectile's trajectory and present appropriate mathematical models to execute the purpose.

The study is intended to promote knowledge of the projectile as it is in flight by using an in depth study which takes apart the trajectory and analyzes its components and their influence on the trajectory. The mechanics of projectile motion are investigated and the physical properties describing projectile motion are developed for the purpose of examination.

.

1.0 Introduction

1.1 Purpose

The purpose of the report is designed to promote a better understanding of projectile motion, and the physical properties that projectile motion possesses. These physical properties include air resistance (drag), velocity vectors, gravity and cross sectional area, just to name a few.

The project seeks to investigate namely two things. The first is a comparison between different methods of simulating and calculating a projectile's trajectory. This shall use a common method of calculating projectile motion, the vacuum method, where air resistance is negligible, and compare the results against a more complex, yet accurate method, where air resistance is included.

In doing so, the first stage of the project shall highlight the differences between the two methods of projectile motion.

The second part of the project is investigating and simulating the affects of wind on a projectile in flight. The purpose of this study is to simulate and highlight the affects that wind can have on a projectile, whether it be an increase in time or windspeed.

1.2 Scope

The project develops two Java (Sun-Microsystems) programs, each with its own specific task. The first program executes the corresponding purpose as stated above, as well as the second which executes its own corresponding purpose. The mathematics and physics of projectile motion are studied through equations that simulate the behavior of a projectile in flight.

1.3 Computer Program

The program is written using the Java (Sun-Microsystems) language. In actuality, there exists two programs which together constitutes the project itself. The computer program looks for specific values contained at a certain time interval as specified by the user to output. The output can be graphed easily by using Microsoft Excel or a similar program.

2.0 Project Methods

2.1 The Programs

The first program was a java class that contained two main methods, with other miscellaneous methods. The main two methods were the air resistance and vacuum trajectory equations.

The table below (1.1) shows the variables used in the program for calculating a vacuum trajectory (sans air resistance). To the far left are the variables in their traditional mathematical format. To the right of that are their names, followed by units and to the far left are their coded names as used in the program:

1-1: SANS AIR RESISTANCE VARIABLES:

Variable	Name	Units	Coded Name
v_0	Initial Velocity	m/s	InVel
×	Horizontal Displacement	m	HztDsp
y	Vertical Displacement	m	VrtDsp
x_0	Initial Horizontal Displacement	m	InHztDsp
y_0	Initial Vertical Displacement	m	InVrtDsp
t	Time (0.001)	s	t
V_x	Horizontal Velocity	m/s	HztVel
V_y	Vertical Velocity	m/s	VrtVel
g	Gravitational Acceleration	m/s^{-1}	g
θ	Angle of Launch	°	LauAng
θ	New Angle	°	NAng

The table below (1.2) shows the same variables used again in the mathematical format to create the model for calculating the vacuum trajectory. To the left are the variables, and to their right are their traditional mathematical equations. To the right of those are the equations as they appear in the program.

1-2: SANS AIR RESISTANCE OUTPUT FORMULAS:

Variable	Equation	Coded Equation
V_y	$v_0\sin\theta + g(t)$	VrtVel = (Invel * (Math.sin (LauAng))) + (g * t)
V_x	$v_0\cos\theta$	HztVel = Invel * (Math.cos (LauAng))
y	$y_0 + (v_0\sin\theta) + 0.5(gt^2)$	VrtDsp = InVrtDsp + (InVel * (Math.sin (LauAng))) + (0.5 * t * g ^2)
x	$x_0 + (v_0\cos\theta)t$	HztDsp = InHztDsp + (InVel * (Math.cos (LauAng)) * t);

The table below shows the variables used in air resistance models, where the extra variables used are shaded in gray.

2-1: AIR RESISTANCE VARIABLES:

Variable	Name	Units	Coded Name
v_0	Initial Velocity	m/s	InVel
×	Horizontal Displacement	m	HztDsp
y	Vertical Displacement	m	VrtDsp
x_0	Initial Horizontal Displacement	m	InHztDsp
y_0	Initial Vertical Displacement	m	InVrtDsp
t	Time (0.001)	s	t
V_x	Horizontal Velocity	m/s	HztVel
V_y	Vertical Velocity	m/s	VrtVel
g	Gravitational Acceleration	m/s^{-1}	g
θ	Angle of Launch	°	LauAng
A_x	Horizontal Acceleration	m/s^{-1}	NAng
A_y	Vertical Acceleration	m/s^{-1}	VrtAcc
A	Cross Sectional Area	m^2	CSArea
F_x	Horizontal Force	N	HztFor
F_y	Vertical Force	N	VrtFor
C	Drag Coefficient	N/A	DrgCo
R	Drag Force	N	DrgFor
m	Mass	Kg	m
ρ	Air Density	Kg/m^3	ArD

The table below shows the equations used in air resistance models:

2-1: AIR RESISTANCE OUTPUT FORMULAS:

Variable	=	Equation	Coded Equation
R	=	$-0.5(C\rho)\ AV^2$	DrgFor = -0.5 * (DrgCo * ArD) * CSA * V ^ 2
A_x	=	R/m	HztAcc = (DrgFor / m)
x	=	$x_0 + Vx(t) + 0.5(A_x)\ 2t$	HztDsp = (InHztDsp + (HztVel * t) + (0.5 * HztAcc * t * 2))
V_x	=	$V_0\cos\theta + A_x(t)$	HztVel = (Invel * (Math.cos (LauAng)) + (HztAcc * t))
F_y	=	m(g) + R	VrtFor = ((m * g) + DrgFor)
A_y	=	[(m)g + R] / m	VrtAcc = (((m * g) + DrgFor) / m)
y	=	$y_0 + Vy(t) + 0.5(A_y2)$	VrtDsp = (InVrtDsp + (VrtVel * t) + (0.5 * VrtAcc * 2))
V_y	=	$V_0\cos\theta + A_y(t)$	VrtVel = (InVel * (Math.cos (LauAng))) + (VrtAcc * t)

The program itself is contained in the Appendix.

3.0 Results & Conclusion

It is most unfortunate, however, that the project could not be accomplished within the time given. The path to finding a suitable model in which to model our program has been a tortuous road, where one source's method contradicted the other and no possible solution seemed eminent. It is nevertheless our deepest regrets in not doing so; despite such a setback, the team is resolute and does plan to have respectable results in by the time of the Expo.

APPENDIX

Program Files:

```
import javax.swing.*;
//
/** AirProjectile.java  */
//
/*   Methods included are
 *      calculateAirDensity  (only used inside of AirProjectile class)
 *      inputVaribles
 *      runTrajectory
 *      startTrajectory
 */
/**
 *
 * @author Abraham Sanogo
 */
public class AirProjectile {

    double CroSecArea;                  //Cross-Sectional Area
    double Velocity;                //Velocity vector direction X,Y
    double Density;                 //Density of the bullet, used for Mass
    double Mass;              //
    double Caliber;                 //User needs enter caliber only, not mass
    double Launch_Angle;            //Initial Angle of Launch
    double Altitude, latitude;
     //
//Input variables
//
//Altitude, both initial and cumulative
//
// Variables that the user never sees
//
    double DCon;              //A constant used in Drag_Force
    double Air_Density;                 //
    double Acceleration_X;          //
    double Acceleration_Y;          //
    double Velocity_X;                  //Velocity vector direction X
    double Velocity_Y;                  //Vector Y
    double DragForce;                   //
    double Displacement_X;          //Vector X
    double Displacement_Y;          //Vector Y
    double New_Angle;
    double balCoef;//
//
//Output Varaible[s]
//
    double Distance;                //Cumulative distance of the projectile
//Altitude should be here, but doubles as an intial input

//
```

```
    double g;                           //Gravity, initialized to 0, reset by
g(n)
//
    final double G1 = 9.80665;            //Gravity values based on altitude,
    final double G2 = 9.80655;            // "
    final double G3 = 9.80645;            // "
//
    double Air_Den=0;                   //Air Density, initialized to0, reset
by Air_Den(n)
//
    final double Air_Den1 = 1.2255; //Air Density based on altitude
    final double Air_Den2 = 1.1677; // "
    final double Air_Den3 = 1.112;  // "
    final double Air_Den4 = 1.0583; // "
    final double Air_Den5 = 1.0067; // "
    final double Air_Den6 = 0.957;  // "
//
    final double timeInterval = 0.01;                //Time interval

//------------------------------------------- //
//                  STRINGS                   //
//------------------------------------------- //
    String Velocity2;                         //
    String Caliber2;                          //
    String Density2;                          //
    String Altitude2;                         //
    String Latitude2;                         //
    String Launch_Angle2;                   //
//-------------------------------------------//
  //
  /** Creates a new instance of AirProjectile */
  //
    public AirProjectile() {
    CroSecArea=0;       //Cross-Sectional Area
    Velocity=0;               //Velocity vector direction X,Y
    Density=0;                //Density of the bullet, used for Mass
    Mass=0;             //
    Caliber=0;                //User needs enter caliber only, not mass
    Launch_Angle=0;           //Initial Angle of Launch
    Altitude=0;
    latitude = 0;
     //
 //Input variables
 //
                  //Altitude, both initial and cumulative
//
// Variables that the user never sees
//
    DCon=0;             //A constant used in Drag_Force
    Air_Density=0;            //
    Acceleration_X=0;         //
    Acceleration_Y=0;         //
    Velocity_X = 0;           //Velocity vector direction X
    Velocity_Y=0;       //Vector Y
    DragForce = 0;            //
    Displacement_X=0;         //Vector X
    Displacement_Y=0;         //Vector Y
```

```
    New_Angle=0;
    balCoef = -.5;            //Ballistic Coefficient
//
//Output Varaible[s]
//
    Distance=0;               //Cumulative distance of the projectile
//Altitude should be here, but doubles as an intial input

//
    g = 0;

 }  // End of instance variables

//
//   Methods ******************
//


 void startTrajectory()
    {
    Mass = ( (4/3 * Math.PI) * Math.pow (Caliber / 2, 2) ) * Density;
    this.calculateAirDensity();

    //
    DCon = (Air_Den * balCoef * CroSecArea) / 2;
    DragForce = DCon * (Math.pow(Velocity, 2));
//
//Iteration 1-----------------------------------------------------------------
------------------------------

    Velocity_X = (Math.cos(Launch_Angle)) * Velocity * timeInterval;
    Velocity_Y= (Math.sin(Launch_Angle)) * Velocity * timeInterval;
    Altitude = Velocity_Y * timeInterval;
    Acceleration_X = -(DCon / Mass)* (Velocity * Velocity_X);
    Acceleration_Y =  (-g)  - (DCon/Mass * Velocity * Velocity_Y);

    Distance = Velocity_X * timeInterval;

    Displacement_X= (Velocity_X * timeInterval) +
            (0.5 * Acceleration_X * (Math.pow(timeInterval, 2)) );
    Displacement_Y= (Velocity_Y*timeInterval) +
            (0.5 * Acceleration_Y * (Math.pow(timeInterval, 2)) );

    New_Angle =  Math.atan(Displacement_X / Displacement_Y);
    }
//
//This is our main method:
//Iterations 2(+)-------------------------------------------------------------
----------------------------
void  runTrajectory()
    {
    if (Altitude>=-10)
        {
      Velocity_X = (Math.cos(New_Angle)) * Velocity * timeInterval;
      Velocity_Y= (Math.sin(New_Angle)) * Velocity * timeInterval;
      Velocity = Math.sqrt(  (Math.pow(Velocity_Y, 2)) +
(Math.pow(Velocity_X, 2)) );
```

```
                        //Velocity = a^2 + b^2

      Altitude = Altitude + Displacement_Y;
      JOptionPane.showMessageDialog(null,"Y=" + Altitude);
        this.calculateAirDensity();

      DCon = (Air_Den * balCoef * CroSecArea) / 2;
      DragForce = DCon * (Math.pow(Velocity, 2));

      Acceleration_X = -(DCon / Mass)* (Velocity * Velocity_X);
      Acceleration_Y = (-g) - (DCon/ Mass * Velocity * Velocity_Y);

      Displacement_X= (Velocity_X*timeInterval) +
                (0.5 * Acceleration_X * (Math.pow(timeInterval, 2)));
      Displacement_Y= (Velocity_Y*timeInterval) +
                (0.5 * Acceleration_Y * (Math.pow(timeInterval, 2)));

      Distance = Distance + Displacement_X;
            JOptionPane.showInputDialog("X=" + Distance);
                  //OK, here's where I use JOption instead of "return" to
return a value
      New_Angle =  Math.atan(Displacement_X / Displacement_Y);
        }
    else

      JOptionPane.showInternalMessageDialog(null," Thank You For Using
Penguins " +
                "International Program Software");
    }

//End

//
//Input variables method
//
    void  inputVariables ()
    {
     Velocity2 = JOptionPane.showInputDialog("Please Enter Muzzle
Velocity:");
      Velocity = Double.parseDouble( Velocity2 );
    Caliber2 =JOptionPane.showInputDialog("Please Enter Caliber:");
      Caliber = Double.parseDouble( Caliber2 );
    Density2 = JOptionPane.showInputDialog("Please Enter Bullet Density:");
      Density = Double.parseDouble( Density2 );
    Altitude2 = JOptionPane.showInputDialog("Please Enter Initial
Altitude:");
      Altitude = Double.parseDouble( Altitude2 );
    Latitude2 = JOptionPane.showInputDialog("Please Enter Latitude:");
      latitude = Double.parseDouble( Latitude2 );
    Launch_Angle2 = JOptionPane.showInputDialog("Please Enter Angle of
Launch:");
      Launch_Angle = Double.parseDouble( Launch_Angle2 );
    }


//
//  This initializes the air density based on altitude
```

```
//
void  calculateAirDensity()
    {

      if (Altitude==0 && Altitude<500){
                  Air_Den = Air_Den1;
                  g = G1;     }
            if (Altitude==500 && Altitude<1000){
                  Air_Den = Air_Den2;
                  g = G1;     }
            if (Altitude==1000 && Altitude<1500){
                  Air_Den = Air_Den3;
                  g = G2;     }
            if (Altitude==1500 && Altitude<2000){
                  Air_Den = Air_Den4;
                  g = G2;     }
            if (Altitude==2000 && Altitude<2500){
                  Air_Den = Air_Den5;
                  g = G3;     }
            if (Altitude==2500 && Altitude<3000){
                  Air_Den = Air_Den6;
                  g = G3;     }
    return;
    }
}//---------------------------------
```

```
import javax.swing.*;
/*
 * TestAirProjectile.java
 *
 * Created on March 23, 2006, 8:10 AM
 *
 */

/**
 *
 * @author Abraham Sanogo
 */
public class TestAirProjectile {

    public static void main(String[] args) {
        AirProjectile projectile;               // create a variable of type
AirProjectile
        projectile = new AirProjectile();       // creates the AirProjectile
object and
                                                //    references it by the
name projectile
        projectile.inputVariables();            // calls the method
inputVariables
```

```
        projectile.runTrajectory();             // calls the method
runTrajectory
        projectile.startTrajectory();           // calls the method
startTrajectory

        JOptionPane.showMessageDialog(null, "Your Projectile's Trajectory " +
                 "Is Currently Being Calculated…Please Standby:");
//
    }

}
```