Rocket Sled Trajectory Simulation Language (RSTSL)

Project Proposal

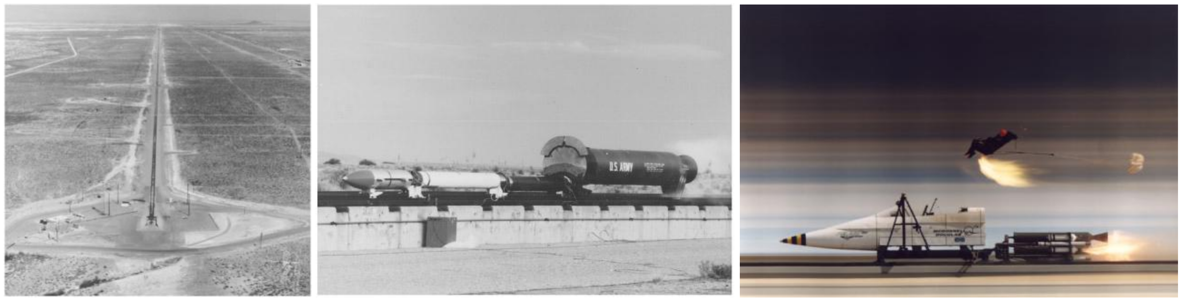
COMS W4115 – Programming Languages and Translator

Rui Chen

9/29/2010

1. Introduction

The purpose of Rocket Sled Trajectory Simulation Language (RSTSL) is to provide an engineering tool for constructing one degree of freedom (1DOF) trajectory simulations of rocket sleds. Rocket sled trajectory simulations are performed at several weapon test ranges across the country to support testing of aircraft ejection seats, munitions, and other weapon systems at subsonic to hypersonic speeds. Two of the most famous rocket sled test facilities are Holloman High Speed Test Track (HHSTT) in Holloman AFB, NM[1], and the Supersonic Naval Ordinance Research Track (SNORT) in China Lake, CA[2].



(a)[3] (b)[3] (c)[4]

Figure 1: Illustrations of Rocket Sled Testing

Rocket sled tests are performed on rail tracks such as those shown in Fig. 1(a) which is a photograph of the 10-mile long test track at Holloman High Speed Test Track. One example of rocket sled testing is supersonic impact testing of “bunker busters”. As shown in Fig. 1(b), a warhead is mounted on a single track and accelerated to supersonic speeds (for example Mach 3.6, 4000ft/s) by three different solid-propellant rocket motor stages. A typical objective of this type of test is to evaluate the ability of a warhead to destroy hardened underground bunkers protected by layers of reinforced concrete and soil. Another typical example is the test of ejection seats at transonic flight conditions. Figure 1(c) shows a manikin being ejected from an F-16 aircraft forebody to validate ejection seat design.

Sled trajectory simulations play a pivotal role in rocket sled testing. For example, trajectory simulations need to be performed to design the propulsion configuration so that the test articles can achieve the desired velocity. Trajectory simulations are also used to ensure high-speed cameras are placed at the correct locations and configured to the correct speed to record events (such as seat ejection or store separation) at the desired velocity.

2. RSTSL Language Overview

Rocket Sled Trajectory Simulation Language (RSTSL) is intended to be a simulation environment specifically designed for performing rocket sled trajectory simulations. It will allow the user to rapidly develop programs that simulate single or multiple sled runs. For example, the user can write a program in RSTSL to find the best combination of sled propulsion configuration and initial conditions to achieve a certain test objective. For supersonic impact tests, the user may want to write a program to iterate through several solid rocket motor types, staging sequences, and launch positions to find the best way to accelerate the warhead to the desired velocity at impact. The user may also write a RSTSL program to study the effect of uncertainties in aerodynamic drag and rocket motor thrust predictions on the sled's trajectory.

The RSTSL will be able to handle multiple-stage sled train consisting of an unpowered forebody sled and a single or multiple "pusher" sled(s) on which rocket motors are mounted. An example of this kind of sled train is shown in Fig.1(b), which consists of three pusher sleds and one forebody sled. Since the pusher sleds are not linked, they will separate from the sled train after burnout. RSTSL will also be able to handle ejection-seat sleds where a forebody sled is connected to a pusher sled which does not separate, and several rocket motors are fired in stages to achieve the desired velocity and acceleration.

3. Simulating Sled Trajectories with RSTSL

This section will use an example rocket sled simulation to illustrate the syntax and application of RSTSL. As shown in Fig. 2, the sled train used for this example consists of a 500lb warhead mounted on a forebody sled and a pusher sled with a single "Type A" motor mounted on it. ("Type A" is an arbitrary designation given to a representative rocket motor. RSTSL will have 3 different types of representative rocket motors –Type "A", "B" and "C" from which the user may choose.) The objective of this simulation is to determine if the warhead can be accelerated to the desired impact velocity at the end of the track where the warhead will fly a short distance to impact a target. In addition, this example will also calculate the effect of 20% over and under predication of aerodynamic drag on sled trajectory.

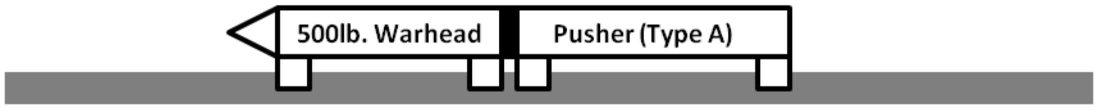


Figure 2: Example Sled Train Configuration

3.1. Constructing the Simulation

To simulate a single sled run, the user first creates a *Sim* object. The *Sim* object manages the entire 3-step simulation process: inputs, solving equation of motion (EOM), and outputs. For inputs, the *Sim* object contains data such as the initial conditions (velocity and position) of the run, atmospheric conditions (pressure and temperature for aerodynamic force calculations.) and time-step of the simulation. *Sim* object also contains one or multiple *Stage* objects which contain data on how each stage of the sled run is modeled. After all required input data have been stored in the *Sim* object, the *Sim* object numerically solves the 1DOF translation EOM with variable mass using the *Sim.run()* function which utilizes 4th order Runge-Kutta method. The outputs of *Sim.run()* method are arrays containing acceleration, velocity and position of the sled as a function of time. A single *Sim* object handles a single run. Multiple *Sim* objects need to be created to handle multiple runs.

To model a sled train's characteristics which include weight, aerodynamic drag, thrust and staging information, the user creates a *Stage* object for each stage or state of the sled train. In this example, two stage objects need to be created. Stage#1 models the characteristics of the forebody sled plus the pusher sled. Stage#2 models the characteristics of the forebody (after the pusher burns out and separates from the sled train). After all *Stage* objects have been initialized with the required data, they are added to *Sim* objects.

RSTSL also contains build-in commands for outputting simulation results on-screen or in an output file. RSTSL's loop structures will allow the user to iterate through multiple sled runs.

3.2. Syntax and Example Program

The syntax of RSTSL will be similar to that of MATLAB. White space can be used to separate keywords, operators and variables. The end of a line marks the end of each operation or command. The start of comments is marked by "%". Variables can be scalar or a vector. Vectors are constructed using syntax "[a, b, c…]" where "[ ]" encloses the vector and each element of the vector is separated by ",". Syntax "v(i)" is used to access *i*th element in vector v. 2D Matrices can be created using the following syntax: "m=[a,b,c;e,f,g]" where "," separates the columns and ";" separates the rows of a matrix. Syntax "m(i,j)" is used to access item in the *i*th row and *j*th column in matrix m. *Sim* and *Stage* objects are constructed by calling their constructors with the required argument, such as sledSim=Sim() and stage1=Stage(1500, "A"). The following is a representative program to solve the example problem describe in the beginning of this section. The comments explain each line of the program.

```
aeroFactors = [1, 0.8, 1.2] %define factors to apply to drag for sensitivity study
%The following defines aerodynamic drag table. The table has two rows, the first row contains the Mach number
%the second row contains the drag coefficients or CDA, the first and second row are separated by ";"
%the drag tables are interpolated linearly for drag calculations
aeroStage1 = [0.3, 0.8, 1.5, 2, 3; 0.8, 2, 1.6, 1.5]  %define nominal aerodynamic drag coefficients for Stage#1
aeroStage2 = [0.3, 0.8, 1.5, 2, 3; 0.6, 1.4, 1.2, 1.1] %define nominal aerodynamic drag coefficients for Stage#2
for i=1:1:3             %loop thru i=1 to i=3 with increment of 1
   sledSim = Sim()    %initialize new Sim object
   sledSim.initPos =30000    % initial track station of the warhead's tip in ft. (HHSTT end of track is at station 50788ft)
   sledSim.initV = 0    %initial velocity
   sledSim.dt = 0.001    %time step used by Runga-Kutta method to solve the EOM
   sledSim.atmP= 13.17    %ambient pressure in psi
   sledSim.atmT = 59    %ambient temperature in F
   stage1=Stage(1500, "A")    %defines new Stage object stage#1 with predefined rocket motor  Type "A"
   %the weight of stage1 is set to 1500lb(Forebody + Pusher weights)
   stage1.aero=aeroStage1    %assign aerodynamic table for stage1
   stage1.startTime=0.0    %define initial ignition or start time of stage1
   stage1.endTime=9.0    %define the burnout or end time of stagel
   stage1.aeroFact=aeroFactors(i) %define multiplication factor for aero coefficients
   stage2=Stage(700, "null")    %defines new Stage object stage#2 for the forebody with no propulsion
   %the weight of stage2 is set to 700lb (payload + sled structure weights, after the pusher sled separates)
   stage2.aero=aeroStage2    %assign aerodynamic table for stage2;
   stage2.aeroFact=aeroFactors(i)    %define multiplication factor for aero coefficients
   %no need to define startTime and endTime for stage2 since the start time of stage2 is the same as the end time for
   %stage1, and the simulation will terminate after the forebody reaches the end of track.
   sledSim.addStage ([stage1, stage2])    %add the stages to Sim object
   %run simulation and store the time, position, velocity and acceleration  data in arrays t,x,v,a
```

```
         [t, x, v, a]=sledSim.run(0.01);     %the results are saved every 0.01 seconds
        writeToFile("testRun"+i+ ".out", [t, x, v, a])    %write the results of the current run to a result file testRun#.out
    end
```

After executing the above program, three files: testRun1.out, testRun2.out, and testRun3.out will be generated. These files contain the warhead's position, velocity and acceleration as a function of time every 0.01 seconds from ignition to impact at the end of track. The results in testRun1.out shows the trajectory of the warhead with the nominal sled train predicted drag defined in aeroStage1 and aeroStage2. The results in testRun2.out show the trajectory of the warhead with 80% of the nominal predicted sled train drag. The results in testRun3.out show the trajectory of the warhead with 120% of the nominal predicted sled train drag. Using these results, a test engineer will be able to determine whether if the sled train configuration can achieve the desired velocity and assess the effect of inaccurate drag prediction on warhead impact velocity.

References:

1. Fact Sheets: 846th Test Squadron: http://www.holloman.af.mil/library/factsheets/factsheet.asp?id=5924
2. Supersonic Naval Ordinance Research Track, http://www.navair.navy.mil/ranges/LAND/docs/LAND/SNORT.pdf.
3. Dan Marren and Paul Zarchan, "Advanced Hypersonic Test Facilities", AIAA, 2002
4. NAVAIR Test Ranges, http://www.navair.navy.mil/ranges/LAND/index.htm