Trajectory Model User Guide (v0.06)

Oleg Mazonka, July-Aug 2017, personal notes

Contents

This document describes Trajectory Model (Artill) interface. This program is designed for calculating ballistic trajectories of projectiles flying through Earth's atmosphere. For more see "Empirical Data to Determine Transonic Drag Coefficient" paper.

Common interface

Trajectory Model is a program written in C++. It can compile into a standalone executable, static library, or shared library (DLL/SO). By design it has only one interface for three different usages as:

1. a standalone executable;
2. C++ library;
3. Java library using JNI.

The difference between usage as executable and library is the following. The executable has command line commands and the input and output files. It parses the command given from the command line, then loads necessary input data from files, executes, and finally writes the results into the output files. The library does not work with disk files. Instead the library interface has a function to set data as if it is a file – kind of virtual file stored in memory. Then the library user issues another call to the library consisting of a string representing the same command as it would be given in the command line. And finally the library user reads back the data from the virtual files using another library call.

This particular design makes support of the interfaces safe from desynchronization errors. The downside of this is that a higher level library interface has to be created for simpler usage. The following table shows the difference between the different usages of the interface.

Executable	C++ library	Java library
`> test`	`WorkClass x;` `x.run("test");`	`DragMod x = new DragMod();` `x.run("test");`

Creating a file "`hi`" with a content "`hello`"	`x.setio("hi");`
	`x.set_input("hello");`
	`cout << x.get_output();`
		`x.setio("hi");`
		`x.set_input("hello");`
		`System.out.println(x.get_output());`

`> range max`
read file "`out/trres.dat`"
	`x.run("range max");`
	`x.setio("out/trres.dat");`
	`cout << x.get_output();`
		`x.run("range max ");`
		`x.setio("out/trres.dat");`
		`System.out.println(x.get_output());`

The library API has 4 methods:

- Set input/output virtual file as current
  `C++: void setio(const string & name);`
  `Java: int setio(String name);`

- Set content of the current virtual file
  `C++: void set_input(const string & data);`
  `Java: int set_input(String data);`

- Get content of the current virtual file
  `C++: string get_output();`
  `Java: String get_output();`

- Execute command
  `C++: int run(const string & cmd);`
  `Java: int run(String cmd);`

The commands of executable are: **`help`**, **`model`**, **`solve`**, **`shoot`**, **`range`**, **`build`**, **`maps`**, and **`frep`**.

Optimisation

One of the features of the program is finding drag function given empirical trajectory data. This feature uses optimisation algorithm. At the moment of writing the optimisation solution is selected as a combination of several things. First, the local optimisation algorithm is Bobyqa taken from Dlib library. The objective function w consists of a difference function u, smooth function s, and smooth coefficient k:

$$w = u + us + ks$$

This particular form was selected after efficiency experiments. The difference function is a sum of relative differences between given values x and obtained values y:

$$u = \frac{1}{1000}\sum_{i=1}^{N}\left(\ln x_i - \ln y_i\right)^2$$

The smooth function is a combination of the average squares of first, second and fourth derivatives (for more details see the code). The smooth coefficient is ultimately selected zero. However to emulate global-ness of optimisation, different attempts are selected with different sequences of smooth coefficient before the final optimisation with $s = 0$.

Units

All physical units by default use SI system. Angles are by default defined in degrees. They also can be defined in radians or mils.

Commands

Execution is initialised by reading input files, and finalised by writing output files. All input files have the equivalent output files so they are not mentioned explicitly in the following description.

help
This command prints brief usage information.

shoot [angle]
This command makes a simple shot or shots at particular angles. If the angle is specified at the command the angles from the input file are ignored.

Input files required are:

```
cd<type>.dat
consts.dat
integr.dat
proj.dat
shot.dat
```

Output files are:

```
cdgrplt.dat
earth.dat
progr.dat
shots_summary.dat
trajs.dat
trres.dat
```

range [number|max]
This command solves the elevation angles for a particular range. If argument is missing it is assumed to be “max”. If the argument is “max”, then the program solves the angle for the maximal range. If the a number value is given, then the result is two elevation angles producing the required range. Input files required are the same as for “shoot” command. Output file is:

```
trres.dat
```

This file will have only two lines. The first line is for the flat trajectory, and the second line is for the high trajectory. If “max” is solved, the second line contains rubbish data.

model
This command loads dataset file and calculates the closes trajectory for each records in the dataset. For each known value in each record the difference function *u* is calculated (see formula in Optimisation section) and the overall result is printed. It is not stored in any output file because it does not represent any physical value. The function *u* is useful for optimisation, or as a hint of how bad drag function describes the empirical data in dataset. For details about records in dataset see the description of dataset.dat file. Output file:

`dataset.dat`

contains the calculated values for the given drag function.

solve
This command tries to find better parametrisation of the drag function. Input files are the same as for the command 'model'. Additional optional input file

`optim.dat`

defines one or more optimisation scenarios. Output file is the one corresponding to the input of the drag function.

build
This command does optimisation and increases fidelity of drag function representation. If the grad function is represented with a number of points, then after each round of optimisation new points are injected and optimisation is repeated. At this moment this process does not have a final condition, so it has to be interrupted by user.

frep
This is an ancillary functionality for re-parametrisation of functions. It has the format:

```
inputFile argument outputFile
```

Input or output file is a list of XY pairs. The argument can be either a number, a file name, or `@`number.

If the argument is a number, then a new function parametrisation is built with this number of equidistant X points. Y values of the points are adjusted so to minimise the quadratic difference of the resulted function from the original function assuming their linear interpolation between points. This is done by calling the same optimisation algorithm that is used for optimisation of drag function. The integral under the function is conserved.

If the argument is a file name, then the same process occurs as in the previous case, but X points are taken from the file, while Y values are ignored.

If the argument is a `@`number, then the output function is the closest function to the original with some points of the original function removed. The number specifies how many points should be left. No optimisation occurs in this case. The number of points can be less in the result function than the number asked. This may happened if more points can be removed without changing the value of the function. As in the other case the closeness if based on quadratic deviation assuming linear interpolation between points.

Input files

proj.dat

File describing projectile

- **mass** – The mass of the projectile [kilogram].
- **diameter** – The diameter of the projectile [meter].
- **length** – The length of the projectile [meter]. This has an effect only if wind is used.
- **velocity** – Initial (muzzle) velocity of the projectile [m/s].
- **cdtype** – The type of the drag function parametrisation. Can be 0, 1, 2, or 3. The value of cdtype defines which file is to load for Cd function
  0. cdfixed.dat
  1. cdgraph.dat
  2. cdpoint.dat
  3. cdalpha.dat

shot.dat

File describing factors related to shot

- **angle_deg** – Elevation angle of the shot [degrees].
- **angle_dlt** – Step in elevation angle for each subsequent trajectory if number of trajectories is greater than 1 [degrees].
- **height** – The initial height of the trajectory (gun), [meter]
- **ntraj** – Number of trajectories to calculate. If the step in elevation or turbulence are not defined (equal to 0), then there is no point to calculate more than one trajectory, since they would be the same.
- **turbulence** – Turbulence strength [m/s]. This is the average speed that the body would experience without any forces (including gravitational) due to turbulences in the air.
- **zwind** – Side wind [m/s]
- **xwind** – Frontal wind [m/s]

consts.dat

This file contains physical constants that should not be changed

Earth_radius 6378137
Earth_mass 5.9722e24
Ggrav 6.67408e-11 – Gravitational constant
std_press 101325 – sea level standard pressure, [Pa]
temp_rate 6.5 – temperature lapse rate, [K/km]
Earth_atm_R 6356.766 – the radius of the earth, [km]
std_temp 288.15 – sea level standard temperature, [K]
grav_atm 9.80665 – gravitational constant, [m/sec2]
molec_w 28.9644 – molecular weight of dry air, [gm/mol]
gas_const 8.31432 – gas constant, [J/(mol*K)]
speed_1k 20.046 – speed of sound at 1K [m/s]

integr.dat
File describing integration and optimisation parameters

- **max_len** – maximal length of the trajectory [m]. This is a condition to stop calculation of trajectories if the trajectory is too long, e.g. fired in space
- **dt0** – initial time step [s]
- **dtmax** – maximal time step [s]. This parameter is useful to enforce more precise calculations in expense of calculation time.
- **precision** – relative precision of integration. Integration is relative to the initial kinetic energy of the projectile
- **dEpm_rej** – reject step factor. Step is rejected if the relative error in energy exceeds this value.
- **dEpm_max** – maximal error. Step time is reduced if the relative error exceeds this value.
- **dEpm_min** – minimal error. Stop time is increased if the relative error is under this value.
- **dh_up** – increase of step time
- **dh_dn** – decrease of step time
- **renormE** – number of steps without renormalisation of energy
- **dragcare** – this parameter controls step time by reducing it if exponent deviates too much from linear approximation. Drag coefficient γ and step time Δt work in exponential way on a long time scale. If the drag coefficient is too big (for any reason) comparing to the time step, i.e. their product is much greater than 1, then the drag force would have a computational error of integration. This parameter is introduced to restrict step time to grow greater than the integration of drag force introduces significant computational error. Basically it is safe to set it to some value less than 1.
- **maxeval** – maximal number of objective function evaluations

cdfixed.dat
This file represents the drag function as pairs of XY values. In the optimisation only Y values are allowed to change.

cdgraph.dat
This file represents the drag function as a sequence of Y values. X values are present but ignored by the program. The first line starting with '#' specifies the X range. All X points are equidistant.

cdpoint.dat
Same as cdfixed.dat except that X values are allowed to change during the optimisation.

cdalpha.dat
Alpha parametrisation. For more details see the "Drag" paper.

dataset.dat
Dataset file contains fire tables. One file describes one set of data for a particular projectile. The file can contain one or more tables in different formats. The format of

each table is defined in its header. Each table consists of a number of lines: the first line is the header; the second line can represent a separation line and is ignored; the subsequent lines are records – a list of numbers separated by spaces. The header is a list of column names. The program recognizes some predefined keywords. All columns of unrecognized names are loaded and saved together with other columns, but their values do not participate in calculation of function *u*.

Valid column names are:
V – initial velocity;
Max – max range;
R – range;
Angle, AngMil, AngRad – elevation angle;
Fall, FallMil, FallRad – angle of fall;
topH, topR – hight and range of the trajectory apex;
Drop – the shortest distance between the aim line and the last point of trajectory;
Time – time of flight;
Vf – final velocity;
Kind – type of the trajectory: 0,1,2,3 – unknown, flat, high, max;
Hint – elevation angle hint used to solve range

Types of records
Each table is one of the following 3 types: MAX, RANGE, or ANGLE. The type is defined according to the following rules:

- if column Max is defined, then the type is MAX, and no R can be defined;
- else if R is defined, then the type is RANGE, and no Max can be defined;
- else the type is ANGLE, and Angle must be defined, and no R and no Max can be defined.

These types define how the records are processed. For MAX, max range is found.

For RANGE, the angle is solved for the specific range. There are two solutions for each range. If the Kind of the trajectory is not defined as 1 or 2, then each solution is tested (u function is calculated) and the better match is selected. For example, the following table shows input and output of dataset file for a bullet.

R Time	R	Time	Kind	Hint
1000 5	1000	1.47114	1	0.528811
1000 50	1000	99.244	2	87.0727

The first trajectory is guessed as ***flat*** and the second trajectory is guessed as ***high***.

For ANGLE type, a trajectory is calculated straightforward using the elevation angle.

Dataset file can contain comments. The first character '#' on the line make this line a comment. The first character on the line '{' starts a comment block; and the first character on the line '}' ends the comment block.

Sometimes it is useful to have indirect references to other files. If the first character on the line is '@', then the rest of the line is considered to be a filename and that file is loaded as if it would be included at this place.

optim.dat
This is an optional file defining the optimisation plan. Each line represents an optimisation scenario. After all scenarios are executed the best one is selected and then the normal optimisation is run on that one. Each positive number in the line represents the smoother coefficient *k* in the optimisation (see section Optimisation). Optimisation works sequentially for each number and sequentially (or in parallel if implemented later) for each scenario. The negative value sets the number of function evaluations for the current scenario. For example the line '-400 1000 1 -1000 1' will run 400 evaluations with *k*=1000, then 400 evaluations with *k*=1, the 1000 evaluations with *k*=1.

Output files

All output files with the same name as input have the same or similar format as input files. They can be copied into input area and used as a new input. Other files are described below.

cdgrplt.dat
Cd in the form of XY pairs with the step in X 0.01.

earth.dat
Trajectory projected on Earth's surface.

progr.dat
Progress data: log of actions.

shots_summary.dat
List of some trajectory values, one line per one trajectory with the following columns:
ElevMils – elevation angle in mils
Range – earth's surface distance to the hit point
Z – Z-coordinate of the final point (sidewise)
Y – Y-coordinate of the final point, geometrical height from the coordinate system origin. Normally it is negative due to Earth's curvature.
TOF – time of flight
MaxH – maximal height of the trajectory from the Erath's surface
Vfin – final velocity
AOFdeg – Angle of fall in degrees
EK0 – initial kinetic energy [J]
EKfin – final kinetic energy [J]
RejectRate% – percentage of rejected steps of integration

trajs.dat
Trajectory data for each trajectory separated by empty line between different trajectories. The columns are named: Time, X, Y, Z, vX, vY, vZ, dt, Length, Range, HeatKJ, Mach.

They are time, 3 coordinates and 3 components of velocity, integration step time, length of the trajectory, range of the trajectory on the Earth's surface, accumulated heat produced by the air resistance, and velocity as Mach number at the current projectile position.

trres.dat
List of some trajectory values, one line per one trajectory with the following columns:
v0 – initial velocity
Angle_deg – elevation angle
Range – range on Earth's surface
Time – time of flight
Fall_deg – angle of fall in degrees
vFinal – final velocity
topHeight – maximal altitude of the trajectory
topRange – range to the projection of apex point of the trajectory
Drift – deviation in Z direction from the straight line
HeatKJ – accumulated heat in kJ
KinE0 – initial kinetic energy
KinEf – final kinetic energy
TrajLength – length of the trajectory
Elevation0 – initial height of the trajectory (gun)
RejectRate – percentage of rejected steps of integration