Using l_script to define L1d simulations.

Mechanical Engineering Report 2005/09
P. A. Jacobs
Centre for Hypersonics
The University of Queensland.

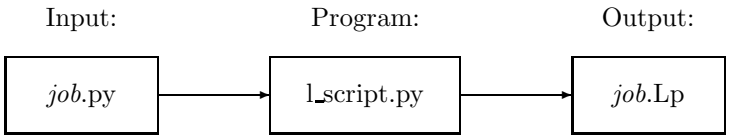
July 2005

Contents

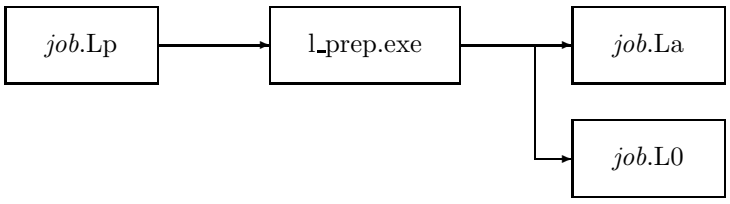
1 Simulation Overview

Setting up a simulation is mostly an exercise in writing a textual description of your experimental facility and the gas slugs contained within it. This description is presented to the l_script program as a Python script, and is assumed to have a ".py" extension. Once you have prepared your simulation script, the simulation data is generated in a number of stages:

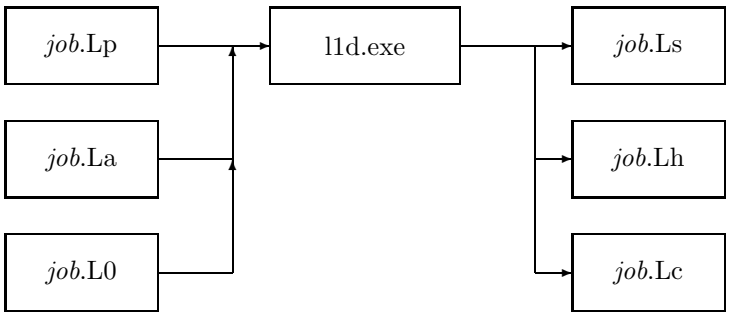
1. Create the input parameter file *job*.Lp with the command.
   `$ l_script.py -f job`



2. Generate an initial (*i.e.* $t = 0$) flow solution in file *job*.L0 and tube description file *job*.La.
   `$ l_prep.exe -f job`



3. Run the simulation code to produce flow data at subsequent times. The whole-of-tube data are saved in *job*.Ls while history data at selected locations and cells are saved in files *job*.Lh and *job*.Lc respectively.
   `$ l1d.exe -f job`



4. Extract subsets of the flow solution data for postprocessing. The specific commands for this stage depend very much on what you want to do. The flow solution data is cell-averaged data associated with cell centres. You may extract the flow data for all cells at a particular time using `l_post.exe` and save it in a form ready for display with GNU-Plot or for further calculation. The data for all cells over a range of times may be extracted with the program `sptime.exe` and written into a form ready to produce an (x, t)-diagram via a contouring program. The program `l_hist.exe` can be used to extract data for individual history locations and cells while `piston.exe` can be used to get motion data for a specific piston. See the shell scripts in the examples for ideas on what can

be done. Since the output of this stage is always a text file, you may look at the head of each file for hints as to what data is present.

2 Module l_script

Python program to write the input parameter file for L1d.

It is intended for the user to define their particular facility and flow in terms of the data objects defined in this module. As part of its initialization, this program will execute a user-specified job file that contains, in Python, the user's script that defines both facility geometry and gas-path details.

Usage:

```
$ l_script.py -f <job>
```

The simulation control data is then organised via the classes: GlobalData, GasSlug, Piston and Diaphragm. These classes provide places to store the configuration information and their function/method names appear as commands in the user's job description file.

When setting up a new simulation, first define the tube as a set of (x,d) break-points and identify regions of head-loss and regions where the wall-temperature varies from the nominal value. Create the GasSlugs, Pistons, and Diaphragms that will make up the gas path. Note that places where two GasSlugs join will need a GasInterface to be defined. Once all of the components have been created, assemble the gas path and then set any of the time-stepping parameters for which you want values other than the default.

Here is an example script for the Sod shock-tube problem:

```
# sod.py
gdata.title = 'Sods ideal shock tube, 06-Jul-05'
gdata.gas_name = 'perf_air_14'

Define the tube walls.
add_break_point(0.0, 0.01)
add_break_point(3.0, 0.01)

# Create the gas-path.
left_wall = VelocityEnd(x0=0.0, v=0.0)
driver_gas = GasSlug(p=100.0e3, u=0.0, T=348.4, nn=100,
                     to_end_R=1, cluster_strength=1.1,
                     hcells=1)
interface = GasInterface(x0=0.5)
driven_gas = GasSlug(p=10.0e3, u=0.0, T=278.7, nn=100,
                     hcells=1)
right_wall = VelocityEnd(x0=1.0, v=0.0)
assemble_gas_path(left_wall, driver_gas, interface, driven_gas, right_wall)

# Set some time-stepping parameters
gdata.dt_init = 1.0e-7
gdata.max_time = 0.6e-3
gdata.max_step = 5000
add_dt_plot(0.0, 10.0e-6, 5.0e-6)
add_history_loc(0.7)
```

This script should define the gas path:

```
.     |+----- driver-gas -----+|+----- driven-gas -----+|
.     |                       |                        |
.     |                       |                        |
. left-wall               interface               right-wall
```

and can be invoked with the command:

```
$ l_script.py -f sod
```

Upon getting to the end of the user's script, this program should then write a complete simulation parameter file (sod.Lp) in the traditional (i.e. ugly) format. Because this program just gathers the data in order to write the input parameter file, the old documentation for that file is still relevant (despite a few small name changes).

2.1 Functions

add_break_point(*x*, *d*, *transition_flag*=0)

Add a break-point tuple to the tube-diameter description contained in `GlobalData`.
The tube is described as a set of (x,d)-coordinate pairs that define break points in the profile of the tube wall.

Parameters

`x:` x-coordinate (in metres) of the break point
(type=float)

`d:` diameter (in metres) of the tube wall at the break-point.
(type=float)

`transition_flag:` Indicates the variation in diameter between this break-point and the next. 1=linear, 0=Hermite-cubic.

Return Value

Number of break points defined so far.

add_dt_plot(*t_change*, *dt_plot*, *dt_his*)

Add a dt tuple to the dt_plot tuple list in `GlobalData`.

Parameters

`t_change:` The time (in seconds) at which this dt_plot and dt_his should take effect.
(type=float)

`dt_plot:` Time interval between writing whole solutions (for later plotting).
(type=float)

`dt_his:` Time interval between writing data to history file.
(type=float)

add_history_loc(*x*)

Add a location to the history-location list in `GlobalData`.

Parameters

`x:` x-coordinate (in metres) of the sample point.
(type=float)

Return Value

Number of sample points defined so far.

add_loss_region(*xL*, *xR*, *K*)

Add a head-loss region to the tube description in `GlobalData`.
There is a momentum-sink term much like the so-called minor-loss terms in the fluid mechanics text books. The effect of the loss is spread over a finite region so that the cells are gradually affected as they pass through the region

Parameters

`xL`: Left-end location (in metres) of the loss region.
(type=float)

`xR`: Right-end location (in metres) of the loss region.
(type=float)

`K`: Head-loss coefficient. A value of 0.25 seems to be good for a reasonably smooth contraction such as the T4 main diaphragm station.
(type=float)

Return Value

Number of loss regions defined so far.

add_T_patch(*xL*, *xR*, *T*)

Add a temperature patch for a region where the wall temperature is different from the nominal value.

Parameters

`xL`: Left-end location (in metres) of the loss region.
(type=float)

`xR`: Right-end location (in metres) of the loss region.
(type=float)

`T`: Wall temperature in degrees K.
(type=float)

Return Value

Number of temperature patches defined so far.

assemble_gas_path(**components*)

Assembles a gas path by makeing the logical connections between adjacent components.
The components are assembled left-to-right, as they are supplied to this function.

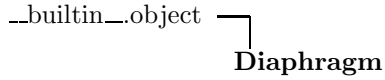
Parameters

`components`: An arbitrary number of arguments representing individual components or lists of components. Each component may be a `GasSlug`, `Piston`, or any other gas-path object, however, it doesn't always make sense to connect arbitrary components. For example, connecting a `GasSlug` to a `Piston` is reasonable but connecting a `Piston` to a `Diaphragm` without an intervening `GasSlug` does not make sense in the context of this simulation program.
(type=mixed gas-path objects and lists of gas-path objects)

2.2 Variables

Name	Description
gdata	Contains the `GlobalData` information describing the simulation. Note that there is one such variable set up by the main program and the user's script should directly set the attributes of this variable to adjust settings for the simulation. **Value:** `<l_script.GlobalData object at 0xb7c0956c>` *(type=`GlobalData`)*

2.3 Class Diaphragm



Contains the information for a diaphragm which controls the interaction of two `GasSlug`s.

2.3.1 Methods

__init__(*self*, *x0*, *p_burst*, *is_burst*=0, *dt_hold*=0.0, *dt_blend*=0.0, *dx_blend*=0.0, *dxL*=0.0, *dxR*=0.0, *label*='')

Creates a diaphragm with specified properties.
The connections to GasSlugs are made later via the function assemble_gas_path.

Parameters

- x0: x-position in the tube, metres. This value is used to determine the end-points of the GasSlugs. *(type=float)*
- p_burst: Pressure (in Pa) at which rupture is triggered. *(type=float)*
- is_burst: Flag to indicate the state of diaphragm. A value of 0 indicates that the diaphragm is intact while a value of 1 indicates that the diaphragm is ruptured and the GasSlugs are interacting. *(type=int)*
- dt_hold: Time delay (in seconds) from rupture trigger to actual rupture. *(type=float)*
- dt_blend: Time delay (in seconds) from rupture to a blend event. This models the mixing of the two gas slugs some time after rupture of the diaphragm. Blending events are seldom used so this is usually set to 0.0. *(type=float)*
- dx_blend: Distance (in metres) over which blending occurs. Set to 0.0 to have no effective blending. *(type=float)*
- dxL: The distance over which p is averaged on left of the diaphragm. The pressure difference between the left- and right-sided of the diaphragm is used to trigger rupture. The default value of 0.0 will cause the pressure in the gas cell immediately adjacent to the diaphragm to be used. *(type=float)*
- dxR: The distance (in metres) over which p is averaged on right-side of the diaphragm. *(type=float)*
- label: A label that will appear in the parameter file for this diaphragm. *(type=string)*

Overrides: __builtin__.object.__init__

2.4 Class FreeEnd

__builtin__.object
└ **FreeEnd**

Contains the information for a free-end condition.

2.4.1 Methods

__init__(*self*, *x0*)

Creates a `GasSlug` end-condition with a specified location.

Parameters

`x0`: Initial position (in metres).
(type=float)

Overrides: __builtin__.object.__init__

2.5 Class GasInterface

__builtin__.object
└ **GasInterface**

Contains the information for an interface between two slugs.

The primary use of this class is to locate the ends of the connected `GasSlug`s. Implicitly, the logical connections are also made via the function `assemble_gas_path`.

2.5.1 Methods

__init__(*self*, *x0*)

Creates as interface between two `GasSlug`s at specified location.

Parameters

`x0`: Initial position (in metres).
(type=float)

Overrides: __builtin__.object.__init__

2.6 Class GasSlug

__builtin__.object
└ **GasSlug**

Contains the gas properties and discretisation for each gas slug.

The user may create more than one gas slug to describe the initial gas properties throughout the facility.

Note: A slug needs to have appropriate end-conditions. This is achieved by creating end-condition objects such as `FreeEnd` and `VelocityEnd` objects and then assembling the gas-path via a call to `assemble_gas_path`.

2.6.1 Methods

__init__(*self*, *p*=100000.0, *u*=0.0, *T*=300.0, *mf*=[1.0], *Tv*=None, *Te*=None, *label*='', *nn*=10, *to_end_L*=0, *to_end_R*=0, *cluster_strength*=0.0, *viscous_effects*=0, *adiabatic_flag*=0, *hcells*=[])

Creates a gas slug with user-specified properties.
Most parameters have default properties so that only the user needs to override the ones that they wish to set differently.

Parameters

`p`: Pressure in Pa.
(type=float)

`u`: Velocity in m/s.
(type=float)

`T`: Temperature in degrees K.
(type=float)

`mf`: Mass fractions. The number of mass fraction values should match the number of species expected by the selected gas model. See `GlobalData.gas_name`.
(type=list of floats)

`Tv`: Vibrational temperatures in degrees K.
(type=list of floats)

`Te`: Electron temperature in degrees K.
(type=float)

`label`: Optional label for the gas slug.
(type=string)

`nn`: Number of cells within the gas slug.
(type=int)

`to_end_L`: Boolean flag to indicate that cells should be clustered to the left end.
(type=int)

`to_end_R`: Boolean flag to indicate that cells should be clustered to the right end.
(type=int)

`cluster_strength`: As this value approaches 1.0 from above, the clustering gets stronger. A value of zero indicates no clustering.
(type=float)

`viscous_effects`: A nonzero value activates the viscous effects. 0 = inviscid equations only; 1 = include viscous source terms F_wall_, loss_, q, friction factor for pipe flow; 2 = use Con Doolan's laminar mass-loss model if the mass within a cell is greater than MINIMUM_MASS as set in l1d.h; 3 = use Con Doolan's turbulent mass-loss model if the mass within a cell is greater than MINIMUM_MASS as set in l1d.h; 4 = include viscous source terms F_wall_, loss_, q, friction factor for flat plate; 5 = use David Buttsworth's mass-loss model with pipe-flow friction factor; 6 = use David Buttsworth's mass-loss model with flat-plate friction factor; 7 = include viscous source terms F_wall_, loss_, q, friction factor for pipe flow; half heat flux.
(type=int)

`adiabatic_flag`: Boolean flag to indicate that there should be no heat transfer at the tube wall.
(type=int)

`hcells`: Either the index of a single cell or a list of indices of cells for which the data are to be written every `dt_his` seconds, as set by `add_dt_plot`.
(type=int or list of int)

Overrides: __builtin__.object.__init__

Note: Locations of the ends of the slug are communicated through end-condition objects that are attached during assembly of the gas path.

2.7 Class GlobalData

__builtin__.object
└ **GlobalData**

Contains the global data that defines the tube and simulation control parameters.

The user's script should not create one of these but should specify the simulation parameters by altering the attributes of the global object `gdata` that already exists by the time the user's script executes.

2.7.1 Properties

Name	Description
gas_name	The name of the gas.

2.7.2 Instance Variables

Name	Description
case_id	Specifies a special case that has custom C-code in the main simulation. See l1d.h for possible values. Use a value of 0 for a generic simulation; this is the usual case. **Value:** <member 'case_id' of 'GlobalData' objects> *(type=int)*
cfl	Largest allowable CFL number. The time step is adjusted to ensure that this value is not exceeded in any particular cell. A typical value of 0.25 seems to work well for simulations with sudden events such as diaphragm bursting, while a value as high as 0.5 should be considered only for well-behaved flows. **Value:** <member 'cfl' of 'GlobalData' objects> *(type=float)*
dt_init	The size of the time-step that will be used for the first few simulation steps. After a few steps, the cfl condition takes over the determination of a suitable time-step. **Value:** <member 'dt_init' of 'GlobalData' objects> *(type=float)*
dt_plot_list	Specifies the frequency of writing complete solutions (for later plotting, maybe) and also for the writing of data at history locations. It may be convenient to have different frequencies of writing such output at different stages of the simulation. For example, free-piston driven shock tunnels have a fairly long period during which the piston travels the length of the compression tube and then a relatively short period, following diaphragm rupture, when all the interesting things happen. It is good to have low-frequency output during most of the compression process and higher-frequency output starting just before diaphragm rupture. Arranging good values may require some trial and error. Add entries to this list via `add_dt_plot`. **Value:** <member 'dt_plot_list' of 'GlobalData' objects> *(type=list of tuples)*

continued on next page

Name	Description
gas_name	selects the thermo-chemical model (which, in turn, sets the number of species). Look in the file gas.h for a current list of the gas models, their names and their integer indices.
hloc_list	List of x-coordinates for the history locations. Add entries via the function `add_history_loc`. **Value:** <member 'hloc_list' of 'GlobalData' objects> *(type=list of float)*
loss_region_list	List of head-loss regions, usually associated with sudden changes in tube cross-section and diaphragm stations. Add regions via the function `add_loss_region`. **Value:** <member 'loss_region_list' of 'GlobalData' objects> *(type=list of tuples)*
max_step	The simulation will be stopped if it reaches this number of steps. This is mostly used to catch the problem of the calculation taking a very long time (measured by one's patience), possibly because the time-step size has decreased to an extremely small value. **Value:** <member 'max_step' of 'GlobalData' objects> *(type=int)*
max_time	The simulation will stop if it reaches this time. It is most usual to use this critereon to stop the simulation. **Value:** <member 'max_time' of 'GlobalData' objects> *(type=float)*
n	The number of small segments that will be used to describe the tube's area distribution internal to the simulation. To enable a fast lookup process for the area calculation, the area variation between equally-spaced x-positions is taken to be linear. The default value is 4000 and probably won't need to be changed except for geometries with rapidly changing cross-sections. **Value:** <member 'n' of 'GlobalData' objects> *(type=int)*
reacting_flag	If set to 1, Rowan's finite-rate chemistry will be active. (Default is 0) **Value:** <member 'reacting_flag' of 'GlobalData' objects> *(type=int)*
T_nominal	The nominal wall temperature (in degrees K) in the absence of a patch of differing temperature. **Value:** <member 'T_nominal' of 'GlobalData' objects> *(type=float)*
t_order	1=Euler time-stepping. This is generally cheap and nasty. 2=predictor-corrector time-stepping, nominally second order. This is the default setting. It is, however, twice as CPU intensive as Euler time-stepping. **Value:** <member 't_order' of 'GlobalData' objects> *(type=int)*
T_patch_list	Regions of the tube wall that have temperature different to the nominal value can be specified via the function `add_T_patch`. **Value:** <member 'T_patch_list' of 'GlobalData' objects> *(type=list of tuples)*

continued on next page

Name	Description
title	Short title string for embedding in the parameter and solution files. **Value:** <member 'title' of 'GlobalData' objects> *(type=string)*
x_order	1=use cell averages without high-order reconstruction. Use this only if the second-order calculation is showing problems. 2=use limited reconstruction (nominally second order). This is the default selection. **Value:** <member 'x_order' of 'GlobalData' objects> *(type=int)*
xd_list	List of break-point tuples defining the tube wall. Add elements to the list via the function `add_break_point`. **Value:** <member 'xd_list' of 'GlobalData' objects> *(type=member_descriptor)*

2.8 Class Piston

Contains the information for a piston.

Notes:

- The left- and right-end positions of the piston are also used to locate the ends of adjoining `GasSlug`s.
- The generic piston model (type_of_piston=0) has inertia but no friction. However, to make accurate simulations of a particular facility, it is usually important to have some account of the friction caused by gas-seals and guide-rings that may be present on the piston. Piston models that have been encoded are listed in `l1d.h`.

2.8.1 Methods

__init__(*self*, *m*, *d*, *xL0*, *xR0*, *v0*, *type_of_piston*=0, *p_restrain*=0.0, *is_restrain*=0, *with_brakes*=0, *brakes_on*=0, *x_buffer*=10000000.0, *hit_buffer*=0, *label*='')

Create a piston with specified properties.

Parameters

`m`: Mass in kg.
(type=float)

`d`: Face diameter, metres.
(type=float)

`xL0`: Initial position of left-end, metres. The initial position of the piston centroid is set midway between xL0 and xR0 while piston length is the difference (xR0 - xL0).
(type=float)

`xR0`: Initial position of right-end, metres.
(type=float)

`v0`: Initial velocity (of the centroid), m/s.
(type=float)

`type_of_piston`: Usually 0 but, for others, see the header file `l1d.h`.
(type=int)

`p_restrain`: Pressure at which restraint will release. Some machines, such as two-stage light-gas guns, will hold the projectile in place with some form of mechanical restraint until the pressure behind the piston reaches a critical value. The piston is then allowed to slide.
(type=float)

`is_restrain`: Status flag for restraint. 0=free-to-move, 1=restrained.
(type=int)

`with_brakes`: Flag to indicate the presence of brakes. 0=no-brakes, 1=piston-does-have-brakes. Such brakes, as on the T4 shock tunnel, allow forward motion of the piston but prevent backward motion by locking the piston against the tube wall.
(type=int)

`brakes_on`: Flag to indicate the state of the brakes. 0=off, 1=on.
(type=int)

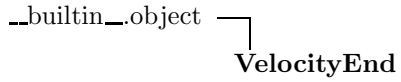
`x_buffer`: Position of the stopping buffer in metres. This is the location of the piston centroid at which the piston would strike the buffer (or brake, in HEG terminology). Note that it is different to the location of the front of the piston at strike.
(type=float)

`hit_buffer`: Flag to indicate state of buffer interaction. A value of 0 indicates that the piston has not (yet) hit the buffer. A value of 1 indicates that it has. Details of the time and velocity of the strike are recorded in the *event* file.
(type=int)

`label`: A bit of text for corresponding line in the Lp file.
(type=string)

Overrides: __builtin__.object.__init__

2.9 Class VelocityEnd



Contains the information for a fixed-velocity end condition for a `GasSlug`.

2.9.1 Methods

__init__(*self*, *x0*, *v*=`0.0`)

Creates a `GasSlug` end-condition with a specified location and velocity.

Parameters

`x0:` Initial position (in metres).
(type=float)

`v:` Velocity (in m/s) of the end-point of the `GasSlug`.

Overrides: __builtin__.object.__init__

Index