Verification and validation of Hobyah

Ewan Bennett

August 2024 (v1.2)

Contents

1 Introduction **4**

1.1 Verification and validation 5

1.2 Verification methods 1 5

1.3 Verification methods 2 6

1.4 Validation methods 7

2 Requirements **7**

3 Programs **11**

3.1 Primary programs/spreadsheets 11

3.2 Secondary programs 11

3.3 Ancillary programs 12

4 Traceability **13**

4.1 Traceability in curve data file names 14

4.2 Traceability data inside curve data files 15

5 Verification of `SESconv.py`'s annulus volume flow calculation **23**

5.1 Annulus volume flow at solitary trains 25

5.2 Combinations of trains 26

5.3 Description of the calculation 26

5.4 Manual calculation 30

6 Verification of `SESconv.py`'s annulus air velocity calculation 37

7 Verification of `SESconv.py`'s air density corrections 43

7.1 Verification of `SESconv.py`'s line segment density correction . . . 44

7.2 Verification of `SESconv.py`'s vent segment density correction . . 50

8 Verification of the quadratics spreadsheet 55

8.1 Introduction 55

8.2 The original equation 55

8.3 Verification activities 56

8.4 Sample printout 57

8.5 Verifying the three calculations 57

8.6 Verifying friction 60

8.7 Verifying the blockage correction calculation 65

8.8 Verification of variant inputs 65

8.9 Verifying switching equations 65

8.10 Verifying balanced forces 74

9 Loss factor conversions in Hobyah 77

9.1 The `loss1` keyword 77

9.2 The `loss2` keyword 79

10 Hobyah with one tunnel 81

10.1 Hobyah/quadratics spreadsheet steady-state comparison 81

10.2 Hobyah/SES transient comparison 83

11 Hobyah with traffic 87

11.1 Steady-state comparison with pressures at portals 88

11.2 Steady-state comparison with fixed flow at one portal 90

12 Unverified features 92

13 `UScustomary.py` **94**

14 Requirements and their validation **97**

15 Check of the error messages **105**

1 Introduction

Anyone trying to convince someone to use new engineering software has to be prepared for the question, "that all sounds great...what work did you do to show that it is correct?" A second question is "does it do what it needs to do?"

The first question concerns verification, the second relates to validation. This document is my answer to those two questions as they apply to Hobyah and its associated scripts and spreadsheets.

This document is not complete (verification and validation activities never end, because every time a new feature is added new questions can be asked). Further work may also be needed when someone asks a question that the documentation doesn't answer.

This document assumes that the reader has some familiarity with Hobyah terminology. If you get lost in the text because you don't know the definition of a particular phrase (such as `.hbn`, `.sbn`, "section", "segment", "subsegment", "subpoint", "join" or "node"), then it's best to read the Glossary in the Hobyah User Manual before reading this document.

This document was written at the same time as the software/spreadsheets it verifies and validates. In this document, I take all the methods I've used over the years to catch mistakes in my professional work, and apply them to the calculations in these programs.

In some sections the verification evidence is a three-page arithmetic calculation that quotes information from a datafile and has a tedious calculation of one data point out of thousands—a specimen calculation to show how each calculation was done. These sections include transcripts from Python sessions that show you how to interrogate the contents of the `.hbn` and `.sbn` binary files by using the classes that interpret them.

A couple of suggestions for those readers who want to work their way through this document in detail looking for flaws:

- Make a copy of this document. In many sections a claim is made in a paragraph on one page and the evidence to back it up is given in a Figure on a different page. Open this document and its copy: you can see the paragraph with a claim in the original document and the Figure with the evidence in the copy. On large or dual monitors you can look at the two side-by-side.
- The sections that verify the calculation of annulus volume flows, annulus areas and annulus air velocities in `SESconv.py` are tedious (if you're bored reading them, consider how bored I was writing them). In those sections I suggest looking at one or more `.pdf` flipbooks before reading the text. Flipbooks are like a video, but with you controlling the frame rate and back/forwards direction of play with the arrow keys on your keyboard. They let you grok how the movement of trains affects volume flow and air velocities.

1.1 Verification and validation

Here are the definitions of "verification" and "validation" used in this document:

- *Verification* should answer the question, "Are we writing the code correctly?" Once verified (usually by comparing a calculation in code to a simpler calculation), the answer is yes.
- *Validation* should answer the question, "Does the code do what we want?" Once validated (usually through a set of files that test the desired behaviour), the answer is also yes.

I have seen different online sources switch the meanings of these two terms. I can't be bothered to figure out which is the One True Definition, so the definitions above are the ones I've used in this document.

1.2 Verification methods 1

Verification is done in three ways.

The first is to write up hand calculations of something that a piece of code does thousands of times and compare the results. Examples in this document include the calculations in the quadratics spreadsheet, the annulus volume flow calculations in `SESconv.py`, the annulus area calculations in `SESconv.py`, the annulus air velocity calculations in `SESconv.py` and the density correction calculations in `SESconv.py`. You can find these in Sections 5 to 8 of this document.

The second way is to compare calculations from a more complex calculation to the results of a simpler calculation method that has been verified. For example:

- I wrote a quadratics spreadsheet that calculates incompressible, steady-state flow.
- That spreadsheet is verified in Section 8 of this document (by comparing it to hand calcs).
- I generated a set of compressible, transient input files for `Hobyah.py` that replicate the calculations in the incompressible, steady-state spreadsheet and ran them for several thousands seconds (so that the transient calculation converges on a steady value).
- As long as the density changes in the compressible calculation are low, you can compare the two and—while they won't match exactly because one calculation is compressible and the other is not—this comparison of incompressible and compressible calculations can (and did) flush out errors in the compressible flow calculation and the spreadsheet. See Section 10.1 for an example of this type of verification.

Another example of the second way to verify is to compare the results of Hobyah transient calculations to the equivalent transient calculations in SES v4.1. Once again these won't be exact comparisons (SES has transient, incompressible flow and Hobyah has transient, compressible flow) but the general shape of how airflow varies with time and the steady-state conditions can be compared. See Section 10.2 for an example of this.

The third way is to compare Hobyah calculations to full-scale test data in conference proceedings and technical reports that have graphs of full-scale test data.

Most full-scale tests are deliberately simple and make good test cases for software. Some have already been used to verify the calculations in existing compressible codes like the Mott MacDonald Aero program and ThermoTun. Examples include the full-scale tests in Patchway tunnel in the UK (single-track, no shafts, one train), Stanton tunnel (double-track, three shafts, two passing trains) and Milford tunnel (double-track, with/without shafts[1], two passing trains),

1.3 Verification methods 2

I've always found that a good way of finding mistakes in my calculations is to ask myself "How can I prove that I got the wrong answer?", then spend at least five minutes sitting there, thinking about that question and making a list of things I could do to prove I got the wrong answer. Each time something occurs to me, I add something to the list.

Note that this is not "fill in a checklist written by my firm's QA department or given to me by my boss" or "find that list of checking activities we used on that other project that was kind of similar to this project".

It is *"sit there for five minutes thinking about how I can prove that I got the wrong answer"*. This means not checking email, definitely not checking my phone, not engaging in conversations, not getting a cup of tea.
It's just *"sitting there for five minutes thinking about activities I could do that might prove I got the calculation wrong and writing them down"*. I do draw on my experience of past projects, but for this five-minute task it's best to start from a blank slate and look for ways to prove that the calculation is wrong.

Then I carry those activities out, and any new activities that suggest themselves during the work. If I can't prove that I got the wrong answer, maybe I have the right answer. I find that carrying out those activities leads me to do dull-but-worthy programming tasks like adding more output data to text files, binary files and log files. All of these additions are useful cross-checks that might lead me to think "that's odd" when I see something I didn't expect in an output and

[1] Milford tunnel has four open shafts: British Rail's Research Division ran trains through the tunnel with the shafts open and with inflatable rubber bags blocking all four shafts. I'd like to take a moment here to acknowledge that BR Research (later Transmark), SNCF and Deutsche Bahn did some badass full-scale tunnel tests in the 1970s and 1980s and took great pains to put their test results out into the world in technical papers and reports. We all benefit from their work.

am prompted to investigate the oddity.

You should consider following this process in your day-to-day work. And if you record in detail the actions you take (as I have done by writing this document whilst writing my code) you can make a good case to disinterested observers that your calculations are correct.

You might not catch everything, but you'll have done your best and now have evidence that you've used all due care, skill, diligence and other buzzwords that your PI insurers love to hear about.

1.4 Validation methods

Validation is a trickier thing to nail down than verification.

Validation is a high-level activity (no detailed calculations).

Validation starts by writing a set of requirements stating what the code should be able to do. See Section 2 for the requirements I wrote for my codes.

When you have a set of input files that you can point at and claim that those input files validate that the code meets a particular requirement, then you can claim to have validated that requirement.

Now, your readers (if any) may disagree with you.

But because you have listed your evidence, the dispute can be carried out based on documentation (see Section 14). Having evidence on hand often quashes a British pantomime-style "Oh no it isn't"—"Oh yes it is" exchange of opinions that I've often seen third party checkers try to start on projects.

Another good way to forestall disputes about validation is to play Devil's Advocate: consider "how could I rationally object to this validation evidence if I were seeing it for the first time?" and try to answer the objections you can foresee whilst wearing your Devil's Advocate hat.

2 Requirements

The following is a high-level list of requirements that the tunnel ventilation software must meet.

1. Trap errors and have informative error messages for each.
2. Some errors don't have test files. Document why those error messages have no test files.
3. Calculate compressible, isothermal airflow in a network of tunnels by the method of characteristics (MoC).

4. Handle portals with fixed pressure, fixed volume flow and fixed velocity.
5. Handle pressure changes with elevation.
6. Handle area changes in the tunnels, with fixed losses.
7. Handle junctions with up to six air paths attached, with fixed losses in each branch.
8. Allow junctions to be created partway along the length of tunnels, with up to four tunnel ends socketed into each junction (this makes building networks easier).
9. Handle friction factors in a way that encourages users to learn the difference between Fanning friction factor and Darcy friction factor if they don't already know it.
10. Handle fan characteristics that have stall humps.
11. Handle fan characteristics generated by four points and a spline fit calculation.
12. When a fan duty point goes off the end of a fan characteristic into the reverse flow or freewheeling zones, use linear extrapolation. When plotting on the flow-pressure plane, show the extrapolated lines.
13. Handle fixed-pitch fans that run at constant speed, have one start time and one stop time. This is for compatibility with SES.
14. Handle fixed-pitch fans that start, stop, change speed and reverse multiple times in each run (set by the run time).
15. Handle fixed-pitch fans that follow a sequence of starts, stops, speed changes and reversals (triggered by the actions of trains).
16. Handle jet fans that start, stop, change speed and reverse in each run (set by the run time).
17. Handle jet fans that follow a sequence of starts, stops, speed changes and reversals (triggered by the actions of trains).
18. Handle dampers that follow time series of area and k-factors (set by the run time). Both area and k-factors must vary.
19. Handle dampers that follow a timed sequence of opening and closing (triggered by the actions of trains). These are platform screen doors.
20. Handle dampers that open (triggered by train location) and close using a different trigger event. These are portal doors.
21. Handle Saccardo nozzles.
22. Handle road traffic drag.
23. Handle road traffic in a way that makes it easy for data from point-to-point traffic modelling to be imported.

24. Distinguish between road traffic routes that share common lanes in a tunnel and road traffic routes that move traffic on different lanes in a tunnel.
25. Calculate the density of stationary road traffic in a tunnel that has traffic from multiple overlapping routes.
26. Handle road traffic pollution by PIARC methods.
27. Handle the aerodynamic effects of non-leaky, incompressible trains passing through tunnels.
28. Handle trains at constant speed.
29. Handle trains with predefined speed-time curves and predefined speed-distance curves.
30. Handle trains with preset acceleration and deceleration curves that obey speed limits in routes and stop at predefined station locations.
31. Handle junctions with three air paths and losses that vary with the flow, following the rules used by SES.
32. Allow variable print timesteps.
33. Allow multiple junctions along the length of a tunnel to be defined in one line of input, with procedurally-generated names.
34. Allow multiple tunnels to be defined in one line of input, with procedurally-generated names (this is great for cross-passages).
35. Plot properties at a fixed point against time.
36. Plot profiles of properties along routes at a given time.
37. Plot properties at a moving point against time and against distance.
38. Plot fan characteristics and system characteristics in the flow-pressure plane.
39. Plot icons for in-tunnel features like trains, fires and jet fans at the correct locations in routes.
40. Allow plotting in SI and US units.
41. Generate plotted output to `.pdf` files by feeding `.plt` files to Gnuplot.
42. Generate multiple individual `.png` files from multi-page `.pdf` files.
43. Allow Gnuplot commands to be written directly to the `.plt` file.
44. Allow plotting from multiple binary files on the same graph.
45. Allow plotting from external `.csv` files and from blocks of csv-style data in the input file.
46. Store run data in binary files and give examples of how to interrogate them.

47. Store plotted curve data in text files with traceability information.
48. Allow the generation of SES input files with the same tunnels, routes and gradients as the Hobyah geometry.
49. Make it difficult to write output files over input files if Hobyah and SES files have the same filestem.

At the time of issue of this document, not all of these requirements have been implemented.

The following is a list of features that could be included in the above list, but are not:

1. Temperatures and heat transfer. There is nothing to stop someone forking this code and implementing a method of characteristics that uses the energy equation instead of the isentropic relationship.
2. A graphical user interface (GUI). If I tried, I would code it badly.
3. Variation of air density with temperature and heat transfer to the wall (a fire model).
4. Implicitly-calculated train speeds (the calculation of train speeds based on gradients, train drag, Davis equation resistances and tractive effort curves).
5. Air leakage between tunnels and the interiors of trains.
6. Changes of train cross-section.
7. Fan characteristics for fans that have variable pitch-in-motion impellers or variable pitch inlet vanes.
8. Fan curves input as fan static pressure.
9. Micro-pressure waves.

Some of these may be moved out of this list as and when they become of enough interest.

A software requirement can be considered "validated" once we show it in use in test files and we have documentation that verifies that the code handles the feature correctly (if necessary). Section 14 gives the above set of requirements and the evidence for the validation of the requirements that have been implemented.

3 Programs

There are three categories of program in the Hobyah suite:

1. Primary calculation programs and spreadsheets
2. Secondary calculation programs
3. Ancillary programs

3.1 Primary programs/spreadsheets

The primary programs/spreadsheets carry out engineering calculations from base data.

- The Python script `Hobyah.py` calculates homentropic airflow by the method of characteristics. It also holds most of the plotting code.
- The Fortran code `compressible.f95` replicates some of the method of characteristics calculations in `Hobyah.py`, but is faster than the Python code. Its calculations are not verified yet.
- The spreadsheet `slugflow-quadratics.ods` calculates incompressible airflow in a simple tunnel (constant area) with jet fans, traffic, friction and losses. It solves the airflow by three quadratic equations and chooses which solution to display. Its calculations are verified in Section 8.
- The spreadsheet `slugflow.ods` calculates incompressible airflow in a simple tunnel with area changes, fixed flow at shafts, traffic, jet fans, friction and losses. It uses the spreadsheet program's goal seek function to vary one airflow until the pressure is balanced. Its calculations are not verified yet.
- The spreadsheet `mean-friction.ods` calculates mean roughness of an air path based on the duct area and multiple surfaces with different friction factors or roughness heights. It figures out what roughness to use to get a friction factor at a given air velocity, using various friction factor approximations. Its calculations are not verified yet.

3.2 Secondary programs

These programs either convert data from one form to another or were written to test certain types of calculation.

- The Python script `SESconv.py` mostly just processes numbers, but it does a few of its own engineering calculations. It calculates annulus volume flows around trains, air velocities around trains in SES, air velocities accounting for the lighter density of hot air in subsegments and the density

corrections for when hot air passes through vent segments. These calculations are verified in Sections 5, 6, 7.1 and 7.2. The density correction calculations in those Sections replicate calculations carried out inside SES and the verification of these shows the SES code alongside the `SESconv.py` code and hand calculations.

- The Python script `UScustomary.py` is a converter between SI units and US customary units. Its conversions are verified in Section 13.
- The spreadsheet `MoC-simple.ods` calculates compressible flow in one tunnel. It was used to develop the MoC calculation in `Hobyah.py`.
- The program `offline-SES` is a private fork of SES version 4.1. I use it to test things out in legacy Fortran. For example, when verifying the density correction calculations in `SESconv.py` (Sections 7.1 and 7.2) it was necessary to look at values that SES v4.1 calculates but does not print. Code to print them was written and compiled into `offline-SES` and the output is presented here.

3.3 Ancillary programs

These programs just move information around or automate quality assurance tasks.

- `classHobyah.py` reads binary files full of data generated from Hobyah runs. It can be called by other programs to get plotting data in `.csv` format and can be loaded as a module in a Python interactive session to let users familiar with the command line interrogate the contents of a `.hbn` binary file.
- `classSES.py` reads binary files full of data generated by `SESconv.py`. It can be called by other programs to get plotting data in `.csv` format and can be loaded as a module in a Python interactive session to let users familiar with the command line interrogate the contents of a `.sbn` binary file. It has a class method that can be used in Python scripts to generate new SES input files from the values stored in a known-to-be-good `.sbn` file.
- `generics.py` holds a set of common routines used by all the other programs.
- `syntax.py` holds code used to check the block structure of Hobyah input files. It raises error messages about mismatched `begin...end` blocks.
- `_error-statements.py` scans the text of the other Python scripts alongside a transcript of error messages. It flags up error messages that do not have test files without being flagged as an error message that cannot have a test file. The reasons why some error messages cannot have test files are recorded in the code comments of this script.

4 Traceability

I reckon that every engineer should aspire to write calculations in such a way that the provenance of those calculations can be traced two decades later. Let's start at the sharp end. We engineers need to generate graphs to show something to our clients. We use those graphs to convince our clients to make a particular decision, or to convince our client's Engineer that our client is not ripping their client off. Those graphs are usually in technical reports.

Years or decades later, those graphs may need to be recreated to support or disprove a claim being made during litigation or a public inquiry into a disaster. How should the program that is generating such graphs with a "we might need to recreate this one day" requirement provide traceability? Let's look at a bad example and a good example.

Badly-designed programs may plot graphs for you without giving you access to the base data. They may give you text files full of numbers with little supplementary information about where the numbers came from. They don't identify the name of the file that plotted the data, where the data came from, the date and time it was created, or who plotted it.

Well-designed programs plot graphs that include traceability information in small text in the margins of the image. They store the values they plot in text files that you can load into spreadsheets. The text files that hold the plotted values include traceability information in the body of the text files.

Many of the Figures in this document are graphs plotted from Hobyah. Each Figure plotted from Hobyah contains traceability information in the image (date, time, username, name of the file that created it, names of the files the data came from). For example, Figure 1 is an image that contains text recording where the data on it came from, which file generated it, who generated it and when.

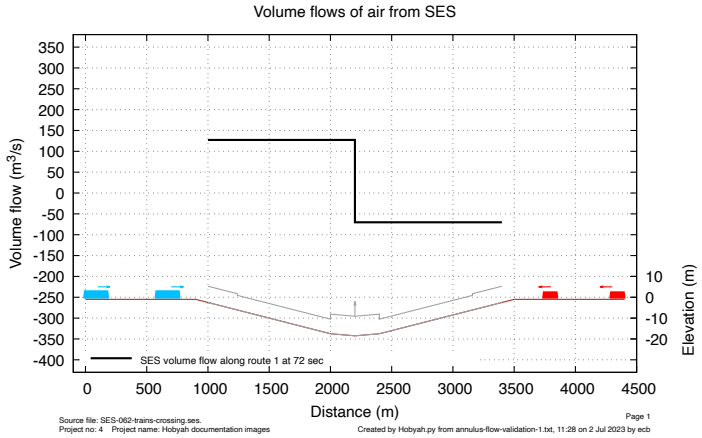

Figure 1: Figures from Hobyah have traceability information in small text

The traceability information is in small text at the base of the image and it is clearly visible if you zoom in.

When Hobyah plots its curves, it creates text files (curve data files), each containing the data for one curve. These are held in the `ancillaries` subfolder (because there can be thousands of them). The files are in `.csv` form, to make it easy to load them into spreadsheets.

The curve data files have two types of traceability information: traceability information in the file name and traceability information in the file's contents.

4.1 Traceability in curve data file names

The names of the curve data files point to the file, page, graph and curve they are used in. Take the following filename as an example:

`ok-038-Channel-Tunnel-lp1-p950-g3-c1.txt`

This is one of 12,012 curve data files that generated a 1001-page `.pdf` flipbook with 12 curves on each page. Let's break down the file name, starting from the right-hand side.

`ok-038-Channel-Tunnel-lp1-p950-g3-c1.txt`

`c1` is the curve number (1st curve)

`g3` is the graph number (on the 3rd graph)

`p950` is the page number (on the 950th page)

`lp1` signifies "loop plot #1"[2]

`ok-038-Channel-Tunnel` is the filestem.

The `Hobyah.py` file that created it was named `ok-038-Channel-Tunnel.txt`.

This build-up of the curve data file name lets you know that this curve data file holds the data for the 1st curve on the 3rd graph on the 950th page of the first loop plot from the Hobyah file named `ok-038-Channel-Tunnel.txt`.

If you want to investigate the curve data for a particular graph on a particular page of a `.pdf` file, the above rules for building the curve data file names lets you narrow down the list of candidates to look inside.

If you say "that's odd" to yourself as you look at one of the graphs on the 595th page of the `ok-038` Channel Tunnel flipbook, the naming scheme above lets you narrow your search down to 12 text files out of over 12,000.

[2] One Hobyah run can create multiple `.pdf` files, so we need a way to distinguish between them. Appending `lp1` for the first loop plot, `lp2` for the second etc. is how I've chosen to do this.

4.2 Traceability data inside curve data files

Curve data files are text files in `.csv` format. Each curve data file written by Hobyah contains three blocks of QA data at the top of the file.

The first block of QA data gives the details of the run that wrote the curve data file. The block includes the line number and text of the line in the input file that caused the curve data file to be created. It also gives the name of the curve data file. For example:

```
# Hobyah source file, "ok-017d-one-tunnel-fixed-Q.txt"
# Timestamp & user, "19:37 on 13 Jun 2023 by tester"
# Project number, "2"
# Project name, "Software testing"
# Project description, "Files that work and test different capabilities"
# 132nd line, "transient ptot file2      Mainline1@10072  DP with Q_outflow +150 at back end"
# Data file name, "ok-017d-one-tunnel-fixed-Q-p3-g4-c2.txt"
```

The second block gives the details of the file that the data originally came from (this may not be the same file as the file that created this curve data file). It describes the type of the curve (transient, profile, fan char etc.), some QA data for the binary file, the property being plotted and the locator. For example:

```
# Hobyah transient QA
# Name of binary, "ok-017b-one-tunnel-fixed-Q.hbn"
# Source name, "ok-017b-one-tunnel-fixed-Q.txt"
# Project no., "2"
# Project name, "Software testing"
# Description, "Files that work and test different capabilities"
# Date created, "18:25 on 27 May 2023 by tester"
# Property plotted, ptot
# Property description, total pressure
# Original locator, Mainline1@10072
# Modified locator, (0, 10075.0)
```

The above example has the lines for a transient curve at a fixed point in a tunnel in Hobyah. There are different blocks for fan characteristics, profiles, train properties and tunnel properties seen by passing trains.

The last two entries (`Original locator` and `Modified locator`) are worth explaining. `Original locator` is the instruction in the input file, in this case *plot at whichever gridpoint is closest to ch. 10072 in the tunnel or route called* `Mainline1`. The `modified locator` turns that into a tuple of the internal segment index (0) and the distance of the nearest gridpoint (which happens to be at 10075.0). Putting the two locators into the curve data file are useful for program development and debugging.

The third block gives details of the SI to US conversion factors, the key texts, the multipliers, offsets and clipping applied to limit the extent of data plotted. For example:

```
# X conversion key, null, 1.0
# Y conversion key, "press1", converting Pa to IWG by dividing by 249.08853408
# Auto key, total pressure in tunnel "mainline1" at 10075 m
# Key used, "DP with Q_outflow +150 at back end"
# Multiplier on the X axis (applied first), 1.0
# Offset added to the X axis (applied second), 0.0
# Multiplier on the Y axis (applied first), 1.0
# Offset added to the Y axis (applied second), 0.0
# X value at which data starts being plotted, 6.7
# X value at which data stops being plotted, 7.2
```

The most useful entry in this third block is the `Auto key`. You may think that you have asked the program to plot something at a particular location, but you made a mistake and really asked it to plot something else (or somewhere else). The `auto key` is an automated description of what is being plotted (in this case *total pressure in tunnel "mainline1" at 10075 m*). You may have set the curve description to "air velocity in the Eastbound tunnel" and be thoroughly puzzled when it looks wrong because you told the program to plot pressure. The auto key of a curve the first thing I check when I'm looking into a curve whose shape looks wrong.

The block of `csv` data in the file is in two parts: the original data in SI and US units in columns on the left-hand side, then the adjusted data (after applying the multipliers, offsets and cropping rules) on the right-hand side.

It starts with a descriptive header on two lines, followed by the lines of data:

```
Time,        Value,                Value,          Plot time,       Plot value
 (s),         (Pa),                 (IWG),            (s)    ,         (Pa)
6.6,    141.4887617125205,    0.5680259921842826
6.7,    130.85950492086704,    0.5253533865145265,    6.7,    130.85950492086704
6.8,    99.05861751813791,    0.39768437308448396,    6.8,    99.05861751813791
6.9,    54.388143594856956,    0.21834864376932367,    6.9,    54.388143594856956
7.0,    3.451668088120641,    0.01385719379203567,    7.0,    3.451668088120641
7.1,    -49.813083382032346,    -0.19998143859176526,    7.1,    -49.813083382032346
7.2,    -101.24822388395842,    -0.4064748474188717,    7.2,    -101.24822388395842
7.3,    -144.39982932335988,    -0.5797128713960911,
7.4,    -171.73205673908524,    -0.6894418379126592,
```

In the example above, the first three columns are the unadjusted numbers from the source of the data. The fourth and fifth columns contain the numbers after the multipliers, offsets and cropping rules have been applied; these are the numbers that are used to plot the curve. Note that in this example the cropping rule caused the plotted values to be at times between 6.7 and 7.2 seconds.

It is useful to have the values in US units when plotting from SES data; the numbers in US units can be directly compared to the values in the corresponding SES output file to check whether they have been processed correctly.

Data files may have between four and eight columns, as follows:

- Four columns are used for data taken from `.csv` files or `data` blocks.
- Five columns are used for transient data at a fixed location (like the example above).
- Six columns are used for fan performance on the flow–pressure plane.
- Seven columns are used for two types of plot: plots along routes at a fixed time and plots of train performance data.
- Eight columns are used for plots of in-tunnel properties seen by passing trains.

The following five pages give examples of the layout and QA data in each type of curve data file. These have been edited to make them typeset in LaTeX: the actual curve data files are significantly more untidy and are probably best loaded into a spreadsheet if you want to do some serious interrogation of curve data files.

Note that the blocks of QA data are padded out with blank lines so that the data always starts on the same row of the spreadsheet. This comes in useful when you decide to use a spreadsheet program to process Hobyah output.

Example of QA for data from a `.csv` file or a `begin data` block (4 columns)

```
# Hobyah source file, "ok-009.txt"
# Timestamp & user, "15:54 on 13 Jun 2023 by tester"
# Project number, "2"
# Project name, "Software testing"
# Project description, "Files that work and test different capabilities"
# 1085th line, "userdata csvc  1  2   lw:=4  Static pressure 100 m into the tunnel"
# Data file name, "ok-009-p28-g1-c1.txt"
#
# Hobyah .csv data QA
# Name of source .csv file, 1976-G-Fig-3c-100m-pstat.csv
# Nickname of source .csv file: csvc
# Column used for the X axis was the 1st.  It had no nickname.
# Column used for the Y axis was the 2nd.  It had no nickname.
#
#
#
#
#
#
#
# X conversion factor, 1.0
# Y conversion factor, 1.0
# Auto key, 1976-G-Fig-3c-100m-pstat.csv: 1st column and 2nd column
# Key used, "Static pressure 100 m into the tunnel"
# Multiplier on the X axis (applied first), 1.0
# Offset added to the X axis (applied second), 0.0
# Multiplier on the Y axis (applied first), 1.0
# Offset added to the Y axis (applied second), 0.0
# X value at which data starts being plotted, -inf
# X value at which data stops being plotted, inf
1st column,  2nd column,  Plot X,  Plot Y
(-),  (-),  (-),  (-)
0.01,             4.8,          0.01,       4.8
0.21,             191.8,        0.21,      191.8
0.29,             407.9,        0.29,      407.9
0.35,            1177.0,        0.35,     1177.0
0.41,            1686.4,        0.41,     1686.4
0.78,            1704.7,        0.78,     1704.7
0.97,            1987.8,        0.97,     1987.8
1.13,            2006.6,        1.13,     2006.6
1.29,            2073.5,        1.29,     2073.5
1.83,            2254.8,        1.83,     2254.8
2.01,            2388.9,        2.01,     2388.9
2.29,            2493.9,        2.29,     2493.9
2.66,            2685.3,        2.66,     2685.3
2.76,            2718.7,        2.76,     2718.7
2.87,            2641.4,        2.87,     2641.4
2.87,            2434.7,        2.87,     2434.7
2.89,            2232.7,        2.89,     2232.7
2.91,            1780.7,        2.91,     1780.7
2.96,            1343.1,        2.96,     1343.1
3.02,            1309.3,        3.02,     1309.3
```

Example of QA for transient data at a fixed point (5 columns)

```
# Hobyah source file, "ok-009.txt"
# Timestamp & user, "15:54 on 13 Jun 2023 by tester"
# Project number, "2"
# Project name, "Software testing"
# Project description, "Files that work and test different capabilities"
# 1086th line, "transient -DP  P1976-1.0  sec810  xoffset := -80  Total pressure at 100 m, SES v4.1"
# Data file name, "ok-009-p28-g1-c2.txt"
#
# SES transient QA
# Name of binary, "Patchway-76-01a.sbn"
# SES file, "Patchway-76-01a.ses"
# SES header, "SES VER 4.10             Patchway tunnel pressure tests from Figure 3 of Gawthorpe & Pope,
# SES footer, "FILE: Patchway-76-01a.ses                                                    SIMULATION TI
# Program, SES 4.10
# Date created, "12:39 on 13 Jun 2023 by tester"
# Property plotted, -DP
# Property description, total pressure
# Original locator, sec810
# Modified locator, sec810
#
#
#
# X conversion key, null, 1.0
# Y conversion key, "press1", converting Pa to IWG by dividing by 249.08853408.
# Auto key, Total pressure in section 810
# Key used, "Total pressure at 100 m, SES v4.1"
# Multiplier on the X axis (applied first), 1.0
# Offset added to the X axis (applied second), -80.0
# Multiplier on the Y axis (applied first), 1.0
# Offset added to the Y axis (applied second), 0.0
# X value at which data starts being plotted, 0
# X value at which data stops being plotted, inf
Time,         Value,          Value,          Plot time,     Plot value
 (s) ,         (Pa),          (IWG),             (s)   ,        (Pa)
0.0,           -0.0,          -0.0,
79.0,          -0.0,          -0.0,
80.0,          -0.0,          -0.0,              0.0,           0.0
81.0,       1708.577963,      6.85932,           1.0,        1708.577963
82.0,       2146.024756,      8.61551,           2.0,        2146.024756
83.0,        938.000165,      3.76573,           3.0,         938.000165
84.0,        282.842521,      1.13551,           4.0,         282.842521
```

The excerpt above illustrates what happens when the amount of data is being cropped. In this example, an SES file was run in which a train entered the tunnel at 80 seconds. It was plotted alongside full-scale test data in which the train entered at zero seconds (the data on the previous page, as it happens). The curve has an X offset of -80 seconds and X value at which data starts to be plotted is zero seconds, so the first two rows have three entries instead of five (Gnuplot can handle being asked to plot numbers that are not present).

Example of QA for fan data (6 columns)

```
# Hobyah source file, "ok-009.txt"
# Timestamp & user, "15:54 on 13 Jun 2023 by tester"
# Project number, "2"
# Project name, "Software testing"
# Project description, "Files that work and test different capabilities"
# 1115th line, "fandata fanchar  calc  Main-extract@9999"
# Data file name, "ok-009-p29-g1-c1.txt"
#
# Hobyah system/fan performance QA
# Name of binary, "ok-009.hbn"
# Source name, "ok-009.txt"
# Project no., "2"
# Project name, "Software testing"
# Description, "Files that work and test different capabilities"
# Date created, "12:39 on 13 Jun 2023 by tester"
# Property plotted, fanchar
# Property description, fan characteristic
# Original locator, Main-extract@9999
# Modified locator, Main-extract@9999.0
#
#
#
# X conversion key, "volflow", converting m^3/s to CFM by dividing by 0.0004719474432.
# Y conversion key, "press2", converting Pa to IN. HG by dividing by 3386.42425928967
# Auto key, P_t characteristic of fan "main-extract" at 700 sec (forwards)
# Key used, '"{/*0.65 P\\_t characteristic of fan \"main-extract\" at 700 sec (forwards)}"'
# Multiplier on the X axis (applied first), 1.0
# Offset added to the X axis (applied second), 0.0
# Multiplier on the Y axis (applied first), 1.0
# Offset added to the Y axis (applied second), 0.0
# X value at which data starts being plotted, -inf
# X value at which data stops being plotted, inf
Volume flow,    Value,      Volume flow,    Value,   Plot Volume flow,    Plot value
 (m3/s),        (Pa),          (cfm),      (IN. HG),      (m3/s),            (Pa)
0.0,          2481.235484,   0.0,          0.732700,        0.0,          2481.235484
10.0,         2431.610775,   21188.8,      0.718046,       10.0,          2431.610775
20.0,         2372.061123,   42377.6,      0.700461,       20.0,          2372.061123
30.0,         2302.586529,   63566.4,      0.679946,       30.0,          2302.586529
40.0,         2243.036878,   84755.2001,   0.662361,       40.0,          2243.036878
50.0,         2183.487226,   105944.0001,  0.644776,       50.0,          2183.487226
60.0,         2084.237807,   127132.8001,  0.615468,       60.0,          2084.237807
80.0,         1786.489549,   169510.4002,  0.527544,       80.0,          1786.489549
100.0,        1439.116581,   211888.0003,  0.424966,      100.0,          1439.116581
120.0,        1042.118903,   254265.6003,  0.307734,      120.0,          1042.118903
140.0,        496.2470969,   296643.2004,  0.146540,      140.0,          496.2470969
```

Example of QA for a profile along a route at a fixed time (7 columns)

```
# Hobyah source file, "ok-038-Channel-Tunnel.txt"
# Timestamp & user, "10:50 on 14 Jun 2023 by tester"
# Project number, "2"
# Project name, "Software testing"
# Project description, "Files that work and test different capabilities"
# 280th line, "profile Ptot  calc  RT-North@*time  xdiv:= 1000"
# Data file name, "ok-038-Channel-Tunnel-lp1-p998-g2-c1.txt"
#
# Hobyah profile (transient data) QA
# Name of binary, "ok-038-Channel-Tunnel.hbn"
# Source name, "ok-038-Channel-Tunnel.txt"
# Project no., "2"
# Project name, "Software testing"
# Description, "Files that work and test different capabilities"
# Date created, "10:20 on 16 May 2023 by tester"
# Property plotted, Ptot
# Property description, total pressure
# Original locator, RT-North@*time
# Modified locator, RT-North@997.0
# Plot time wanted, 997.0
# Plot time used, 997.0
#
# X conversion key, "dist1", converting m to FT by dividing by 0.3048
# Y conversion key, "press1", converting Pa to IWG by dividing by 249.08853408
# Auto key, total pressure along tunnel "rt-north" at 997 sec
# key used, "{/*0.65 total pressure along tunnel \"rt-north\" at 997 sec}"
# Multiplier on the X axis (applied first), 0.001
# Offset added to the X axis (applied second), 0.0
# Multiplier on the Y axis (applied first), 1.0
# Offset added to the Y axis (applied second), 0.0
# X value at which data starts being plotted, -inf
# X value at which data stops being plotted, inf
Distance,      Value,   Distance,      Value, Plot Distance, Plot value
   (m),        (Pa),      (ft),        (IWG),      (m),         (Pa),      Location
10000.0,   -0.7117804, 32808.39895, -0.0028575, 10.0,      0.7117804, # rt_north@10000.0
10014.705, -0.7917811, 32856.64659, -0.0031787, 10.014705, 0.7917811, # rt_north@10014.71
10029.411, -0.8717613, 32904.89424, -0.0034998, 10.029411, 0.8717613, # rt_north@10029.41
10044.117, -0.9517432, 32953.14188, -0.0038209, 10.044117, 0.9517432, # rt_north@10044.12
10058.823, -1.0317271, 33001.38953, -0.0041420, 10.058823, 1.0317271, # rt_north@10058.82
10073.529, -1.1117135,  3049.63717, -0.0044631, 10.073529, 1.1117135, # rt_north@10073.53
10088.235, -1.1917024, 33097.88482, -0.0047842, 10.088235, 1.1917024, # rt_north@10088.24
10102.941, -1.2716944, 33146.132468, -0.005105, 10.102941, 1.2716944, # rt_north@10102.94
```

Note that the units in the header text of the plot columns are the base units: they do not reflect any multipliers applied to the values in the plot columns. There is an example of this above, in the `Plot distance` column. The unit for the distances is stored in metres. But a multiplier of 0.001 was applied (to convert the distances into km) for the plot units. The units text at the top of the column of X data to be plotted is still `m`, not `km`. It is easy to forget about this and be misled by the units text in the rightmost two columns.

Example of QA for the intrinsic properties of a train (7 columns)

```
# Hobyah source file, "ok-009.txt"
# Timestamp & user, "10:59 on 14 Jun 2023 by tester"
# Project number, "2"
# Project name, "Software testing"
# Project description, "Files that work and test different capabilities"
# 818th line, "transient  speed slugflow1  train1@0   xaxis:=chainage"
# Data file name, "ok-009-p18-g1-c2.txt"
#
# SES moving train data QA (time or distance on X axis)
# Name of binary, "SES-060-normal-ops-sample.sbn"
# SES file, "SES-060-normal-ops-sample.ses"
# SES header, "SES VER 4.10         SES v4.1 file with a simple rail tunnel (up line, down line
# SES footer, "FILE: SES-060-normal-ops-sample.ses                                          SI
# Program, SES 4.10
# Date created, "15:53 on 13 Jun 2023 by tester"
# Property plotted, speed
# Property description, speed of train
# Original locator, train1@0
# Modified locator, train1
#
#
#
# X conversion key, "dist1", converting m to FT by dividing by 0.3048.
# Y conversion key, "speed2", converting km/h to MPH by dividing by 1.609344.
# Auto key, Speed of train 1
# key used, "{/*0.65 Speed of train 1}"
# Multiplier on the X axis (applied first), 1.0
# Offset added to the X axis (applied second), 0.0
# Multiplier on the Y axis (applied first), 1.0
# Offset added to the Y axis (applied second), 0.0
# X value at which data starts being plotted, -inf
# X value at which data stops being plotted, inf
Time,       Distance,       Value,    Distance,     Value,    Plot distance,   Plot value
 (s) ,        (m)   ,      (km/h),       (ft)  ,    (MPH),        (m)    ,     (km/h)
5.0,      3292.254528,   18.62011008,  10801.36,    11.57,     3292.254528,    18.62011008
10.0,     3334.271208,   41.55326208,  10939.21,    25.82,     3334.271208,    41.55326208
15.0,     3404.667816,   59.04683136,  11170.17,    36.69,     3404.667816,    59.04683136
20.0,     3497.037456,   73.74014208,  11473.22,    45.82,     3497.037456,    73.74014208
25.0,     3606.21072,    80.00049024,  11831.4,     49.71,     3606.21072,     80.00049024
30.0,     3717.322512,   80.00049024,  12195.94,    49.71,     3717.322512,    80.00049024
35.0,     3828.434304,   80.00049024,  12560.48,    49.71,     3828.434304,    80.00049024
40.0,     3939.546096,   80.00049024,  12925.02,    49.71,     3939.546096,    80.00049024
45.0,     4050.657888,   80.00049024,  13289.56,    49.71,     4050.657888,    80.00049024
50.0,     4161.76968,    80.00049024,  13654.1,     49.71,     4161.76968,     80.00049024
55.0,     4272.881472,   80.00049024,  14018.64,    49.71,     4272.881472,    80.00049024
```

This is a transient plot, but the X values being plotted are distance. What gives? This is a plot on a moving train, so the train is at different locations in each time step. The text `xaxis:=chainage` in the curve definition is an optional entry in train plots that tells the software to not plot the X axis as time (which is the default) but as the chainage of the down end of the train.

5 Verification of `SESconv.py`'s annulus volume flow calculation

`SESconv.py` has code that calculates volume flow of air in the annulus around moving trains. The purpose of this section is to show the workings of the calculation in two ways:

- Plot the properties that are used in the calculation in an example SES run and explain the calculation process, with examples of how to interrogate the binary file.
- Calculate a spot value, quoting extracts from the SES `.PRN` file that the data comes from. Compare that to the value produced by the code in `SESconv.py`.

The file used is `SES-062-trains-crossing.ses`. This models a twin-track tunnel with trains of different sizes and different lengths running in both directions approximately every 50 seconds. There is a vent shaft at the midpoint extracting air.

Figure 2 shows the geometry and the volume flow of air before any trains enter: in from both portals and out at the vent shaft. Positive values of volume flow represent flow from left to right on the Figure; negative values of volume flow represent flow right to left.

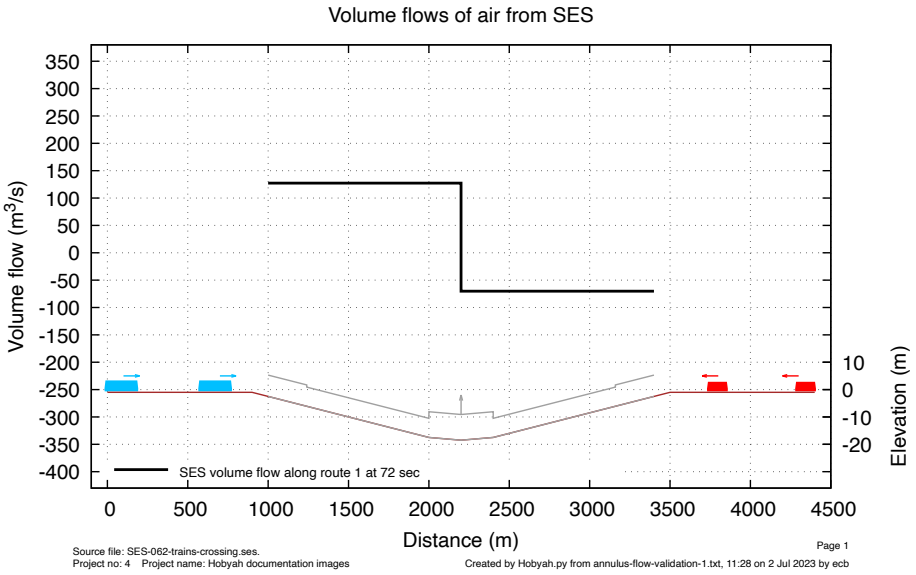

Figure 2: General arrangement of the example file

At the base of Figure 2 are vertical profiles that show the train routes and the segments. The upper vertical profile illustrates changes of area along the

length of the tunnel and the location of the vent shaft. The lower vertical profile illustrates track levels in the routes. The blue polygons represent trains that move left to right through the tunnel, the red polygons represent trains that move right to left.

The whole sequence of train movements and the resulting airflow profiles can be seen in the `.pdf` flipbook named `annulus-flow-validation-1-lp1.pdf`. You should flip back and forward through that file before reading any further: one `.pdf` flipbook illustrating how trains move and affect airflow is worth a thousand of the boring paragraphs in this section of this verification document.

Figure 3 shows the variation of train speeds versus distance as they travel through the model. The speeds have been converted to m/s and multiplied by the relevant train areas (m^2) so that the curves are volume flows of trains (in m^3/s). This may sound insane and irrelevant; it should make sense in a few pages.

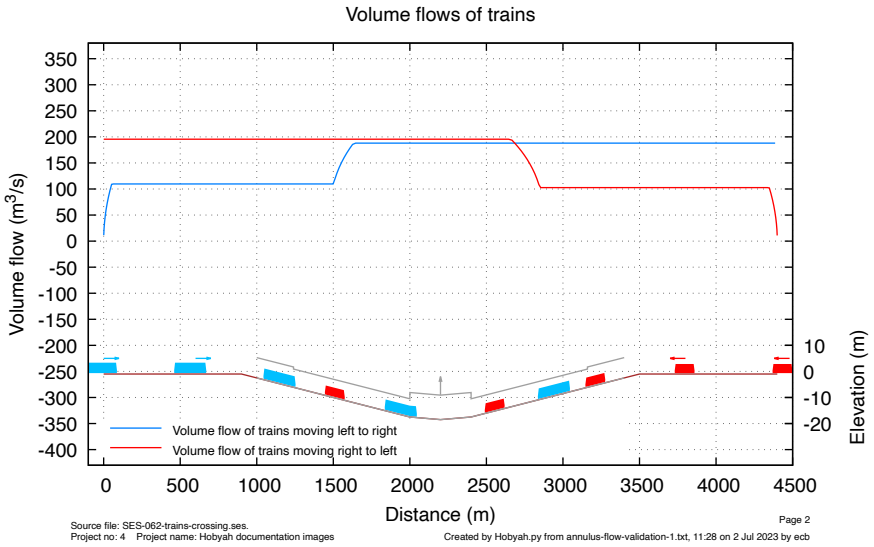

Figure 3: Variation of train volume flow with distance (loci are nose locations)

The trains accelerate to the permitted speed at the start of their respective routes, then increase speed shortly after they enter the tunnels (this speed change was included to check that the speed at each time is being handled correctly).

The combination of air volume flows and train volume flows allows the volume flows in the annulus around trains to be calculated.

5.1 Annulus volume flow at solitary trains

Figure 4 shows three volume flows relevant to the calculation, at 213 seconds:

1. Volume flow in the open tunnel (printed out by SES)—the black line
2. Volume flows of trains (calculated in `classSES.py`)—the blue and red lines
3. Volume flows in the annulus around trains (calculated in `SESconv.py`)—the grey line

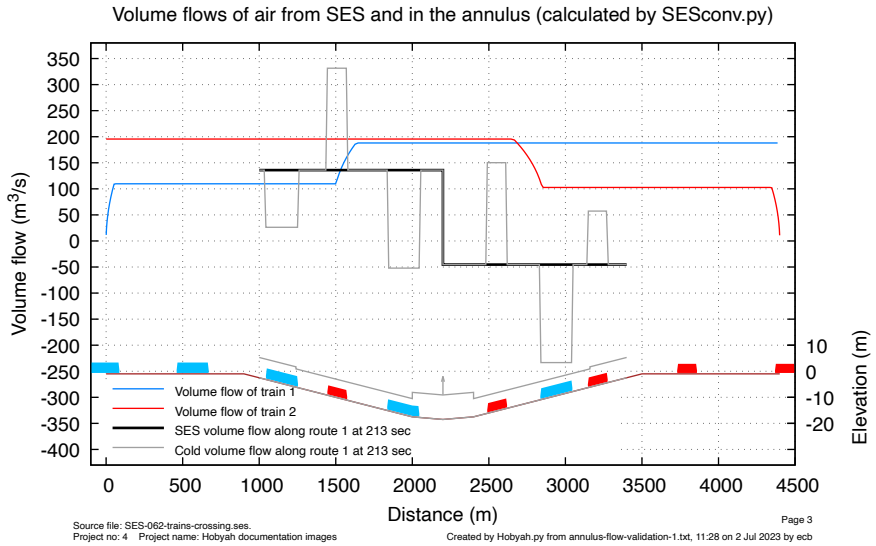

Figure 4: Volume flows versus distance

The blue train at 1250 m is travelling slowly, at a rate of approximately 110 m^3/s. The volume flow in the annulus around that train drops by the same amount.

The other two blue trains in the tunnel are travelling faster, and cause a change in volume flow of approximately 190 m^3/s. Once again the change in volume flow in the annulus around those trains is correct. The rightmost blue train in the tunnel is in a segment that is reversed in the route: this is evidence that the sign of the flows is handled correctly regardless of the orientation of the segments in the route.

The red trains are all moving left to right. The same points (small change in annulus airflow where the train is moving slowly, large change where trains are moving quickly; segment orientation is handled correctly) can be made about the red trains. The only difference is that the flowrates of the red trains cause air volume flow to increase, as you would expect, given that the red trains are moving in the up direction as considered by route 1 (from 4500 to 0 on the X axis).

5.2 Combinations of trains

Next we turn our attention to the combining the effects of moving trains, showing what happens when trains pass one another inside a tunnel. Figure 5 shows three trains crossing in the tunnels at 272 seconds. One pair are crossing under the vent shaft, making the airflows alongside them complex.

Consider the leftmost pair of passing trains (at approximate chainage 1250). The blue train is moving slowly and the red train is moving quickly, so the combined effect is to make the airflow more positive.

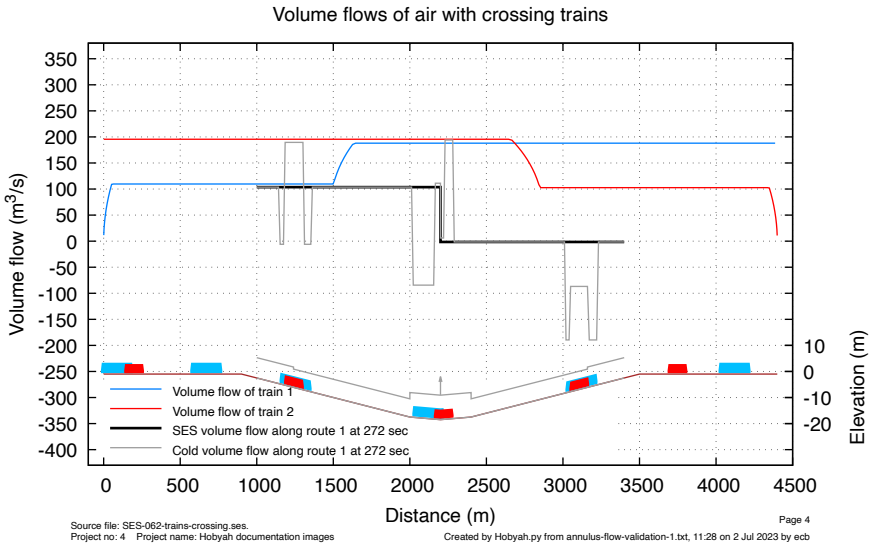

Figure 5: Volume flows versus distance

Consider the rightmost pair of crossing trains (at approximate chainage 3100). The blue train is moving quickly and the red train is moving slowly, so the combined effect is to make the airflow more negative.

The pair in the middle are both moving fast, but the red train is moving slightly more train volume than the blue train (195 m^3/s compared to 188 m^3/s). The combined effect is a change of approximately +7 m^3/s, which is what is shown on both sides of the vent shaft (zoom in and compare the grey line to the black line at the vent shaft).

5.3 Description of the calculation

It may be useful to give an example of how engineers can look at the data calculated by `SESconv.py`. `SESconv.py` writes a text file in SI units, but it also stores its results in a binary file (a Python pickle file). Included in the Hobyah suite is a Python module (`classSES.py`) that can be used to interrogate these files. It

is used by the Hobyah program to get plot data out of binary files; it can also be imported into a Python interpreter and its methods used interactively. The following is a transcript showing the commands used (starting from a Terminal session in macOS). The Hobyah Python scripts are held in a directory called `/Users/tester/Documents/sandboxes/Hobyah` and the test files are in a folder called `/Users/tester/Documents/sandboxes/Hobyah/files-SES`

```
% cd /Users/tester/Documents/sandboxes/Hobyah
% python3
>>> import classSES as clS
>>> folder = "/Users/tester/Documents/sandboxes/Hobyah/files-SES"
>>> fname = "SES-062-trains-crossing.sbn"
>>> log = open("log_discard.txt", "w")
>>> s062 = clS.SESdata(folder, fname, log)
```

The above code changes to a suitable directory on the machine, starts a Python interactive session, creates two strings with the names of the folder and the file, opens a log file to which a class instance can write error messages, imports the class then creates a class instance by passing the folder, filename and log file handle to the class. The class (which is called `s062`) has two functions to describe and show the entries: `s062.Describe()` and `s062.PrettyClassPrint()`:

```
>>> s062.Describe("seg_flows")
A pandas DataFrame giving the volume flows in the open
tunnel in every segment at every timestep.  The columns
are integers (the segment numbers).  The indices are the
print times given as floats.
The values are in m^3/s.
>>> s062.PrettyClassPrint("seg_flows")
Printing the contents of "seg_flows" in "SES-062-trains-crossing.sbn".
Rounding to 3 decimal places.
  pandas DataFrame "self.seg_flows":
            101      102      103     104     106     105      901
0.0       0.000    0.000    0.000   0.000   0.000   0.000    0.000
1.0       0.001    0.001    0.001  -0.001   0.001   0.001    0.003
2.0       0.003    0.003    0.003  -0.003   0.003   0.003    0.006
3.0       0.004    0.004    0.004  -0.004   0.004   0.004    0.008
4.0       0.006    0.006    0.006  -0.006   0.006   0.006    0.011
...         ...      ...      ...     ...     ...     ...      ...
291.0   159.495  159.495  159.495 -17.081  17.081  17.081  176.576
292.0   159.760  159.760  159.760 -17.891  17.891  17.891  177.651
293.0   159.350  159.350  159.350 -18.556  18.556  18.556  177.906
294.0   158.526  158.526  158.526 -19.512  19.512  19.512  178.038
295.0   151.968  151.968  151.968 -22.364  22.364  22.364  174.332
```

The `PrettyClassPrint` function takes an optional count of decimal places:

```
>>> r062.PrettyClassPrint("seg_flows", 6)
Printing the contents of "seg_flows" in "SES-062-trains-crossing.sbn".
Rounding to 6 decimal places.
  pandas DataFrame "self.seg_flows":
               101         102         103         104         106         105         901
```

```
0.0      0.000000    0.000000    0.000000   0.000000   0.000000   0.000000    0.000000
1.0      0.001369    0.001369    0.001369  -0.001369   0.001369   0.001369    0.002784
2.0      0.002784    0.002784    0.002784  -0.002784   0.002784   0.002784    0.005569
3.0      0.004153    0.004153    0.004153  -0.004153   0.004153   0.004153    0.008353
4.0      0.005569    0.005569    0.005569  -0.005569   0.005569   0.005569    0.011138
...           ...         ...         ...        ...        ...        ...         ...
291.0  159.495110  159.495110  159.495110 -17.080533  17.080533  17.080533  176.575643
292.0  159.759967  159.759967  159.759967 -17.891150  17.891150  17.891150  177.651164
293.0  159.349939  159.349939  159.349939 -18.556313  18.556313  18.556313  177.906252
294.0  158.525636  158.525636  158.525636 -19.512337  19.512337  19.512337  178.037973
295.0  151.968351  151.968351  151.968351 -22.364032  22.364032  22.364032  174.332336
```

The downside of that is that the pandas print function cuts out columns to fit in the width of the column. You can't have it all.

The DataFrame used in the example commands above is `seg_flows`, the volume flow in segments.

`SESconv.py` takes the contents of this DataFrame and builds a new DataFrame with the same volume flows at each time, but with values at every subpoint rather than at every segment (the DataFrame is called `subpoint_coldflows`).

Next, `SESconv.py` generates a DataFrame (`subpoint_trainflows`) of the same size as `subpoint_coldflows` and fills it with zeros. It then runs a looped calculation that turns the zeros into the net volume flow of trains passing each subpoint. The calculation has three levels of looping:

1. The outer level loops over each time
2. The middle level loops over each train active at the current time
3. The inner level loops over each subpoint identifier

This code is near the end of function `ReadTimeSteps` in `SESconv.py`. The interesting stuff happens in the inner loop. The code checks if a part of the current train is alongside the current subpoint. If it is, it adds or subtracts the train volume flow to/from the value for that subpoint at the current time, using the following rules:

- If the train is moving from the back end of the subsegment to the forward end it adds the train volume flow.
- If the train is moving in the opposite direction or is stationary it subtracts the train volume flow.

At the end of the calculation, the DataFrame holds the cumulative volume flows of trains at each subpoint in m^3/s:

```
>>> s062.PrettyClassPrint("subpoint_trainflows")
Printing the contents of "subpoint_trainflows" in "SES-062-trains-crossing.sbn".
Rounding to 3 decimal places.
  pandas DataFrame "self.subpoint_trainflows":
  101-1b   101-1m   101-1f   101-2b  ...  901-15f  901-16b  901-16m  901-16f
0.0      0.000    0.000    0.000    0.000  ...      0.0      0.0      0.0      0.0
1.0      0.000    0.000    0.000    0.000  ...      0.0      0.0      0.0      0.0
2.0      0.000    0.000    0.000    0.000  ...      0.0      0.0      0.0      0.0
```

```
3.0      0.000    0.000    0.000    0.000  ...      0.0      0.0      0.0      0.0
4.0      0.000    0.000    0.000    0.000  ...      0.0      0.0      0.0      0.0
...        ...      ...      ...      ...  ...      ...      ...      ...      ...
291.0    0.000    0.000    0.000    0.000  ...      0.0      0.0      0.0      0.0
292.0  109.676    0.000    0.000    0.000  ...      0.0      0.0      0.0      0.0
293.0  109.676  109.676    0.000    0.000  ...      0.0      0.0      0.0      0.0
294.0  109.676  109.676  109.676  109.676  ...      0.0      0.0      0.0      0.0
295.0  109.676  109.676  109.676  109.676  ...      0.0      0.0      0.0      0.0
```

Note the series of non-zero value starting at 292 seconds and moving into the tunnel: this is a train entering the left-hand side of segment 101 with a train volume flow of +109.676 m^3/s. Recall that at 292 seconds the inflow of air in segment 101 was 159.76 m^3/s.

Once `SESconv.py` finishes calculating the net flow of trains at each subpoint, the program subtracts the DataFrame `subpoint_trainflows` from the initial values in `subpoint_coldflows`. The effect is to make `subpoint_coldflows` hold the annulus volume flows around moving trains:

```
>>> r062.PrettyClassPrint("subpoint_coldflows")
Printing the contents of "subpoint_coldflows" in "SES-062-trains-crossing.sbn".
Rounding to 3 decimal places.
  pandas DataFrame "self.subpoint_coldflows":
        101-1b   101-1m   101-1f   101-2b  ...  901-15f  901-16b  901-16m  901-
16f
0.0      0.000    0.000    0.000    0.000  ...    0.000    0.000    0.000    0.000
1.0      0.001    0.001    0.001    0.001  ...    0.003    0.003    0.003    0.003
2.0      0.003    0.003    0.003    0.003  ...    0.006    0.006    0.006    0.006
3.0      0.004    0.004    0.004    0.004  ...    0.008    0.008    0.008    0.008
4.0      0.006    0.006    0.006    0.006  ...    0.011    0.011    0.011    0.011
...        ...      ...      ...      ...  ...      ...      ...      ...      ...
291.0  159.495  159.495  159.495  159.495  ...  176.576  176.576  176.576  176.576
292.0   50.084  159.760  159.760  159.760  ...  177.651  177.651  177.651  177.651
293.0   49.674   49.674  159.350  159.350  ...  177.906  177.906  177.906  177.906
294.0   48.850   48.850   48.850   48.850  ...  178.038  178.038  178.038  178.038
295.0   42.292   42.292   42.292   42.292  ...  174.332  174.332  174.332  174.332
```

The numbers at 292 seconds and later show the progress of a train as it enters the left-hand side of the tunnel. This maps to the change in annulus volume flow at chainage 1000.

Figure 6 shows the state of the system at 294 seconds, with the nose of a blue train just inside the left-hand tunnel portal (segment 101) and a step in the curve of cold volume flow. If you find all this difficult to follow, I recommend (once again) looking at the `.pdf` flipbook named `annulus-flow-validation-1-lp1.pdf` and flipping back and forward through its pages. I've often thought that this automated calculation was wrong, looked into it by loading a class instance, then concluded that my suspicions were unfounded.

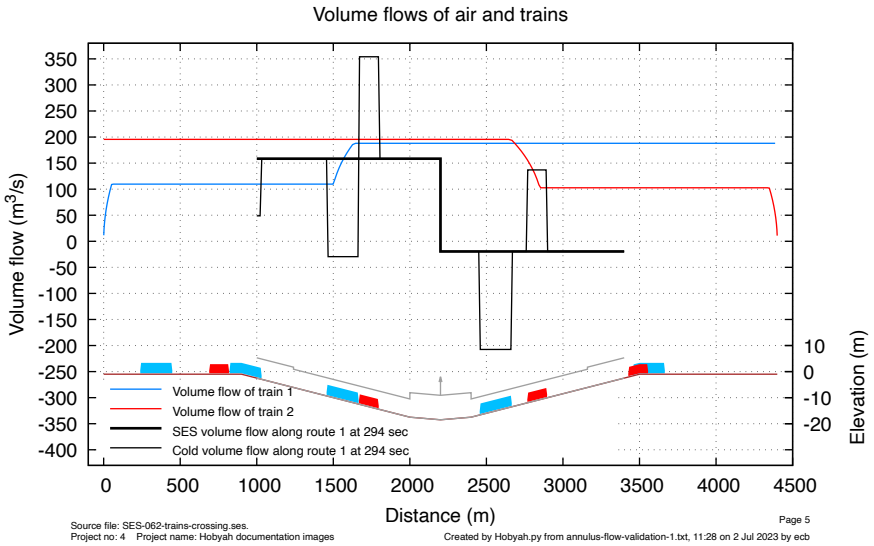

Figure 6: Volume flows versus distance at 294 seconds

5.4 Manual calculation

This section has a manual calculation of one of the locations at which two trains are passing each other in opposite directions. The SES output file that is being quoted is called `SES-062-trains-crossing.PRN`.

The time selected is 272 seconds, the location is the midpoint of segment 105, subsegment 5. This is alongside the rightmost pair of passing trains on Figure 7 (on the following page).

The calculation takes the following values from the `.PRN` file:

- Area of the segment
- Lengths of the subsegments
- Type of each train
- Areas of the train types
- Lengths of the train types
- Location of the selected subsegment in the routes
- Speed of each train
- Volume flow in the segment

From Figure 7 we expect the manual calculation to have a volume flow in the open tunnel of just under zero and the combined effect of the train movements to cause the annulus volume flow to be about -90 m^3/s.

First get the area of segment 105 and the length of its subsegments from `SES-062-trains-crossing.PRN`:

- `INPUT VERIFICATION FOR LINE SEGMENT 105 -105 28 m^2, 760 m tunnel FORM 3A`
  `LINE SEGMENT TYPE 1 (TUNNEL)`
  `LENGTH 2493.43 FT`
  `CROSS SECTION AREA 301.4 SQ FT`
  `NUMBER OF SUBSEGMENTS 38`

You can check these yourself by loading `SES-062-trains-crossing.PRN` into a text editor.

Turning the values into SI units, they become:

- The tunnel area is `301.4 ft`$^2 \equiv 301.4 \times 0.3048^2 = 28.0$ m^2.
- The tunnel length is `2493.43 ft` $\equiv 2493.43 \times 0.3048 = 760$ m.
- There are 38 subsegments, so each subsegment is $760/38 = 20$ m long.

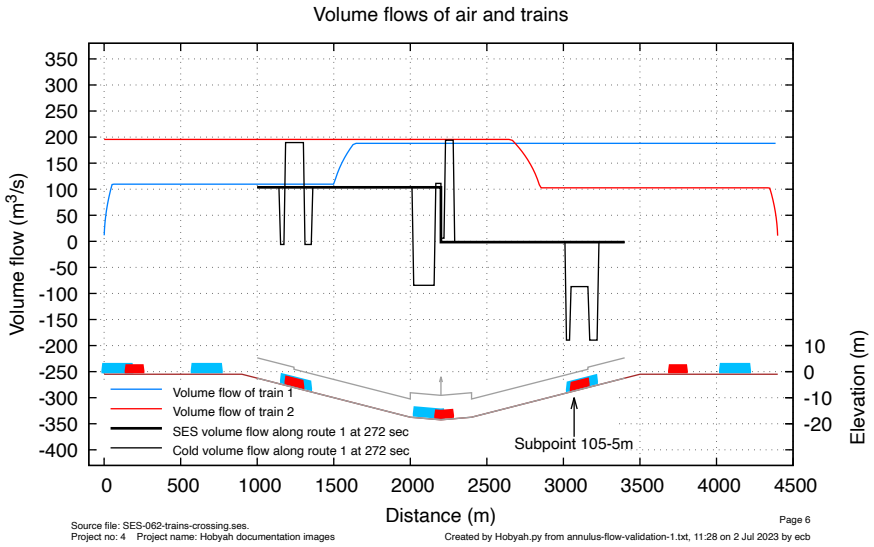

Figure 7: Location and time of the hand calculation

Next are the printouts for train types (form 9). There are two train types in the `.PRN` file: large, long trains (used on route 1, the blue train icons) and small, short trains (used on route 2, the red train icons).

- Areas and lengths of the train types, from form 9A in the SES output file:

  `INPUT VERIFICATION FOR TRAIN TYPE 1 Large area train FORM 9A`
  `TOTAL LENGTH OF TRAIN 689.0 FT`
  `FRONTAL AREA OF TRAIN 101.18 SQ FT`

  `INPUT VERIFICATION FOR TRAIN TYPE 2 Small area train FORM 9A`
  `TOTAL LENGTH OF TRAIN 426.5 FT`
  `FRONTAL AREA OF TRAIN 94.72 SQ FT`

From these:

- Trains of type 1 have length `689 ft`$^2 \equiv 689 \times 0.3048 = 210$ m.
- Trains of type 1 have area `101.18 ft`$^2 \equiv 101.18 \times 0.3048^2 = 9.4$ m^2.
- Trains of type 2 have length `426.5 ft`$^2 \equiv 426.5 \times 0.3048 = 130$ m.
- Trains of type 2 have area `94.72 ft`$^2 \equiv 94.72 \times 0.3048^2 = 8.8$ m^2.

Now we figure out where our specimen subpoint `105-5m` is on the train routes. In route 1 (left to right on the graphs) the output of form 8F is:

```
      SECTIONS AND SEGMENTS THROUGH WHICH ROUTE PASSES
SECTION             SEGMENT          LOCATION OF SEGMENT ALONG ROUTE
NUMBER              NUMBER              (FEET)             (FEET)
 101                 101                3280.8    TO      4068.2
                     102                4068.2    TO      6561.7
 103                 103                6561.7    TO      7217.8
 104                 104                7217.8    TO      7874.0
-105                -105                7874.0    TO     10367.4
                    -106               10367.4    TO     11154.8
```

Segment 105 is reversed in route 1. Its forward end is at `7874.0 ft` (2400 m) and the back end is at `10367.4 ft` (3160 m). We subtract the length of four and a half subsegments to get to the midpoint of subsegment 5: $3160 - 4.5 \times 20 = 3070$ m.

For route 2 (right to left on the graphs) the output of form 8F is:

```
      SECTIONS AND SEGMENTS THROUGH WHICH ROUTE PASSES
SECTION             SEGMENT          LOCATION OF SEGMENT ALONG ROUTE
NUMBER              NUMBER              (FEET)             (FEET)
 105                 106                3280.8    TO      4068.2
                     105                4068.2    TO      6561.7
-104                -104                6561.7    TO      7217.8
-103                -103                7217.8    TO      7874.0
-101                -102                7874.0    TO     10367.4
                    -101               10367.4    TO     11154.8
```

Segment 105 is not reversed in route 2. Its back end is at `4068.2 ft` (1240 m) on route 2 and the forward end is at `6561.7 ft` (2000 m). We add the length of four and a half subsegments to get to the midpoint of subsegment 5: $1240 + 4.5 \times 20 = 1330$ m.

Now take the runtime printout of train speeds and locations at 272 seconds:

```
TIME    272.00 SECONDS                          11 TRAIN(S) ARE OPERATIONAL
      R T                                 AIR
TRAIN T Y  LOCATION  SPEED  ACCELERATION  DRAG
 NO.  E P   (FEET)   (MPH)   (MPH/SEC)   (LBS)

  1   1 1  13864.39  44.74      0.00     1021.
  2   2 2  13996.98  49.71      0.00      490.
  3   1 1  10583.63  44.74      0.00     5993.
  4   2 2  10570.88  49.71      0.00     5277.
  5   1 1   7302.77  44.74      0.00      683.
  6   2 2   7362.28  49.71      0.00     1076.
```

```
 7   1 1   4457.37  26.10      0.00    -3430.
 8   2 2   4457.37  26.10      0.00    -3191.
 9   1 1   2543.52  26.10      0.00      347.
10   2 2   2352.14  26.10      0.00      135.
11   1 1    629.66  26.10      0.00      347.
```

From this,

- Train 3 on route 1 has its nose at 10583.63 feet ($\equiv 10583.63 \times 0.3048 = 3225.89$ m).
- Train 3 is of train type 1 (210 m long) so its tail is at 3015.89 m. It must lie alongside subpoint `105-5m` (which is at 3070 m on route 1).
- Train 3 is travelling at 44.74 mph ($\equiv 44.74 \times 1.609344 = 72$ km/h). 72 km/h is 20 m/s and the train has area 9.4 m^2. The volume flow of train is $20 \times 9.4 = 188$ m^3/s.
- Train 8 on route 2 has its nose at 4457.37 feet ($\equiv 4457.37 \times 0.3048 = 1358.61$ m).
- Train 8 is of train type 2 (130 m long) so its tail is at 1228.61 m. It must lie alongside subpoint `105-5m` (which is at 1330 m on route 2).
- Train 8 is travelling at 26.1 mph ($\equiv 26.1 \times 1.609344 = 42$ km/h). 42 km/h is $11.66\dot{6}$ m/s and the train has area 8.8 m^2, so the volume flow of train is $8.8 \times 11.66\dot{6} = 102.66\dot{6}$ m^3/s.

The final piece of information from the `.PRN` file is volume flow in the open tunnel, from the runtime printout of conditions in segment 105 at 272 seconds:

```
            SYSTEM              AIR       HUMIDITY     AIR         AIR     TRAIN POSITION
 LENGTH   PARTITIONING      TEMPERATURE    RATIO       FLOW     VELOCITY   0 0 0 0 0 0 0 0 0 0 1 1 1 1 1
  (FT)                       (DEG F)     (LB/LB)     (CFM)       (FPM)    1 2 3 4 5 6 7 8 9 0 1 2 3 4
 2493.4  105 -105     (TUNNEL)                  28 m^2, 760 m tunnel
         105 -105 -  1         105.33     0.01560      3218.4      10.7    3 8
         105 -105 -  2         105.70     0.01560                          3 8
         105 -105 -  3         105.91     0.01560                          3 8
         105 -105 -  4         105.97     0.01560                          3 8
         105 -105 -  5         105.53     0.01559                          3 8
         105 -105 -  6         103.98     0.01556                          3 8
         105 -105 -  7         102.93     0.01554                          3
         105 -105 -  8         102.60     0.01552                          3
         105 -105 -  9         102.49     0.01548
         105 -105 - 10         102.42     0.01543
```

Note that the above transcript has been edited to remove the sensible and latent heat loads so that the train position entries fit on this page.

The indicators of "train positions" in the transcript confirm that trains 3 and 8 are alongside subsegment 5 of segment 105. The volume flow of air in segment 105 is 3218.4 cfm, which is $3218.4/60 \times 0.3048^3 = 1.52$ m^3/s. Because the segment is reversed in the route along which volume flow was plotted (route 1), this is shown as -1.52 m^3/s when plotted as the SES volume flow in Figure 7, which is just below zero from the central shaft to the portal at the right-hand side.

At this point we have all the necessary figures to calculate annulus volume flow:

- Volume flow of air in the open tunnel in route 1 is -1.52 m^3/s.
- Volume flow of train 3 is 188 m^3/s. Train 3 is going from left to right in Figure 7 (positive train volume flow), so we subtract it from the volume flow of air in the open tunnel.
- Volume flow of train 8 is $102.66\dot{6}$ m^3/s. Train 8 is going right to left in Figure 7 (negative train volume flow), so we add it to the volume flow of air in the open tunnel.
- Net volume flow in the annulus is $-1.52 - 188 + 102.67 = -86.85$ m^3/s.

The following two pages give extracts of the curve data files used to plot the SES volume flow and the cold volume flow in Figure 5. They have been edited to line up the columns and focus on the area of interest (segment 105).

```
# Hobyah source file, "annulus-flow-validation-1.txt"
# Timestamp & user, "11:20 on 14 Jun 2023 by tester"
# Project number, "4"
# Project name, "Hobyah documentation images"
# Project description, "Files that generate images used in the documentation"
# 148th line, "profile  qSES   normal  route1@time3 lt:=20"
# Data file name, "annulus-flow-validation-1-p4-g1-c3.txt"
#
# SES profile (transient data) QA
# Name of binary, "SES-062-trains-crossing.sbn"
# SES file, "SES-062-trains-crossing.ses"
# SES header, "SES VER 4.10      SES v4.1 file.  A bidirectional rail tunnel with an open shaft at the mi
# SES footer, "FILE: SES-062-trains-crossing.ses                                                  SIMULATION TI
# Program, SES 4.10
# Date created, "08:54 on 7 Jun 2023 by tester"
# Property plotted, qSES
# Property description, SES volume flow
# Original locator, route1@time3
# Modified locator, route1@272.0
# Plot time wanted, 272.0
# Plot time used, 272.0
#
#
#
# Auto key, SES volume flow along route 1 at 272 sec
# key used, "{/*0.65 SES volume flow along route 1 at 272 sec}"
# Multiplier on the X axis (applied first), 1.0
# Offset added to the X axis (applied second), 0.0
# Multiplier on the Y axis (applied first), 1.0
# Offset added to the Y axis (applied second), 0.0
# X value at which data starts being plotted, -inf
# X value at which data stops being plotted, inf
Distance,          Value,      Distance,   Value,   Plot Distance,    Plot value
   (m),           (m^3/s),      (ft),      (CFM),       (m),           (m^3/s),     Location
999.98784,    103.6364492841,   3280.8,   219593.2,    999.98784,   103.6364492841,   # 101
1239.98736,   103.6364492841,   4068.2,   219593.2,   1239.98736,   103.6364492841,   # 101
1239.98736,   103.6364492841,   4068.2,   219593.2,   1239.98736,   103.6364492841,   # 102
1999.98482,   103.6364492841,   6561.63,  219593.2,   1999.984824,  103.6364492841,   # 102
1999.98482,   103.6364492841,   6561.63,  219593.2,   1999.984824,  103.6364492841,   # 103
2199.98544,   103.6364492841,   7217.8,   219593.2,   2199.98544,   103.6364492841,   # 103
2199.98544,   -1.51891565119,   7217.8,    -3218.4,   2199.98544,   -1.51891565119,   # 104
2399.98605,   -1.51891565119,   7873.97,   -3218.4,   2399.986056,  -1.51891565119,   # 104
2399.98605,   -1.51891565119,   7873.97,   -3218.4,   2399.986056,  -1.51891565119,   # -105
3159.98352,   -1.51891565119,  10367.4,    -3218.4,   3159.98352,   -1.51891565119,   # -105
3159.98352,   -1.51891565119,  10367.4,    -3218.4,   3159.98352,   -1.51891565119,   # -106
3399.98304,   -1.51891565119,  11154.8,    -3218.4,   3399.98304,   -1.51891565119,   # -106
```

```
# Hobyah source file, "annulus-flow-validation-1.txt"
# Timestamp & user, "11:20 on 14 Jun 2023 by tester"
# Project number, "4"
# Project name, "Hobyah documentation images"
# Project description, "Files that generate images used in the documentation"
# 149th line, "profile  qcold  normal  route1umd@time3 lt:=6 lw:=1"
# Data file name, "annulus-flow-validation-1-p4-g1-c4.txt"
#
# SES profile (transient data) QA
# Name of binary, "SES-062-trains-crossing.sbn"
# SES file, "SES-062-trains-crossing.ses"
# SES header, "SES VER 4.10      SES v4.1 file.  A bidirectional rail tunnel with an open shaft
# SES footer, "FILE: SES-062-trains-crossing.ses                                              SI
# Program, SES 4.10
# Date created, "08:54 on 7 Jun 2023 by tester"
# Property plotted, qcold
# Property description, cold volume flow
# Original locator, route1umd@time3
# Modified locator, route1umd@272.0
# Plot time wanted, 272.0
# Plot time used, 272.0
#
#
# Auto key, Cold volume flow along route 1 at 272 sec
# key used, "{/*0.65 Cold volume flow along route 1 at 272 sec}"
# Multiplier on the X axis (applied first), 1.0
# Offset added to the X axis (applied second), 0.0
# Multiplier on the Y axis (applied first), 1.0
# Offset added to the Y axis (applied second), 0.0
# X value at which data starts being plotted, -inf
# X value at which data stops being plotted, inf
 Distance,        Value,      Distance,    Value,    Plot Distance,   Plot value
   (m),         (m^3/s),       (ft),        (CFM),        (m),         (m^3/s),  Location
3009.98402,    -1.518915,  9875.27565,    -3218.4,     3009.98402,   -1.518915, # -105-8m
3019.98398,  -189.522861,  9908.083947, -401576.2016,  3019.98398, -189.522861, # -105-8b
3019.98398,  -189.522861,  9908.083947, -401576.2016,  3019.98398, -189.522861, # -105-7f
3029.98395,  -189.522861,  9940.892236, -401576.2016,  3029.98395, -189.522861, # -105-7m
3039.98392,  -189.522861,  9973.700526, -401576.2016,  3039.98392, -189.522861, # -105-7b
3039.98392,  -189.522861,  9973.700526, -401576.2016,  3039.98392, -189.522861, # -105-6f
3049.98388,   -86.849328, 10006.508815, -184023.3056,  3049.98388,  -86.849328, # -105-6m
3059.98385,   -86.849328, 10039.317105, -184023.3056,  3059.98385,  -86.849328, # -105-6b
3059.98385,   -86.849328, 10039.317105, -184023.3056,  3059.98385,  -86.849328, # -105-5f
3069.98382,   -86.849328, 10072.125394, -184023.3056,  3069.98382,  -86.849328, # -105-5m
```

The last line above (at 3069.98 m) is the one we want—note that it ends with `-105-5m` in the "location" column. The cold volume flow is -86.849 m^3/s. These numbers are close to the calculated distance of 3070 m and the calculate volume flow of -86.85 m^3/s.

It is also worth noting that the midpoint of subsegment 105-7 (location `105-7m`) has only train 3 in it, and its volume flow is -189.52 m^3/s, which is $-1.52 - 188$.

Finally, the volume flow at the midpoint of subsegment 105-8 has no trains in it and the volume flow there is -1.519 m^3/s. That line also shows -3218.4 cfm, the same as in the SES runtime output at 272 seconds. This is a good example of providing traceability: -3218.4 cfm appears in the SES printout and in this curve data file.

6 Verification of `SESconv.py`'s annulus air velocity calculation

`SESconv.py` has code that calculates air velocity in the annulus around trains. The purpose of this section is to show the workings of the calculation in two ways:

- Plot the properties that are used in the calculation at one time.
- Calculate a spot value, with quoted extracts from the SES `.PRN` file.

Once again, the file used is `SES-062-trains-crossing.ses` at 272 seconds. The same loop that calculated the volume flow in the annulus (described in Section 5) also calculated the annulus area at each subpoint. Figure 8 shows the annulus area profile.

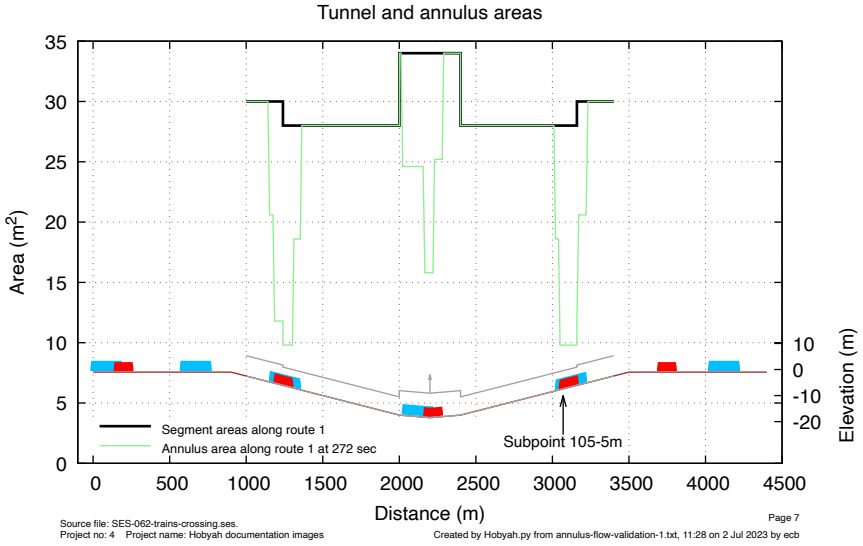

Figure 8: Tunnel areas and annulus areas at 272 seconds

The annulus areas are calculated by generating a DataFrame holding the open tunnel areas at every subpoint and at every time. The DataFrame is called `subpoint_areas`.

The calculation iterates over every time, train and subpoint (it is the same calculation loop given in Section 5.2):

1. The outer level loops over each time
2. The middle level loops over each train active at the current time
3. The inner level loops over each subpoint identifier

When the inner loop finds a train crossing a subpoint, it subtracts that train's area from the area at that subpoint at the current time and stores it in `subpoint_areas`. In this way, the annulus area when multiple trains cross a subpoint is calculated correctly.

After the loop completes, `SESconv.py` creates a new DataFrame called `subpoint_coldvels` by dividing `subpoint_coldflows` by `subpoint_areas`. Dividing pandas DataFrames of the same size by one another is easy.

The hand calculation will be done at subpoint 105-5m at 272 seconds, so that the values used in the volume flow verification can be re-used.

We have the following values from the manual calculation for annulus volume flow (see Section 5):

- Train 3 and train 8 are both alongside subpoint 105-5m.
- Train 3 has area 9.4 m^2.
- Train 8 has area 8.8 m^2.
- The tunnel area is 28.0 m^2.
- The annulus volume flow at subpoint 105-5m is -86.849 m^3/s.

The annulus area is $28 - 9.4 - 8.8 = 9.8$ m^2, which is what is shown on Figure 8 at chainage 3070. The annulus air velocity is $-86.85 \div 9.8 = -8.86$ m/s.

Figure 9 shows the profile of annulus air velocity. At the end of this section are extracts of the annulus volume flow, annulus area and annulus air velocity datafiles that were used to create the annulus velocity curve on Figure 9.

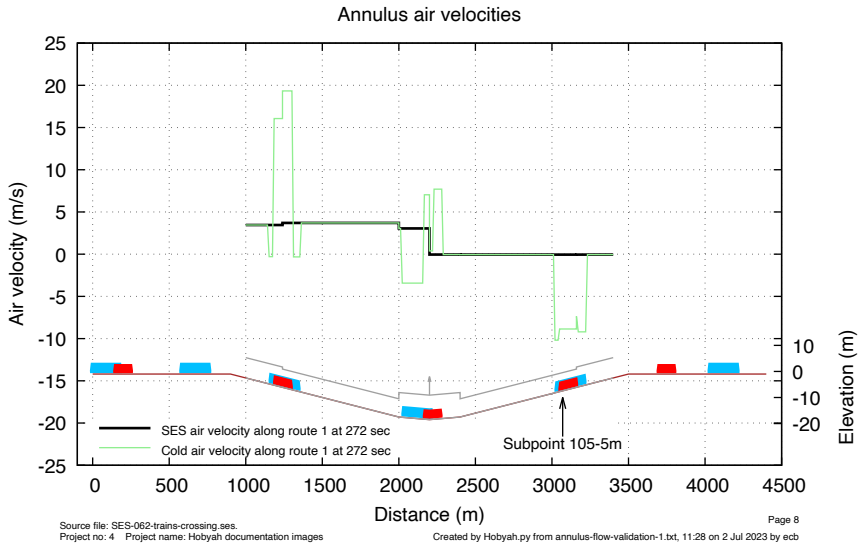

Figure 9: Annulus air velocities at 272 seconds

The extracts of the curve data files on the following three pages show the values at all the subpoints that span across train 3 (from `105-6m` to `106-10f`).

```
# Hobyah source file, "annulus-flow-validation-1.txt"
# Timestamp & user, "11:20 on 14 Jun 2023 by tester"
# Project number, "4"
# Project name, "Hobyah documentation images"
# Project description, "Files that generate images used in the documentation"
# 209th line, "profile  qcold  normal  route1umd@time3 lw:=1"
# Data file name, "annulus-flow-validation-1-p6-g1-c4.txt"
#
# SES profile (transient data) QA
# Name of binary, "SES-062-trains-crossing.sbn"
# SES file, "SES-062-trains-crossing.ses"
# SES header, "SES VER 4.10      SES v4.1 file.  A bidirectional rail tunnel with an open shaft at the mid
# SES footer, "FILE: SES-062-trains-crossing.ses                                                  SIMULATION TI
# Program, SES 4.10
# Date created, "08:54 on 7 Jun 2023 by tester"
# Property plotted, qcold
# Property description, cold volume flow
# Original locator, route1umd@time3
# Modified locator, route1umd@272.0
# Plot time wanted, 272.0
# Plot time used, 272.0
#
#
#
# Auto key, Cold volume flow along route 1 at 272 sec
# Key used, '"{/*0.65 Cold volume flow along route 1 at 272 sec}"'
# Multiplier on the X axis (applied first), 1.0
# Offset added to the X axis (applied second), 0.0
# Multiplier on the Y axis (applied first), 1.0
# Offset added to the Y axis (applied second), 0.0
# X value at which data starts being plotted, -inf
# X value at which data stops being plotted, inf
Distance,   Value,  Distance,    Value, Plot Distance, Plot value
  (m),    (m^3/s),    (ft),      (CFM),       (m),     (m^3/s),    Location
3039.983,-189.522,  9973.7,    -401576.201, 3039.983,  -189.522,  # -105-6f
3049.983, -86.849, 10006.5088, -184023.305, 3049.983,  -86.8493,  # -105-6m
3059.983, -86.849, 10039.3171, -184023.305, 3059.983,  -86.8493,  # -105-6b
3059.983, -86.849, 10039.3171, -184023.305, 3059.983,  -86.8493,  # -105-5f
3069.983, -86.849, 10072.1253, -184023.305, 3069.983,  -86.8493,  # -105-5m
3079.983, -86.849, 10104.9336, -184023.305, 3079.983,  -86.8493,  # -105-5b
3079.983, -86.849, 10104.9336, -184023.305, 3079.983,  -86.8493,  # -105-4f
3089.983, -86.849, 10137.7419, -184023.305, 3089.983,  -86.8493,  # -105-4m
3099.983, -86.849, 10170.5502, -184023.305, 3099.983,  -86.8493,  # -105-4b
3099.983, -86.849, 10170.5502, -184023.305, 3099.983,  -86.8493,  # -105-3f
3109.983, -86.849, 10203.3585, -184023.305, 3109.983,  -86.8493,  # -105-3m
3119.983, -86.849, 10236.1668, -184023.305, 3119.983,  -86.8493,  # -105-3b
3119.983, -86.849, 10236.1668, -184023.305, 3119.983,  -86.8493,  # -105-2f
3129.983, -86.849, 10268.9751, -184023.305, 3129.983,  -86.8493,  # -105-2m
3139.983, -86.849, 10301.7834, -184023.305, 3139.983,  -86.8493,  # -105-2b
3139.983, -86.849, 10301.7834, -184023.305, 3139.983,  -86.8493,  # -105-1f
3149.983, -86.849, 10334.5917, -184023.305, 3149.983,  -86.8493,  # -105-1m
3159.983, -86.849, 10367.4,    -184023.305, 3159.983,  -86.8493,  # -105-1b
3159.983, -86.849, 10367.4,    -184023.305, 3159.983,  -86.8493,  # -106-10f
3171.983,-189.522, 10406.77,   -401576.201, 3171.983,  -189.522,  # -106-10m
```

```
# Hobyah source file, "annulus-flow-validation-1.txt"
# Timestamp & user, "11:20 on 14 Jun 2023 by tester"
# Project number, "4"
# Project name, "Hobyah documentation images"
# Project description, "Files that generate images used in the documentation"
# 238th line, "profile  annulus  normal  route1umd@time3 lt:=3 lw:=1"
# Data file name, "annulus-flow-validation-1-p7-g1-c2.txt"
#
# SES profile (transient data) QA
# Name of binary, "SES-062-trains-crossing.sbn"
# SES file, "SES-062-trains-crossing.ses"
# SES header, "SES VER 4.10      SES v4.1 file.  A bidirectional rail tunnel with an open shaft
# SES footer, "FILE: SES-062-trains-crossing.ses                                              SI
# Program, SES 4.10
# Date created, "08:54 on 7 Jun 2023 by tester"
# Property plotted, annulus
# Property description, annulus area
# Original locator, route1umd@time3
# Modified locator, route1umd@272.0
# Plot time wanted, 272.0
# Plot time used, 272.0
#
#
#
# Auto key, Annulus area along route 1 at 272 sec
# Key used, '"{/*0.65 Annulus area along route 1 at 272 sec}"'
# Multiplier on the X axis (applied first), 1.0
# Offset added to the X axis (applied second), 0.0
# Multiplier on the Y axis (applied first), 1.0
# Offset added to the Y axis (applied second), 0.0
# X value at which data starts being plotted, -inf
# X value at which data stops being plotted, inf
Distance,  Value,    Distance,   Value, Plot Distance, Plot value
   (m),     (m^2),     (ft),    (SQ FT),      (m),       (m^2),    Location
3039.983,  18.601,   9973.700,   200.2,    3039.983,   18.601,   # -105-6f
3049.983,   9.801,   10006.508,  105.5,    3049.983,    9.801,   # -105-6m
3059.983,   9.801,   10039.317,  105.5,    3059.983,    9.801,   # -105-6b
3059.983,   9.801,   10039.317,  105.5,    3059.983,    9.801,   # -105-5f
3069.983,   9.801,   10072.125,  105.5,    3069.983,    9.801,   # -105-5m
3079.983,   9.801,   10104.933,  105.5,    3079.983,    9.801,   # -105-5b
3079.983,   9.801,   10104.933,  105.5,    3079.983,    9.801,   # -105-4f
3089.983,   9.801,   10137.741,  105.5,    3089.983,    9.801,   # -105-4m
3099.983,   9.801,   10170.550,  105.5,    3099.983,    9.801,   # -105-4b
3099.983,   9.801,   10170.550,  105.5,    3099.983,    9.801,   # -105-3f
3109.983,   9.801,   10203.358,  105.5,    3109.983,    9.801,   # -105-3m
3119.983,   9.801,   10236.166,  105.5,    3119.983,    9.801,   # -105-3b
3119.983,   9.801,   10236.166,  105.5,    3119.983,    9.801,   # -105-2f
3129.983,   9.801,   10268.975,  105.5,    3129.983,    9.801,   # -105-2m
3139.983,   9.801,   10301.783,  105.5,    3139.983,    9.801,   # -105-2b
3139.983,   9.801,   10301.783,  105.5,    3139.983,    9.801,   # -105-1f
3149.983,   9.801,   10334.591,  105.5,    3149.983,    9.801,   # -105-1m
3159.983,   9.801,   10367.4,    105.5,    3159.983,    9.801,   # -105-1b
3159.983,  11.798,   10367.4,    127.0,    3159.983,   11.798,   # -106-10f
3171.983,  20.598,   10406.77,   221.7,    3171.983,   20.598,   # -106-10m
```

```
# Hobyah source file, "annulus-flow-validation-1.txt"
# Timestamp & user, "11:20 on 14 Jun 2023 by tester"
# Project number, "4"
# Project name, "Hobyah documentation images"
# Project description, "Files that generate images used in the documentation"
# 267th line, "profile  vcold  normal  route1umd@time3 lt:=3 lw:=1"
# Data file name, "annulus-flow-validation-1-p8-g1-c2.txt"
#
# SES profile (transient data) QA
# Name of binary, "SES-062-trains-crossing.sbn"
# SES file, "SES-062-trains-crossing.ses"
# SES header, "SES VER 4.10      SES v4.1 file.  A bidirectional rail tunnel with an open shaft at the mid
# SES footer, "FILE: SES-062-trains-crossing.ses                                                    SIMULATION TI
# Program, SES 4.10
# Date created, "08:54 on 7 Jun 2023 by tester"
# Property plotted, vcold
# Property description, cold air velocity
# Original locator, route1umd@time3
# Modified locator, route1umd@272.0
# Plot time wanted, 272.0
# Plot time used, 272.0
#
#
#
# Auto key, Cold air velocity along route 1 at 272 sec
# Key used, '"{/*0.65 Cold air velocity along route 1 at 272 sec}"'
# Multiplier on the X axis (applied first), 1.0
# Offset added to the X axis (applied second), 0.0
# Multiplier on the Y axis (applied first), 1.0
# Offset added to the Y axis (applied second), 0.0
# X value at which data starts being plotted, -inf
# X value at which data stops being plotted, inf
Distance, Value,    Distance,     Value, Plot Distance, Plot value
  (m),      (m/s),    (ft),        (FPM),     (m),      (m/s),    Location
3039.983, -10.188,   9973.700,  -2005.674,  3039.983, -10.188,   # -105-6f
3049.983,  -8.861,  10006.508,  -1744.296,  3049.983,  -8.861,   # -105-6m
3059.983,  -8.861,  10039.317,  -1744.296,  3059.983,  -8.861,   # -105-6b
3059.983,  -8.861,  10039.317,  -1744.296,  3059.983,  -8.861,   # -105-5f
3069.983,  -8.861,  10072.125,  -1744.296,  3069.983,  -8.861,   # -105-5m
3079.983,  -8.861,  10104.933,  -1744.296,  3079.983,  -8.861,   # -105-5b
3079.983,  -8.861,  10104.933,  -1744.296,  3079.983,  -8.861,   # -105-4f
3089.983,  -8.861,  10137.741,  -1744.296,  3089.983,  -8.861,   # -105-4m
3099.983,  -8.861,  10170.550,  -1744.296,  3099.983,  -8.861,   # -105-4b
3099.983,  -8.861,  10170.550,  -1744.296,  3099.983,  -8.861,   # -105-3f
3109.983,  -8.861,  10203.358,  -1744.296,  3109.983,  -8.861,   # -105-3m
3119.983,  -8.861,  10236.166,  -1744.296,  3119.983,  -8.861,   # -105-3b
3119.983,  -8.861,  10236.166,  -1744.296,  3119.983,  -8.861,   # -105-2f
3129.983,  -8.861,  10268.975,  -1744.296,  3129.983,  -8.861,   # -105-2m
3139.983,  -8.861,  10301.783,  -1744.296,  3139.983,  -8.861,   # -105-2b
3139.983,  -8.861,  10301.783,  -1744.296,  3139.983,  -8.861,   # -105-1f
3149.983,  -8.861,  10334.591,  -1744.296,  3149.983,  -8.861,   # -105-1m
3159.983,  -8.861,  10367.4,    -1744.296,  3159.983,  -8.861,   # -105-1b
3159.983,  -7.36,   10367.4,    -1449.002,  3159.983,  -7.36,    # -106-10f
3171.983,  -9.2,    10406.77,   -1811.186,  3171.983,  -9.2,     # -106-10m
```

Subpoint 105-5m and values at the subpoints around 3160 m (where there is a change of tunnel area). When I first plotted Figure 9, I thought that the spike in air velocity just to the right of the selected subpoint was an error, an incorrect calculation. If you are equally suspicious (and who could blame you?) below is an edited extract of the plot data files showing annulus volume flows, annulus areas and annulus air velocities at the points where segment 106 ends and segment 105 starts:

```
Chainage    Annulus volume flow    Annulus area   Air velocity
   (m)            (m^3/s)              (m^2)         (m/s)          Subpoint
3149.983,        -86.849,              9.801,        -8.861,       # -105-1m
3159.983,        -86.849,              9.801,        -8.861,       # -105-1b
3159.983,        -86.849,             11.798,         -7.36,       # -106-10f
3171.983,       -189.522,             20.598,          -9.2,       # -106-10m
```

Segment 106 has an area of 30 m^2 (larger than the area of segment 5), so when both trains are alongside subpoint `106-10f`, the annulus there has an area of $30-9.4-8.8 = 11.8$ m^2. Only train 8 is alongside subpoint `106-10b`, so the annulus there has an area of $30 - 9.4 = 20.6$ m^2.

You can get a better understanding of these by looking at the `.pdf` flipbook named `annulus-flow-validation-1-lp2.pdf`. Such spikes are not uncommon, and the timesteps just before them show how each develops. Figure 10 shows the state of things at the previous timestep, when part of both trains is alongside two subpoints (`106-10f` and `106-10m`).

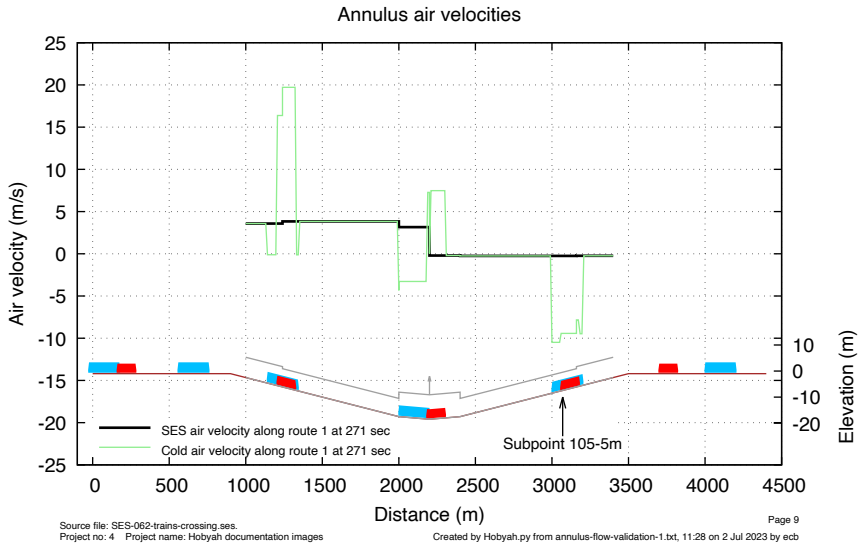

Figure 10: Annulus air velocities at 271 seconds

7 Verification of `SESconv.py`'s air density corrections

The SES fire model has a simple approach to density: it assumes that a change in air density is inversely proportional to the change in absolute dry-bulb temperature (absolute temperature is the Kelvin scale in SI units and the Rankine scale in US customary units).

Density correction factors are stored at every thermal subnode and are applied when air flows from that node into a line segment. The air density corrections in line segments and vent segments are handled very differently. For line segments:

- Density correction factors are stored for every line subsegment and for every thermal subnode.
- When calculating friction, the correction factor in each line subsegment is applied to frictional pressure drop in that subsegment.
- When calculating pressure losses at the back end of line segments, the correction factor in the first subsegment is used if the flow in the segment is negative. If the flow in the segment is positive, the correction factor in the thermal node or subsegment attached at the back end is used instead.
- When calculating pressure losses at the forward end of line segments, the factor in the last subsegment is used if the flow in the segment is positive. If the flow in the segment is negative, the factor in the thermal node or subsegment attached at the forward end is used instead.

In vent segments:

- One correction factor is calculated. It is calculated from the mean temperature in all the vent segment's subsegments.
- Vent segments do not have friction, so calculating a value for each vent subsegment is pointless.
- When calculating pressure rises due to fans, the mean correction factor is used to adjust the fan total pressure rise.
- When calculating pressure losses at the ends of vent segments, the mean correction factor is used regardless of the direction of flow (I think).

The file used to verify the density calculation is `SES-065-density-adjustments-offline-SES.ses`. This file has a line segment with a fire in it connected to a vent segment with an extract fan in it.

The outside air temperature is set in form 1F:

```
                          DESIGN HOUR WEATHER DATA
AMBIENT AIR DRY-BULB TEMPERATURE                78.8   DEG F          FORM 1F
```

7.1 Verification of `SESconv.py`'s line segment density correction

In SES, the array named `TSSTAB` holds the ratio of absolute subsegment air temperature to absolute outside air temperature (one value per subsegment). This is taken to be the ratio of outside air density to air density in the subsegment.

The following extracts are from SES routines and the descriptions of variables in `Record.txt` and show how the values in `TSSTAB` are built up.

1. SES has a factor to convert degrees Fahrenheit to degrees Rankine, `ABTEMP`. Its description (in `Record.txt`) and its definition (in `DSES.FOR`) are given below.

   ```
   From Record.txt:
    ABTEMP  IS A CONSTANT USED TO CONVERT 'USUAL' TEMPERATURES TO
            'ABSOLUTE' TEMPERATURES - THAT IS DEGREES FAHRENHEIT TO
            DEGREES RANKINE.  ( C,R ).

   In DSES.FOR:
   C*****ABSOLUTE TEMPERATURE CONVERSION CONSTANT  DEG F TO DEG R
         ABTEMP=459.67
   ```

   On the Rankine scale absolute zero is -459.67 °F, so 459.67 is what you add to convert °F to °R.

2. The dry-bulb temperature of air outside the tunnel is read from the input file in form 1F, in degrees Fahrenheit. Its description (in `Record.txt`) and its definition are given below.

   ```
   From Record.txt:
    TDBAMB  IS THE AMBIENT ( ATMOSPHERIC ) DRY-BULB AIR TEMPERATURE.
           DEG. F.  ( I/O,R ).

   In INPUT.FOR:
   C---- FORM 1F
   C
         READ (IN,30) TDBAMB,TWBAMB,PAMB,TAMBA,TWAMBM,TAMBM,TWAMBE,ANNAMP
   ```

3. SES has a variable to store the outside air dry-bulb temperature in degrees Rankine, `ABSAMB`. Its description (in `Record.txt`) and its calculation (in `INPUT.FOR`) are given below.

   ```
   From Record.txt:
    ABSAMB  IS THE ABSOLUTE AMBIENT TEMPERATURE.   DEG R.   ( C,R ).

   In INPUT.FOR:
   C*****COMPUTE AMBIENT DRY BULB TEMPERATURE IN DEGREES RANKINE
         ABSAMB = TDBAMB+ABTEMP
   ```

4. SES has a one-dimensional array variable to store the dry-bulb temperature in each subsegment in degrees Fahrenheit, `TDBSS()`. Its description (in `Record.txt`) is given below. It is calculated in multiple routines, depending on whether the subsegment is in an air-conditioned zone (`ACEST2.FOR`), not in an air-conditioned zone (`RKTHRM.FOR`), after an environmental control zone estimate (`DTHS2.FOR`) and after a simulation error (`THDERV.FOR`).

   ```
   From Record.txt:
   TDBSS(ISS) IS THE DRY-BULB AIR TEMPERATURE IN A SUBSEGMENT. DEG F. ( C,R ).
   ```

5. SES has a one-dimensional array variable to store the ratio of dry-bulb temperature in each subsegment in degrees Rankine to the outside air temperature in degrees Rankine, `TSSTAB()`. Its description (in `Record.txt`) is given below. It is calculated in `Pinpnt.for`.

   ```
   From Record.txt:
    TSSTAB(ISS)  IS THE RATIO OF THE ABSOLUTE SUBSEGMENT AIR TEMPERATURE
           TO THE ABSOLUTE AMBIENT TEMPERATURE.   ( C,R ).

   In Pinpnt.for:
     241 DO 248 ISS = ISSL,ISSH
   C --  COMPUTE AND STORE THE RATIO OF THE ABSOLUTE SUBSEGMENT AIR
   C --  TEMPERATURE AND ABSOLUTE AMBIENT TEMPERATURE FOR USE IN THIS
   C --  SUBROUTINE, OMEGA1 AND OMEGA4
         TSSTAB(ISS) = ( TDBSS(ISS) + ABTEMP) / ABSAMB
   ```

   This loops over each subsegment in a segment calculating the ratio of air temperatures for the subsegment. It is inside another loop that iterates over each segment. By the end, the array `TSSTAB` contains a ratio for every line subsegment in the model. Other routines use the inverse of this ratio as the ratio of air densities.

The following are the mandatory printout of data for the line segment 201 at 300 seconds in `SES-064-density-adjustments.PRN`.

```
TIME    300.00 SECONDS                          0 TRAIN(S) ARE OPERATIONAL
            SYSTEM        SENSIBLE     LATENT          AIR     HUMIDITY   AIR        AIR
 LENGTH   PARTITIONING     HEAT LOAD   HEAT LOAD    TEMPERATURE  RATIO     FLOW    VELOCITY
  (FT)                     (BTU/SEC)   (BTU/SEC)      (DEG F)   (LB/LB)   (CFM)     (FPM)
  328.1  101 -201     (TUNNEL)                  30 m^2, 100 m tunnel
         101 -201 -  1      0.2839         0.0         78.80   0.01723  308248.4   954.6
         101 -201 -  2      0.2839         0.0         78.81   0.01723
         101 -201 -  3  28460.4707         0.0        367.90   0.01723
         101 -201 -  4      0.6511         0.0        355.71   0.01723
         101 -201 -  5      0.6326         0.0        344.05   0.01723
```

This shows that there is an airflow of just under 5 m/s and a fire of 30 MW in the third segment. There is a small amount of heat gain in the segments upwind of the fire (friction losses turned into heat), that causes a slight rise in air temperature in those subsegments. In the third, fourth and fifth subsegments the air temperature is high, dropping towards the forward end of the segment due to heat transfer to the tunnel walls.

The file has been told to print additional thermodynamic data (supplementary print option in form 1C set to 4). One of the tables prints the values of `TSSTAB`:

```
 ISCTX-NUMLS-ISS  BUOYS(ISCT)  TSSTAB(ISS)  RELSS(ISS)  TSFSS(ISS)  QWALSS(ISS)  QRADSS(ISS)
  101               0.0000
  101 -201 -  1                 1.000006     1643314.     78.80       0.00000        0.00
  101 -201 -  2                 1.000011     1643309.     78.80       0.00000        0.00
  101 -201 -  3                 1.536889     1205549.    188.13       0.45017     1406.40
  101 -201 -  4                 1.514235     1217919.    132.86       0.22766      147.07
  101 -201 -  5                 1.492582     1230343.    129.92       0.21728      136.78
```

The absolute outside air temperature is $78.8 + 459.67 = 538.47$ °R.

Looking at the hottest subsegment (subsegment 3, 367.90 °F):

- Absolute temperature in the subsegment is $367.9 + 459.67 = 827.57$ °F
- Ratio of absolute temperatures is $827.57/538.47 = 1.536892$

1.536892 is close enough to the value printed (1.536889) to be acceptable. The discrepancy is due to the subsegment temperatures being printed to two decimal places and the internal calc using more decimal places.

We apply the same logic to all five subsegments in segment 201 and compare the calculated figures (from the temperature printouts) to the supplementary printout. They are:

- subsegment 1: 1.0 from hand calculation, 1.000006 printed (OK due to rounding)
- subsegment 2: 1.000019 calculated, 1.000011 printed (OK due to rounding)
- subsegment 3: 1.536892 calculated, 1.536889 printed (OK due to rounding)
- subsegment 4: 1.514253 calculated, 1.514235 printed (not OK)
- subsegment 5: 1.492599 calculated, 1.492852 printed (not OK)

There is something not right with these last two numbers, the ones in subsegments 4 and 5. The temperature printed in the `.PRN` file for subsegment 4 is 355.71 °F (rounded to two decimal places). The true value of the temperature could be anywhere between 355.705 °F and 355.71499 °F. If we take the lowest of these and calculate the ratio, we get 1.514244, which is above the figure of 1.514235 in the supplementary printout. The same thing happens in subsegment 5, where the lowest temperature ratio (344.045 + 459.67/538.47) is 1.492590.

In short, the ratios in the supplementary printout must have been calculated from slightly lower temperatures than the printed temperatures. What's going on here?

SES does its calculations in several steps, and the temperature at an intermediate step may be being used to calculate the temperature ratio. To look into this, I wrote a short Fortran routine (VentTempRatios) to print the subsegment temperatures and the calculated ratios, compiled it into `offline-SES`, and called it from both `PRINT.FOR` and `PINPNT.FOR`.

The following are the transcripts printed by the calls from the two routines.

```
Line section 101 temperature ratio calc (called from PINPNT.FOR):
        Outside air temperature (deg F):    78.800003
Factor to add to convert deg F to deg R:   459.670013
        Outside air temperature (deg R):   538.470032
Seg-Sub   Temperature     Absolute temperature     Ratio
 201-1     78.802994 deg F  538.473022 deg R      1.000006
 201-2     78.805862 deg F  538.475891 deg R      1.000011
 201-3    367.898804 deg F  827.568848 deg R      1.536889
 201-4    355.700165 deg F  815.370178 deg R      1.514235
 201-5    344.040894 deg F  803.710938 deg R      1.492582

Line section 101 temperature ratio calc (called from PRINT.FOR):
```

```
        Outside air temperature (deg F):    78.800003
Factor to add to convert deg F to deg R:   459.670013
        Outside air temperature (deg R):   538.470032
Seg-Sub   Temperature     Absolute temperature     Ratio  PINPNT.FOR ratio
 201-1     78.802994 deg F  538.473022 deg R     1.000006     1.000006
 201-2     78.805862 deg F  538.475891 deg R     1.000011     1.000011
 201-3    367.904358 deg F  827.574341 deg R     1.536900     1.536889
 201-4    355.707001 deg F  815.377014 deg R     1.514248     1.514235
 201-5    344.048859 deg F  803.718872 deg R     1.492597     1.492582
```

This explains the difference: the subsegment temperatures change very slightly between the calculation of the temperature ratio in `PINPNT.FOR` and when the temperatures are printed in `PRINT.FOR`. The difference in calculated temperature ratios are less than 0.001%, so it's safe enough to use the temperatures printed in the output file to calculate temperature ratios in `SESconv.py`.

In `SESconv.py`, a similar calculation is carried out on pandas DataFrames. The DataFrame containing the air temperatures (in °C) is called `subseg_temps`. This has entries for every subsegment at every timestep. The following is the record of a Terminal session starting in the folder that the file "classSES.py" is in that shows how to load the class, and make an instance of it, then print the contents of the DataFrame:

```
% python3
>>> import classSES as clS
>>> log = open("log_discard.txt", "w")
>>> folder = "/Users/tester/Documents/sandboxes/Hobyah/files-SES"
>>> fname = "SES-064-density-adjustments.sbn"
>>> s064 = clS.SESdata(folder, fname, log)
>>> s064.PrettyClassPrint("subseg_temps")
Printing the contents of "subseg_temps" in "SES-064-density-adjustments.sbn".
Rounding to 3 decimal places.
  pandas DataFrame "self.subseg_temps":
       201-1   201-2    201-3    201-4    201-5    801-1    801-2    801-3
0.0    26.000  26.000   26.000   26.000   26.000   26.000   26.000   26.000
20.0   26.000  26.006   26.006   26.006   26.006   26.000   26.000   26.000
40.0   26.006  26.006   26.011   26.011   26.017   26.011   26.011   26.011
60.0   26.006  26.006   26.011   26.011   26.017   26.017   26.017   26.011
80.0   26.006  26.006   26.011   26.011   26.017   26.017   26.017   26.011
100.0  26.006  26.006   26.011   26.011   26.017   26.017   26.017   26.011
120.0  26.000  26.006  161.906  152.000  142.011  129.111  107.361   83.406
140.0  26.000  26.006  178.017  169.861  161.928  156.333  148.811  139.556
160.0  26.000  26.006  182.644  175.150  167.961  163.544  158.689  153.561
180.0  26.000  26.006  184.194  176.950  170.022  165.900  161.756  157.650
200.0  26.000  26.006  184.911  177.794  171.000  166.944  162.978  159.139
220.0  26.000  26.006  185.372  178.344  171.633  167.589  163.661  159.889
240.0  26.000  26.006  185.739  178.783  172.144  168.083  164.161  160.400
260.0  26.000  26.006  186.056  179.167  172.583  168.517  164.583  160.817
280.0  26.000  26.006  186.344  179.517  172.989  168.906  164.961  161.189
300.0  26.000  26.006  186.611  179.839  173.361  169.261  165.311  161.522
```

The code in `SESconv.py` takes a copy of `subseg_temps` and stores the copy in a new DataFrame, `subseg_denscorr`.

273.15 K is added to every value in this new DataFrame. It calculates the absolute

outside air temperature (in this case, 299.15 K), then divides every value in `subseg_denscorr` by that.

The Python code is compact (in most calculations involving DataFrames there is no need to iterate):

```
subseg_denscorr = subseg_temps.copy()
# Convert all entries in the dataframe from degrees Celsius to Kelvin.
subseg_denscorr += 273.15
outside_temp = 273.15 + settings_dict["ext_DB"]
# Divide all entries in the dataframe by the outside temperature
# in Kelvin.
subseg_denscorr /= outside_temp
```

The following is the result of printing the contents of the DataFrame `subseg_denscorr`:

```
>>> s064.PrettyClassPrint("subseg_denscorr")
Printing the contents of "subseg_denscorr" in "SES-064-density-adjustments.sbn".
Rounding to 6 decimal places.
  pandas DataFrame "self.subseg_denscorr":
        201-1     201-2     201-3  ...     801-1     801-2     801-3
0.0    1.000000  1.000000  1.000000  ...  1.000000  1.000000  1.000000
20.0   1.000000  1.000019  1.000019  ...  1.000000  1.000000  1.000000
40.0   1.000019  1.000019  1.000037  ...  1.000037  1.000037  1.000037
60.0   1.000019  1.000019  1.000037  ...  1.000056  1.000056  1.000037
80.0   1.000019  1.000019  1.000037  ...  1.000056  1.000056  1.000037
100.0  1.000019  1.000019  1.000037  ...  1.000056  1.000056  1.000037
120.0  1.000000  1.000019  1.454306  ...  1.344680  1.271974  1.191896
140.0  1.000000  1.000019  1.508162  ...  1.435679  1.410534  1.379594
160.0  1.000000  1.000019  1.523632  ...  1.459784  1.443553  1.426412
180.0  1.000000  1.000019  1.528813  ...  1.467658  1.453804  1.440080
200.0  1.000000  1.000019  1.531209  ...  1.471150  1.457890  1.445057
220.0  1.000000  1.000019  1.532750  ...  1.473304  1.460174  1.447564
240.0  1.000000  1.000019  1.533976  ...  1.474957  1.461846  1.449273
260.0  1.000000  1.000019  1.535034  ...  1.476405  1.463257  1.450666
280.0  1.000000  1.000019  1.536000  ...  1.477705  1.464520  1.451910
300.0  1.000000  1.000019  1.536892  ...  1.478894  1.465690  1.453024
```

The pandas print routine cuts out entries to enable the array to fit into the window it is printing in. It is often useful to be able to pick out a particular slice of a pandas database. The following command accesses the variable `subseg_denscorr` directly and prints only the slice of the pandas arrays that needs to be shown here (values in the subsegments of segment 201):

```
>>> s064.subseg_denscorr.loc[:, s064.subseg_denscorr.columns.str.startswith("201")]
        201-1     201-2     201-3     201-4     201-5
0.0    1.000000  1.000000  1.000000  1.000000  1.000000
20.0   1.000000  1.000019  1.000019  1.000019  1.000019
40.0   1.000019  1.000019  1.000037  1.000037  1.000056
60.0   1.000019  1.000019  1.000037  1.000037  1.000056
80.0   1.000019  1.000019  1.000037  1.000037  1.000056
```

```
100.0  1.000019  1.000019  1.000037  1.000037  1.000056
120.0  1.000000  1.000019  1.454306  1.421193  1.387802
140.0  1.000000  1.000019  1.508162  1.480900  1.454380
160.0  1.000000  1.000019  1.523632  1.498579  1.474548
180.0  1.000000  1.000019  1.528813  1.504596  1.481438
200.0  1.000000  1.000019  1.531209  1.507419  1.484707
220.0  1.000000  1.000019  1.532750  1.509258  1.486824
240.0  1.000000  1.000019  1.533976  1.510725  1.488532
260.0  1.000000  1.000019  1.535034  1.512006  1.489999
280.0  1.000000  1.000019  1.536000  1.513176  1.491355
300.0  1.000000  1.000019  1.536892  1.514253  1.492599
```

The values for the subsegments in segment 201 at 300 seconds exactly match the manual calculations above, so the calculation can be considered verified.

7.2 Verification of `SESconv.py`'s vent segment density correction

Section 7.1 showed how SES corrects for density in individual line subsegments and how that calculation is replicated in `SESconv.py`. This section shows how SES corrects for density in vent segments (it is very useful to be able to show the change in fan performance as fans start passing hot smoke).

Fans in SES are placed in vent segments, but are not in a particular location. `PINPNT.FOR` calculates the mean absolute temperature of all the subsegments in the vent segment, divides it by the absolute outside air temperature, then stores it in the index in the `TSSTAB` array that is for the first subsegment in the vent segment.

To get the mean temperature, it creates a variable `TSUM` to hold the sum of the subsegment temperatures and runs a loop that starts at the first subsegment of the vent segment (index `ISSL` in the arrays) and ends at the last subsegment (index `ISSH` in the arrays). Note that some calculations have been removed from this transcript as they are not relevant to the verification process. Here is the edited code in `PINPNT.FOR`:

```
C---- LOOP OVER EACH SUBSEGMENT IN VENT SHAFT
  160 ISSL = TABL5( IVS )
      ISSH = TABL6( IVS )
      TSUM = 0.0
      DO 163 ISS=ISSL,ISSH

C**** CHECK FIRE SIMULATION OPTION -- IF A FIRE IS BEING SIMULATED,
C**** COMPUTE AND STORE THE AVERAGE SUBSEGMENT TEMPERATURE (STORE AS THE
C**** RATIO OF AVERAGE TEMPERATURE TO AMBIENT TEMPERATURE)
C
      IF (FIROP2 .LE. 0)  GO TO 163
C**** FIRE SIMULATION
      TSUM = TSUM+TDBSS(ISS)
163   CONTINUE
```

After the loop finishes, `TSUM` contains the sum of the temperatures in the subsegments. The next step is to calculate the mean (divide by the count of subsegments), convert the mean to absolute temperature and divide by the outside air temperature:

```
C
C**** STORE THE AVERAGE TEMPERATURE RATIO IN THE ARRAY LOCATION
C**** CORRESPONDING TO THE LOWEST SUBSEGMENT WITHIN THE VENT SECTION
C**** FOR USE IN SUBROUTINE OMEGA1
C
      IF( FIROP2 .GT. 0 )
     1TSSTAB(ISSL) = (TSUM/FLOAT(ISSH-ISSL+1)+ABTEMP)/ABSAMB
C
```

A few notes about parsing the above code (old-style Fortran 66 code):

- In Fortran 66, numbered labels are in the first five characters on the line, if the 6th character is not blank then the line is a continuation of the previous line and the code is in the 7th to 72nd characters on the line.
- `FIROP2` is negative in non-fire runs and above zero in fire runs, which is why it appears in the `IF` statements: the calculations are only carried out in fire runs.

- Fortran 66 has a form of `IF` statement that doesn't need an `END IF`: this form is `IF <condition> <action>`. Both `IF` statements above are of this type: one jumps to the line labelled 163, the other executes the calculation on the continuation line.

The mean value of temperature in vent segments is never printed to the `.PRN` file, so there is no way in SES v4.1 to verify the calculation.

However, code has been added to `offline-SES` to print it next to the buoyancy term in the supplementary printout. The following shows the differences between the format fields, print statements and printouts in SES v4.1 and the modified equivalents in `offline-SES`.

The SES v4.1 code in `PRINT.FOR` is below, followed by the printout.

```
884   FORMAT(//T8, 'ISCTX-NUMLS-ISS',T34,'BUOYS(ISCT)',
     1T51,'TSSTAB(ISS)',T68,'RELSS(ISS)',T84,'TSFSS(ISS)',T100,
     2'QWALSS(ISS)',T117,'QRADSS(ISS)',/)
      WRITE (OUT,884)
885   FORMAT(T8, I3, T32, F12.4 )
C-----WRITE OUT BUOYANCY IN SECTIONS
      WRITE (OUT,885)  ISCTX, BUOYS(ISCT)
```

```
  ISCTX-NUMLS-ISS  BUOYS(ISCT)  TSSTAB(ISS)  RELSS(ISS)  TSFSS(ISS)  QWALSS(ISS)  QRADSS(ISS)
   901            2147.0000
```

In offline-SES, the code followed by the printout. The `offline-SES` code in `PRINT.FOR` is below, followed by its printout.

```
884   FORMAT(//T8, 'ISCTX-NUMLS-ISS',T34,'BUOYS(ISCT)',
     1T51,'TSSTAB(ISS)',T68,'RELSS(ISS)',T84,'TSFSS(ISS)',T100,
     2'QWALSS(ISS)',T117,'QRADSS(ISS)',/)
      WRITE (OUT,884)
881     format(T8, I3, T32, F12.4, 1X, F16.6,
     &         ' (mean of all vent subsegments)')
        ilssl = tabl5(-isegl)
        write (out,881) isctx, buoys(isct), tsstab(ilssl)
```

```
  ISCTX-NUMLS-ISS  BUOYS(ISCT)  TSSTAB(ISS)  RELSS(ISS)  TSFSS(ISS)  QWALSS(ISS)  QRADSS(ISS)
       901        2147.0000      1.465857 (mean of all vent subsegments)
```

This table is actually a mix of the properties of line segments and vent segments, so the full printout from `offline-SES` has data for line segment 201 and vent section 901:

```
  ISCTX-NUMLS-ISS  BUOYS(ISCT)  TSSTAB(ISS)  RELSS(ISS)  TSFSS(ISS)  QWALSS(ISS)  QRADSS(ISS)
   101               0.0000
   101 -201 -  1                1.000006     1643314.     78.80       0.00000        0.00
   101 -201 -  2                1.000011     1643309.     78.80       0.00000        0.00
   101 -201 -  3                1.536889     1205549.    188.13       0.45017     1406.40
   101 -201 -  4                1.514235     1217919.    132.86       0.22766      147.07
   101 -201 -  5                1.492582     1230343.    129.92       0.21728      136.78
       901        2147.0000      1.465857 (mean of all vent subsegments)
```

Let's take a look at the temperatures printed out for the vent segment at 300 seconds:

```
            SYSTEM        SENSIBLE     LATENT          AIR     HUMIDITY   AIR        AIR
 LENGTH   PARTITIONING     HEAT LOAD   HEAT LOAD    TEMPERATURE  RATIO     FLOW    VELOCITY
  (FT)                     (BTU/SEC)   (BTU/SEC)      (DEG F)   (LB/LB)   (CFM)     (FPM)
  328.1  101 -201      (TUNNEL)                 30 m^2, 100 m tunnel
         101 -201 -  1      0.2839      0.0000         78.80   0.01723  308248.4   954.6
         101 -201 -  2      0.2839      0.0000         78.81   0.01723
         101 -201 -  3  28460.4707      0.0000        367.90   0.01723
         101 -201 -  4      0.6511      0.0000        355.71   0.01723
         101 -201 -  5      0.6326      0.0000        344.05   0.01723
  246.1  901 -801      (VENTILATION SHAFT)       75 m vertical ventilation shaft
         901 -801 -  1                                336.67   0.01723  308248.4   923.8
         901 -801 -  2                                329.56   0.01723
         901 -801 -  3                                322.74   0.01723
```

Section 901 (vent segment 801) has three subsegments, and the mean of the three subsegment temperatures is $(336.67 + 329.56 + 322.74)/3 = 329.556\dot{6}$ °F.

$329.656\dot{6}$ °F is $789.326\dot{6}$ °R. Divide by the absolute outside air temperature (calculated in Section 7.1 as 538.47 °R) to get the ratio, 1.465869. As you might expect from reading Section 7.1, this is not quite the same as the value printed in the output file, because the air temperatures used by `PINPNT.FOR` during the run are not the same as the air temperatures printed by `PRINT.FOR`.

The Fortran routine VentTempRatios has this calculation and prints the values calculated from the calls in `PINPNT.FOR` and `PRINT.FOR`:

```
Vent section 901 temperature ratio calc (called from PINPNT.FOR):
Seg-Sub     Temperature       Cumulative sum
 801-1    336.662689 deg F    336.662689 deg F
 801-2    329.550262 deg F    666.212952 deg F
 801-3    322.737030 deg F    988.949951 deg F

Mean of subseg air temperatures (deg F):   329.649994
        Outside air temperature (deg F):    78.800003
Factor to add to convert deg F to deg R:   459.670013
Mean of subseg air temperatures (deg R):   789.320007
        Outside air temperature (deg R):   538.470032

          Mean of subseg air temperatures (deg R)
TSSTAB = ---------------------------------------
              Outside air temperature (deg R)

          789.320007
        = ---------- = 1.465857
          538.470032

Vent section 901 temperature ratio calc (called from PRINT.FOR):
Seg-Sub     Temperature       Cumulative sum
 801-1    336.670441 deg F    336.670441 deg F
 801-2    329.557800 deg F    666.228271 deg F
 801-3    322.744324 deg F    988.972595 deg F
```

```
Mean of subseg air temperatures (deg F):   329.657532
        Outside air temperature (deg F):    78.800003
Factor to add to convert deg F to deg R:   459.670013
Mean of subseg air temperatures (deg R):   789.327515
        Outside air temperature (deg R):   538.470032

         Mean of subseg air temperatures (deg R)
TSSTAB = ---------------------------------------
             Outside air temperature (deg R)

         789.327515
       = ---------- = 1.465871
         538.470032
Value in TSSTAB(ISSL): 1.465857
```

The calculated values differ slightly (1.465869 in the hand calc vs. 1.465871 in the call from `PRINT.FOR`) due to rounding the temperatures to two decimal places in the hand calc vs. single precision accuracy in the machine calculation. The value calculated in `PINPNT.FOR` differs because the subsegment temperatures in the interim calculation are slightly different (same as in Section 7.1).

In `SESconv.py` a pandas DataFrame (`seg_meandenscorr`) is created. The columns are integers (the vent segment numbers). The indices are the print times given as floats. The values are determined from the contents of `subseg_denscorr`, by taking a slice of the values in for the subsegments of each vent segment, calculating the mean at every timestep and storing it in `seg_meandenscorr`.

The code below is from `SESconv.py`. This is heavily commented because it uses pandas syntax that I was using for the first time.

```
    # Make a tuple of tuples of zeros for the vent segment mean density
    # corrections then turn it into a pandas dataframe that we can
    # set values in below.
    all_zeros = (  (0.0,)*len(vent_segs),) * len(print_times)
    seg_meandenscorr = pd.DataFrame( all_zeros, columns = vent_segs,
                                     index = print_times)
    # Now we populate the vent segment mean density correction array.
    for seg_num in vent_segs:
        # Take the subsegments of each vent segment in turn and
        # calculate the mean of the density corrections.
        # First we get the segment number as a string so we can
        # select all the columns keys that start with that number
        # seg_num 201 (integer) becomes "201" so we can take
        # the keys "201-1", "201-2", "201-3", "201-4".
        seg_str = str(seg_num)
        # Pandas syntax is complex, this is a crib for future me.
        # The ".loc[:," means "take all the times."
        # "subseg_denscorr.columns.str.startswith(seg_str)" means
        # "take all the columns whose identifiers starts with the
        # segment number".
        # The ".mean("axis=1")" entry means "calculate a mean value
        # from the subsegments at each time".  Using "axis=0" in the
        # mean() function would give a mean in each subsegment over
```

```
        # all time, which we don't want.
        seg_meandenscorr[seg_num] = subseg_denscorr.loc[:,
            subseg_denscorr.columns.str.startswith(seg_str)].mean(axis=1)
```

There is only one vent segment in the test file. It has the following density ratios in each subsegment at each timestep:

```
>>> s064.subseg_denscorr.loc[:, s064.subseg_denscorr.columns.str.startswith("801")]
          801-1     801-2     801-3
0.0    1.000000  1.000000  1.000000
20.0   1.000000  1.000000  1.000000
40.0   1.000037  1.000037  1.000037
60.0   1.000056  1.000056  1.000037
80.0   1.000056  1.000056  1.000037
100.0  1.000056  1.000056  1.000037
120.0  1.344680  1.271974  1.191896
140.0  1.435679  1.410534  1.379594
160.0  1.459784  1.443553  1.426412
180.0  1.467658  1.453804  1.440080
200.0  1.471150  1.457890  1.445057
220.0  1.473304  1.460174  1.447564
240.0  1.474957  1.461846  1.449273
260.0  1.476405  1.463257  1.450666
280.0  1.477705  1.464520  1.451910
300.0  1.478894  1.465690  1.453024
```

And it has the following mean density corrections in each vent segment:

```
>>> s064.PrettyClassPrint("seg_meandenscorr")
Printing the contents of "seg_meandenscorr" in "SES-064-density-adjustments.sbn".
Rounding to 6 decimal places.
  pandas DataFrame "self.seg_meandenscorr":
             801
0.0    1.000000
20.0   1.000000
40.0   1.000037
60.0   1.000050
80.0   1.000050
100.0  1.000050
120.0  1.269517
140.0  1.408602
160.0  1.443250
180.0  1.453848
200.0  1.458032
220.0  1.460348
240.0  1.462025
260.0  1.463443
280.0  1.464712
300.0  1.465869
```

The density correction at 300 seconds (1.465869) exactly matches the hand-calculation at 300 seconds using the temperatures in Fahrenheit, so the calculation can be considered verified.

8 Verification of the quadratics spreadsheet

8.1 Introduction

Take a tunnel with constant area, incompressible flow, pressure differences, jet fans, moving traffic, friction and fixed pressure losses. The air velocity can be calculated from an expression that can be reduced to one of three quadratic equations for air velocity.

The spreadsheet `slug-flow-quadratics.ods` has been written to take the inputs to that equation, solve the three quadratic equations and choose which to show as the result.

The spreadsheet is able to handle three types of traffic (all moving at the same speed).

8.2 The original equation

Start from a simple calculation of steady-state, incompressible airflow in a tunnel with constant area, loss factors, traffic drag and jet fans. The general expression to calculate the air velocity v_t is

$$\begin{aligned} P_2 - P_1 \quad &+ \quad \frac{1}{2}\rho\left(\zeta_{in} + \zeta_{out} + \frac{\lambda L}{D_h}\right) v_t|v_t| \\ &- \quad \frac{1}{2}\rho\frac{\Sigma(N_v c_d A_v)}{A_t}\left(v_v - v_t\right)|v_v - v_t| - \frac{N_f T_f \eta_f}{|v_f| A_t}(v_f - v_t) = 0 \end{aligned} \qquad (1)$$

where

$$\begin{aligned} P_1 &= \text{Pressure at the traffic entry portal } (Pa) \\ P_2 &= \text{Pressure at the traffic exit portal } (Pa) \\ v_t &= \text{Tunnel air velocity } (m/s) \\ \rho &= \text{Standard air density } (1.2\ kg/m^3) \\ \zeta_{in} &= \text{Entry portal loss } (-) \\ \zeta_{out} &= \text{Exit portal loss } (-) \\ L &= \text{Tunnel length } (m) \\ A_t &= \text{Tunnel area } (m^2) \\ D_h &= \text{Tunnel hydraulic diameter } (m) \\ \lambda &= \text{Darcy friction factor } (-), \\ N_v &= \text{Number of vehicles in the tunnel} \\ c_d &= \text{Vehicle drag coefficient } (-) \\ A_v &= \text{Vehicle cross-sectional area } (m^2) \\ v_v &= \text{Vehicle speed (which is always positive) } (m/s) \\ N_f &= \text{Number of jet fans in the tunnel} \\ T_f &= \text{Static thrust of the jet fans } (N) \\ \eta_f &= \text{Installation efficiency of the jet fans } (-) \\ v_f &= \text{Discharge velocity of the jet fans } (m/s) \end{aligned}$$

This equation appears all over the technical literature in one form or another. A free source is the 1995 PIARC technical report on road tunnels [1].

All the terms have known values except air velocity v_t. The terms in (1) can be broken into the following groups:

$$P_2 - P_1 \qquad \text{External pressure difference} \tag{2}$$

$$\frac{1}{2}\rho\left(\zeta_{in} + \zeta_{out} + \frac{\lambda L}{D_h}\right) v_t|v_t| \qquad \text{Friction and fixed losses} \tag{3}$$

$$\frac{1}{2}\rho\frac{\Sigma(N_v c_d A_v)}{A_t}\left(v_v - v_t\right)|v_v - v_t| \qquad \text{Traffic drag} \tag{4}$$

$$\frac{N_f T_f \eta_f}{|v_f| A_t}(v_f - v_t) \qquad \text{Jet fan thrust} \tag{5}$$

The presence of $|v_f|$ in the jet fan term is to account for jet fans running in reverse (negative jet velocity).

The presence of the absolute value terms $|v_v - v_t|$ and $|v_t|$ means that when you turn (1) into quadratic equations with v_t as the subject you get three quadratics, with different signs in front of the friction and traffic terms:

$$P_2 - P_1 + \frac{1}{2}\rho\left(\zeta_{in} + \zeta_{out} + \frac{\lambda L}{D_h}\right) v_t^2 + \frac{1}{2}\rho\frac{\Sigma(N_v c_d A_v)}{A_t}\left(v_v - v_t\right)^2 - \frac{N_f T_f \eta_f}{|v_f| A_t}(v_f - v_t) = 0 \tag{6}$$

$$P_2 - P_1 + \frac{1}{2}\rho\left(\zeta_{in} + \zeta_{out} + \frac{\lambda L}{D_h}\right) v_t^2 - \frac{1}{2}\rho\frac{\Sigma(N_v c_d A_v)}{A_t}\left(v_v - v_t\right)^2 - \frac{N_f T_f \eta_f}{|v_f| A_t}(v_f - v_t) = 0 \tag{7}$$

$$P_2 - P_1 - \frac{1}{2}\rho\left(\zeta_{in} + \zeta_{out} + \frac{\lambda L}{D_h}\right) v_t^2 - \frac{1}{2}\rho\frac{\Sigma(N_v c_d A_v)}{A_t}\left(v_v - v_t\right)^2 - \frac{N_f T_f \eta_f}{|v_f| A_t}(v_f - v_t) = 0 \tag{8}$$

Each is valid over a different range of tunnel air velocity v_t. All are valid only for positive values of traffic velocity v_v.

(6) is used when traffic and air are moving in the same direction and the traffic is slower than the air.

(7) is used when traffic and air are moving in the same direction and the traffic is faster than the air.

(8) is used when air is flowing in the opposite direction to the traffic. This could be caused by high adverse pressure difference, lots of jet fans in reverse, or both.

8.3 Verification activities

The `slug-flow-quadratics.ods` spreadsheet on github takes the inputs, solves the three quadratic equations and selects the correct one to display to the user. A subsidiary spreadsheet, `quadratics.xlsx`, is a copy of it that Excel may be happier with[3].

The approach in this section is to take several different sets of inputs that cause the spreadsheet to use each of its three equations, plug the values into the general equation (1) and show that it gives the same answer.

[3]Microsoft seem to hate and fear the `.ods` (Open Document Spreadsheet) format and have features in Excel that seem to have been intentionally designed to misinterpret `.ods` files.

Like all the programs in the suite, the spreadsheet gives you the opportunity to use Darcy friction factor or Fanning friction factor. The use of both to calculate the same result is demonstrated.

Traffic drag factors C_d may be from sources that give drag factors in the open air or inside the tunnel (it is often difficult to tell which is being given in sources like TRRL reports though). The spreadsheet has a switch to activate or de-activate the blockage correction. Its operation shown as well.

A few edge cases are also given. These cover the region at which the spreadsheet switches from showing the result of (6) to showing the result of (7) and the region at which it switches from (7) to (8). This is to demonstrate that there are no step changes in the calculated air velocity as the spreadsheet switches from one quadratic to the other.

A number of cases are set up in which two opposing forces exactly balance:

- Jet fan thrust balanced against wind pressure
- Jet fan thrust balanced against traffic drag
- Traffic drag balanced against wind pressure

In all three cases, the calculated air velocity should be zero (and is).

The spreadsheet has three types of traffic in it. Cases are given to demonstrate that they all act the same:

- Cases with only one type of traffic
- Cases with three types of traffic in different permutations

8.4 Sample printout

Figure 11 (a printout of the spreadsheet on a page of its own) shows the general layout of the spreadsheet. Cells that can be set by the user are in blue text. At the top is a box for comments.

There are four groups of input, for the tunnel, wind, jet fans and traffic. The result is given in the box near the bottom right. Below it is a check calculation that gives the results of the four terms in equation (1) and a check of whether or not they sum up to zero.

8.5 Verifying the three calculations

Let's take Figure 11 and plug the relevant values on it (the input values and v_t) into the general equation (1).

First we do some arithmetic on the three traffic flows to change them from vehicle flowrate and speed into the count of each type of vehicle in the tunnel.

- Vehicle speed = 10 km/h $\equiv 10 \div 3.6 = 2.77\dot{7}$ m/s
- 500 cars/hr $\div$ 10 km/h = 50 cars/km

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_1

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example that uses Darcy friction factor, positive airflow and traffic slower than air.

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Wind input

P at left portal	0	Pa
P at right portal	20	Pa

Jet fan input

Count of fans	10	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Traffic input

Traffic speed (+ve)	10	km/h
Car flowrate	500	cars/hr
LCV flowrate	50	LCVs/hr
HGV flowrate	100	HGVs/hr
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.8	—
HGV area	6	m²
HGV drag factor	0.9	—

Use blockage correction in calc? N (Y/N)

Calculation (vehicles slower than air)

Air velocity	3.152227 m/s
Volume flow	157.611 m³/s

Independent check calc of pressures

Wind pressure	20.00 Pa
Friction pressure	77.62 Pa
Traffic pressure	0.37 Pa
Jet fan pressure	-97.99 Pa
Is the sum zero?	TRUE

Figure 11: Quadratics spreadsheet: +ve airflow and air faster than traffic

- 50 LCVs/hr ÷ 10 km/h = 5 LCVs//km
- 100 HGVs/hr ÷ 10 km/h = 10 HGVs/km
- 50 cars/km × 2 km long tunnel = 100 cars in the tunnel
- 5 LCVs/km × 2 km long tunnel = 10 LCVs in the tunnel
- 100 HGVs/km × 2 km long tunnel = 20 HGVs in the tunnel

The four parts of the general equation (1) are:

$P_2 - P_1$	Wind pressure
$+\frac{1}{2}\rho\left(\zeta_{in} + \zeta_{out} + \frac{\lambda L}{D_h}\right) v_t \lvert v_t \rvert$	Friction & losses
$-\frac{1}{2}\rho\frac{\Sigma(N_v c_d A_v)}{A_t}\left(v_v - v_t\right)\lvert v_v - v_t \rvert$	Traffic drag
$-\frac{N_f T_f \eta_f}{\lvert 30 \rvert \times A_t}\left(30 - v_t\right)$	Jet fans

Plug the numbers in Figure 11 (positive airflow, traffic slower than air) into (1).

$$
\begin{aligned}
= \quad & 20 - 0 \\
+ \quad & \frac{1}{2} \times 1.2 \times \left(0.5 + 1.0 + \frac{0.036 \times 2000}{6.25}\right) \times 3.152227 \times \lvert 3.152227 \rvert \\
- \quad & \frac{1}{2} \times 1.2 \times \frac{100 \times 2 \times 0.4 + 10 \times 4 \times 0.8 + 20 \times 6 \times 0.9}{50} \left(2.77\dot{7} - 3.152227\right) \lvert 2.77\dot{7} - 3.152227 \rvert \\
- \quad & \frac{10 \times 730 \times 0.75}{30 \times 50} \left(30 - 3.152227\right) \\
= \quad & 20 \\
+ \quad & 0.6 \times 13.02 \times 9.936535 \\
- \quad & 0.6 \times 4.4 \times (-0.374449) \times \lvert -0.374449 \rvert \\
- \quad & 3.65 \times 26.84773 \\
= \quad & 20 \\
+ \quad & 77.624212 \\
+ \quad & 0.37016 \\
- \quad & 97.994397 \\
= \quad & 0
\end{aligned}
$$

This sums up to zero and matches the numbers in the check calc box in Figure 11 (`20.00`, `77.62`, `0.37` and `-97.99`). So the calculation for positive airflow in slow traffic is verified.

Next plug in the numbers in Figure 12 (positive airflow, traffic faster than air). The only difference between Figure 12 and Figure 11 is that Figure 12 has fewer jet fans (2 instead of 10).

$$
\begin{aligned}
= \quad & 20 - 0 \\
+ \quad & \frac{1}{2} \times 1.2 \times \left(0.5 + 1.0 + \frac{0.036 \times 2000}{6.25}\right) \times 1.065254 \times \lvert 1.065254 \rvert \\
- \quad & \frac{1}{2} \times 1.2 \times \frac{100 \times 2 \times 0.4 + 10 \times 4 \times 0.8 + 20 \times 6 \times 0.9}{50} \left(2.77\dot{7} - 1.065254\right) \lvert 2.77\dot{7} - 1.065254 \rvert \\
- \quad & \frac{2 \times 730 \times 0.75}{\lvert 30 \rvert \times 50} \left(30 - 1.065254\right) \\
= \quad & 20 \\
+ \quad & 0.6 \times 13.02 \times 1.134766
\end{aligned}
$$

$$
\begin{aligned}
&- \quad 0.6 \times 4.4 \times 1.712524 \times |1.712524| \\
&- \quad 0.73 \times 28.934746 \\
= \quad & 20 \\
&+ \quad 8.864792 \\
&- \quad 7.7412427 \\
&- \quad 21.122365 \\
= \quad & 0
\end{aligned}
$$

This sums up to zero and matches the numbers in the check calc box in Figure 12, so the calculation for positive airflow in fast traffic is verified.

Finally, plug in the numbers in Figure 13 (negative airflow). The only difference between Figure 13 and Figure 12 is that Figure 13 has a higher adverse pressure at the portal the traffic exits from.

$$
\begin{aligned}
= \quad & 100 - 0 \\
&+ \quad \frac{1}{2} \times 1.2 \times \left(0.5 + 1.0 + \frac{0.036 \times 2000}{6.25} \right) \times (-1.726343) \times | - 1.726343| \\
&- \quad \frac{1}{2} \times 1.2 \times \frac{100 \times 2 \times 0.4 + 10 \times 4 \times 0.8 + 20 \times 6 \times 0.9}{50} \left(2.77\dot{7} - (-1.726343)\right) |2.77\dot{7} - (-1.726343)| \\
&- \quad \frac{2 \times 730 \times 0.75}{|30| \times 50} \left(30 - (-1.726343)\right) \\
= \quad & 100 \\
&- \quad 0.6 \times 13.02 \times 2.98026 \\
&- \quad 0.6 \times 4.4 \times 4.504121 \times |4.504121| \\
&- \quad 0.73 \times 31.726343 \\
= \quad & 100 \\
&- \quad 23.281792 \\
&- \quad 53.557953 \\
&- \quad 23.16023 \\
= \quad & 0
\end{aligned}
$$

This sums up to zero and matches the numbers in the check calc box in Figure 13, so the calculation for negative airflow is verified.

8.6 Verifying friction

Figure 11 showed a calculation that uses Darcy friction factor λ.

Figure 14 shows the same calculation but with Fanning friction factor c_f, which is one-quarter of λ.

In Figure 11 the input is friction factor $\lambda = 0.036$ and in Figure 14 the input is friction factor $c_f = 0.009$. Both give the same answer, both in the main calculation and the independent check calculation.

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_2

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example that uses Darcy friction factor, positive airflow and traffic faster than air.

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Traffic input

Traffic speed (+ve)	10	km/h
Car flowrate	500	cars/hr
LCV flowrate	50	LCVs/hr
HGV flowrate	100	HGVs/hr
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.8	—
HGV area	6	m²
HGV drag factor	0.9	—

Wind input

P at left portal	0	Pa
P at right portal	20	Pa

Use blockage correction in calc? N (Y/N)

Jet fan input

Count of fans	2	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Calculation (vehicles faster than air)

Air velocity	1.065254	m/s
Volume flow	53.263	m³/s

Independent check calc of pressures

Wind pressure	20.00	Pa
Friction pressure	8.86	Pa
Traffic pressure	-7.74	Pa
Jet fan pressure	-21.12	Pa
Is the sum zero?	TRUE	

Figure 12: Quadratics spreadsheet: −ve airflow

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_3

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example that uses Darcy friction factor and negative airflow.

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Traffic input

Traffic speed (+ve)	10	km/h
Car flowrate	500	cars/hr
LCV flowrate	50	LCVs/hr
HGV flowrate	100	HGVs/hr
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.8	—
HGV area	6	m²
HGV drag factor	0.9	—

Wind input

P at left portal	0	Pa
P at right portal	100	Pa

Use blockage correction in calc? N (Y/N)

Jet fan input

Count of fans	2	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Calculation (reverse airflow)

Air velocity	-1.726343	m/s
Volume flow	-86.317	m³/s

Independent check calc of pressures

Wind pressure	100.00 Pa
Friction pressure	-23.28 Pa
Traffic pressure	-53.56 Pa
Jet fan pressure	-23.16 Pa
Is the sum zero?	TRUE

Figure 13: Quadratics spreadsheet: −ve airflow

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_4

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example that uses Fanning friction factor, positive airflow and traffic slower than air.

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Fanning	(Darcy/Fanning)
Fanning fricfac c_f	0.009	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Traffic input

Traffic speed (+ve)	10	km/h
Car flowrate	500	cars/hr
LCV flowrate	50	LCVs/hr
HGV flowrate	100	HGVs/hr
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.8	—
HGV area	6	m²
HGV drag factor	0.9	—

Wind input

P at left portal	0	Pa
P at right portal	20	Pa

Use blockage correction in calc? N (Y/N)

Jet fan input

Count of fans	10	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Calculation (vehicles slower than air)

Air velocity	3.152227 m/s
Volume flow	157.611 m³/s

Independent check calc of pressures

Wind pressure	20.00 Pa
Friction pressure	77.62 Pa
Traffic pressure	0.37 Pa
Jet fan pressure	-97.99 Pa
Is the sum zero?	TRUE

Figure 14: Quadratics spreadsheet: Giving input in Fanning friction factor

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_5

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example with the traffic blockage correction term turned off.

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Wind input

P at left portal	20	Pa
P at right portal	0	Pa

Jet fan input

Count of fans	4	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Traffic input

Traffic speed (+ve)	50	km/h
Car flowrate	1800	cars/hr
LCV flowrate	200	LCVs/hr
HGV flowrate	500	HGVs/hr
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.8	—
HGV area	6	m²
HGV drag factor	0.9	—

Use blockage correction in calc? N (Y/N)

Calculation (vehicles faster than air)

Air velocity	5.347430	m/s
Volume flow	267.371	m³/s

Independent check calc of pressures

Wind pressure	-20.00 Pa
Friction pressure	223.38 Pa
Traffic pressure	-167.39 Pa
Jet fan pressure	-35.99 Pa
Is the sum zero?	TRUE

Figure 15: Quadratics spreadsheet: blockage correction off

8.7 Verifying the blockage correction calculation

The traffic drag calculation includes an optional correction factor based on the blockage correction used in wind tunnel experiments. Some tunnel ventilation programs always exclude it (e.g. Camatt [2]) and some always include it (e.g. IDA [3]). So it makes sense to write programs & spreadsheets that can include it or exclude it.

Traffic drag without the blockage correction term is

$$\frac{1}{2}\rho\frac{N_v c_d A_v}{A_t}\left(v_v - v_t\right)\left|v_v - v_t\right|. \tag{9}$$

Traffic drag with the blockage correction term is

$$\frac{1}{2}\rho\frac{N_v c_d A_v}{A_t\left(1-\frac{A_v}{A_t}\right)^2}\left(v_v - v_t\right)\left|v_v - v_t\right|. \tag{10}$$

Figure 15 shows a calculation with traffic and the blockage correction off. For the three types of traffic used for the spreadsheet, the correction factors are:

$$\begin{aligned}
\text{Cars:} \quad & \text{Divide by } \left(1-\frac{2}{50}\right)^2 \equiv \text{divide by } 0.9216 \\
\text{LCVs:} \quad & \text{Divide by } \left(1-\frac{4}{50}\right)^2 \equiv \text{divide by } 0.8464 \\
\text{HGVs:} \quad & \text{Divide by } \left(1-\frac{6}{50}\right)^2 \equiv \text{divide by } 0.7744
\end{aligned}$$

Figure 16 shows a similar calculation with the blockage correction turned on. In this calculation, the drag factors used for the vehicles have been manually adjusted to cancel out the blockage correction. The two calculations have the same answer, so the blockage correction calculation is verified.

8.8 Verification of variant inputs

This section just verifies that when inputs are moved around, they are still handled correctly. In Figure 17 the types of traffic have been moved. The answer is the same as Figure 16. In Figure 18 the types of traffic have been moved again and the pressure differences have been altered (same pressure difference, just different expressions of it). The answer is the same as Figures 16 and 17.

This indicates that the three types of traffic are being handled in the same way. The pressure differentials are handled correctly as well. This might seem like the dullest kind of verification but the misuse of the `$` symbol in spreadsheets is so widespread that it's definitely worth including.

8.9 Verifying switching equations

The spreadsheet calculates three separate quadratic equations and picks one to display. It is important to show that when it switches from one equation to the next, there is no step-change in air velocity.

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_6

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example with the traffic blockage correction term turned on and the drag factors adjusted to cancel out the correction.

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Wind input

P at left portal	20	Pa
P at right portal	0	Pa

Jet fan input

Count of fans	4	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Traffic input

Traffic speed (+ve)	50	km/h
Car flowrate	1800	cars/hr
LCV flowrate	200	LCVs/hr
HGV flowrate	500	HGVs/hr
Car area	2	m²
Car drag factor	0.36864	—
LCV area	4	m²
LCV drag factor	0.67712	—
HGV area	6	m²
HGV drag factor	0.69696	—

Use blockage correction in calc? Y (Y/N)

Calculation (vehicles faster than air)

Air velocity	5.347430	m/s
Volume flow	267.371	m³/s

Independent check calc of pressures

Wind pressure	-20.00	Pa
Friction pressure	223.38	Pa
Traffic pressure	-167.39	Pa
Jet fan pressure	-35.99	Pa
Is the sum zero?	TRUE	

Figure 16: Quadratics spreadsheet: blockage correction on

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_7

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example with the traffic flow and drag data switched around.

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Traffic input

Traffic speed (+ve)	50	km/h
Car flowrate	500	cars/hr
LCV flowrate	1800	LCVs/hr
HGV flowrate	200	HGVs/hr
Car area	6	m²
Car drag factor	0.9	—
LCV area	2	m²
LCV drag factor	0.4	—
HGV area	4	m²
HGV drag factor	0.8	—

Wind input

P at left portal	20	Pa
P at right portal	0	Pa

Use blockage correction in calc? N (Y/N)

Jet fan input

Count of fans	4	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Calculation (vehicles faster than air)

Air velocity	5.347430	m/s
Volume flow	267.371	m³/s

Independent check calc of pressures

Wind pressure	-20.00	Pa
Friction pressure	223.38	Pa
Traffic pressure	-167.39	Pa
Jet fan pressure	-35.99	Pa
Is the sum zero?	TRUE	

Figure 17: Quadratics spreadsheet: traffic inputs switched around

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_8

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example with the traffic flow and drag data switched again and the pressures modified

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Wind input

P at left portal	0	Pa
P at right portal	-20	Pa

Jet fan input

Count of fans	4	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Traffic input

Traffic speed (+ve)	50	km/h
Car flowrate	200	cars/hr
LCV flowrate	500	LCVs/hr
HGV flowrate	1800	HGVs/hr
Car area	4	m²
Car drag factor	0.8	—
LCV area	6	m²
LCV drag factor	0.9	—
HGV area	2	m²
HGV drag factor	0.4	—

Use blockage correction in calc? N (Y/N)

Calculation (vehicles faster than air)

Air velocity	5.347430 m/s
Volume flow	267.371 m³/s

Independent check calc of pressures

Wind pressure	-20.00 Pa
Friction pressure	223.38 Pa
Traffic pressure	-167.39 Pa
Jet fan pressure	-35.99 Pa
Is the sum zero?	TRUE

Figure 18: Quadratics spreadsheet: traffic inputs and pressure switched around

Figures 19 and 20 show the switch from "traffic slower than the air" to "traffic faster than the air". The traffic speed is 18 km/h (5 m/s). In Figure 19 the calculated air velocity is 5.000061 m/s. In Figure 20 the calculated air velocity is 4.99939 m/s and the spreadsheet changes to using the equation for traffic faster than air (caused by a change in wind pressure of 0.01 Pa). So those two match up at the point where the calculation switches from one to the other.

Figures 21 and 22 show the switch from "traffic faster than the air" to "negative airflow". In these cases the adverse wind pressure is such that the resulting air velocities are on either side of zero.

In Figure 21 the calculated air velocity is +0.000051 m/s and the spreadsheet uses the equation for traffic faster than air. In Figure 22 the adverse pressure is 0.01 Pa higher and the calculated air velocity is −0.000103 m/s. So those two also match at the point where they pass from one to the other.

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_9

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example with the traffic just slower than the airflow

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Wind input

P at left portal	94.92	Pa
P at right portal	0	Pa

Jet fan input

Count of fans	11	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Traffic input

Traffic speed (+ve)	18	km/h
Car flowrate	1800	cars/hr
LCV flowrate	200	LCVs/hr
HGV flowrate	500	HGVs/hr
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.8	—
HGV area	6	m²
HGV drag factor	0.9	—

Use blockage correction in calc? N (Y/N)

Calculation (vehicles faster than air)

Air velocity	4.999939 m/s
Volume flow	249.997 m³/s

Independent check calc of pressures

Wind pressure	-94.92 Pa
Friction pressure	195.30 Pa
Traffic pressure	0.00 Pa
Jet fan pressure	-100.38 Pa
Is the sum zero?	TRUE

Figure 19: Quadratics spreadsheet: traffic just slower than the air

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_10

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example with the traffic just faster than the airflow

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Traffic input

Traffic speed (+ve)	18	km/h
Car flowrate	1800	cars/hr
LCV flowrate	200	LCVs/hr
HGV flowrate	500	HGVs/hr
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.8	—
HGV area	6	m²
HGV drag factor	0.9	—

Wind input

P at left portal	94.93	Pa
P at right portal	0	Pa

Use blockage correction in calc? N (Y/N)

Jet fan input

Count of fans	11	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Calculation (vehicles slower than air)

Air velocity	5.000061 m/s
Volume flow	250.003 m³/s

Independent check calc of pressures

Wind pressure	-94.93 Pa
Friction pressure	195.30 Pa
Traffic pressure	0.00 Pa
Jet fan pressure	-100.37 Pa
Is the sum zero?	TRUE

Figure 20: Quadratics spreadsheet: traffic just faster than the air

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_11

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example with the air velocity positive and just above zero.

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Traffic input

Traffic speed (+ve)	18	km/h
Car flowrate	200	cars/hr
LCV flowrate	500	LCVs/hr
HGV flowrate	1800	HGVs/hr
Car area	4	m²
Car drag factor	0.8	—
LCV area	6	m²
LCV drag factor	0.9	—
HGV area	2	m²
HGV drag factor	0.4	—

Wind input

P at left portal	0	Pa
P at right portal	192.18	Pa

Use blockage correction in calc? N (Y/N)

Jet fan input

Count of fans	3	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Calculation (vehicles faster than air)

Air velocity	0.000051 m/s
Volume flow	0.003 m³/s

Independent check calc of pressures

Wind pressure	192.18 Pa
Friction pressure	0.00 Pa
Traffic pressure	-159.33 Pa
Jet fan pressure	-32.85 Pa
Is the sum zero?	TRUE

Figure 21: Quadratics spreadsheet: air velocity just above zero

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_12

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example with the air velocity negative and just below zero.

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Traffic input

Traffic speed (+ve)	18	km/h
Car flowrate	1800	cars/hr
LCV flowrate	200	LCVs/hr
HGV flowrate	500	HGVs/hr
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.8	—
HGV area	6	m²
HGV drag factor	0.9	—

Wind input

P at left portal	0	Pa
P at right portal	192.19	Pa

Use blockage correction in calc? N (Y/N)

Jet fan input

Count of fans	3	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Calculation (reverse airflow)

Air velocity	-0.000103	m/s
Volume flow	-0.005	m³/s

Independent check calc of pressures

Wind pressure	192.19	Pa
Friction pressure	0.00	Pa
Traffic pressure	-159.34	Pa
Jet fan pressure	-32.85	Pa
Is the sum zero?	TRUE	

Figure 22: Quadratics spreadsheet: air velocity just below zero

8.10 Verifying balanced forces

A number of situations can be generated in which two forces exactly balance one another and lead to $v_t = 0$.

Take the first one as wind pressure balancing jet fan thrust. Jet fan thrust (equation 5) is a pressure difference. When $v_t = 0$ we can calculate the pressure rise caused by four 730 N jet fans with jet speed 30 m/s and installation efficiency 0.75 in a tunnel of area 50 m^2:

$$\begin{aligned} \Delta P &= \frac{N_f T_f \eta_f}{|v_f| A_t}(v_f - v_t) \\ \Rightarrow &= \frac{4 \times 730 \times 0.75}{|30| \times 50}(30 - 0) \\ \Rightarrow &= 43.8 \text{ Pa} \end{aligned}$$

Figure 23 shows a calculation with jet fans that have those properties and an adverse wind pressure of 43.8 Pa. The air velocity calculated by the spreadsheet is zero.

Take the second as traffic drag balancing jet fan thrust. We'll use the same jet fans as before but blowing in reverse.

$$\begin{aligned} \Delta P &= \frac{N_f T_f \eta_f}{|v_f| A_t}(v_f - v_t) \\ \Rightarrow &= \frac{4 \times 730 \times 0.75}{|-30| \times 50}(-30 - 0) \\ \Rightarrow &= -43.8 \text{ Pa} \end{aligned}$$

There is one traffic type at 36 km/h (10 m/s). We set a count of cars in the tunnels such that the pressure generated is 43.8 Pa (it turns out to be 45.625 cars):

$$\begin{aligned} \Delta P &= \frac{1}{2}\rho \frac{N_v c_d A_v}{A_t}(v_v - v_t)\,|v_v - v_t| \\ \Rightarrow &= \frac{1}{2} \times 1.2 \frac{45.625 \times 0.4 \times 2}{50}(10 - 0)\,|10 - 0| \\ \Rightarrow &= 43.8 \text{ Pa} \end{aligned}$$

45.625 cars in the tunnel (which is 2 km long) mean that the density of traffic is 22.8125 cars/km. At a traffic speed of 36 km/h this equates to 821.25 veh/hr, which is what is put into the spreadsheet for the flowrate of cars.

Figure 24 shows that the air velocity calculated by the spreadsheet is zero.

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_13

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example in which the jet fan thrust exactly balances the wind pressure difference.

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Traffic input

Traffic speed (+ve)	50	km/h
Car flowrate	0	cars/hr
LCV flowrate	0	LCVs/hr
HGV flowrate	0	HGVs/hr
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.8	—
HGV area	6	m²
HGV drag factor	0.9	—

Wind input

P at left portal	0	Pa
P at right portal	43.8	Pa

Use blockage correction in calc? N (Y/N)

Jet fan input

Count of fans	4	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Calculation (vehicles faster than air)

Air velocity	0.000000 m/s
Volume flow	0.000 m³/s

Independent check calc of pressures

Wind pressure	43.80 Pa
Friction pressure	0.00 Pa
Traffic pressure	0.00 Pa
Jet fan pressure	-43.80 Pa
Is the sum zero?	TRUE

Figure 23: Quadratics spreadsheet: wind pressure exactly balancing jet fans

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-quadratics-verification.ods'#$quadratics_14

Single tunnel of constant area with losses, friction, jet fans, traffic drag and portal pressures.
Example in which the traffic drag exactly balances the jet fan thrust.
This case also verifies that jet fans blowing backwards are handled correctly.

Tunnel geometry input

Length	2000	m
Area	50	m²
Perimeter	32	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.036	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	6.25	m

Traffic input

Traffic speed (+ve)	36	km/h
Car flowrate	821.25	cars/hr
LCV flowrate	0	LCVs/hr
HGV flowrate	0	HGVs/hr
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.8	—
HGV area	6	m²
HGV drag factor	0.9	—

Wind input

P at left portal	0	Pa
P at right portal	0	Pa

Use blockage correction in calc? N (Y/N)

Jet fan input

Count of fans	4	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	-30	m/s
Installation efficiency	0.75	—

Calculation (vehicles faster than air)

Air velocity	0.000000 m/s
Volume flow	0.000 m³/s

Independent check calc of pressures

Wind pressure	0.00 Pa
Friction pressure	0.00 Pa
Traffic pressure	-43.80 Pa
Jet fan pressure	43.80 Pa
Is the sum zero?	TRUE

Figure 24: Quadratics spreadsheet: traffic drag exactly balancing jet fans

9 Loss factor conversions in Hobyah

Fixed loss factors in Hobyah (the `loss1` and `loss2` keywords) are converted from their input values into pressure loss factors on the area of the tunnel they are in.

It can be disconcerting to look at the contents of the `log` file and see a k-factor set at the end of a segment that you don't remember putting into the input file. The following sections explain why that happens, verify the calculations used in the conversion and show the traceability data included in the log file for them.

9.1 The `loss1` keyword

The `loss1` keyword takes input consisting of an area and two pressure loss factors (one loss for positive airflow, one for negative airflow). The area the losses are applied to does not need to be the same as the area of the current sectype.

When Hobyah splits tunnels up into segments, it is convenient to convert the pressure loss factors on the area in the `loss1` entry into pressure loss factors on the segment area instead.

This section verifies this in excruciating detail, because there's always going to be someone out there who needs that level of explanation (a case in point: me back when I was a graduate engineer).

Let's take an example in test file `ok-032-fans-in-parallel.txt`. The `sectype` block and the definition of the tunnel named `system_res` are as follows:

```
begin sectypes
  systemduct   40     25   -0.036   # area  perimeter  lambda
end sectypes

begin tunnel system_res
  back  10000 portal +20   systemduct    # +20 Pa gauge air pressure.
  loss1 10030   20.0    30   56          # distance  area  zeta_bf  zeta_fb
  fwd   10100 node plenum
end tunnel
```

This creates a 100 m long tunnel (called `system_res`) with a loss set around a third of the way along. The tunnel uses a sectype named `systemduct`, which has area 40 m^2.

The loss has an area of 20 m^2, a loss factor for +ve airflow of 30 and a loss factor for −ve airflow of 56. Those loss factors are applied to the air velocity through the area of the loss:

$$\Delta P = \frac{1}{2}\rho\zeta_{loss}v_{loss}^2 \tag{11}$$

$$\Delta P = \frac{1}{2}\rho\zeta_{loss}\frac{Q^2}{A_{loss}^2} \tag{12}$$

where

$$\begin{aligned} \Delta P &= \text{Pressure drop across the loss } (Pa) \\ \rho &= \text{air density } (kg/m^3) \end{aligned}$$

$$
\begin{aligned}
Q &= \text{Volume flow } (m^3/s) \\
\zeta_{loss} &= \text{loss coefficient applied to } A_{loss} \ (-) \\
v_{loss} &= \text{Air velocity through the loss } (m/s) \\
A_{loss} &= \text{Area at the loss } (m^2)
\end{aligned}
$$

There is a loss factor ζ_t that applies at the tunnel air velocity that creates the same pressure drop (Pa):

$$\Delta P = \frac{1}{2}\rho\zeta_t v_t^2 \tag{13}$$

$$\Delta P = \frac{1}{2}\rho\zeta_t \frac{Q^2}{A_t^2} \tag{14}$$

where

$$
\begin{aligned}
\Delta P &= \text{Pressure drop across the loss } (Pa) \\
\rho &= \text{air density } (kg/m^3) \\
Q &= \text{Volume flow } (m^3/s) \\
\zeta_t &= \text{loss coefficient applied to } A_t \ (-) \\
v_t &= \text{Air velocity through the tunnel } (m/s) \\
A_t &= \text{Area of the tunnel } (m^2)
\end{aligned}
$$

We make the equations that use volume flow (12) and (14) equal to one another, eliminate the common terms and plug in areas $A_{loss} = 20$ m^2 and $A_t = 40$ m^2:

$$\frac{1}{2}\rho\zeta_t \frac{Q^2}{A_t^2} = \frac{1}{2}\rho\zeta_{loss} \frac{Q^2}{A_{loss}^2} \tag{15}$$

$$\Rightarrow \frac{1}{2}\rho Q^2 \frac{\zeta_t}{A_t^2} = \frac{1}{2}\rho Q^2 \frac{\zeta_{loss}}{A_{loss}^2} \tag{16}$$

$$\Rightarrow \frac{\zeta_t}{A_t^2} = \frac{\zeta_{loss}}{A_{loss}^2} \tag{17}$$

$$\Rightarrow \zeta_t = \frac{A_t^2}{A_{loss}^2}\zeta_{loss} \tag{18}$$

$$\Rightarrow \zeta_t = \frac{40^2}{20^2}\zeta_{loss} \tag{19}$$

$$\Rightarrow \zeta_t = 4\zeta_{loss} \tag{20}$$

So in the context of this example, the loss values that will appear in the log file are four times the values in the input file: $4 \times 30 = 120$ and $4 \times 56 = 224$.

The following is an extract of the log file for one of the segments in `ok-032-fans-in-parallel.txt`:

```
Seg_ID 1, 1st segment of tunnel "system_res":
             Back end: 10000.0 m (portal with gauge pressure 20.0 Pa)
          Forward end: 10030.0 m (pressure loss: area = 40 m^2, zeta_bf = 30, zeta_fb = 56)
       Segment length: 30.0 m
         Segment area: 40.0 m^2
            Perimeter: 25.0 m
   Hydraulic diameter: 6.4 m
    Fixed Fanning c_f: 0.009
   Fixed Darcy lambda: 0.036
```

```
     Fixed Atkinson k: 0.0054 kg/m^3
     Back inflow zeta: 0.5    (zeta_bf)
    Back outflow zeta: 1.0    (zeta_fb)
 Forward outflow zeta: 120.0    (zeta_bf)
  Forward inflow zeta: 224.0    (zeta_fb)
       Count of cells: 1
          Cell length: 30.0 m
  Count of gridpoints: 2
```

The lines for pressure loss factors at the forward end in the transcript match the calculation above, so the adjustment of losses from loss area to tunnel area in the `loss1` keyword is verified. Note that the numbers given in the `loss1` keyword are included in the text describing what is at the forward end, for completeness.

9.2 The `loss2` keyword

The `loss2` keyword takes input consisting of two Atkinson resistances (one loss for positive airflow, one for negative airflow).

Hobyah converts the Atkinson resistances into pressure loss coefficients applying to the air velocity in the segment at the end of the segment.

Let's take an example, also from test file `ok-032-fans-in-parallel.txt`. Again, we do this mind-numbing detail; but that might be a good thing, considering how averse many tunnel ventilation engineers are to Atkinson resistance. The tunnel named `fan2_duct` has its sectype definition and tunnel definition as follows:

```
begin sectypes
  fanduct  10.2     12.8   0.012   # area  perimeter  roughness
end sectypes

begin tunnel fan2_duct
  back    1100 node plenum fanduct
  loss2   1120  0.015  0.015          # distance  R_bf  R_fb
  damper2 1140  D2  D2_ops  time  atk1  atk1
  fan2    1160  2  my_char  fan2ops   T   speed
  fwd     1200 node discharge
end tunnel
```

The sectype named `fanduct` has area 10.2 m^2. The duct has a `loss2` keyword to represent the pressure losses in the nozzle and diffuser, a fan isolation damper that follows a predefined resistance profile (the `damper2` keyword) and a fan that follows a predefined fan operational profile (the `fan2` keyword).

The loss has Atkinson resistance $R = 0.015$ gauls for flow in both directions. These can be converted into a pressure loss factor on the air velocity through the tunnel area by considering two expressions for pressure drop:

$$\Delta P = RQ^2 \tag{21}$$

$$\Delta P = \frac{1}{2}\rho\zeta_t\frac{Q^2}{A_t^2} \tag{22}$$

where

$$\begin{aligned} \Delta P &= \text{Pressure drop across the loss } (Pa) \\ R &= \text{Atkinson resistance } (Pa\text{-}s^2/m^6) \\ \rho &= \text{air density } (kg/m^3) \\ Q &= \text{Volume flow } (m^3/s) \\ \zeta_t &= \text{loss coefficient applied to tunnel air velocity}(-) \\ A_t &= \text{Area of the tunnel } (m^2) \end{aligned}$$

We equate the right hand sides of (21) and (22), eliminate the common terms and plug in area $A_t = 10.2$ m^2, $R = 0.015$ gauls and $\rho = 1.2$ kg/m^3 (R is valid at that density):

$$RQ^2 = \frac{1}{2}\rho\zeta_t\frac{Q^2}{A_t^2} \quad (23)$$

$$\Rightarrow R = \frac{\rho\zeta_t}{2A_t^2} \quad (24)$$

$$\Rightarrow \zeta_t = \frac{2RA_t^2}{\rho} \quad (25)$$

$$\Rightarrow \zeta_t = \frac{2 \times 0.015 \times 10.2^2}{1.2} \quad (26)$$

$$\Rightarrow \zeta_t = 2.601 \text{ applied to } 10.2 \text{ m}^2. \quad (27)$$

The following is an extract of the log file for `ok-032-fans-in-parallel.txt`:

```
Seg_ID 8, 1st segment of tunnel "fan2_duct":
             Back end: 1200.0 m (tunnel end at a node named "plenum")
          Forward end: 1180.0 m (fixed Atkinson resistances: R_bf = 0.015 gauls, R_fb = 0.015
       Segment length: 20.0 m
         Segment area: 10.2 m^2
            Perimeter: 12.8 m
   Hydraulic diameter: 3.1875 m
     Roughness height: 0.012 m
   Relative roughness: 0.0037647
         >>> Fully turbulent Fanning c_f : 0.006981 Colebrook's 1939 approximation (used)
         >>> Fully turbulent Fanning c_f : 0.006983 Colebrook-White (exact)
         >>> Fully turbulent Fanning c_f : 0.007182 Moody's 1947 approximation
         >>> Fully turbulent Fanning c_f : 0.007117 SES's 1974 approximation
         >>> Fully turbulent Darcy lambda: 0.027923 Colebrook's 1939 approximation (used)
         >>> Fully turbulent Atkinson k  : 0.0041885 kg/m^3
    Back inflow zeta: 0.0    (zeta_bf)
   Back outflow zeta: 0.0    (zeta_fb)
 Forward outflow zeta: 2.601    (zeta_bf)
  Forward inflow zeta: 2.601    (zeta_fb)
       Count of cells: 1
          Cell length: 20.0 m
  Count of gridpoints: 2
```

The lines for pressure loss factors at the forward end in the transcript match the calculation above, so the conversion of Atkinson resistances to a pressure loss factor in the `loss2` keyword is verified. Note that the numbers given in the `loss2` keyword are included in the text describing what is at the forward end for completeness.

10 Hobyah with one tunnel

This section verifies the Hobyah calculation of flow in one tunnel driven by a pressure difference in two ways:

- compare the steady-state results of a Hobyah calculation to the results of calculations by the quadratics spreadsheet.
- compare the transient results of a (slightly different) Hobyah calculation to the transient results of calculations by SES.

The first comparison shows that when the transient compressible calculation converges, it gets to the same calculation as the quadratics spreadsheet.

The second comparison shows that the rates of change of air velocity from zero flow to steady-state in the compressible calculation is similar to the incompressible calculation in SES.

10.1 Hobyah/quadratics spreadsheet steady-state comparison

First we choose a geometry. Let's use area 92 m^2, perimeter 45 m, length 1500 m and fixed Darcy friction factor 0.01. We use a fixed pressure difference across the tunnel of 20 Pa.

This is the arrangement of the tunnel named `mainline1` in test file `ok-022-sectype-changes-verification.txt`. The losses at the back end are $\zeta_{bf} = 2.6$ and $\zeta_{fb} = 0.85$ and the losses at the forward end are $\zeta_{bf} = 0.85$ and $\zeta_{fb} = 0.24$.

The Hobyah calculation runs for 3000 seconds so that it is as close to convergence as possible. Figure 25 shows how the compressible calculation develops and the final pressure profile.

The final air velocities can be found by interrogating the contents of `ok-022-sectype-changes-verification.hbn` in a Python session:

```
% python
Python 3.9.4 (default, Apr 23 2021, 13:22:46)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import classHobyah as clH
>>> folder = "verification+validation"
>>> fname = "ok-022-sectype-changes-verification.hbn"
>>> log = open("log_discard.txt", "w")
>>> ok022 = clH.Hobyahdata(folder, fname, log)
>>> ok022.PrettyClassPrint("tuns2frames")
Printing the contents of "tuns2frames" in "ok-022-sectype-changes-verification.hbn".
  dict "self.tuns2frames":
     mainline1 : [0]
     mainline2 : [1]
     mainline3 : [2, 3]
     mainline4 : [4, 5]
Printed the contents of "tuns2frames" only.
>>>
```

This prints the dictionary that holds the correlation between the tunnels and the segments in the file. `Mainline1` is the first segment (segment zero) and `mainline2` is the second (segment 1). Tunnels `mainline3` and `mainline4` are not relevant here.

First we print the air velocities in the first segment, which correlates to the tunnel named `mainline1`:

```
>>> ok022.v_bin[0]
        0.000000     36.585366    ...  1463.414634  1500.000000
0.0        0.000000     0.000000  ...     0.000000     0.00000
0.1        0.048471     0.000000  ...     0.000000     0.00000
0.2        0.048459     0.045553  ...     0.000000     0.00000
0.3        0.048458     0.048284  ...     0.000000     0.00000
0.4        0.048458     0.048448  ...     0.000000     0.00000
...             ...          ...  ...          ...         ...
2992.0     2.511457     2.511460  ...     2.511577     2.51158
2994.0     2.511457     2.511460  ...     2.511577     2.51158
2996.0     2.511457     2.511460  ...     2.511577     2.51158
2998.0     2.511457     2.511460  ...     2.511577     2.51158
3000.0     2.511457     2.511460  ...     2.511577     2.51158

[3401 rows x 42 columns]
```

This shows that the air velocity ranges between 2.511457 m/s and 2.51158 m/s in `mainline1` (the difference is due to compressibility). Figure 26 shows the equivalent calculation in the quadratics spreadsheet: the value is 2.511857 m/s.

A similar call prints the velocities in `mainline2`:

```
>>> ok022.v_bin[1]
        0.000000     36.585366    ...  1463.414634  1500.000000
0.0        0.000000     0.000000  ...     0.000000     0.000000
0.1        0.000000     0.000000  ...     0.000000    -0.048471
0.2        0.000000     0.000000  ...    -0.045557    -0.048467
0.3        0.000000     0.000000  ...    -0.048292    -0.048466
0.4        0.000000     0.000000  ...    -0.048456    -0.048466
...             ...          ...  ...          ...         ...
2992.0    -2.792555    -2.792551  ...    -2.792390    -2.792386
2994.0    -2.792555    -2.792551  ...    -2.792390    -2.792386
2996.0    -2.792555    -2.792551  ...    -2.792390    -2.792386
2998.0    -2.792555    -2.792551  ...    -2.792390    -2.792386
3000.0    -2.792555    -2.792551  ...    -2.792390    -2.792386

[3401 rows x 42 columns]
```

This shows that the air velocity ranges between -2.792555 m/s and -2.792386 m/s in `mainline2`. Figure 27 shows the equivalent calculation in the quadratics spreadsheet: the value is -2.792608 m/s.

The calculation is run for 3000 seconds so that it is as close to convergence as possible. Figure 25 shows how the compressible calculation develops and the final pressure profile.

The velocities in the compressible and incompressible calculations differ by less than 0.02%, so we can say that the steady-state of the compressible calculation driven by wind pressure is verified.

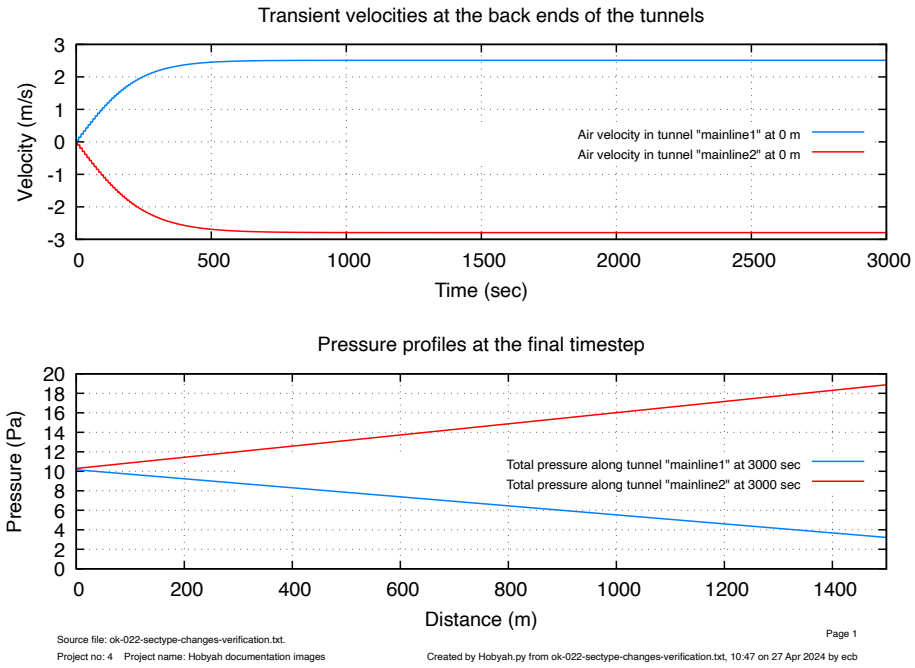

Figure 25: Hobyah calculation convergence

10.2 Hobyah/SES transient comparison

First we choose a geometry. Let's use the same as the previous section, but set a roughness height of 0.05 m, as SES cannot use fixed friction factors. We tell Hobyah to use Moody's approximation to calculate friction factors from roughness height, because SES uses Moody's approximation in segments without trains and SES's approximation in segments with trains.[4]

The test file `ok-041-risetime-verification.txt` has two tunnels (again named `mainline1` and `mainline2`) with forward and reverse flow respectively.

The equivalent calculations are in SES runs `SES-067-sectype-changes-pos.ses` and `SES-068-sectype-changes-neg.ses`. SES v4.1 does not have the ability to exert fixed pressure, so a workaround is needed. The standard way is to use a jet fan with a very high jet velocity so that the derating factor is negligible and the jet fan's thrust is effectively constant. A jet velocity of 999,999 fpm (5,080 m/s) is used. The air velocity in the tunnels reaches 2 m/s, so the jet fan derating factor ends up as

[4] I learned that SES uses Moody's friction approximation when there are no trains present and uses SES's friction approximation when there are trains present while writing this section of the verification document. You learn something new every day.

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-ok-022-sectype-changes-verification.ods'#$quadratics_2

Single tunnel of constant area with losses and friction. This is a slug-flow calculation matching the compressible calculation in the tunnel named "Mainline1" in the Hobyah file "ok-022-area-changes.txt". The compressible calculation ends up with air velocity +2.511457 m/s, this incompressible spreadsheet has +2.511587 m/s.

Tunnel geometry input

Length	1500	m
Area	92	m²
Perimeter	45	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.01	—
Left portal loss	2.6	—
Right portal loss	0.85	—
D_h (info only, not used)	8.18	m

Traffic input

Traffic speed (+ve)	0	km/h
Car density	0	cars/km
LCV density	0	LCVs/km
HGV density	0	HGVs/km
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.9	—
HGV area	6	m²
HGV drag factor	0.8	—

Wind input

P at left portal	20	Pa
P at right portal	0	Pa

Use blockage correction in calc? N (Y/N)

Jet fan input

Count of fans	0	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Calculation (vehicles slower than air)

Air velocity	2.511587 m/s
Volume flow	231.066 m³/s

Independent check calc of pressures

Wind pressure	-20.00 Pa
Friction pressure	20.00 Pa
Traffic pressure	0.00 Pa
Jet fan pressure	0.00 Pa
Is the sum zero?	TRUE

Figure 26: Spreadsheet calculation, positive airflow

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-ok-022-sectype-changes-verification.ods'#$quadratics_1

Single tunnel of constant area with losses and friction. This is a slug-flow calculation matching the compressible calculation in the tunnel named "Mainline2" in the Hobyah file "ok-022-area-changes.txt". The compressible calculation ends up with air velocity -2.792555 m/s, this incompressible spreadsheet has -2.792608 m/s.

Tunnel geometry input

Length	1500	m
Area	92	m²
Perimeter	45	m
Friction type	Darcy	(Darcy/Fanning)
Darcy fricfac λ	0.01	—
Left portal loss	2.2	—
Right portal loss	0.24	—
D_h (info only, not used)	8.18	m

Traffic input

Traffic speed (+ve)	0	km/h
Car density	0	cars/km
LCV density	0	LCVs/km
HGV density	0	HGVs/km
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.9	—
HGV area	6	m²
HGV drag factor	0.8	—

Wind input

P at left portal	0	Pa
P at right portal	20	Pa

Use blockage correction in calc? N (Y/N)

Jet fan input

Count of fans	0	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Calculation (reverse airflow)

Air velocity	-2.792608	m/s
Volume flow	-256.920	m³/s

Independent check calc of pressures

Wind pressure	20.00 Pa
Friction pressure	-20.00 Pa
Traffic pressure	0.00 Pa
Jet fan pressure	0.00 Pa
Is the sum zero?	TRUE

Figure 27: Spreadsheet calculation, negative airflow

$1 - 2/5080 = 0.9996$.

The final air velocities in the Hobyah run can be found by interrogating the contents of `ok-041-comparison.hbn` in a Python session. As with all the previous transcripts of Python sessions, it imports the class, sets variable names, opens a log file for writing, then imports the contents of the `.hbn` file as a class instance:

```
% python
Python 3.9.4 (default, Apr 23 2021, 13:22:46)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import classHobyah as clH
>>> subf = "verification+validation"
>>> log = open("log_discard.txt", "w")
>>> ok041 = clH.Hobyahdata(subf, "ok-041-risetime-verification.hbn", log)
>>> ok041.PrettyClassPrint("tuns2frames")
Printing the contents of "tuns2frames" in "ok-041-risetime-verification.hbn".
  dict "self.tuns2frames":
     mainline2 : [0]
     mainline1 : [1]
Printed the contents of "tuns2frames" only.
>>>
```

Note that in this file, `mainline1` maps to the second segment rather than the first segment. This is because I edited `ok-041-risetime-verification.txt` so that the definition of tunnel `mainline2` is before the definition of `mainline1` in `ok-041-risetime-verification.txt`. This is just to drive home the message that the correlation between the segment numbers and the tunnels depends on the order in which the `begin tunnel <name>` blocks appear in the input file.

First we print the air velocities in `mainline1`:

```
>>> ok041.v_bin[1]
        0.000000     36.585366    ...  1463.414634  1500.000000
0.0        0.000000     0.000000  ...     0.000000     0.000000
0.1        0.048471     0.000000  ...     0.000000     0.000000
0.2        0.048459     0.045553  ...     0.000000     0.000000
0.3        0.048458     0.048284  ...     0.000000     0.000000
0.4        0.048458     0.048447  ...     0.000000     0.000000
...             ...          ...  ...          ...          ...
2992.0     1.875064     1.875068  ...     1.875228     1.875232
2994.0     1.875064     1.875068  ...     1.875228     1.875232
2996.0     1.875064     1.875068  ...     1.875228     1.875232
2998.0     1.875064     1.875068  ...     1.875228     1.875232
3000.0     1.875064     1.875068  ...     1.875228     1.875232

[3401 rows x 42 columns]
>>>>
```

This shows that the air velocity ranges between 1.875064 m/s and 1.875232 m/s in `mainline1` (the difference is due to compressibility). The equivalent calculation in SES run `SES-066-sectype-changes.ses`: the value is 369.6 fpm (1.878 m/s).

A similar call prints the velocities in `mainline2`:

```
>>> ok041.v_bin[0]
        0.000000     36.585366    ...  1463.414634  1500.000000
0.0        0.000000     0.000000  ...     0.000000     0.000000
0.1        0.000000     0.000000  ...     0.000000    -0.048471
0.2        0.000000     0.000000  ...    -0.045557    -0.048467
0.3        0.000000     0.000000  ...    -0.048292    -0.048466
0.4        0.000000     0.000000  ...    -0.048455    -0.048466
...             ...          ...  ...          ...          ...
2992.0    -1.983952    -1.983947  ...    -1.983758    -1.983753
2994.0    -1.983952    -1.983947  ...    -1.983758    -1.983753
2996.0    -1.983952    -1.983947  ...    -1.983758    -1.983753
2998.0    -1.983952    -1.983947  ...    -1.983758    -1.983753
3000.0    -1.983952    -1.983947  ...    -1.983758    -1.983753

[3401 rows x 42 columns]
```

This shows that the air velocity ranges between −1.983952 m/s and −1.983753 m/s in `mainline1` (the difference is due to compressibility). The equivalent calculation in SES run `SES-067-sectype-changes.ses`: the value is −391.1 fpm (−1.987 m/s), about 0.15% different.

These velocities are all close enough that we can put the differences down to compressible versus incompressible.

Hobyah and SES both carry out transient flow. Figure 28 shows how the airflows develop in Hobyah and SES; they match each other well. One feature of the compressible calculation is that the velocity rises in steps, at intervals. The interval between the step changes in velocity is the time it takes for a pressure wave to travel up and down the 1500 m long tunnel: 2×1500 m $\div$ 343.8 m/s = 8.726 sec. The lower graph in Figure 28 shows it in more detail.

11 Hobyah with traffic

This section verifies Hobyah's calculation of road traffic drag (moving and stationary traffic).

Hobyah's calculations work in the same basic way as the traffic in the quadratics spreadsheet, but are more complex. The main differences are:

- Any number of vehicle types can be used in Hobyah (the spreadsheet is limited to three vehicle types).
- Vehicle types can be told to slow on steep upgrades in Hobyah (the spreadsheet has all vehicles at the same speed).
- Stationary traffic can be limited to part of the route in Hobyah (in the spreadsheet, vehicles fill the tunnel end to end).
- Traffic at different speeds can pass through tunnels in Hobyah.
- Passenger Carrier Unit (PCU) values can be assigned to vehicle types at different speeds. PIARC's current guidelines on tunnel ventilation (2019) assign 3 PCU to heavy goods vehicles (HGVs) when they are travelling at 10 km/h and slower and 2 PCU when HGVs are moving faster than 10 km/h.

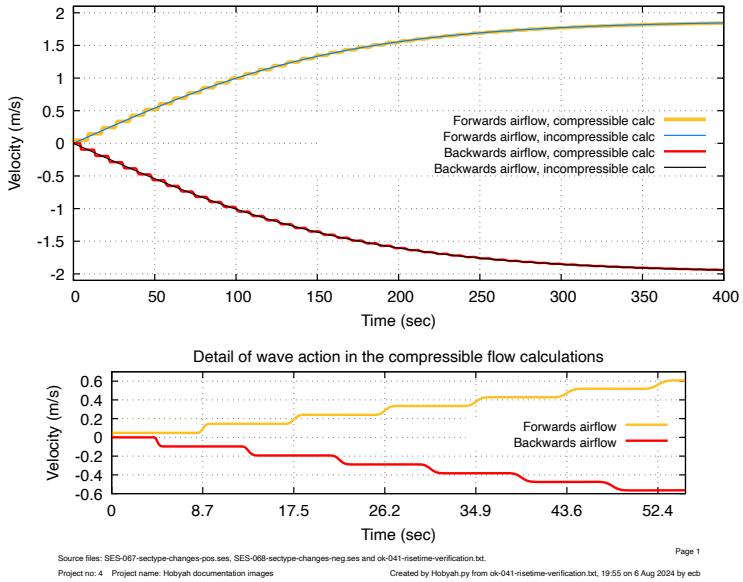

Figure 28: Comparison of compressible and incompressible transient flow

11.1 Steady-state comparison with pressures at portals

This section takes a tunnel full of stationary traffic with a pressure difference driving the flow. The details of the tunnel and the traffic are shown in the quadratics spreadsheet printout in Figure 29. The spreadsheet has a tunnel full of traffic and the blockage correction turned off. The resulting air velocity in the tunnel is 4.525630 m/s.

The Hobyah file `ok-039-traffic-verification.txt` has an equivalent calculation. The final air velocities in the Hobyah run can be found by interrogating the contents of `ok-039-traffic-verification.hbn` in a Python session. As with all the previous transcripts of Python sessions, it imports the class, sets variable names, opens a log file for writing, then imports the contents of the `.hbn` file as a class instance:

```
% python
Python 3.9.4 (default, Apr 23 2021, 13:22:46)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import classHobyah as clH
>>> log = open("log_discard.txt", "w")
>>> subfolder = "verification+validation"
>>> ok039 = clH.Hobyahdata(subfolder, "ok-039-traffic-verification.hbn", log)
>>> ok039.v_bin
(          0.0       40.0      80.0      120.0     160.0     200.0
0.0     0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
1.0     0.144943  0.156413  0.170146  0.170173  0.156462  0.144975
2.0     0.327348  0.326242  0.324528  0.324565  0.326338  0.327478
3.0     0.493308  0.491767  0.489272  0.489317  0.491912  0.493542
4.0     0.654937  0.655038  0.655184  0.655247  0.655232  0.655238
```

Slug flow in a simple road tunnel by quadratic equations. Solution of the equations in section IV of PIARC report 05.02.B, "Vehicle emissions, air demand, environment, longitudinal ventilation". Handles stationary traffic, moving traffic (vehicle speed must not be negative) jet fans and wind. Selects the solution from a range of different quadratics, depending on the inputs. The check calculation uses the source formula.

Slug flow spreadsheet v1.3 'slug-flow-ok-039-traffic-verification.ods'#$quadratics_1

Single tunnel of constant area with losses and friction. This is a slug-flow calculation matching the compressible calculation in the tunnel named "filled" in the Hobyah file "ok-039-stationary-traffic.txt". The compressible calculation ends up with air velocities between 4.525234 and 4.525922 m/s, this incompressible spreadsheet has +4.525630 m/s.

Tunnel geometry input

Length	200	m
Area	103	m²
Perimeter	55	m
Friction type	Fanning	(Darcy/Fanning)
Fanning fricfac c_f	0.008	—
Left portal loss	0.5	—
Right portal loss	1	—
D_h (info only, not used)	7.49	m

Traffic input

Traffic speed (+ve)	0	km/h
Car density	197.16206	cars/km
LCV density	25.261389	LCVs/km
HGV density	35.85885	HGVs/km
Car area	2	m²
Car drag factor	0.4	—
LCV area	4	m²
LCV drag factor	0.9	—
HGV area	6	m²
HGV drag factor	1	—

Wind input

P at left portal	20	Pa
P at right portal	-20	Pa

Use blockage correction in calc? N (Y/N)

Jet fan input

Count of fans	0	—
Static thrust	730	N at 1.2 kg/m³
Jet velocity	30	m/s
Installation efficiency	0.75	—

Calculation (vehicles slower than air)

Air velocity	4.525630	m/s
Volume flow	466.140	m³/s

Independent check calc of pressures

Wind pressure	-40.00 Pa
Friction pressure	28.93 Pa
Traffic pressure	11.07 Pa
Jet fan pressure	0.00 Pa
Is the sum zero?	TRUE

Figure 29: Quadratics spreadsheet: simple tunnel with stationary traffic

```
...          ...       ...       ...       ...       ...       ...
2996.0  4.525234  4.525371  4.525509  4.525647  4.525784  4.525922
2997.0  4.525234  4.525371  4.525509  4.525647  4.525784  4.525922
2998.0  4.525234  4.525371  4.525509  4.525647  4.525784  4.525922
2999.0  4.525234  4.525371  4.525509  4.525647  4.525784  4.525922
3000.0  4.525234  4.525371  4.525509  4.525647  4.525784  4.525922

[3001 rows x 6 columns],)
>>>
```

This shows us that the air velocity at 3000 seconds lies between 4.525234 m/s and 4.525922 m/s in the compressible calculation. These differ from the incompressible result (4.525630 m/s) by less than 0.01%, so we can conclude that they match. This verifies the calculation when vehicles are slower than the air.

11.2 Steady-state comparison with fixed flow at one portal

The routine that calculates pressure from fixed flow at portals (`def FixedVel`) is different to the routine that calculates flow from fixed pressure at portals (`def OpenEnd`).

There is a need to verify that the calculation of traffic drag in both routines is the same.

The way to do this is to take three tunnels in which the traffic drag is the main contributor to the losses:

- The first tunnel has stationary traffic drag and fixed pressure values at its two portals.
- The second tunnel has stationary traffic drag, fixed velocity at the back portal and fixed pressure at the forward portal.
- The third tunnel has stationary traffic drag, fixed velocity at the forward portal and fixed pressure at the back portal.

The file `ok-040-traffic-verification.txt` has three tunnels the same as the tunnel in `ok-039-traffic-verification.txt`. The tunnel named "first" has pressures at its two portals and calculated air velocities of 8.739636 m/s at the back end and 8.742097 m/s at the forward end. The tunnel named "second" has an air velocity at the back portal of 8.739636 m/s and a pressure of -20 Pa at the forward end. The tunnel named "third" has a pressure of $+20$ Pa at the back end and an air velocity at the forward portal of 8.742097 m/s.

If the traffic drag calculation in `def OpenEnd` matches that in `def FixedVel`, the three tunnels will have the same pressure profile.

The following transcript shows how to interrogate the contents of the `.hbn` file to show the velocities and pressures in the three tunnels at the final timestep (5000 seconds).

```
% python
Python 3.9.4 (default, Apr 23 2021, 13:22:46)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> import classHobyah as clH
>>> subf = "verification+validation"
>>> log = open("log_discard.txt", "w")
>>> ok040 = clH.Hobyahdata(subf, "ok-040-traffic-verification.hbn", log)
>>> ok040.PrettyClassPrint("tuns2frames")
Printing the contents of "tuns2frames" in "ok-040-traffic-verification.hbn".
  dict "self.tuns2frames":
     first : [0]
     second : [1]
     third : [2]
Printed the contents of "tuns2frames" only.

()
>>> ok040.v_bin[0].loc[5000.]   # Air velocities at 5000 seconds, has fixed pressures at both ends
0.0      8.739636
40.0     8.740128
80.0     8.740620
120.0    8.741112
160.0    8.741604
200.0    8.742097
Name: 5000.0, dtype: float64
>>> ok040.v_bin[1].loc[5000.]   # Air velocities at 5000 seconds, has fixed velocity at 0.0 m
0.0      8.739636
40.0     8.740128
80.0     8.740620
120.0    8.741112
160.0    8.741604
200.0    8.742097
Name: 5000.0, dtype: float64
>>> ok040.v_bin[2].loc[5000.]   # Air velocities at 5000 seconds, has fixed velocity at 200.0 m
0.0      8.739636
40.0     8.740128
80.0     8.740620
120.0    8.741112
160.0    8.741605
200.0    8.742097
Name: 5000.0, dtype: float64
>>> ok040.p_tot_bin[0].loc[5000.]   # Gauge total pressures at 5000 seconds
0.0      19.992597
40.0     11.993495
80.0      3.993944
120.0    -4.006058
160.0   -12.006509
200.0   -20.007410
Name: 5000.0, dtype: float64
>>> ok040.p_tot_bin[1].loc[5000.]
0.0      19.992598
40.0     11.993495
80.0      3.993944
120.0    -4.006058
160.0   -12.006509
200.0   -20.007410
Name: 5000.0, dtype: float64
>>> ok040.p_tot_bin[2].loc[5000.]
```

```
0.0      19.992597
40.0     11.993495
80.0      3.993942
120.0    -4.006060
160.0   -12.006512
200.0   -20.007413
Name: 5000.0, dtype: float64
>>>
```

The velocities and pressures are the same down to the fifth decimal place, so this proves that the traffic drag calculations in the fixed pressure portal calculation routine (`def OpenEnd`) matches that in the fixed velocity portal calculation (`def FixedVel`).

12 Unverified features

This is the verification to-do list. Each time a feature is implement but is not yet verified, it gets added here. Unimplemented features remain in the list of requirements (Section 2).

1. Friction factor spreadsheet
2. Slug flow spreadsheet
3. Flow split at junctions. The files exist, the test files all demonstrate that tunnel orientation doesn't affect flow split, but the section needs to be written up in this document.
4. Demonstrate that nodes generated from `join` keywords behave exactly like nodes generated from the ends of individual tunnels coming together.
5. Conversion of Hobyah route data into SES route data (both forwards and reversed SES routes) in terms of route gradients/speed limits and plots of stack elevations.
6. Moving traffic slower than air
7. Moving traffic faster than air
8. Jet fans blowing in the direction of air flow
9. Jet fans blowing against the direction of air flow
10. Placement of the outlet of jet fans with non-zero fan carcass length

References

[1] PIARC report 02.05.B, *Vehicle Emissions, Air Demand, Environment, Longitudinal Ventilation*, 1995

[2] *Camatt Guide de l'utilisation – Troisième Édition*, Cetu, 2003

[3] *IDA Tunnel Theoretical Reference*, EQUA Simulation AB, 2019

13 UScustomary.py

`UScustomary.py` contains a dictionary of conversion factors and four functions, as follows:

- `ConvertToUS()`, which takes a value and a key (to the dictionary of conversion factors) and converts the value to US customary units. It returns the units and the units text.
- `ConvertToSI()`, which does the opposite
- `ConversionDetails()`, which takes a key and turns it into a sentence like "converting deg C to DEG F by dividing by 0.555555556 and adding 32".
- `ConversionTest()`, which has a list of values for every conversion factor used and produces a transcript of the conversions that can be checked manually.

The following is the result of calling `ConversionTest()` and asking it to convert from SI values to US values.

```
Key:        "tdiff"    5.0 deg C          = 9.0 DEG F
Key:         "temp"   35.0 deg C          = 95.0 DEG F
Key:    "tempzero1" 0.0055 deg C          = 0.0 DEG F
Key:    "tempzero2"  0.055 deg C          = 0.0 DEG F
Key:    "tempzero3"  0.055 deg C          = 0.0 DEG F
Key:       "press1"  250.0 Pa             = 1.0036592046412995 IWG
Key:       "press2" 101325 Pa             = 29.920940863225965 IN. HG
Key:       "press3"101.325 kPa            = 29.92094086322596 IN. HG
Key:        "dens1"    1.2 kg/m^3         = 0.07491355269137354 LB/CU FT
Key:        "dens2"    1.2 kg/m^3         = 0.07491355269137354 LBS/CUFT
Key:        "dist1" 0.3048 m              = 1.0 FT
Key:        "dist2"  30.48 cm             = 1.0 FT
Key:        "dist3"  304.8 mm             = 1.0 FT
Key:        "dist4" 0.3048 m              = 12.0 IN.
Key:        "dist5"   2.54 cm             = 1.0 IN.
Key:        "dist6"   25.4 mm             = 1.0 IN.
Key:         "area"   20.0 m^2            = 215.27820833419443 SQ FT
Key:        "mass1"  0.454 kg             = 1.000898670319344 LB
Key:        "mass2"  0.454 kg             = 1.000898670319344 LBS
Key:        "mass3"    1.0 tonnes         = 1.1023113109243878 TONS
Key:        "mass4"  907.2 kg             = 1.0000168212706046 TONS
Key:       "speed1"    1.0 m/s            = 196.85039370078738 FPM
Key:       "speed2"   80.0 km/h           = 49.709695378986716 MPH
Key:        "accel"    1.0 m/s^2          = 2.2369362920544025 MPH/SEC
Key:      "volflow"    1.0 m^3/s          = 2118.8800032893155 CFM
Key:     "volflow2"    1.0 m^3/s          = 35.31466672148859 CFS
Key:     "massflow"    1.0 kg/s           = 2.2046226218487757 lb/s
Key:     "IT_watt1"    1.0 W        = 3.4121416331279417 BTU/hr
Key:     "IT_watt2" 1000.0 W        = 0.9478171203133172 BTU/sec
Key:     "IT_kwatt"    1.0 kW       = 3412.141633127942 BTU/hr
Key:     "IT_Mwatt"    1.0 MW       = 3412141.633127942 BTU/hr
Key:     "IT_thcon"    1.0 W/m-K    = 0.5777893165429983 BTU/ft-hr-deg F
Key:  "IT_specheat"    1.0 J/kg-K   = 0.00023884589662749592 BTU/lb-deg F
Key:      "IT_SHTC"    1.0 W/m^2-K  = 0.17611018368230585 BTU/hr-ft^2-deg F
Key:     "IT_SHTC2"    1.0 W/m^2-K  = 4.891949546730718e-05 BTU/sec-ft^2-deg F
```

```
Key:   "IT_wattpua"    1.0 W/m^2    = 8.805509184115292e-05 BTU/sec-ft^2
Key:     "IT_wperm"    1.0 W/m      = 0.0002888946582714991 BTU/sec-ft
Key:    "IT_energy"    1.0 J        = 1055.05585262 BTU
Key:    "v41_watt1"    1.0 W        = 3.415176 BTU/HR
Key:    "v41_watt2" 1000.0 W        = 0.94866 BTU/SEC
Key:    "v41_kwatt"    1.0 kW       = 3415.176 BTU/HR
Key:    "v41_Mwatt"    1.0 MW       = 3415176.0 BTU/HR
Key:    "v41_thcon"    1.0 W/m-K    = 0.578303136 BTU/FT-HR-DEG F
Key: "v41_specheat"    1.0 J/kg-K   = 0.00023905829873566668 BTU/LB-DEG F
Key:     "v41_SHTC"    1.0 W/m^2-K  = 0.17626679585280003 BTU/HR-FT^2-DEG F
Key:    "v41_SHTC2"    1.0 W/m^2-K  = 4.896299884800001e-05 BTU/SEC-FT^2-DEG F
Key:  "v41_wattpua"    1.0 W/m^2    = 8.81333979264e-05 BTU/SEC-FT^2
Key:    "v41_wperm"    1.0 W/m      = 0.000289151568 BTU/SEC-FT
Key:   "v41_energy"    1.0 J        = 1054.1184407480025 BTU
Key:         "diff"    1.0 m^2/s          = 38750.077500155 FT SQUARED/HR
Key:      "Aterm1a"    1.0 N/kg           = 203.94355038311204 LBS/TON
Key:      "Aterm1b"    1.0 N/tonne        = 0.20394355038311202 LBS/TON
Key:       "Aterm2"    1.0 N              = 0.22480928237738218 LBS
Key:      "Bterm1a"    1.0 N/(kg-m/s)     = 91.17092476326717 LBS/TON-MPH
Key:      "Bterm1b"    1.0 N/(kg-km/h)    = 328.21532914775906 LBS/TON-MPH
Key:      "Bterm1c"    1.0 N/(tonne-m/s)  = 91170.9247632664 LBS/TON-MPH
Key:      "Bterm1d"    1.0 N/(tonne-km/h) = 0.328215329147759 LBS/TON-MPH
Key:      "rotmass"   9.65 % tare mass    = 8.8008 (LBS/TON)/(MPH/SEC)
Key:       "momint"    1.0 kg-m^2         = 23.730360404231824 LBS-FT SQUARED
Key:            "W"    1.0 kg/kg          = 1.0 LB/LB
Key:           "RH"   60.0 %              = 60.0 PERCENT
Key:         "perc"   92.0 percent        = 92.0 PERCENT
Key:         "null"    1.0               = 1.0
Key:          "atk"    1.0 gauls          = 16.746881584866482 atk.
Key:       "volume"    1.0 m^3            = 35.31466672148859 ft^3
Key:        "buoys"    1.0 m^2/s^2        = 10.763910416709722 ft^2/s^2
Key:     "nulldash"    1.0 -              = 1.0 -
Key:      "seconds"   60.0 s              = 60.0 s
Key:       "Amotor"    1.0 amps/motor     = 1.0 amps/motor
Key:         "Acar"    1.0 amps/pwd car   = 1.0 amps/pwd car
Key:          "rpm"    1.0 rpm            = 1.0 rpm
```

The following is the result of calling `ConversionTest()` and asking it to convert from US values to SI values.

```
Key:        "tdiff"       9.0 DEG F              = 5.0 deg C
Key:         "temp"      95.0 DEG F              = 35.0 deg C
Key:    "tempzero1"    0.0099 DEG F              = 0.0 deg C
Key:    "tempzero2"     0.099 DEG F              = 0.0 deg C
Key:    "tempzero3"     0.099 DEG F              = 0.0 deg C
Key:       "press1"       1.0 IWG                = 249.08853408 Pa
Key:       "press2"      29.9 IN. HG             = 101254.08535276112 Pa
Key:       "press3"      29.9 IN. HG             = 101.25408535276112 kPa
Key:        "dens1"     0.075 LB/CU FT           = 1.2013847530470103 kg/m^3
Key:        "dens2"     0.075 LBS/CUFT           = 1.2013847530470103 kg/m^3
Key:        "dist1"       1.0 FT                 = 0.3048 m
Key:        "dist2"       1.0 FT                 = 30.48 cm
Key:        "dist3"       1.0 FT                 = 304.8 mm
```

```
Key:        "dist4"      12.0 IN.                = 0.3048 m
Key:        "dist5"       1.0 IN.                = 2.54 cm
Key:        "dist6"       1.0 IN.                = 25.4 mm
Key:         "area"       1.0 SQ FT              = 0.09290304 m^2
Key:        "mass1"       1.0 LB                 = 0.45359237 kg
Key:        "mass2"       1.0 LBS                = 0.45359237 kg
Key:        "mass3"       1.0 TONS               = 0.90718474 tonnes
Key:        "mass4"       1.1 TONS               = 997.903214 kg
Key:       "speed1"    196.85 FPM                = 0.999998 m/s
Key:       "speed2"      50.0 MPH                = 80.4672 km/h
Key:        "accel"      2.24 MPH/SEC            = 1.0013696 m/s^2
Key:      "volflow"   2118.88 CFM                = 0.999999998447616 m^3/s
Key:     "volflow2"    35.315 CFS                = 1.00000943739648 m^3/s
Key:     "massflow"       1.0 lb/s               = 0.45359237 kg/s
Key:     "IT_watt1"     3.415 BTU/hr             = 1.000837704638139 W
Key:     "IT_watt2"     3.415 BTU/sec            = 3603.0157366973 W
Key:     "IT_kwatt"    3415.2 BTU/hr             = 1.0008963188521731 kW
Key:     "IT_Mwatt" 3415176.0 BTU/hr             = 1.0008892851464892 MW
Key:     "IT_thcon"  1.73 BTU/ft-hr-deg F     = 2.994170972822506 W/m-K
Key:  "IT_specheat"   1.0 BTU/lb-deg F        = 4186.8 J/kg-K
Key:      "IT_SHTC"   1.0 BTU/hr-ft^2-deg F   = 5.678263341113487 W/m^2-K
Key:     "IT_SHTC2"   1.0 BTU/sec-ft^2-deg F  = 20441.748028008555 W/m^2-K
Key:   "IT_wattpua"   1.0 BTU/sec-ft^2        = 11356.526682226975 W/m^2
Key:     "IT_wperm"       1.0 BTU/sec-ft          = 3461.469332742782 W/m
Key:    "IT_energy"       1.0 BTU                 = 0.000947817120313317 J
Key:    "v41_watt1"     3.415 BTU/HR              = 0.9999484653206745 W
Key:    "v41_watt2"     3.415 BTU/SEC             = 3599.8144751544287 W
Key:    "v41_kwatt"    3415.2 BTU/HR              = 1.0000070274562716 kW
Key:    "v41_Mwatt" 3415176.0 BTU/HR              = 1.0 MW
Key:    "v41_thcon"  1.73 BTU/FT-HR-DEG F     = 2.9915106668209384 W/m-K
Key: "v41_specheat"   1.0 BTU/LB-DEG F        = 4183.080049045808 J/kg-K
Key:     "v41_SHTC"   1.0 BTU/HR-FT^2-DEG F   = 5.673218232406616 W/m^2-K
Key:    "v41_SHTC2"   1.0 BTU/SEC-FT^2-DEG F  = 20423.585636663818 W/m^2-K
Key:  "v41_wattpua"   1.0 BTU/SEC-FT^2        = 11346.436464813232 W/m^2
Key:    "v41_wperm"       1.0 BTU/SEC-FT          = 3458.393834475073 W/m
Key:   "v41_energy"       1.0 BTU                 = 0.00094866 J
Key:         "diff"       1.0 FT SQUARED/HR       = 2.58064e-05 m^2/s
Key:      "Aterm1a"       1.0 LBS/TON             = 0.0049033176 N/kg
Key:      "Aterm1b"       1.0 LBS/TON             = 4.9033176 N/tonne
Key:       "Aterm2"       1.0 LBS                 = 4.448214902093424 N
Key:      "Bterm1a"       1.0 LBS/TON-MPH   = 0.010968409090909 N/(kg-m/s)
Key:      "Bterm1b"       1.0 LBS/TON-MPH   = 0.003046780303030303 N/(kg-km/h)
Key:      "Bterm1c"       1.0 LBS/TON-MPH   = 1.09684090909e-05 N/(tonne-m/s)
Key:      "Bterm1d"       1.0 LBS/TON-MPH   = 3.046780303030303 N/(tonne-km/h)
Key:      "rotmass"       8.8 (LBS/TON)/(MPH/SEC) = 9.649122807017543 % tare mass
Key:       "momint"       1.0 LBS-FT SQUARED      = 0.042140110093805 kg-m^2
Key:            "W"       1.0 LB/LB               = 1.0 kg/kg
Key:           "RH"      60.0 PERCENT             = 60.0 %
Key:         "perc"      92.0 PERCENT             = 92.0 percent
Key:         "null"       1.0                     = 1.0
Key:          "atk"    16.747 atk.                = 1.0000070708766235 gauls
Key:       "volume"    35.315 ft^3                = 1.00000943739648 m^3
Key:        "buoys"       1.0 ft^2/s^2            = 0.09290304 m^2/s^2
Key:     "nulldash"       1.0 -                   = 1.0 -
```

```
Key:      "seconds"      60.0 s                  = 60.0 s
Key:       "Amotor"       1.0 amps/motor         = 1.0 amps/motor
Key:         "Acar"       1.0 amps/pwd car       = 1.0 amps/pwd car
Key:          "rpm"       1.0 rpm                = 1.0 rpm
```

Manual checking of early versions of these transcripts turned up occasional discrepancies and the discrepancies were corrected. A further pair of manual checks a few months later found no discrepancies.

Most of these are fairly obvious, like `tdiff` (temperature difference) turning 5 °C into 9 °F and `temp` turning 35 °C into 95 °F.

There are a few duplicates (such as densities `mass1` and `mass2`, whose only difference is the units text). The duplicates are needed because `SESconv.py` checks the units text in the SES files it processes and raises an error if the units don't match the units it expects to find. Different SES programmers used different units text for mass (`LB` and `LBS`) and for density (`LB/CU FT` and `LBS/CUFT`).

There are two sets of work and energy terms (e.g. watts). One has the prefix `IT_`, the other has the prefix `v41_`. These two are needed because the definition of the BTU can vary between different programs[5].

The first set are based on the International Tables BTU. The IT BTU was defined at a 1956 thermodynamics international conference, at which they agreed to define one calorie as 4.1868 Joules. This leads to one IT BTU being exactly 1055.05585262 J.

The second set is based on the watts-to-BTU per second conversion in the SES v4.1 source code (the parameter "WTBTUS". This conversion factor seems to be unique to SES (which is not good) but that's what SES uses to convert electrical losses in trains into heat, so that's what is used when converting SES output from BTUs to watts. It is 1054.11844 J, about 0.09% less than the IT BTU.

The differences are discussed in detail in the comments in the script `UScustomary.py` and in the file "SES-notes.pdf" in the documentation folder.

14 Requirements and their validation

The following is the same high-level list of requirements in Section 2. If a requirement ends in a tick it means that I reckon that the requirement has been validated, and a summary of the evidence follows.

1. Trap errors and have informative error messages for each. ✓
   Evidence:
   - Test files exist for most of the error messages, with the error number in each test file's name. The names of error test files are along the lines of `fault-2123-ProcessSettings.txt`, where `2123` is the error number and `ProcessSettings` is the name of the function definition that the error occurs in.

[5] We're lucky to only need two sets: NIST has a US-SI conversion document that lists six different definitions of the BTU in SI units (SP811, 2008).

- Appendix C of the Hobyah User Manual has a transcript of the error messages that can be triggered by input files. In terms of being informative, error numbers 2161, 2308 (which has two forms), 5041/7041 and 2561 also stand out.

2. Some errors don't have test files. Document why those error messages have no test files. ✓
Evidence:

   - There is a routine (`_error-statements.py`) that reads the Python scripts and a transcript that contains the text of all the error messages raised by test files.
     It makes two lists: one of the error numbers in the transcripts and another of the error numbers in the Python scripts. It then points out any discrepancies between them.
   - See Section 15 for a transcript of the output from `_error-statements.py` when every error message that can have a test file has a test file.
   - `_error-statements.py` has a (manually-updated) list of the error numbers that don't have test files. The list is called `no_testfile`.
   - `_error-statements.py` generates a list of the error numbers that don't have test files but aren't listed in `no_testfile`. The script is run every few months during development to ensure that each error either has a test file or a record of the reason why it doesn't have a test file.
   - The code comments around the definition of `no_testfile` in `_error-statements.py` gives the reasons. The following is an edited extract of the first part of the definition:

```
no_testfile = [
# In the Hobyah.py source code:
      2003, # The input file doesn't exist.
      2006, # Raised if the numpy or pandas Python modules are
            # not installed.
      2084, # Can be raised if a new friction factor approximation
            # is in the middle of being added to the code.
      2090, # Raised if the scipy Python module is not installed.
        ]
```

     Hopefully this sample of the comments makes the reasoning clear.

     – Error 2003 can't have a test file because error 2003 complains that the file doesn't exist. Zen.
     – Errors 2006 and 2090 are only raised if optional Python modules are not available on the computer. Not something that can be caused by a test file.
     – Error 2084 is one that can only turn up during program development.

3. Calculate compressible, isothermal airflow in a network of tunnels by the method of characteristics (MoC). ✓
Evidence:

   - A detailed derivation of the method of characteristics is given in the file `MoC.pdf`.
   - `MoC.pdf` also shows how the various body forces and boundary conditions were worked out (friction, portals, etc.). See the following test files:
     – `ok-013-one-tunnel-fixed-P.txt`, which covers portal pressures and friction.

  - `MoC-simple-020.ods`, a spreadsheet that replicates the calculation in `ok-013-one-tunnel-fixed-P.txt` (the spreadsheet was written first and guided the development of the code).
  - `ok-015f-friction-factor-tests.txt`, which covers different approximations of friction factor and constant friction factor.

4. Handle portals with fixed pressure, fixed volume flow and fixed velocity. ✓
   Evidence:
   - `ok-013-one-tunnel-fixed-P.txt`, which covers the calculation of fixed portal pressures.
   - `ok-017d-one-tunnel-fixed-Q.txt`, which covers fixed volume flows at portals.
   - `ok-016d-one-tunnel-fixed-V.txt`, which covers fixed velocities at portals.
   - `kb-001-portal-zetas.txt`, which triggers a known bug (nonconvergence) in the calculation of fixed portal pressures when there are high pressure loss factors at the portals.
5. Handle pressure changes with elevation.
6. Handle area changes in the tunnels, with fixed losses. ✓
   Evidence:
   - `ok-019-three-changes.txt`, which has two tunnels with positive airflow and two with negative airflow. For each flow direction one of the pair of tunnels has fixed pressure loss factors at the changes of sectype and the other has loss factors of zero. The plotted pressure profiles and velocity profiles match closely to incompressible hand calculations.
7. Handle junctions with up to six air paths attached, with fixed losses in each branch. ✓
   Evidence:
   - There are 242 test files comparing different orientations of tunnels at nodes covering two-, three, four-, five- and six-way nodes.
   - Four different arrangements of two-way junctions (specifically back end to back end, back end to forward end, forward end to back end and forward end) with positive airflow and negative airflow have been generated in `ok-024a-two-way-junction.txt` and `ok-024b-two-way-junction.txt`. Each set of four calculations for each airflow direction have identical airflow.
   - Sixteen different arrangements of three-way junctions have been generated in `ok-025a-three-way-junction.txt` to `ok-025p-three-way-junction.txt`. Each of the sixteen calculations has identical airflow.
   - 32 different arrangements of four-way junctions have been generated in `ok-026aa-four-way-junction.txt` to `ok-026bf-four-way-junction.txt`. Each of the 32 calculations has identical airflow.
   - 64 different arrangements of five-way junctions have been generated in `ok-027aa-five-way-junction.txt` to `ok-027cl-five-way-junction.txt`. Each of the 64 calculations has identical airflow.
   - 128 different arrangements of six-way junctions have been generated in `ok-028aa-six-way-junction.txt` to `ok-028ex-six-way-junction.txt`. Each of the 128 calculations has identical airflow.
   - test file `ok-033-nodes-in-parallel.txt` shows a workaround when more than six air paths need to join at one node.

8. Allow junctions to be created partway along the length of tunnels, with up to four tunnel ends socketed into each junction (this makes building networks easier). ✓
Evidence:
   - The program has a `join` keyword that creates a named junction partway along the length of a tunnel. Up to four other tunnels can end at that junction.
   - *Need to create a test file for four tunnels socketed into the side of another and compare it to a case where four tunnel ends meet!*
   - The program has two keywords, `joins` and `tunnelclones`. The first generates multiple joins, the second generates multiple, near-identical tunnels.
   - `ok-037-tunnelclones.txt`, which demonstrates the use of the `joins` and `tunnelclones` keywords.
   - `ok-038-Channel-Tunnel.txt`, which illustrates how the `joins` and `tunnelclones` keywords can make a huge model easy to manage (Channel Tunnel has ≈190 pressure relief ducts over its 50 km length).
9. Handle friction factors in a way that encourages users to learn the difference between Fanning friction factor and Darcy friction factor if they don't already know it. ✓
Evidence:
   - The documentation doesn't use friction factor f (it is ambiguous). Instead, c_f is used for Fanning friction factor and λ is used for Darcy friction factor.
   - The program requires the user to choose which type of friction factor to use in their input file.
   - If a sectype has a roughness height, the program prints values of different types of turbulent friction factor to the `log` file.
   - The file `friction-rant.pdf` is a discussion of the differences between Fanning friction factor and Darcy friction factor, which the user is pointed to in error message 2123.
10. Handle fan characteristics that have stall humps. ✓
Evidence:
   - The file `ok-032-fans-in-parallel.txt` has a characteristic with a stall hump, and illustrates how fans can get stuck on wrong side of it.
11. Handle fan characteristics generated by four points and a spline fit calculation.
12. When a fan duty point goes off the end of a fan characteristic into the reverse flow or freewheeling zones, use linear extrapolation. When plotting on the flow-pressure plane, show the extrapolated lines. ✓
Evidence:
   - The file `ok-031-fans-in-series.txt` shows how fan characteristics behave when they are forced into the reverse flow and freewheel parts of their characteristics.
13. Handle fixed-pitch fans that run at constant speed, have one start time and one stop time. This is for compatibility with SES. ✓
Evidence:
   - The `fan1` keyword defines a fan in a tunnel with fixed speed, one start time and one stop time, with optional entries for runup time and rundown time. This is similar to the functionality in SES forms 7A, 7B and 5C.

- Test file `ok-031-fans-in-series.txt` has fans defined by the `fan2` keyword.

14. Handle fixed-pitch fans that start, stop, change speed and reverse multiple times in each run (set by the run time). ✓
Evidence:
    - The `fan2` keyword defines a fan in a tunnel and takes the name of a datasource and two columns in it to use for time and fan speed.
    - `ok-029-fan-definitions.txt` and `ok-032-fans-in-parallel.txt` are test files that have fans that use the `fan1` keyword.
15. Handle fixed-pitch fans that follow a sequence of starts, stops, speed changes and reversals (triggered by the actions of trains).
16. Handle jet fans that start, stop, change speed and reverse in each run (set by the run time).
17. Handle jet fans that follow a sequence of starts, stops, speed changes and reversals (triggered by the actions of trains).
18. Handle dampers that follow time series of area and k-factors (set by the run time). Both area and k-factors must vary. ✓
Evidence:
    - The `damper1` keyword takes the name of a datasource and the identifiers of four columns in that datasource to use for time, damper area, +ve flow loss coefficient and −ve flow loss coefficient.
    - The `damper2` keyword takes the name of a datasource and the identifiers of three columns in that datasource to use for time, +ve flow Atkinson resistance and −ve flow Atkinson resistance.
    - `ok-029-fan-definitions.txt` and `ok-032-fans-in-parallel.txt` are test files that use the `damper1` and `damper2` keywords.
19. Handle dampers that follow a timed sequence of opening and closing (triggered by the actions of trains). These are platform screen doors.
20. Handle dampers that open (triggered by train location) and close using a different trigger event. These are portal doors.
21. Handle Saccardo nozzles.
22. Handle road traffic drag. ✓
Evidence:
    - The `traffictypes` block allows any number of traffic types to be defined, each with drag properties, lane occupancy and optional speed limits on gradients (think of how much HGVs slow on steep upgrades). See test files `ok-039-stationary-traffic.txt` and `ok-040-moving-traffic.txt` for examples of it.
    - See Section ?.? for a verification calculation (TBC).
23. Handle road traffic drag in a way that makes it easy for data from point-to-point traffic modelling to be imported. ✓
Evidence:
    - The `trafficsteady` block sets road traffic in routes that pass through multiple tunnels. The input is in the form of a table with a column of vehicle flows, one for each route in the block.
    - See test files `ok-034-Hartwell-tunnel.txt` and `ok-040-moving-traffic.txt` for examples of it.

- See Section 1.27 of the User Manual for more details.

24. Distinguish between road traffic routes that share common lanes in a tunnel and road traffic routes that move traffic on different lanes in a tunnel. ✓
Evidence:
    - Multiple `trafficsteady` block can be used. Each sets traffic in one or more routes.
    - Where routes `trafficsteady` block pass through the same tunnel, they share the same lanes.
    - Traffic in routes in different `trafficsteady` block are in different lanes. For example, one `trafficsteady` block could set eastbound traffic and another block could set westbound traffic.
    - See test files `ok-039-stationary-traffic.txt` and `ok-040-moving-traffic.txt` for examples of it.
    - See Sections 1.26 and 3.20 of the User Manual for more details.
25. Calculate the density of stationary road traffic in a tunnel that has traffic from multiple overlapping routes. ✓
Evidence:
    - Test file `ok-039-stationary-traffic.txt` has three routes that put stationary traffic into a tunnel. The calculation of the resulting density of stationary traffic is verified in Section ?.? (TBC).
26. Handle road traffic pollution by PIARC methods.
27. Handle the aerodynamic effects of non-leaky, incompressible trains passing through tunnels.
28. Handle trains at constant speed.
29. Handle trains with predefined speed-time curves and predefined speed-distance curves.
30. Handle trains with preset acceleration and deceleration curves that obey speed limits in routes and stop at predefined station locations.
31. Handle junctions with three air paths and losses that vary with the flow, following the rules used by SES.
32. Allow variable print timesteps. ✓
Evidence:
    - The `plotcontrol` block allows the user to define a list of numbers, which are times to plot at.
    - The numbers can be generated by a chain of `range`, `startstopstep` and `startstepcount` functions that allow the print time to be varied.
    - An example of its use can be seen in `ok-031-fans-in-series.txt`.
33. Allow multiple junctions along the length of a tunnel to be defined in one line of input, with procedurally-generated names. ✓
Evidence:
    - The joins keyword in the `tunnel` block takes a base name and a list of distances at which to place the nodes.
    - Section 3.14.11 of the User manual explains how the keyword is used.
    - Test file `ok-037-tunnelclones.txt` has an example of generating 23 nodes using the `joins` command.

34. Allow multiple tunnels to be defined in one line of input, with procedurally-generated names (this is great for cross-passages). ✓
Evidence:
    - The tunnelclones block takes a base tunnel name, a tunnel definition and two lists (a list of sectypes to apply in each clone and a list of numbers to insert into the base tunnel name for each tunnel and node names at the tunnel ends.
    - Section 3.15 of the User manual explains how the `tunnelclones` block.
    - Test file `ok-037-tunnelclones.txt` has an example of generating 23 cross-passages using the `tunnelclones` command.
35. Plot properties at a fixed point against time. ✓
Evidence:
    - The `transient` curve type takes a fixed location and generates a transient curve for a property there.
    - The file `docfig-plot-property-examples.txt` has examples of every transient plot type that can be plotted from Hobyah output and from SES output.
36. Plot profiles of properties along routes at a given time. ✓
Evidence:
    - The `profile` curve type takes a route name (route number in SES) and generates a curve for a property along the route's length at a given time.
    - The file `docfig-plot-property-examples.txt` has examples of every profile plot type that can be plotted from Hobyah output and from SES output.
37. Plot properties at a moving point against time and against distance.
38. Plot fan characteristics and system characteristics in the flow-pressure plane. ✓
Evidence:
    - The `fandata` curve type generates fan characteristics and system characteristics for a named fan at a given time.
    - The file `docfig-plot-property-examples.txt` has examples of every profile plot type that can be plotted from Hobyah output and from SES output.
    - The files `ok-031-fans-in-series.txt` and `ok-032-fans-in-parallel.txt` have more informative output showing animations of fan characteristics interacting.
39. Plot icons for in-tunnel features like trains, fires and jet fans at the correct locations in routes. ✓
Evidence:
    - The `icons` behaves like a curve type but generates icons along a route.
    - Test file `docfig-icons-example-1.txt` illustrate the plotting of icons from an SES run.
    - Section 3.25.5 of the User Manual shows an example of the input syntax and the output.
40. Allow plotting in SI and US units. ✓
Evidence:
    - The `units` command in the `plots` block, the `page` block and the `graph` block set the units for those entities respectively.

- The test file `ok-012-timeloop.txt` illustrates the use of the `units` keyword, plotting two graphs on the same page in different units.

41. Generate plotted output to `.pdf` files by feeding `.plt` files to Gnuplot. ✓
Evidence:
    - Test file `ok-009.txt` generates `.pdf` file `ok-009.pdf` automatically.
42. Generate multiple individual `.png` files from multi-page `.pdf` files. ✓
Evidence:
    - The `pngtrim` entry in the `plots` block instructs Hobyah to call ImageMagick to generate `.png` files for each page in the `.pdf` file in a subfolder named textttimages.
    - An example of the use of the `pngtrim` command can be found in test file `ok-007.txt`.
43. Allow Gnuplot commands to be written directly to the `.plt` file. ✓
Evidence:
    - The `verbatim` keyword in the `graph` and `image` blocks allows one line of gnuplot input to be written to the `.plt` file.
    - The `begin verbatim...end verbatim` block in the `graph` and `image` blocks allows multiple lines of gnuplot input to be written to the `.plt` file.
    - The files `ok-031-fans-in-series.txt` and `ok-032-fans-in-parallel.txt` have examples of the use of both.
44. Allow plotting from multiple binary files on the same graph. ✓
Evidence:
    - Each curve definition includes the nickname of the file to take its data from.
    - The example file `ok-021-high-P-succeeds.txt` plots data from its own calculation and from the file `ok-020-high-P-fails.txt` on the same graph.
45. Allow plotting from external `.csv` files and from blocks of csv-style data in the input file. ✓
Evidence:
    - The `userdata` curve type in the `graph` block takes the nickname of a datasource and two column numbers (or names) to use as the X and Y values.
    - The Hobyah input file `docfig-BHR-1976-G+P-3c.txt` illustrates plotting from an external `.csv` file.
    - The Hobyah input file `docfig-datasources-illustration.txt` illustrates plotting from a `data` block inside the file.
46. Store run data in binary files and give examples of how to interrogate them. ✓
Evidence:
    - Test file `ok-013-one-tunnel-fixed-P.txt` is a simple example of a Hobyah run with one tunnel. It generates a binary file named `ok-013-one-tunnel-fixed-P.hbn`.
    - Test file `ok-032-fans-in-parallel.txt` has dampers are test files that have dampers that use the `damper1` and `damper2` keyword.
    - Sections 5.3, 10.1
47. Store plotted curve data in text files with traceability information. ✓
Evidence:

- Sections 4.1 and 4.2 of this document show the traceability information in curve data files.

48. Allow the generation of SES input files with the same tunnels, routes and gradients as the Hobyah geometry. ✓
Evidence:
    - The `SESdata` block allows the user to set some SES forms and the program generates SES forms 1, 2, 3, 4, 5, 6, 7 and 8 from those settings and the Hobyah geometry.
    - The `SESpragmat` entry in `tunnel` block tells the program whether to convert the tunnel to line segments or vent segments, what number to start the section/segment numbering from and which route to take the segment stack heights from.
    - Test file `ok-049-write-SES-file.txt` illustrates the use of the `SESdata` block, generating an SES input file with 75 sections and three routes from a Hobyah input file with three tunnels and 23 cross-passages. The plotted output from the Hobyah file compares the route definition in Hobyah to the route definitions generated for the SES input file and the stack elevations in the segments; they match.
49. Make it difficult to write output files over input files if Hobyah and SES files have the same filestem. ✓
Evidence:
    - Binary files generated by Hobyah are given the filename extension `.hbn`. Binary files generated by SES are given the filename extension `.sbn`. If the SES and Hobyah files have the same filestem, the names of their binary files don't clash.
    - SI output files generated by `SESconv.py` are `.txt` files, but the filestem has the text `_ses` appended to it. For example, if you run `SESconv.py` on an SES output file named `foo.ses` it will generate output files named `foo_ses.txt` and `foo.hbn`. So `SESconv.py` won't overwrite a Hobyah input file named `foo.txt`.
    This is not foolproof. Someone could name their SES file `foo.ses` and name their Hobyah input file `foo_ses.txt`. In that case, `SESconv.py` will overwrite the Hobyah input file. Anyone stupid enough to end the name of a Hobyah input file with `_ses.txt` *deserves* to have `SESconv.py` overwrite their Hobyah input file.

15 Check of the error messages

The following is a transcript written after running the `_error-statements.py` script.

```
The lists below are grouped into ranges and give the name of the
Python file and the function definition that they appear in.
This is an auto-generated list.

Error count in Hobyah.py: 284
Error count in generics.py: 15
Error count in syntax.py: 15
Error count in SESconv.py: 40
```

```
Error count in UScustomary.py: 3
Error count in classSES.py: 63
Error count in classHobyah.py: 34

1001 to 1999   Fault messages for situations which usually occur only
               during code development.  These send a line to the
               screen with a message to tell the user to raise a bug
               report (for when one of them slips through into code
               uploaded to Github).  Then the program exits.

    1021 to 1022    generics.Enth

    1023            generics.ColumnText

    1024            generics.Interpolate

    1025            generics.OpenCSV

    1026            generics.ShoeHornText

    1027 to 1028    generics.CheckNanInf

    1041 to 1044    generics.GetOptionals

    1061 to 1063    generics.WordOrPhrase

    1101            UScustomary.ConvertToUS

    1102            UScustomary.ConvertToSI

    1103            UScustomary.ConversionTest

    1201            Hobyah.CheckRangeAndSI

    1202            Hobyah.RaiseDudSpec

    1203            Hobyah.CheckRangeAndSI


2001 to 4999   Fault calls in the Hobyah program and the syntax checker.

    2001 to 2006    Hobyah.ProcessFile

    2021 to 2022    syntax.CheckBeginEnd

    2023 to 2031    syntax.CheckClosures

    2041            syntax.SplitBlocks

    2042            syntax.GetOneBlock

    2043            syntax.CheckSubDictClashes

    2044            syntax.CheckNestables
```

```
2061 to 2064    Hobyah.GetBegins

2081 to 2090    Hobyah.ProcessCalc

2101            Hobyah.ProcessBlock

2102 to 2103    Hobyah.RaiseTooFew

2104 to 2109    Hobyah.ProcessBlock

2110            Hobyah.Raise2110

2111 to 2113    Hobyah.ProcessBlock

2121 to 2126    Hobyah.ProcessSettings

2141 to 2150    Hobyah.ProcessPlotFiles

2181 to 2184    Hobyah.ProcessConstants

2201 to 2210    Hobyah.ProcessTunnel

2221 to 2227    Hobyah.CheckRangeAndSI

2241 to 2251    Hobyah.ProcessUserData

2261 to 2268    Hobyah.ProcessCsvData

2281 to 2282    Hobyah.ProcessTimeLoop

2301 to 2320    Hobyah.ProcessRoute

2341 to 2351    Hobyah.ProcessNumberList

2361 to 2362    Hobyah.CheckGreater

2363 to 2366    Hobyah.CheckForConstant2

2381 to 2397    Hobyah.CheckListAndRange

2401            Hobyah.ProcessPlotControl

2421            Hobyah.ProcessSectypes

2441            Hobyah.ZetaClash

2461 to 2464    Hobyah.ProcessFanChar

2481 to 2485    Hobyah.GetTabulated

2501 to 2503    Hobyah.BuildFanChar

2521            Hobyah.GetCharData
```

```
    2541 to 2543    Hobyah.CheckTimesIncrease

    2561            Hobyah.CheckEntityName

    2581            Hobyah.AllPositive

    2582            Hobyah.NotNegative

    2621 to 2625    Hobyah.ProcessSchedule

    2661            Hobyah.CheckForPortals

    2681 to 2687    Hobyah.ProcessClones

    2701            Hobyah.CheckAsterisks

    2721 to 2727    Hobyah.ProcessVehTypes

    2741 to 2745    Hobyah.ProcessTraffic1

    2761            Hobyah.ProcessJFTypes

    2841            Hobyah.TrafficInTunnels

    2861 to 2862    Hobyah.DoRoutesMatch

    2881 to 2884    Hobyah.DoSpeedsMatch

    2901            Hobyah.CheckRouteChs

    2921 to 2923    Hobyah.CheckJFThrust

    2941 to 2945    Hobyah.CheckJFBanks

    2961            Hobyah.JFOutletCheck

    2981 to 2983    Hobyah.ProcessFilesLoop

    3001 to 3004    Hobyah.TrafficClash

    3021 to 3022    Hobyah.MovingSpeeds

    3041 to 3043    Hobyah.StationaryDists

    3061 to 3075    Hobyah.ProcessSESData

5001 to 5999   Fault calls in the classSES module.

    5001 to 5008    classSES._ReadSESData

    5021 to 5024    classSES._UnPickleData
```

```
    5041 to 5045    classSES.CheckProperty

    5061 to 5069    classSES._CheckAt

    5081 to 5098    classSES.TransientAt

    5101            classSES._CheckSegNumber

    5121            classSES._CheckSubNumber

    5141            classSES._CheckSecNumber

    5161 to 5162    classSES._CheckRouteNumber

    5181 to 5186    classSES.WriteInputFile

    5201            classSES._ColHeaders

    5221            classSES.FanAt

    5241 to 5245    classSES._CheckZoneStuff

    5261            classSES.RouteAt

6001 to 6999   Fault calls in the plotting routines.

    6001            Hobyah.OpenPltFile

    6002 to 6005    Hobyah.ProcessPlots

    6021            Hobyah.ProcessGraph

    6041 to 6046    Hobyah.ProcessCurves

    6061 to 6064    Hobyah.ReadBinData

    6121 to 6124    Hobyah.SubstituteAt

    6141 to 6143    Hobyah.PickUserData

    6161 to 6164    Hobyah.CheckGnuplotName

    6181 to 6191    Hobyah.ProcessImage

7001 to 7999   Fault calls in the classHobyah module.

    7001 to 7008    classHobyah._ReadHobyahData

    7021 to 7024    classHobyah._UnPickleData

    7041 to 7043    classHobyah.CheckProperty
```

```
    7061 to 7068    classHobyah._CheckAt

    7081 to 7087    classHobyah.TransientAt

    7101            classHobyah._ColHeaders

    7121            classHobyah.FindSegment

    7141 to 7142    classHobyah.FanAt


8001 to 8999   Fault calls in the SESconv program.

    8001 to 8007    SESconv.ProcessFile

    8008            SESconv.ReadTimeSteps

    8022 to 8024    SESconv.FilterJunk

    8025            SESconv.Form1A

    8026 to 8029    SESconv.Form1B

    8030 to 8033    SESconv.ProcessFile

    8041            SESconv.GetValidLine

    8061 to 8067    SESconv.GetReal

    8068            SESconv.GetInt

    8069            SESconv.ConvTwo

    8101            SESconv.CountEntries

    8121            SESconv.AddSecData

    8141            SESconv.ConvertAndChangeOne

    8161            SESconv.Form6

    8181 to 8183    SESconv.ReadSegments

    8201 to 8203    SESconv.GetOpenSESData

    8221            SESconv.Form3

    8241            SESconv.main

    8261            SESconv.Form4

The following 64 errors do not have test files (as intended):
>   1021, 1022, 1023, 1024, 1025, 1026, 1101, 1102, 1103, 2006, 2083,
>   2084, 2090, 2343, 2344, 2347, 2348, 2349, 2362, 5001, 5007, 5008,
```

```
>   5021, 5042, 5044, 5061, 5062, 5063, 5064, 5082, 5083, 5086, 5090,
>   5093, 5096, 5097, 5098, 5181, 5182, 5183, 5184, 5185, 5186, 5201,
>   5221, 6002, 6003, 6004, 6005, 6045, 6124, 7001, 7007, 7008, 7061,
>   7062, 7063, 7064, 7084, 7086, 7101, 8004, 8008 and 8241.
Files exist to trigger these 387 errors:
>   1027, 1028, 1041, 1042, 1043, 1044, 1061, 1062, 1063, 1201, 1202, 1203,
>   2001, 2002, 2003, 2004, 2005, 2021, 2022, 2023, 2024, 2025, 2026, 2027,
>   2028, 2029, 2030, 2031, 2041, 2042, 2043, 2044, 2061, 2062, 2063, 2064,
>   2081, 2082, 2085, 2086, 2087, 2088, 2089, 2101, 2102, 2103, 2104, 2105,
>   2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2121, 2122, 2123, 2124,
>   2125, 2126, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150,
>   2181, 2182, 2183, 2184, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208,
>   2209, 2210, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2241, 2242, 2243,
>   2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2261, 2262, 2263, 2264,
>   2265, 2266, 2267, 2268, 2281, 2282, 2301, 2302, 2303, 2304, 2305, 2306,
>   2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318,
>   2319, 2320, 2341, 2342, 2345, 2346, 2350, 2351, 2361, 2363, 2364, 2365,
>   2366, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391,
>   2392, 2393, 2394, 2395, 2396, 2397, 2401, 2421, 2441, 2461, 2462, 2463,
>   2464, 2481, 2482, 2483, 2484, 2485, 2501, 2502, 2503, 2521, 2541, 2542,
>   2543, 2561, 2581, 2582, 2621, 2622, 2623, 2624, 2625, 2661, 2681, 2682,
>   2683, 2684, 2685, 2686, 2687, 2701, 2721, 2722, 2723, 2724, 2725, 2726,
>   2727, 2741, 2742, 2743, 2744, 2745, 2761, 2841, 2861, 2862, 2881, 2882,
>   2883, 2884, 2901, 2921, 2922, 2923, 2941, 2942, 2943, 2944, 2945, 2961,
>   2981, 2982, 2983, 3001, 3002, 3003, 3004, 3021, 3022, 3041, 3042, 3043,
>   3061, 3062, 3063, 3064, 3065, 3066, 3067, 3068, 3069, 3070, 3071, 3072,
>   3073, 3074, 3075, 5002, 5003, 5004, 5005, 5006, 5022, 5023, 5024, 5041,
>   5043, 5045, 5065, 5066, 5067, 5068, 5069, 5081, 5084, 5085, 5087, 5088,
>   5089, 5091, 5092, 5094, 5095, 5101, 5121, 5141, 5161, 5162, 5241, 5242,
>   5243, 5244, 5245, 5261, 6001, 6021, 6041, 6043, 6044, 6046, 6061, 6062,
>   6063, 6064, 6121, 6122, 6123, 6141, 6142, 6143, 6161, 6162, 6163, 6164,
>   6181, 6182, 6183, 6184, 6185, 6186, 6187, 6188, 6189, 6190, 6191, 7002,
>   7003, 7004, 7005, 7006, 7021, 7022, 7023, 7024, 7041, 7042, 7043, 7065,
>   7066, 7067, 7068, 7081, 7082, 7083, 7085, 7087, 7121, 7141, 7142, 8001,
>   8002, 8003, 8005, 8006, 8007, 8022, 8023, 8024, 8025, 8026, 8027, 8028,
>   8029, 8030, 8031, 8032, 8033, 8041, 8061, 8062, 8063, 8064, 8065, 8066,
>   8067, 8068, 8069, 8101, 8121, 8141, 8161, 8181, 8182, 8183, 8201, 8202,
>   8203, 8221 and 8261.
```