BridgeTrafficLoadSim:

Long Run Simulation Model

for Bridge Loading

Version 1.0.0

User Manual

Dr Colin Caprani

Dublin Institute of Technology

Acknowledgements

This program is based on three separate programs developed as part of the author’s PhD research from 2001. In turn, these were based on work by Dr Samuel Grave, former PhD student at Trinity College Dublin. The current program which encompasses the functions of two of the previous programs has evolved since 2007 and has been much influenced by the work of Dr Bernard Enright, DIT.

For further information, please contact colin.caprani@dit.ie.

Dr Colin Caprani,

July 2012

Contents

1. Introduction

1.1 The BridgeTrafficLoadSim Program

BridgeTrafficLoadSim will be referred to as BTLS hereafter.

BTLS generates artificial traffic and passes it across bridges determining various load effects. The traffic is generated according to a relatively simple model. There are several built-in influence lines for various load effects, but the user can input their own influence line also. The program outputs various quantities of interest, which are controllable by the user.

These programs have been in use in DIT and University College Dublin over a 10-year period. There have been multiple users, and so exhibit a fair degree of maturity. That is, the user should not get unexpected results when used correctly. Whilst the routines have been thoroughly tested many times, with ongoing changes, it is good practice to satisfy oneself as to the accuracy of the programs. Various forms of output should assist with this process.

1.2 The User Manual

Purpose

This User Manual has been written to explain the use of the BTLS program, and to explain its capabilities and limitations.

Notices

Points of significant importance are denoted as:

Important!

Typically, failure to adhere to these points will result in unexpected behaviour or a program crash.

Glossary

Load Effect	The result of a calculation using any influence line. Total load on the bridge is sometimes referred to as a load effect therefore.

1.3 Release History

BridgeTrafficLoadSim Program

Version	Date	Description
1.0.0	??	• Initial release to international users
		•

BridgeTrafficLoadSim Manual

Version	Date	Description
1.0	??	Initial release.

1.4 Installing the Program

BTLS does not require installation, it is a standalone executable program. It has been tested on Windows XP, Vista, and Windows 7.

BTLS can be run in a 64 bit version, which is more efficient that the 32 bit version. The program is a single- threaded application and so cannot take advantage of multi-core processors. Therefore for maximum speed, prefer a computer with a fast single processor over a computer with a multi-core slower processor.

The program does not require much memory because:

- Only a small amount of input information is held in memory;
- Traffic is generated in 1-day blocks on a rolling basis;
- Output to file is made according to user input: this balances accessing eh hard drive (which is slow) and memory requirements. Prefer to use memory than output to the hard drive often.

A computer with 1 GB of RAM is sufficient and other programs can continue to operate successfully.

BTLS uses two folders to operate:

- **Working folder**: the current folder in which configuration files and the executable exist;
- **Traffic folder**: a folder on the computer in which the traffic characteristics for sites are stored.

2. About BTLS

2.1 Introduction

BTLS performs efficient calculations of static traffic actions on bridges. It can generate artificial traffic and write it to file. It can calculate load effects from traffic files. However such simulations are limited by the file size that can be held in the computer memory. In an alternate mode, it can generate traffic and determine load effects simultaneously without recourse outputting traffic to file. In this mode the program can simulate 100s of years of load effect data quite quickly. The exact speed depends on many parameters, but in the worst case, 1000 years has been simulated in under 20 hours. For more typical cases 100 years takes about 1 hour to simulate.

BTLS is provided in two versions:

- `BridgeTrafficLoadSim.exe`: the 32-bit version.
- `BridgeTrafficLoadSim_x64.exe`: the 64-bit version, found to run faster than 32-bit version on 64-bit machines.

2.2 BTLS: Capabilities and Limitations

Capabilities

BTLS is able to:

- Generate artificial traffic from the traffic model of Caprani (2005 & 2012).
- Read in traffic and pass it over influence lines;
- Use lane factors to account for lateral distribution of load effect due to transverse stiffness of the bridge;
- Determine static load effects from generated or read-in traffic passing over defined bridges and either user-defined or built-in influence lines;
- Model one or two directions at the same time, with any number of lanes in each direction;
- Output different types of data for debugging and further analysis, as specified by the user.

Future plans include:

- Fatigue calculation;
- Output data suitable for peaks over threshold analysis;
- Improved traffic model for greater generality;
- Possibly a visual user input interface.

Limitations

BTLS is not able to:

- Generate trucks with more than 5 axles;
- Determine the number of lanes or directions of traffic in the specified input traffic file, in advance of a simulation;
- Determine the input traffic file format (whether CASTOR or BeDIT);
- Determine dynamic load effects;
- Perform extrapolations for return periods.

Note that cars are assumed to be 4 m long and have GVW of 2 tonnes evenly distributed to each axle.

3. BTLS Input

3.1 Introduction

BTLS operates in one of three modes, which are numbered:

1. ***Gen & Sim***: In this mode traffic is generated in the program and simulated crossing the defined bridges;
2. ***Gen***: In this mode traffic is generated and output to file;
3. ***Read & Sim***: In this mode traffic is read from a file and simulated crossing the defined bridges.

Different types of input are required:

1. Traffic model files: for the generation of artificial random traffic;
2. Configuration file: the main user input file which configures each run of the program;
3. Supporting files: define bridges, influence lines and traffic flows to be used in the simulation.

Thus for a successful run the files required are:

Location	**Files**
`C:\Traffic\`[The Site]	Several – see Traffic Data input files section
Working Folder (anywhere on computer)	`BridgeTrafficLoadSim.exe` (or 64 bit version) `BTLSin.txt` Lane flow definition file Bridge definition file Influence line definition file

3.2 Traffic Files

The model describing the physical characteristics of the traffic is defined in a series of files located in a folder, named after the site which is located, which is a sub-folder to `C:\Traffic\`.

Important!

The traffic folder must reside at: "`C:\Traffic\`".

The traffic model is described by Caprani (2005), and is based on Grave (2001). Presently, 13 sites have been modelled accordingly and are indexed by BTLS as follows, and are located in sub-folders of the site name below:

Site Indices	
Index	**Site**
1	Angers
2	Auxerre
3	A196
4	B224
5	A296
6	SAMARIS\D1
7	SAMARIS\D2
8	SAMARIS\D3
9	SAMARIS\S1
10	SAMARIS\S2
11	SAMARIS\S3
12	SAMARIS\D
13	SAMARIS\S

Auxerre is particularly important as the Eurocode load model LM1 was initially calibrated upon this traffic.

Traffic Data Input Files

The files then placed in this folder are of type comma separated values (`*.csv`). These file types are easily created in a spread sheet program, but can also be read or edited in a text editor.

Many of the vehicles properties are modelled with a three-mode normal distribution; that is, the data may be multi-modally normally distributed. There are three parameters required for each of the modes: the weight, ρ; the mean, μ and the standard deviation, σ. The maximum number of modes allowed for is three; hence the 3×3 tabular format of the data. The units of the data are as per the traffic file convention explained in the Output section.

Important!

The files names must be as given for each modelled property.

The input defining the traffic flow and composition is made in the working folder as the executable.

Important!

The current traffic model only accounts for vehicles with up to 5 axles.

Axle Spacing Definition

```
Asall.csv
```

This file stores the axle spacing data for all classes of trucks measured at the site. The values must be separated by commas. An example is:

```
1,50.7,3.7,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0

0.65,34.1,6.9,1,11.5,1.7,0,0,0,0,0,0
0.268,34,1.5,0,0,0,0,0,0,0,0,0
0.082,61.5,6,0,0,0,0,0,0,0,0,0

0.672,30.6,1.5,0.153,34.7,3,0.317,11.8,0.6,0,0,0
0.328,30.2,3.9,0.386,54.8,8.6,0.598,12.1,1.7,0,0,0
0,0,0,0.461,59.5,3.4,0.085,18.3,0.9,0,0,0

0.041,23.2,1.4,0.133,42,5.6,1,10.9,1.7,1,11,1.7
0.959,30.4,1.8,0.867,51.2,3.4,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0
```

This data may be more easily understood viewed in tabular form. The meaning of the rows and columns is also shown in relation to the ti-mode normal distribution adopted.

Class	Line	Spacing 1-2			Spacing 2-3			Spacing 3-4			Spacing 4-5		
		ρ	μ	σ	ρ	μ	σ	ρ	μ	σ	ρ	μ	σ
2-Axle	1	1	50.7	3.7	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	0	0	0	0
	4												
3-Axle	5	0.65	34.1	6.9	1	11.5	1.7	0	0	0	0	0	0
	6	0.268	34	1.5	0	0	0	0	0	0	0	0	0
	7	0.082	61.5	6	0	0	0	0	0	0	0	0	0
	8												
4-Axle	9	0.672	30.6	1.5	0.153	34.7	3	0.317	11.8	0.6	0	0	0
	10	0.328	30.2	3.9	0.386	54.8	8.6	0.598	12.1	1.7	0	0	0
	11	0	0	0	0.461	59.5	3.4	0.085	18.3	0.9	0	0	0
	12												
5-Axle	13	0.041	23.2	1.4	0.133	42	5.6	1	10.9	1.7	1	11	1.7
	14	0.959	30.4	1.8	0.867	51.2	3.4	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0

Axle Weights

```
Aw2&3.csv
```

Two files are used, one for 2 and 3 axle trucks, the other for 4 and 5 axle trucks. This file contains the axle weight information for the 2- and 3- axle trucks of the site. An example is:

```
0.560,33.4,3.7,0.440,59.4,7.4,0.000,0.0,0.0
0.440,40.6,7.4,0.560,66.6,3.7,0.000,0.0,0.0
0.000,0.0,0.0,0.000,0.0,0.0,0.000,0.0,0.0

0.066,20.4,1.5,0.769,34.6,6.8,0.558,30.5,5.9
0.522,26.0,4.9,0.227,39.2,2.2,0.442,37.7,3.5
0.412,38.7,8.6,0.004,54.4,3.7,0.000,0.0,0.0
```

This data is explained as follows:

Class	**Row**	**Weight Axle 1**			**Weight Axle 2**			**Weight Axle 3**		
		ρ	μ	σ	ρ	μ	σ	ρ	μ	σ
2-Axle	1	0.56	33.4	3.7	0.44	59.4	7.4	0	0	0
	2	0.44	40.6	7.4	0.56	66.6	3.7	0	0	0
	3	0	0	0	0	0	0	0	0	0
	4									
3-Axle	5	0.066	20.4	1.5	0.769	34.6	6.8	0.558	30.5	5.9
	6	0.522	26	4.9	0.227	39.2	2.2	0.442	37.7	3.5
	7	0.412	38.7	8.6	0.004	54.4	3.7	0	0	0

```
Aw4&5.csv
```

This file contains the axle weight information for the 4- and 5-axle trucks. It has been found that the axle weights of the 4- and 5-axle trucks depend on the Gross Vehicle Weight (GVW). Thus the data governing these axle weights have been assembled for 12 classes of truck GVW, beginning at 25 kN and increasing in steps of 50 kN.

```
0.0,0.0,0.0,0.0,0.0,0.0
20.9,39.8,39.3,5.2,6.9,7.3
25.6,36.5,38.0,5.4,4.8,5.7
23.9,35.5,40.7,4.3,4.6,5.2
20.3,36.1,43.6,3.6,4.6,5.4
17.4,34.9,47.7,3.0,4.1,5.5
14.8,33.4,51.8,2.1,3.1,4.1
14.5,33.6,51.9,1.5,2.6,3.2
13.9,32.4,53.7,1.3,2.3,3.1
11.9,31.4,56.7,0.9,1.4,0.9
0.0,0.0,0.0,0.0,0.0,0.0
0.0,0.0,0.0,0.0,0.0,0.0

0.0,0.0,0.0,0.0,0.0,0.0
0.0,0.0,0.0,0.0,0.0,0.0
19.1,36.5,44.5,6.0,7.4,7.2
23.6,32.8,43.7,4.6,4.2,5.0
21.4,33.4,45.3,3.2,4.8,5.4
18.1,33.8,48.1,2.4,4.5,5.5
15.7,32.3,52.0,1.8,3.8,4.7
14.3,31.0,54.6,1.5,3.3,3.9
13.4,29.6,57.1,1.2,2.9,3.4
12.7,27.7,59.6,1.0,2.7,3.1
0.0,0.0,0.0,0.0,0.0,0.0
0.0,0.0,0.0,0.0,0.0,0.0
```

A single line separates the 4- and 5-axle data. The six entries for each line, or GVW range of truck, represent the parameters of the single-mode Normal distributions for the first (W1) and second (W2) axles and the total weight of the tandem or tridem (WT) in the following order:

Mean W1	Mean W2	Mean WT	SD W1	SD W2	SD WT

This has resulted from previous research which has found that the weights of the axles in the tandem or tridem of 4- and 5-axle trucks (respectively) are equal and thus the tandem/tridem may be considered as one weight. The calculated tandem/tridem weight are divided by the number of axles to give each axle a weight in the processing of this data. The values must be separated by commas.

Gross Vehicle Weight

```
GVWpdf.csv
```

This file holds the parameters of the distributions that characterize the GVW and speed of each class of truck for both directions. An example of this file is:

```
1,194.5,27.4,0.152,44.2,6.5,0.069,51.2,9.7,0.583,231.1,61.9,0.274,199.9,36.7
0,0,0,0.395,76.4,20.7,0.887,166.3,53.2,0.24,176.6,29.6,0.553,308.7,49.9
0,0,0,0.453,117.4,30.5,0.044,268.4,34.7,0.177,331,30.1,0.173,383.2,35.4

1,181.1,22.4,0.143,46.5,8,0.093,56.4,12.4,0.493,243.6,64.6,0.16,205.3,40.1
0,0,0,0.524,82.9,23.8,0.653,141.5,31.1,0.301,162.1,28.8,0.441,300.6,53.6
0,0,0,0.333,132.3,31.8,0.254,218.5,33.4,0.206,361.9,31.6,0.399,400.4,35.9
```

Again this is best explained by reference to the following table:

	Speed	**2-Axle GVW**	**3-Axle GVW**	**4-Axle GVW**	**5-Axle GVW**
Direction 1	3×3	3×3	3×3	3×3	3×3
Blank Line					
Direction 2	3×3	3×3	3×3	3×3	3×3

In the above table the entry 3×3 refers to the allowance for multi-modal distributions (up to a maximum of three modes) and includes, for each mode, the weight, mean and standard deviation, as explained previously. The values must be separated by commas.

Headway

```
NHM.csv
```

Of the headway models, only the HeDS model requires an input file. This model is defined in OBrien & Caprani (2005). An example is:

```
15,0,0,0
0,0.011855673,-0.014268241,0.004048786
0,0.039251526,-0.05978246,0.02212043
70,-0.004412997,0.054824101,-0.066907905
80,-0.004685721,0.052127816,-0.053475193
90,0.001537014,0.020896587,-0.013787689
100,-0.003853623,0.064555837,-0.069172155
110,-0.002530238,0.054511802,-0.059714977
120,-0.001307981,0.048010242,-0.051645258
130,-0.000487752,0.049738587,-0.057875119
140,-0.004995115,0.081041256,-0.086465967
150,-0.004547469,0.080310658,-0.083351351
160,-0.004938412,0.092219287,-0.105416601
170,-0.005000644,0.086893379,-0.097048852
180,0.001987438,0.052114614,-0.058245039
190,0.003366332,0.044909211,-0.063187142
210,0.000379907,0.068461437,-0.077769612
230,-0.006466786,0.117770005,-0.141174818
```

Line 1 indicates the number of flow-dependent headway models (always less than, or equal to, 24). Lines 2 and 3 give the parameters of the quadratic-fit headway cdf for under 1.0 s and between 1.0 s and 1.5 s respectively. The following lines (of number 15 in this example, from Line 1), return the parameters of the quadratic fit to the headway cdf for that flow (trucks per hour) of the first column. The values must be separated by commas.

3.3 Configuration File

The user interacts with the program through the configuration file.

Important!

The input file must be called “`BTLSin.txt`” and it must be in the working folder.

An example input file is shown next, and each input line explained following.

Line	BTLSin.txt
	`// -------------------------------------------`
	`// START OF BRIDGE TRAFFIC LOAD SIMULATION INPUT`
	`// -------------------------------------------`
	`//`
	`// Program Mode (1 - Gen & Sim, 2 - Gen, 3 - Read & Sim)`
1	`1`
	`//`
	`// TRAFFIC GENERATION PARAMETERS`
	`// -------------------------------------------`
	`// No. of days of traffic simulation:`
2	`100`
	`// Site weight data to be used:`
3	`2`
	`// Headway model to be used:`
	`// (0 - Auxerre NHM, 5 - Congestion (w/ or w/out cars), 6 - free-flow, cars included)`
4	`0`
	`// Lane and flow definition file:`
5	`LaneFlowData_0.csv`
	`// Nominal congested spacing, front to back (m):`
6	`5`
	`// Congested speed (km/h):`
7	`30`
	`// Congested gaps coefficient of variation:`
8	`0.05`
	`//`
	`// TRAFFIC INPUT FILE PARAMETERS`
	`// -------------------------------------------`
	`// Traffic input file to be analysed:`
9	`traffic.txt`
	`// Traffic input file format (CASTOR - 1, BeDIT - 2):`
10	`1`

```
      // Impose constant speed on all vehicles (1 or 0):
 11   0
      // Use average speed of vehicles in file if constant speed
      imposed (1 or 0)
 12   1
      // Constant speed of vehicles if not average used (km/h):
 13   80
      //
      // LOAD EFFECT CALCULATION PARAMETERS
      // ------------------------------------------------
      // Bridge definition file:
 14   68_bridge.txt
      // Influence Line definition file:
 15   IL_68.txt
      // Time step (s):
 16   0.1
      // Block size for maxima (days):
 17   1
      // Block size for maxima (seconds):
 18   0
      // Minimum GVW for inclusion in calculations (t/10):
 19   35
      //
      // OUTPUT PARAMETERS
      // ------------------------------------------------
      // Write vehicle file (1 or 0)
      // WARNING: a large file may result in long-run simulations
 20   0
      // Vehicle file name
 21   BTLSvehicles.txt
      // Vehicle file buffer size
 22   10000
      // Write full time history - slow & large file (1 or 0):
 23   1
      // Write each loading event value (1 or 0):
 24   1
      // Write each event buffer size:
 25   10000
      // Write block max vehicle files (1 or 0):
 26   1
      // Write block max summary files (1 or 0):
 27   1
      // Write block max buffer size:
 28   100
      // ------------------------------------------------
      // END OF BRIDGE TRAFFIC LOAD SIMULATION INPUT
      // ------------------------------------------------
```

\\ Comments:

The program reads all lines of the configuration file except those preceded with C++ style commenting: “\\”. The user is free to add further commenting to the file as they wish, once the order of the input variables is not altered.

Important!

Depending on the program mode, some inputs are redundant. However, they must still be specified as ‘placeholders’ to keep the order of inputs the same.

Line 1:

The user specifies the program mode using 1, 2, or 3 for the modes as defined on page 14.

Line 2:

Specify the number of days of traffic to simulate in Modes 1 or 2. This input is redundant in the case of Mode 3 when the traffic file is specified.

Line 3:

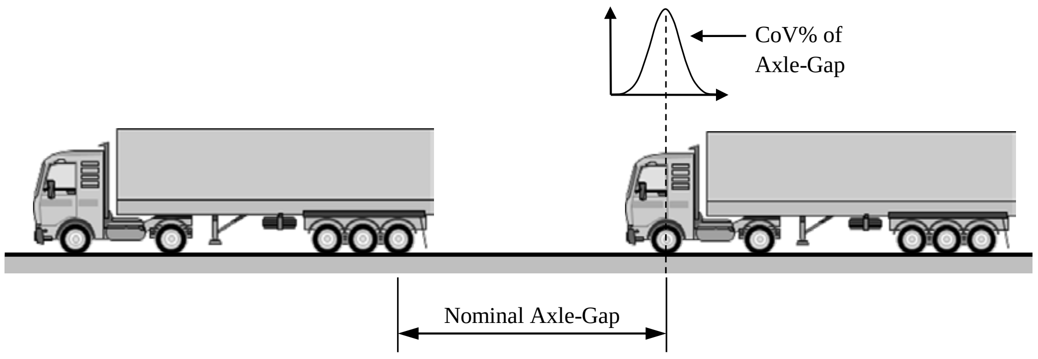
Specify the index of the site in the Traffic folder on which the traffic physical properties will be modelled. At present these are hardcoded into the program as detailed in the Traffic Files section.

Line 4:

Specify the headway model to be used in the generation of artificial traffic. The options are (note the odd-numbering for ‘historical’ reasons):

- “0” – The HeDS (Headway Distribution Statistics) model of OBrien & Caprani (2005). This is suitable for the Auxerre site-measured flowrates only. It is a free-flow model that generates only trucks.

- "5" – Congestion model as per Caprani (2012), summarized in the following diagram. A nominal axle gap is specified (Line 6), along with a coefficient of variation between successive vehicles in all lanes (i.e. trucks and cars) (Line 8) and gaps are then generated using a normal distribution.



- "6" – Free-flow model which uses a Poisson arrival assumption based upon the Normalized Headway Model of Crespo-Minguillón and Casas (1997). This accounts for different flow rates (Q) as follows:

$$F(t) = 1 - \mathrm{e}^{-\lambda t}$$

  Where $1/\lambda$ is the mean headway, i.e. the average time gap between vehicles in the hour of the current total (cars and trucks) flowrate Q and so is given by $Q/3600$.

For all models the program checks that no overlapping of vehicles can occur by ensuring the generated gap is greater than the required minimum gap (taking account of the maximum bridge length, vehicle lengths, and speed difference between them).

Line 5:

Specify the name of the Lane Flow definition file that is in the working folder.

Line 6:

Specify the nominal axle gap for Headway Model 5 – congestion.

Line 7:

Specify the speed of all vehicles for Headway Model 5 – congestion. Note that this, combined with the calculation time step (Line 16), effectively renders a distance-stepping algorithm and so this speed can be notional to achieve a required distance step.

Line 8:

Specify the coefficient of variation of the nominal congested gap for Headway Model 5 – congestion.

Line 9:

Specify the name of the traffic input file in the working folder for Program Mode 3.

Line 10:

Specify the format of the input file: 1 – CASTOR format, 2 – BeDIT format. See Appendix for traffic file format definitions.

Line 11:

BTLS normally passes each vehicle across the bridge according to its own speed when in Program Mode 3. Setting this option to “1” imposes constant speed on all vehicles for comparison with some other algorithms – mostly to do with congestion traffic files.

Line 12:

When constant speed is imposed (Line 11), if this option is “1” then the average speed of all vehicles in the file will be used, otherwise the speed specified in Line 13 will be used.

Line 13:

Specifies the constant speed if Line 11 is “1” and Line 12 is “0”.

Line 14:

Specify the name of the bridge definition file in the working folder.

Line 15:

Specify the name of the influence line definition file in the working folder.

Line 16:

Specify the calculation time step which is used in passing the vehicles over the bridges. 0.1 s has been found a good compromise between accuracy and efficiency. For some very sharp influence lines (e.g. shear forces) a finer step may be required. A sensitivity study is recommended.

Line 17:

For block maximum output, this specifies the block size in days for which the maximum is retained (it can be zero if Line 18 has a number >0).

Line 18:

For block maximum output, this specifies the block size in seconds for which the maximum is retained (it can be zero if Line 17 has a number >0).

Line 19:

To avoid unnecessary computation of smaller vehicles, this specifies the minim GVW for a vehicle’s load effect to be calculated. Its spatial arrangement on the road is not affected if its GVW is less than this number. The units are deci-tonnes (t/10)

Line 20:

For all program modes, this option if “1” will write the generated or read in vehicle file. For long run simulations this should be “0” as very large files can result, filling hard drive space and causing very slow computation. Mostly useful for short debugging or test runs.

Line 21:

The name of the file to be written if Line 20 is “1”.

Line 22:

If the vehicle file is to be written (Line 20 set to “1”), this specifies the number of vehicles that are stored in memory before writing to the hard drive. See section on BTLS Output for more details.

Line 23:

Specify “1” to write a full time history of the load effects – see section on BTLS Output for more details. This should be set to “0” for long simulations due to enormous resulting file size and slow execution.

Line 24:

Specify “1” to write the load effect value for each loading event that occurs. Again this can be a large file and cause slow execution for long-run simulations. See section on BTLS Output for more details.

Line 25:

If each loading event value is to be written (Line 24), this option if set to “1” specifies the number of events that are stored in memory before writing to the hard drive. See section on BTLS Output for more details.

Line 26:

Specify “1” to write block maximum load effect output files, or “0” to not. See section on BTLS Output for more details.

Line 27:

Specify whether to write the block maximum summary files (“1” or “0”). See section on BTLS Output for more details.

Line 28:

If block maximum output is to be written (Line 26), this option if set to “1” specifies the number of events that are stored in memory before writing to the hard drive. See section on BTLS Output for more details.

3.4 Bridge Definition File

The bridges over which vehicles are to pass are defined in the bridge definition file, specified in the configuration file (BTLSin.txt). The file name is arbitrary.

Important!

A bridge definition file must be included in the working folder.

An example bridge definition file is shown:

```
1,   20.0,      4,   2
1,   1,    4,   1.0, 0.8, 0.6, 0.4
2,   0,    2,   -0.2, 0.0, 0.2, 0.4
2,   30.0,      4,   2
1,   1,    4,   1.0, 0.8, 0.6, 0.4
2,   1,    2,   -0.2, 0.0, 0.2, 0.4
```

This file shows that two bridges are defined: Line 1 defines Bridge 1; Lines 2 & 3 define the load effects for Bridge 1; Line 4 defines bridge 2 and Lines 5 and 6 define the two load effects for Bridge 2.

Any number of bridges can be defined, as can any number of load effects for each bridge. Each bridge definition is formatted as follows:

Bridge information: (e.g. "1, 20.0, 4, 2")

This first line specifies some general information about the bridge:

- Column 1: the bridge number, a positive integer – 1 in this case;
- Column 2: the span of the bridge in metres, a real positive number – 20.0 m in this case;
- Column 3: the number of lanes on the bridge, a positive integer – 4 in this case;
- Column 4: the number of load effects to be considered for this bridge, a positive integer – 2 in this case. Each load effect is then defined on separate lines.

Load effect information: (e.g. “`1, 1, 4, 1.0, 0.8, 0.6, 0.4`”)

This line details the information for an individual load effect for the bridge as follows:

- Column 1: the load effect number, a positive integer – e.g. 1.
- Column 2: the type of load effect:
    - 1 if it is a built-in influence line function (see below for built-in functions)
    - 0 if the influence line is specified in the influence line definition file.
- Column 3: the influence line number. If it is a built-in influence line, this is the index of the IL function (see below). If it is a read-in influence line then it is the number of the read-in influence line (see IL definition file description).

The next columns define the lane factors (real numbers) to be applied to this influence line for each lane of the bridge, to determine the load effect. The present bridge has 4 lanes and so there are columns 4 to 7 with lane factors of 1.0, 0.8, 0.6, and 0.4 for lanes 1 through 4 respectively.

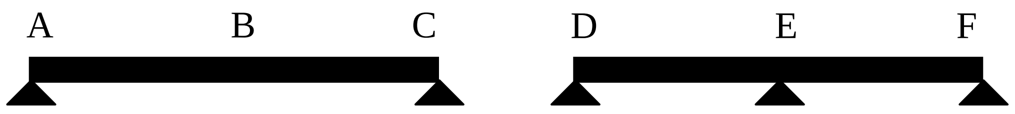
Lane factors represent the proportion of load of the corresponding lane (i.e. lane factor 3 is for lane 3) that contributes to the load effect in the element under consideration. In this way, an influence surface is effectively defined as slices along each lane. Note that this model means that the influence surface must be a scaled version of itself transversely across the bridge – this is not always the case however.

Built-In Influence Functions

The built-in influence functions are mathematical expressions that apply for any bridge length and can be weighted with any value of lane factor. Consequently, these built-in functions execute more quickly than read-in influence lines.

The description and index for the built in functions are:

Index	**Influence Line**	**Location**
1	Mid-span bending moment for a simply supported beam	B
2	Bending moment over the central support of a two-span beam	E
3	Left-hand shear in a simply-supported beam	A
4	Right-hand shear in a simply-supported beam	C
5	Right-hand shear for a two-span beam	F
6	Left-hand shear for a two-span beam	D
7	Total amount of load on the bridge (i.e. the unit influence line)	



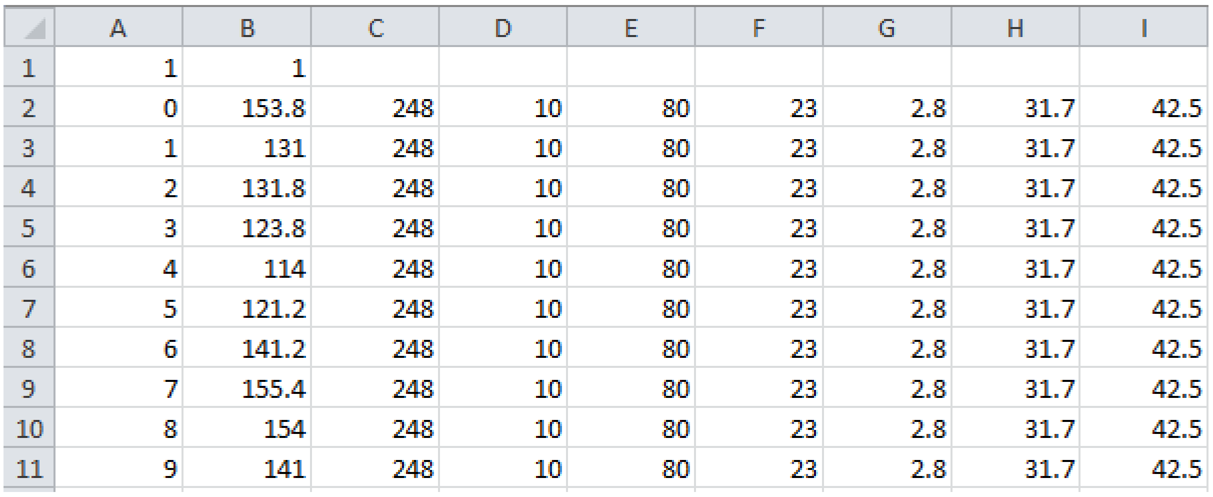
3.5 Lane Flow Data

This file contains all information relating to the number of lanes, the flow in each lane throughout the day, and the traffic composition. It applies when artificial traffic is being created using one of the free-flow headway models.

Important!

A lane flow definition file must be included in the working folder.

This file must be in `*.csv` format (but can have a `*.txt` extension). In `*.csv` format it is easily edited in a spread sheet program as shown:



	A	B	C	D	E	F	G	H	I
1	1	1							
2	0	153.8	248	10	80	23	2.8	31.7	42.5
3	1	131	248	10	80	23	2.8	31.7	42.5
4	2	131.8	248	10	80	23	2.8	31.7	42.5
5	3	123.8	248	10	80	23	2.8	31.7	42.5
6	4	114	248	10	80	23	2.8	31.7	42.5
7	5	121.2	248	10	80	23	2.8	31.7	42.5
8	6	141.2	248	10	80	23	2.8	31.7	42.5
9	7	155.4	248	10	80	23	2.8	31.7	42.5
10	8	154	248	10	80	23	2.8	31.7	42.5
11	9	141	248	10	80	23	2.8	31.7	42.5

Each lane of the simulation is defined by 25 rows of data:

- The first row defines the lane number (sequential) and direction number (1 or 2) in columns 1 and 2 (or A and B in the screenshot);
- The next 24 rows describe the traffic flow for each hour of the day (i.e. rows 2, 3, 4… in the screenshot above).

Note that it is assumed that every day of the simulation is the same. Typically only economic days of traffic are simulated of 5 days per week, 50 weeks per year (250 days per year). It is assumed that each such day has the same properties.

The structure of each hour description is as follows, with numbers given from hour 0 of the screen shot above (at row 2):

- Column 1: The hour identifier, starting at midnight, 0 to 23 (e.g. 0)
- Column 2: The mean truck flow rate in this hour (trucks/hour) (e.g. 153.8)
- Column 3: Mean velocity of traffic (dm/s) (e.g. 248)
- Column 4: Standard deviation of the velocity (dm/s) (e.g. 10)
- Column 5: Percentage of cars in this traffic model (e.g. 80)
- Column 6: Percentage of trucks that are 2 axle (e.g. 23)
- Column 7: Percentage of trucks that are 3 axles (e.g. 2.8)
- Column 8: Percentage of trucks that are 4 axles (e.g. 31.7)
- Column 9: Percentage of trucks that are 5 axles (e.g. 42.5)

The rationale for having the input in this form is the ease of altering the percentage cars and overall flow rate without modifying the truck flow and composition. This is best explained through an example of the calculations the program performs:

1. From Column 5, the truck percentage is 100-80 = 20%;
2. This 20% represents a flow of 153.8 vehicles per hour (Column2);
3. Thus the total flow rate is 153.8/0.2 = 769 vehicles per hour.
4. Of the 20% vehicles that are trucks, for example, 42.5% are 5-axle trucks, thus there will be 0.425*153.8 = 65.4 5-axle trucks on average for this hour.

Changing Column 4 then changes the overall flow rate, without changing the number of trucks that arrive.

Note that a normal distribution is assumed for the speed of all vehicles.

Each lane to be included in the simulation must have the above information. An example file with two lanes, one in each direction is given below:

```
1,1,,,,,,,
0,153.8,248,10,80,23,2.8,31.7,42.5
1,131,248,10,80,23,2.8,31.7,42.5
2,131.8,248,10,80,23,2.8,31.7,42.5
3,123.8,248,10,80,23,2.8,31.7,42.5
4,114,248,10,80,23,2.8,31.7,42.5
5,121.2,248,10,80,23,2.8,31.7,42.5
6,141.2,248,10,80,23,2.8,31.7,42.5
7,155.4,248,10,80,23,2.8,31.7,42.5
8,154,248,10,80,23,2.8,31.7,42.5
9,141,248,10,80,23,2.8,31.7,42.5
10,126.4,248,10,80,23,2.8,31.7,42.5
11,101.6,248,10,80,23,2.8,31.7,42.5
12,95.8,248,10,80,23,2.8,31.7,42.5
13,88.2,248,10,80,23,2.8,31.7,42.5
14,93,248,10,80,23,2.8,31.7,42.5
15,109,248,10,80,23,2.8,31.7,42.5
16,124.2,248,10,80,23,2.8,31.7,42.5
17,151,248,10,80,23,2.8,31.7,42.5
18,141.8,248,10,80,23,2.8,31.7,42.5
19,172.2,248,10,80,23,2.8,31.7,42.5
20,141.4,248,10,80,23,2.8,31.7,42.5
21,148,248,10,80,23,2.8,31.7,42.5
22,157.2,248,10,80,23,2.8,31.7,42.5
23,159.4,248,10,80,23,2.8,31.7,42.5
2,2,,,,,,,
0,92.2,222,10,80,21.9,2.3,31,44.8
1,79.6,222,10,80,21.9,2.3,31,44.8
2,67,222,10,80,21.9,2.3,31,44.8
3,74.8,222,10,80,21.9,2.3,31,44.8
4,81.6,222,10,80,21.9,2.3,31,44.8
5,94.8,222,10,80,21.9,2.3,31,44.8
6,102.4,222,10,80,21.9,2.3,31,44.8
7,121.2,222,10,80,21.9,2.3,31,44.8
8,127.4,222,10,80,21.9,2.3,31,44.8
9,127.2,222,10,80,21.9,2.3,31,44.8
10,112.2,222,10,80,21.9,2.3,31,44.8
11,111.4,222,10,80,21.9,2.3,31,44.8
12,110.2,222,10,80,21.9,2.3,31,44.8
13,146,222,10,80,21.9,2.3,31,44.8
14,160.4,222,10,80,21.9,2.3,31,44.8
15,152,222,10,80,21.9,2.3,31,44.8
16,151,222,10,80,21.9,2.3,31,44.8
17,167.2,222,10,80,21.9,2.3,31,44.8
18,179.6,222,10,80,21.9,2.3,31,44.8
19,164.8,222,10,80,21.9,2.3,31,44.8
20,206.6,222,10,80,21.9,2.3,31,44.8
21,228.4,222,10,80,21.9,2.3,31,44.8
22,189,222,10,80,21.9,2.3,31,44.8
23,138.8,222,10,80,21.9,2.3,31,44.8
```

3.6 Influence Line Definitions

This file stores the definitions of any discrete influence lines that are required. It must be in `*.csv` format (but can have a `*.txt` extension).

> **Important!**
>
> An influence line definition file must be included in the working folder (even if blank and not required).

The first line of the file is the number of influence lines defined within. Subsequently, each influence line is defined with the following structure:

- A first line giving the influence line number (Column 1) and the number of points defining the influence line (Column 2);
- Subsequent lines define the influence line using *x*, *y*, pairs for the location and ordinate values (Columns 1 and 2 respectively).

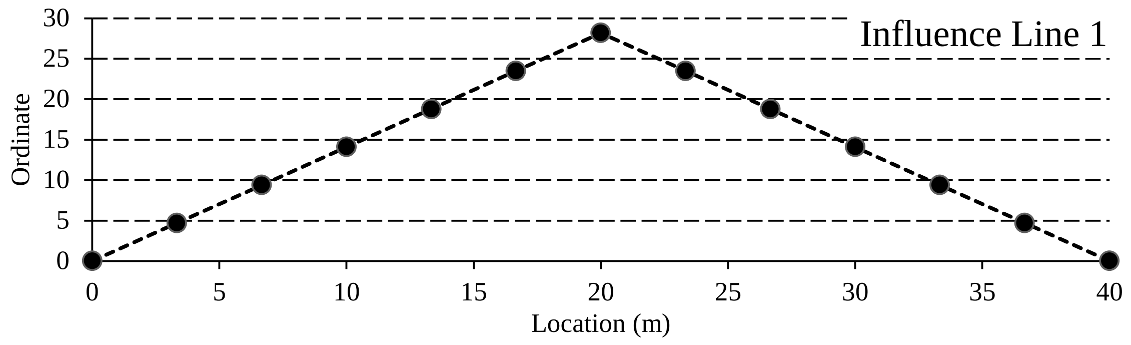
Discrete influence line processing takes longer than built-in expressions. The program must search the vector of *x*-coordinates to find the points surrounding the axle location. Linear interpolation of the ordinates is then used to find the ordinate at the axle location. The spacing of points need not be uniform. Therefore, prefer to use as few points as is necessary where the influence line is linear, and more points where it is curved.

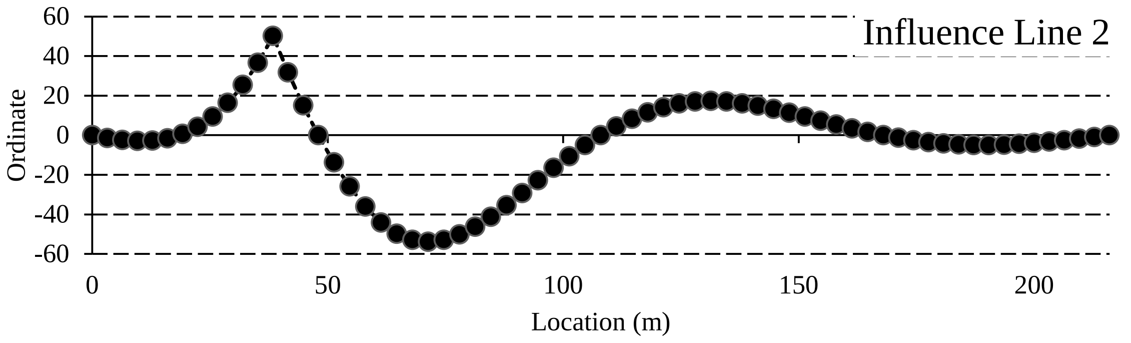
> **Important!**
>
> The program warns if the last *x*-coordinate is not the same as the length of the bridge defined in the Bridge Definition file. Behaviour in this case is generally unpredictable. However this warning can be issued due solely to rounding, and in this case no problems have been observed.

An example file is given below:

- he first influence line is a test of a 40 m simply-supported mid-span bending moment calculation,
- the second is an influence line from the Millau viaduct, courtesy if IFSTTAR, France.

```
2
1, 13
    0.000000,    0.000000
    3.333333,    4.695709
    6.666667,    9.391137
   10.000000,   14.086847
   13.333333,   18.782556
   16.666667,   23.477984
   20.000000,   28.173693
   23.333333,   23.477984
   26.666667,   18.782556
   30.000000,   14.086847
   33.333333,    9.391137
   36.666667,    4.695709
   40.000000,    0.000000
2, 67
    0.000000,    0.000000
    3.200000,   -1.404704
    6.400000,   -2.482683
    9.600000,   -2.967398
   12.800000,   -2.765344
   16.000000,   -1.658346
   19.200000,    0.578218
   22.400000,    4.170048
   25.600000,    9.341767
   28.800000,   16.319074
   32.000000,   25.326593
   35.200000,   36.574975
   38.400000,   50.127631
   41.600000,   31.722458
   44.800000,   15.056238
   48.000000,    0.000000
   51.333333,  -13.793401
   54.666667,  -25.918783
   58.000000,  -36.084887
   61.333333,  -44.196060
   64.666667,  -49.871840
   68.000000,  -52.900498
   71.333333,  -53.815114
   74.666667,  -52.833863
   78.000000,  -50.155575
   81.333333,  -46.221973
   84.666667,  -41.193196
   88.000000,  -35.436810
   91.333333,  -29.230102
   94.666667,  -22.847134
   98.000000,  -16.583464
  101.333333,  -10.646519
  104.666667,   -5.083589
  108.000000,    0.000000
  111.333333,    4.397896
  114.666667,    8.216499
  118.000000,   11.460109
  121.333333,   14.100781
  124.666667,   15.951508
  128.000000,   16.941357
  131.333333,   17.243363
  134.666667,   16.930610
  138.000000,   16.067582
  141.333333,   14.799371
  144.666667,   13.179716
  148.000000,   11.327914
  151.333333,    9.335319
  154.666667,    7.293285
  158.000000,    5.293167
  161.333333,    3.385477
  164.666667,    1.617506
  168.000000,    0.000000
  171.200000,   -1.360639
  174.400000,   -2.545019
  177.600000,   -3.539167
  180.800000,   -4.268925
  184.000000,   -4.746116
  187.200000,   -4.978264
  190.400000,   -4.989011
  193.600000,   -4.803078
  196.800000,   -4.444110
  200.000000,   -3.936826
  203.200000,   -3.307020
  206.400000,   -2.577262
  209.600000,   -1.773345
  212.800000,   -0.904943
  216.000000,    0.000000
```

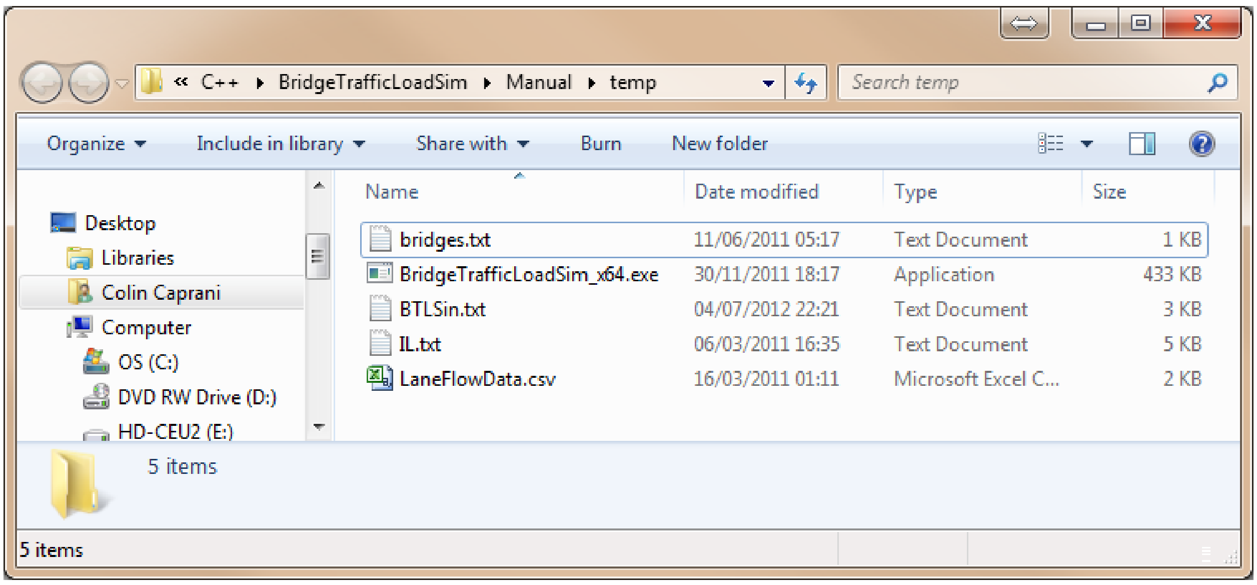

Influence Line 1
Ordinate
Location (m)
0
5
10
15
20
25
30
35
40


Influence Line 2
Ordinate
Location (m)
60
40
20
0
-20
-40
-60
50
100
150
200

4. Using BTLS

4.1 Running the program

The program can be run from any folder as explained previously. An example working folder showing all files necessary to execute the program is given below:



Name	Date modified	Type	Size
bridges.txt	11/06/2011 05:17	Text Document	1 KB
BridgeTrafficLoadSim_x64.exe	30/11/2011 18:17	Application	433 KB
BTLSin.txt	04/07/2012 22:21	Text Document	3 KB
IL.txt	06/03/2011 16:35	Text Document	5 KB
LaneFlowData.csv	16/03/2011 01:11	Microsoft Excel C...	2 KB

Note that the 64 bit version is being used here.

Important!

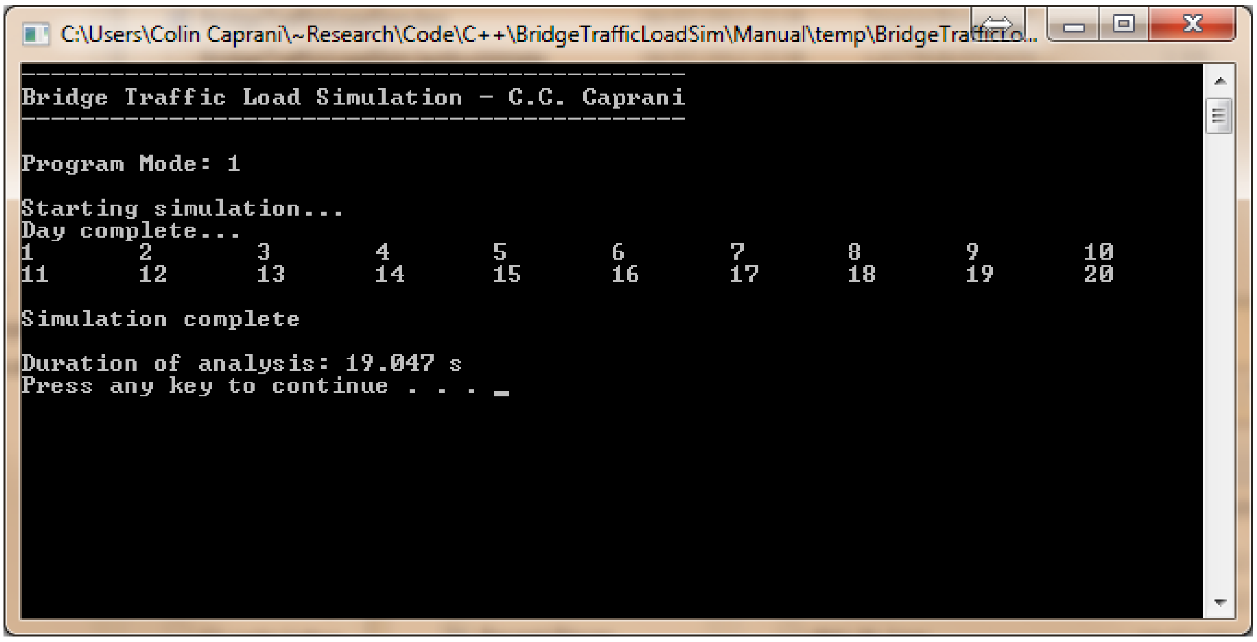
To run the program, double click the executable (`*.exe`) file.

4.2 Console output

Some examples of console output during program execution are given.

Example 1: Program Mode 1

After each day of simulation is complete, the program outputs a notice. At the end of the simulation the elapsed time is displayed for information. In this case, 20 days of traffic and generated and simulated crossing 5 bridges, each with 3 load effects.

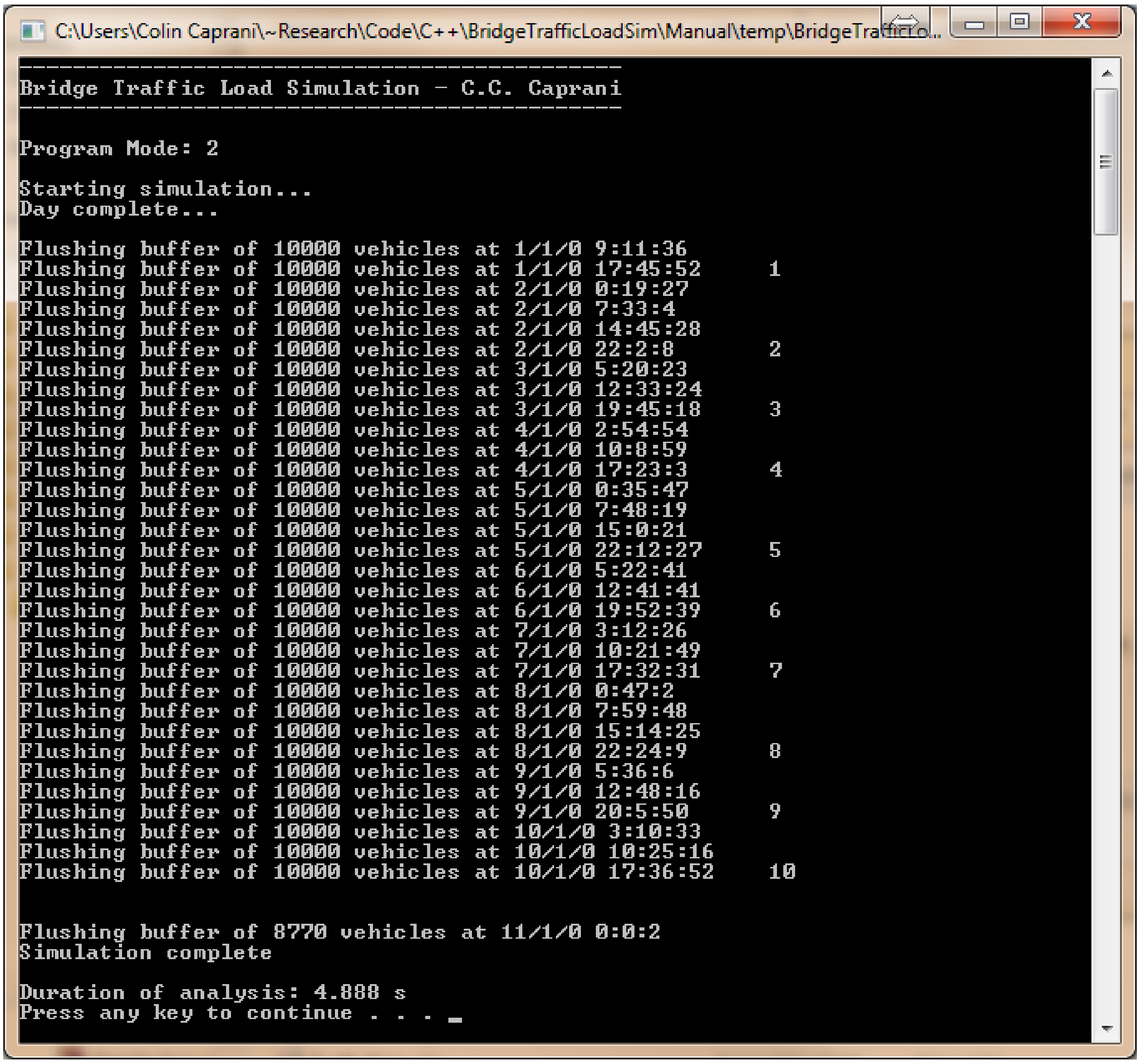


C:\Users\Colin Caprani\~Research\Code\C++\BridgeTrafficLoadSim\Manual\temp\BridgeTrafficLo...

```
---------------------------------------------
Bridge Traffic Load Simulation - C.C. Caprani
---------------------------------------------

Program Mode: 1

Starting simulation...
Day complete...
1       2       3       4       5       6       7       8       9       10
11      12      13      14      15      16      17      18      19      20

Simulation complete

Duration of analysis: 19.047 s
Press any key to continue . . . _
```

Example 1: Program Mode 2

Ten days of vehicles are generated and the current day number is given (right hand columns). Each time the program flushed the vehicle buffer to the file, an output is given of the simulation time at which it occurred.



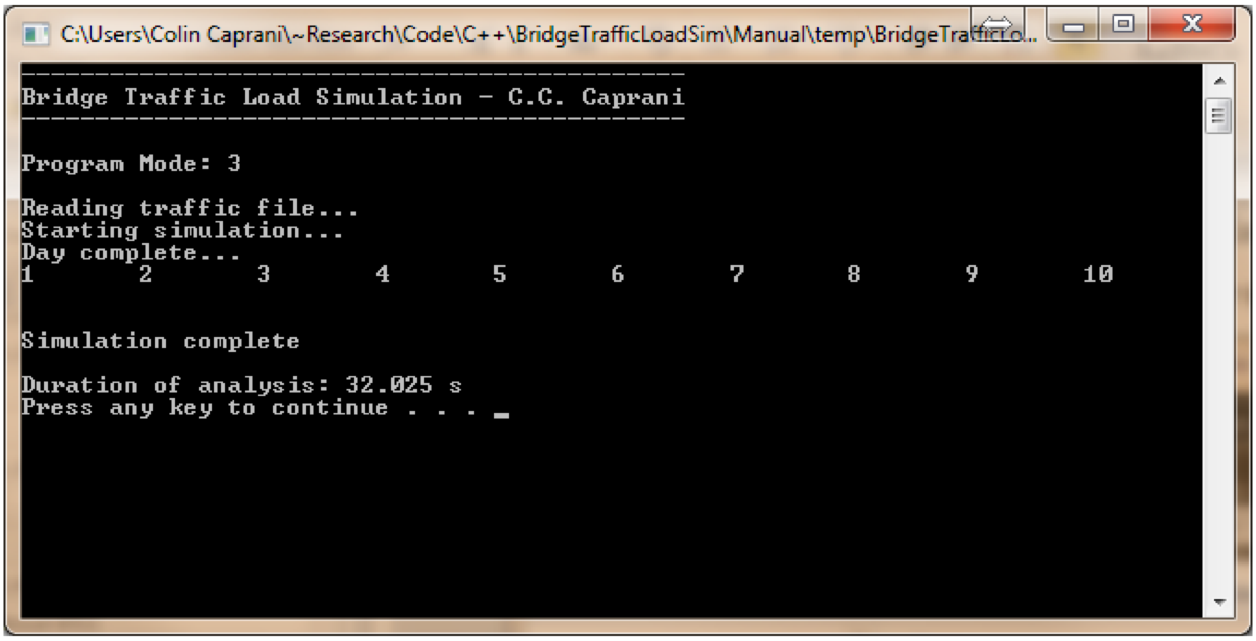
C:\Users\Colin Caprani\~Research\Code\C++\BridgeTrafficLoadSim\Manual\temp\BridgeTrafficLo...

```
--------------------------------------------
Bridge Traffic Load Simulation - C.C. Caprani
--------------------------------------------

Program Mode: 2

Starting simulation...
Day complete...

Flushing buffer of 10000 vehicles at 1/1/0 9:11:36
Flushing buffer of 10000 vehicles at 1/1/0 17:45:52     1
Flushing buffer of 10000 vehicles at 2/1/0 0:19:27
Flushing buffer of 10000 vehicles at 2/1/0 7:33:4
Flushing buffer of 10000 vehicles at 2/1/0 14:45:28
Flushing buffer of 10000 vehicles at 2/1/0 22:2:8       2
Flushing buffer of 10000 vehicles at 3/1/0 5:20:23
Flushing buffer of 10000 vehicles at 3/1/0 12:33:24
Flushing buffer of 10000 vehicles at 3/1/0 19:45:18     3
Flushing buffer of 10000 vehicles at 4/1/0 2:54:54
Flushing buffer of 10000 vehicles at 4/1/0 10:8:59
Flushing buffer of 10000 vehicles at 4/1/0 17:23:3      4
Flushing buffer of 10000 vehicles at 5/1/0 0:35:47
Flushing buffer of 10000 vehicles at 5/1/0 7:48:19
Flushing buffer of 10000 vehicles at 5/1/0 15:0:21
Flushing buffer of 10000 vehicles at 5/1/0 22:12:27     5
Flushing buffer of 10000 vehicles at 6/1/0 5:22:41
Flushing buffer of 10000 vehicles at 6/1/0 12:41:41
Flushing buffer of 10000 vehicles at 6/1/0 19:52:39     6
Flushing buffer of 10000 vehicles at 7/1/0 3:12:26
Flushing buffer of 10000 vehicles at 7/1/0 10:21:49
Flushing buffer of 10000 vehicles at 7/1/0 17:32:31     7
Flushing buffer of 10000 vehicles at 8/1/0 0:47:2
Flushing buffer of 10000 vehicles at 8/1/0 7:59:48
Flushing buffer of 10000 vehicles at 8/1/0 15:14:25
Flushing buffer of 10000 vehicles at 8/1/0 22:24:9      8
Flushing buffer of 10000 vehicles at 9/1/0 5:36:6
Flushing buffer of 10000 vehicles at 9/1/0 12:48:16
Flushing buffer of 10000 vehicles at 9/1/0 20:5:50      9
Flushing buffer of 10000 vehicles at 10/1/0 3:10:33
Flushing buffer of 10000 vehicles at 10/1/0 10:25:16
Flushing buffer of 10000 vehicles at 10/1/0 17:36:52    10


Flushing buffer of 8770 vehicles at 11/1/0 0:0:2
Simulation complete

Duration of analysis: 4.888 s
Press any key to continue . . . _
```

The fodler is then populated with the outputted vehicle file as named in “`BTLSin.txt`”.

Example 3: Program Mode 3

The traffic file created in the last example is read in and passed over 5 bridges, each with 3 load effects. The output is as follows:



C:\Users\Colin Caprani\~Research\Code\C++\BridgeTrafficLoadSim\Manual\temp\BridgeTrafficLo...

```
---------------------------------------------
Bridge Traffic Load Simulation - C.C. Caprani
---------------------------------------------

Program Mode: 3

Reading traffic file...
Starting simulation...
Day complete...
1       2       3       4       5       6       7       8       9       10


Simulation complete

Duration of analysis: 32.025 s
Press any key to continue . . . _
```

Note the slower execution time than for Program Mode 1 which has 20 days of traffic. This is caused by the additional overhead required to manipulate the 25 MB traffic file that has been read into memory.

4.3 Input Errors

When there are errors in the input, the behaviour is unpredictable. More informative user feedback will be built in soon.

Some potential problems:

- The supporting files (e.g. bridge, lane flow, IL) cannot be found in the working folder;
- The traffic folder cannot be found (it should be at "`C:\Traffic`");
- The vehicle file to be read in cannot be found (Program Mode 3).
- Outputs are not matched to Program Mode (e.g. Program Mode 2 – Generate vehicle file, but no vehicle file is to be output – Line 20 of `BTLSin.txt`.)

In each of these cases, the program may:

- Output helpful warnings;
- Flash open and close immediately;
- Remain open and display unusual text, such as "Conversion Error".

Admittedly, all behaviours should be of the first type – this will improve.

5. BTLS Output

5.1 Introduction

BTLS has the ability to produce large amounts of output, especially for long run simulations or heavily congested traffic. Accessing the hard drive often can significantly slow the program’s execution. Therefore keep buffer sizes as large as memory and execution speed can allow.

All outputs are in text file format with specific information layouts and formats for each type of output file.

5.2 Traffic File

A traffic file can be output in any Program Mode if selected on Line 20 of `BTLSin.txt`. For Program Mode 2, this must be selected.

The file is named as specified on Line 21 of `BTLSin.txt`.

The program outputs vehicles in CASTOR format.

An example of the program output for several trucks is given:

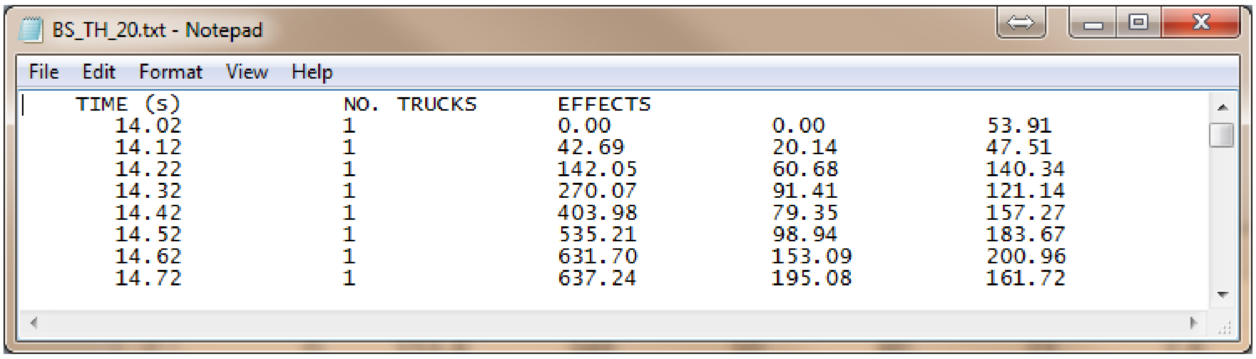
```
1001 1 1 2 0 12618155  54 43211 18 2743 27 0  0 0  0 0  0 0  0 0  0 0  0 0  0
1001 1 1 2 0 2 412133 137 67311 18 5441 6626 17 0  0 0  0 0  0 0  0 0  0 0  0
1001 1 1 2 0 2 598157  64 43211 18 3243 32 0  0 0  0 0  0 0  0 0  0 0  0 0  0
1001 1 1 2 0 44354134 336133511 18 7839 7040 6327 6327 63 0  0 0  0 0  0 0  0
1001 1 1 2 0 93062152 117 67311 18 3941 3926 39 0  0 0  0 0  0 0  0 0  0 0  0
1001 1 1 2 0101765131  97 44211 18 5344 44 0  0 0  0 0  0 0  0 0  0 0  0 0  0
```

5.3 Time History File

If a full time history is selected to be output (Line 23 of `BTLSin.txt`), BTLS creates a single file for each bridge named:

```
BS_TH_L.txt
```

Where *L* is the bridge length. The file gives the load effect at each time step of the simulation for each load effect considered for the bridge. Note that no output is given when there is zero load effect. A sample output is:



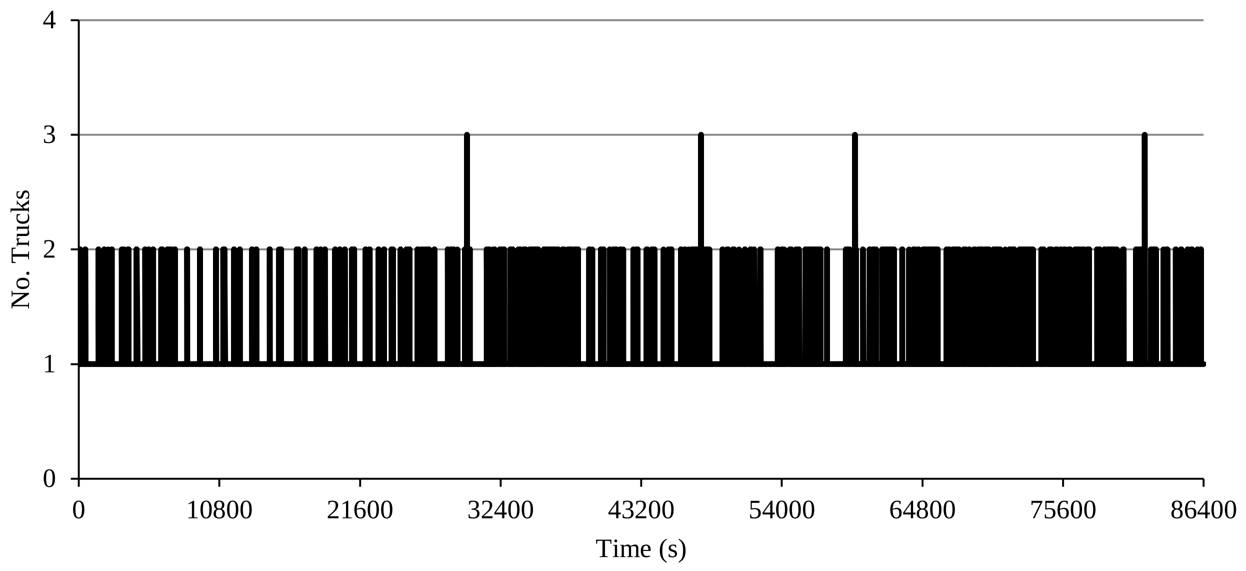
```
   TIME (s)          NO. TRUCKS     EFFECTS
      14.02          1              0.00           0.00           53.91
      14.12          1              42.69          20.14          47.51
      14.22          1              142.05         60.68          140.34
      14.32          1              270.07         91.41          121.14
      14.42          1              403.98         79.35          157.27
      14.52          1              535.21         98.94          183.67
      14.62          1              631.70         153.09         200.96
      14.72          1              637.24         195.08         161.72
```

The format is:

- Column 1: the current time. Note that time starts at the time of arrival of the first vehicle;
- Column 2: The number of trucks currently on the bridge;
- Columns 3+: The current value of each load effect is given, according to the order of the load effects in the bridge definition file.

This file can get extremely large, but for short runs (e.g. 1-day) it is very useful for checking and debugging output.

An example output is given showing the number of trucks on the bridge through the day. As can be seen, four 3-truck events occur on this 20 m bridge.

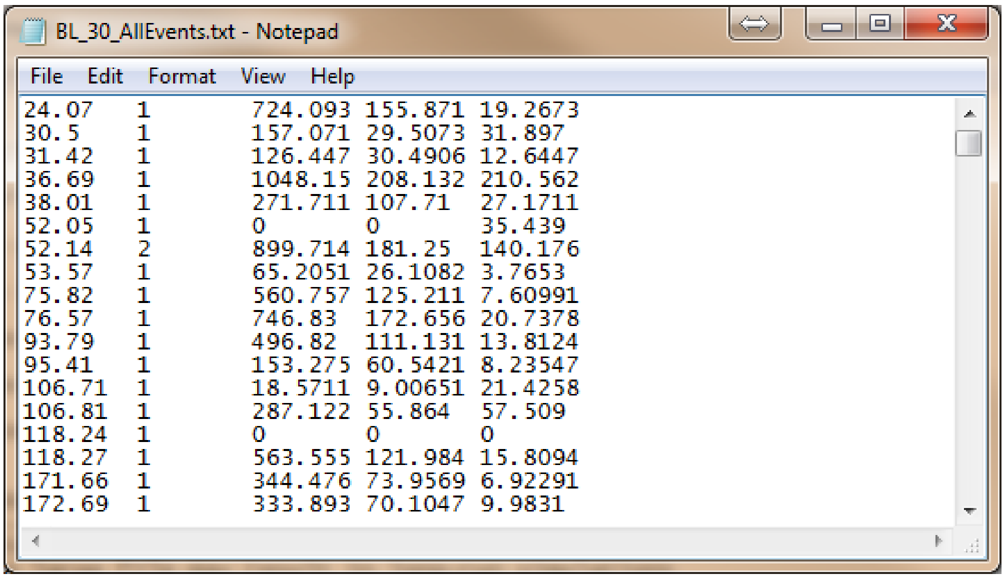


5.4 All Events File

If all loading events are selected to be output (Line 24 of `BTLSin.txt`), BTLS creates a single file for each bridge named:

```
BL_L_AllEvents.txt
```

Where *L* is the bridge length. The file gives the maximum of each calculated load effect recorded during each loading event to occur. A loading event is defined as the loading that occurs between two occasions of zero trucks on the bridge, or the departure or arrival of another vehicle. A sample output is:



```
24.07   1       724.093 155.871 19.2673
30.5    1       157.071 29.5073 31.897
31.42   1       126.447 30.4906 12.6447
36.69   1       1048.15 208.132 210.562
38.01   1       271.711 107.71  27.1711
52.05   1       0       0       35.439
52.14   2       899.714 181.25  140.176
53.57   1       65.2051 26.1082 3.7653
75.82   1       560.757 125.211 7.60991
76.57   1       746.83  172.656 20.7378
93.79   1       496.82  111.131 13.8124
95.41   1       153.275 60.5421 8.23547
106.71  1       18.5711 9.00651 21.4258
106.81  1       287.122 55.864  57.509
118.24  1       0       0       0
118.27  1       563.555 121.984 15.8094
171.66  1       344.476 73.9569 6.92291
172.69  1       333.893 70.1047 9.9831
```

The format is:

- Column 1: the starting time of the loading event.
- Column 2: The number of trucks in the event;
- Columns 3+: The maximum value of each load effect during the event.

This file can get extremely large, but is useful for checking and debugging output.

5.5 Block Maximum Vehicle Files

If block maximum vehicle files are selected to be output (Line 26 of `BTLSin.txt`), BTLS creates two types of file output.

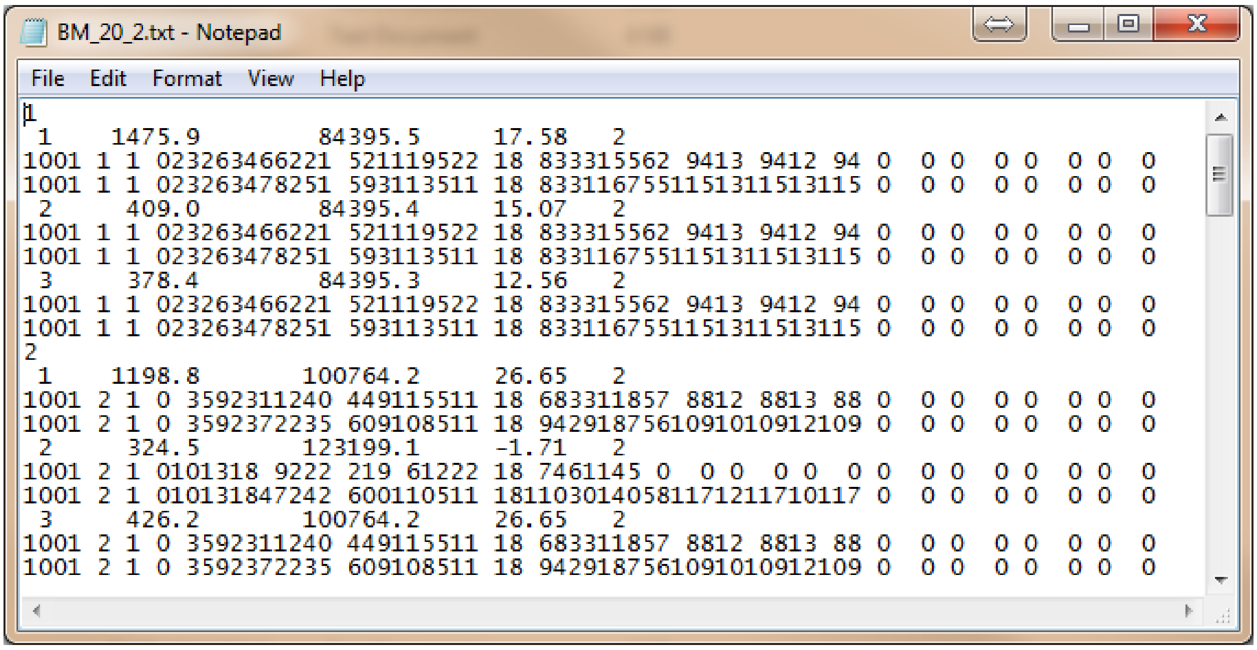
Output by Number of Trucks

The first output type is done for each load effect of each bridge named:

```
BM_L_N.txt
```

Where *L* is the bridge length and *N* is the number of trucks comprising the loading events. In this manner a comprehensive breakdown of the causing of loading can be studied, suitable for application of the CDS method (Caprani et al 2008).

A sample output is shown:



```
1
 1    1475.9         84395.5     17.58   2
1001 1 1 023263466221 521119522 18 833315562 9413 9412 94 0  0 0  0 0  0 0  0
1001 1 1 023263478251 593113511 18 8331167551151311513115 0  0 0  0 0  0 0  0
 2     409.0         84395.4     15.07   2
1001 1 1 023263466221 521119522 18 833315562 9413 9412 94 0  0 0  0 0  0 0  0
1001 1 1 023263478251 593113511 18 8331167551151311513115 0  0 0  0 0  0 0  0
 3     378.4         84395.3     12.56   2
1001 1 1 023263466221 521119522 18 833315562 9413 9412 94 0  0 0  0 0  0 0  0
1001 1 1 023263478251 593113511 18 8331167551151311513115 0  0 0  0 0  0 0  0
2
 1    1198.8        100764.2     26.65   2
1001 2 1 0 3592311240 449115511 18 683311857 8812 8813 88 0  0 0  0 0  0 0  0
1001 2 1 0 3592372235 609108511 18 9429187561091010912109 0  0 0  0 0  0 0  0
 2     324.5        123199.1     -1.71   2
1001 2 1 0101318 9222 219 61222 18 7461145 0  0 0  0 0  0 0  0 0  0 0  0 0  0
1001 2 1 010131847242 600110511 1811030140581171211710117 0  0 0  0 0  0 0  0
 3     426.2        100764.2     26.65   2
1001 2 1 0 3592311240 449115511 18 683311857 8812 8813 88 0  0 0  0 0  0 0  0
1001 2 1 0 3592372235 609108511 18 9429187561091010912109 0  0 0  0 0  0 0  0
```

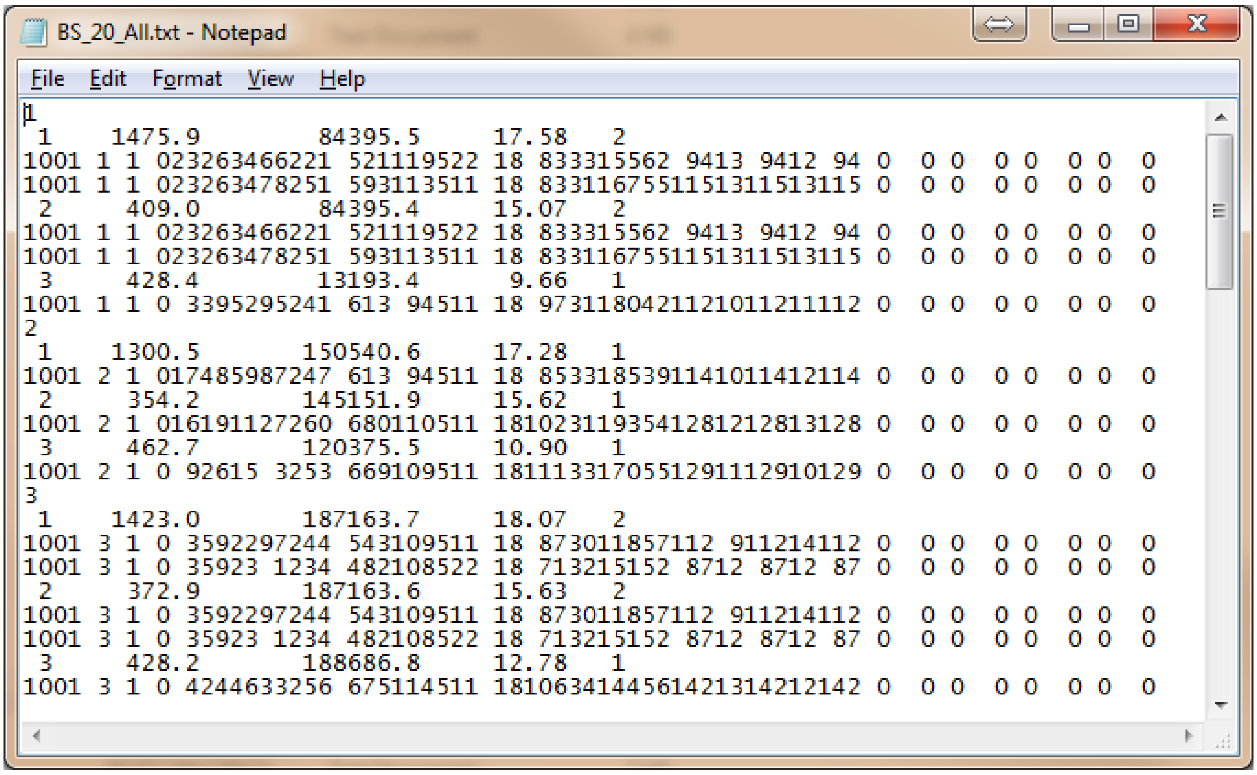
This file structure is termed a Loading Event File, and its structure explained later.

Mixed Output

This form of output represents the conventional form in which the number of trucks comprising the event is not taken into account. Instead the maximum load effect recorded during the block is noted. One such file per bridge is output, named:

```
BS_L_All.txt
```

Where *L* is the bridge length. This file structure is a Loading Event File, explained later. A sample output showing 3 blocks is:



```
1
 1    1475.9        84395.5     17.58   2
1001 1 1 023263466221 521119522 18 833315562 9413 9412 94 0  0 0  0 0  0 0  0
1001 1 1 023263478251 593113511 18 8331167551151311513115 0  0 0  0 0  0 0  0
 2     409.0        84395.4     15.07   2
1001 1 1 023263466221 521119522 18 833315562 9413 9412 94 0  0 0  0 0  0 0  0
1001 1 1 023263478251 593113511 18 8331167551151311513115 0  0 0  0 0  0 0  0
 3     428.4        13193.4      9.66   1
1001 1 1 0 3395295241 613 94511 18 9731180421121011211112 0  0 0  0 0  0 0  0
2
 1    1300.5       150540.6     17.28   1
1001 2 1 017485987247 613 94511 18 8533185391141011412114 0  0 0  0 0  0 0  0
 2     354.2       145151.9     15.62   1
1001 2 1 016191127260 680110511 1810231193541281212813128 0  0 0  0 0  0 0  0
 3     462.7       120375.5     10.90   1
1001 2 1 0 92615 3253 669109511 1811133170551291112910129 0  0 0  0 0  0 0  0
3
 1    1423.0       187163.7     18.07   2
1001 3 1 0 3592297244 543109511 18 873011857112 911214112 0  0 0  0 0  0 0  0
1001 3 1 0 35923 1234 482108522 18 713215152 8712 8712 87 0  0 0  0 0  0 0  0
 2     372.9       187163.6     15.63   2
1001 3 1 0 3592297244 543109511 18 873011857112 911214112 0  0 0  0 0  0 0  0
1001 3 1 0 35923 1234 482108522 18 713215152 8712 8712 87 0  0 0  0 0  0 0  0
 3     428.2       188686.8     12.78   1
1001 3 1 0 4244633256 675114511 1810634144561421314212142 0  0 0  0 0  0 0  0
```

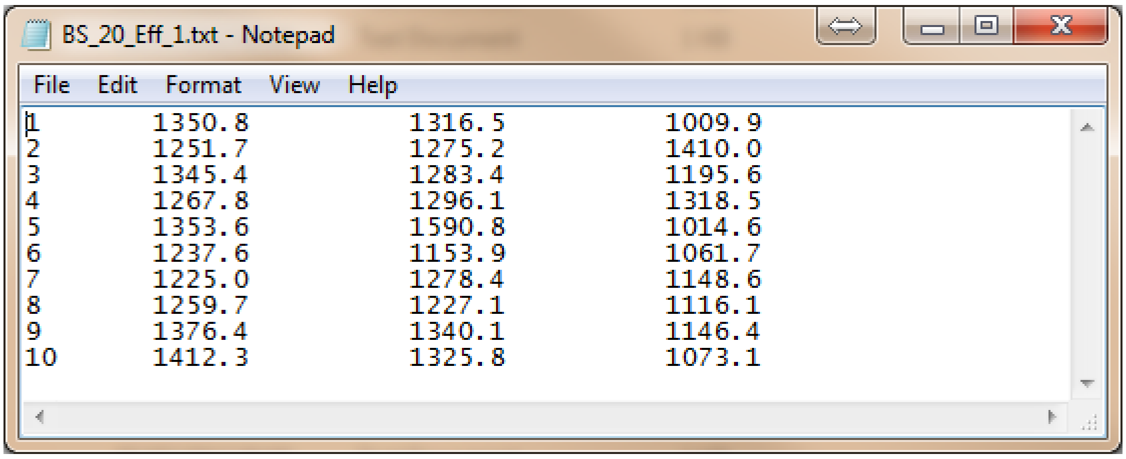
As can be seen, there are a different number of trucks comprising the loading events that cause the maximum of each load effect in the block. Sometimes the same truck(s) loading event causes the maximum of 2 or more load effects, but in general different loading events cause the maximum of each load effect. This is because of the different shapes of the influence lines, and hence critical loading arrangements.

5.6 Block Maximum Summary Files

If block maximum summary files are selected to be output (Line 27 of `BTLSin.txt`), BTLS creates a file for each load effect of each bridge named:

```
BS_L_Eff_E.txt
```

Where *L* is the bridge length and *E* is the load effect number. The file gives the maximum load effect recorded during each block, broken down according to the number of trucks comprising the event. A sample output is:



```
1         1350.8          1316.5          1009.9
2         1251.7          1275.2          1410.0
3         1345.4          1283.4          1195.6
4         1267.8          1296.1          1318.5
5         1353.6          1590.8          1014.6
6         1237.6          1153.9          1061.7
7         1225.0          1278.4          1148.6
8         1259.7          1227.1          1116.1
9         1376.4          1340.1          1146.4
10        1412.3          1325.8          1073.1
```

The format is:

- Column 1: The block index.
- Column 2: The block maximum 1-truck load effect;
- Columns 3: The block maximum 2-truck load effect
- Columns 4+: As appropriate, the block maximum 3-, 4-, … truck load effect.

Taking the maximum across Columns 2+ gives the overall block maximum load effect.

5.7 Loading Event File Structure

This file structure contains all information relating to the event. An example is shown below (from the Block Maximum Vehicle Files, Out by Number of Trucks screenshot).

Line	**Data**
1	`1`
2	`1    1475.9        84395.5     17.58   2`
3	`1001 1 1 023263466221 521119522 18 833315562 9413 9412 94 0  0 0  0 0  0 0  0`
4	`1001 1 1 023263478251 593113511 18 8331167551151311513115 0  0 0  0 0  0 0  0`
5	`2     409.0        84395.4     15.07   2`
6	`1001 1 1 023263466221 521119522 18 833315562 9413 9412 94 0  0 0  0 0  0 0  0`
7	`1001 1 1 023263478251 593113511 18 8331167551151311513115 0  0 0  0 0  0 0  0`
8	`3     378.4        84395.3     12.56   2`
9	`1001 1 1 023263466221 521119522 18 833315562 9413 9412 94 0  0 0  0 0  0 0  0`
10	`1001 1 1 023263478251 593113511 18 8331167551151311513115 0  0 0  0 0  0 0  0`
11	`2`
12	`1    1198.8       100764.2     26.65   2`
13	`1001 2 1 0 3592311240 449115511 18 683311857 8812 8813 88 0  0 0  0 0  0 0  0`
14	`1001 2 1 0 3592372235 609108511 18 9429187561091010912109 0  0 0  0 0  0 0  0`
15	`2     324.5       123199.1     -1.71   2`
16	`1001 2 1 0101318 9222 219 61222 18 7461145 0  0 0  0 0  0 0  0 0  0 0  0 0  0`
17	`1001 2 1 010131847242 600110511 1811030140581171211710117 0  0 0  0 0  0 0  0`
18	`3     426.2       100764.2     26.65   2`
19	`1001 2 1 0 3592311240 449115511 18 683311857 8812 8813 88 0  0 0  0 0  0 0  0`
20	`1001 2 1 0 3592372235 609108511 18 9429187561091010912109 0  0 0  0 0  0 0  0`

Each line is explained as follows.

Line 1:

The index of the current block (if block maximum output), or the index of the particular loading event in legacy files.

Line 2:

This is the load effect information line with 5 fields of data separated by tabs:

- Field 1: The load effect number;

- Field 2: The value of the load effect;
- Field 3: The time at which this load effect was found in seconds;
- Field 4: The distance of the first axle of the first truck on the bridge relative to the bridge datum, at the time of the crossing event maximum effect being reached. This allows one to sketch the positions of the trucks at the time of the load effect.
- Field 5: The number of trucks comprising the event.

Line 3-4:

These lines provide the truck data string in CASTOR format for later processing.

Line 5+:

The format of lines 2-4 continues for each of the effects calculated. Line 11 then provides the information for the start of the second block or loading event, and the format repeats itself.

6. Appendices

6.1 Appendix 1 – Traffic File Formats

CASTOR File Format

In the table below, the Format column gives the storage type of the data. IX refers to an integer of X number of digits, including leading or trailing zeros.

Record	Unit	Format
Head		I4
Day		I2
Month		I2
Year		I2
Hour		I2
Minute		I2
Second		I2
Second/100		I2
Speed	dm/s	I3
Gross Vehicle Weight - GVW	kg/100	I4
Length	dm	I3
Number of Axles		I1
Direction		I1
Lane		I1
Transverse Location In Lane	dm	I3
Weight Axle 1	kg/100	I3
Spacing Axle 1 - Axle 2	dm	I2
Weight Axle 2	kg/100	I3
Spacing Axle 2 - Axle 3	dm	I2
⋮	⋮	⋮
Spacing Axle 8 - Axle 9	dm	I2
Weight Axle 9	kg/100	I3

BeDIT File Format

This file format is similar to CASTOR except that the maximum number of axles possible is 20, the axle spacings are given by a three digit number, and the direction is zero-based.

Record	Unit	Format
Head		I4
Day		I2
Month		I2
Year		I2
Hour		I2
Minute		I2
Second		I2
Second/100		I2
Speed	dm/s	I3
Gross Vehicle Weight - GVW	kg/100	I4
Length	dm	I3
Number of Axles		I1
Direction (zero-based)		I1
Lane		I1
Transverse Location In Lane	dm	I3
Weight Axle 1	kg/100	I3
Spacing Axle 1 - Axle 2	dm	I3
Weight Axle 2	kg/100	I3
Spacing Axle 2 - Axle 3	dm	I3
⋮	⋮	⋮
Spacing Axle 19 - Axle 20	dm	I3
Weight Axle 20	kg/100	I3

SAFT File Format

As for the CASTOR table, the Format column gives the storage type of the data. IX refers to an integer of X number of digits, including leading or trailing zeros. Note that since the SAFT format does not contain direction or lane identifiers, this format is suitable for single lane traffic only.

Record	Unit	Format
Vehicle order		I5
20000 unused number		I5
Day		I2
Month		I2
Year		I2
Hour		I2
Minute		I2
Second		I2
Second/100		I2
Speed	dm/s	I3
Gross Vehicle Weight - GVW	kN	I4
Length	dm	I3
Number of Axles		I1
Weight Axle 1	kN	I3
Spacing Axle 1 - Axle 2	dm	I2
⋮	⋮	⋮
Spacing Axle 8 – Axle 9	dm	I3
Weight Axle 9	kN	I2

6.2 Appendix 2 – References

- Caprani, C.C. (2005), *Probabilistic Analysis of Highway Bridge Traffic Loading*, PhD Thesis, School of Architecture, Landscape and Civil Engineering, University College Dublin, Ireland. http://www.colincaprani.com/research/phd-dissertation/
- Caprani, C.C. (2012), 'Calibration of a congestion load model for highway bridges using traffic microsimulation', *Structural Engineering International*, 22(3), August, in print.
- Caprani, C.C., OBrien, E.J. and McLachlan, G.J. (2008), 'Characteristic traffic load effects from a mixture of loading events on short to medium span bridges', *Structural Safety*, Vol. 30(5), September, pp. 394-404. 10.1016/j.strusafe.2006.11.006
- Grave, S.A.J. (2001), *Modelling of Site-Specific Traffic Loading on Short to Medium Span Bridges*, Ph.D. Thesis, Department of Civil Engineering, Trinity College Dublin.
- OBrien, E.J. and Caprani, C.C. (2005), 'Headway modelling for traffic load assessment of short- to medium-span bridges', *The Structural Engineer*, Vol 83, No. 16, August, pp. 33-36. http://www.istructe.org/thestructuralengineer/HC/Abstract.asp?PID=5494