

Using the VS Table Tool to Create Loadable Tables

Introduction	1
VS Table Tool Resources	2
VS Table Tool Details	2
Example Specifics	3
VS Table Engine Map: Torque vs. Speed	3
VS Table Road Roughness Example	11
Summary	17
References	17

Introduction

The user interface for CarSim, TruckSim, and BikeSim, which is known as the VS Browser, provides a useful way to interact with the configurable functions used by the solver. In the VS Browser, data may be entered as a constant, a coefficient, and as either a one- or two-dimensional table with a variety of interpolation and extrapolation choices. However, there are some situations where the workflow provided by the VS Browser is less efficient. The input data used for some tables (for example, data associated with road features) can have many data points, and the large amount of data can slow down the browser operation. In other situations, data is generated rapidly, or by an automated process, and it is useful to be able to run a simulation with the most recent data quickly without wanting to manually replace the input data using the VS Browser. Additionally, while the CarSim, TruckSim, and BikeSim solvers run in a Linux environment, there is currently no user interface for Linux, so runs must be manually modified in a windows environment and then transferred to the Linux computer to run. One solution for each of these issues is to configure the simulation to load configurable function data from VS Table files generated by the VS Table Tool.

The VS Table Tool is a command line utility located in the Programs folder of the software installation directory. Comma-separated (.csv) files are provided as input, and the tool creates binary (.vsb) and header (.vstb) files that can be read by the VS Solver. These data files can represent VS Road profiles, vehicle component maps, and many of the other parameters that are specified using a configurable function within the VS Browser. The VS Table tool was first released with version 2020.1 of CarSim and TruckSim. The syntax for the VS Table Tool is detailed in the VS Table Tool reference manual, which is a companion to this document, and can be found on the Help menu in the Reference Manuals section.

VS Table Tool Resources

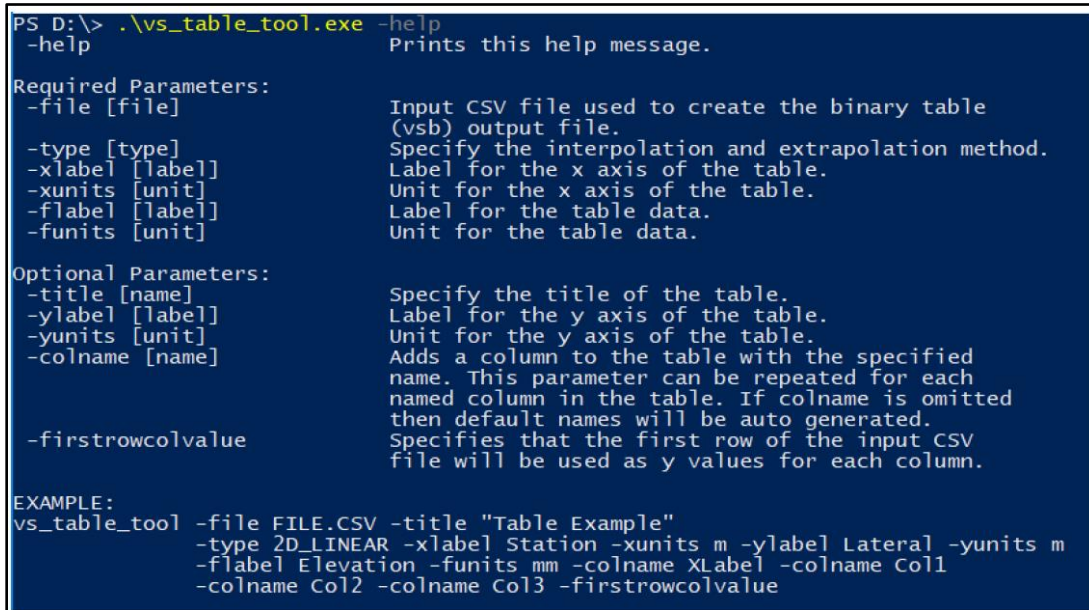
In addition to this reference document, there are a variety of other documents which either reference the VS Table Tool, explain the VS Table Tool in more detail, or reference VS Commands that are used to implement and manipulate VS Tables. These documents are listed below:

- VS Visualizer
- VS Output API Reading and Accessing VS Output Files
- VS Commands Reference Manual
- Extending VS Math Models with VS Commands and the VS API
- VS SDK: The VehicleSim Software Development Kit
- VS Table Tool (in SDK documentation)

It is a good idea to keep these references in mind when questions arise related to the VS Table Tool and its relationship with the VS SDK, VS API, and VS Browser.

VS Table Tool Details

The primary and in-depth details for the VS Table Tool can be found in the VS Table Tool reference guide, located in the Utilities sub-directory of the VS SDK. For a quick introduction, Figure 1 shows how the VS Table Tool appears when accessed via the command line.

A screenshot of a Windows command prompt window with a dark blue background and white text. The prompt shows the command `PS D:\> .\vs_table_tool.exe -help` and its output. The output lists required parameters (-file, -type, -xlabel, -xunits, -flabel, -funits) and optional parameters (-title, -ylabel, -yunits, -colname, -firstrowcolvalue) with their respective descriptions. An example command is also provided at the bottom.

```
PS D:\> .\vs_table_tool.exe -help
-help          Prints this help message.

Required Parameters:
-file [file]    Input CSV file used to create the binary table
                (vsb) output file.
-type [type]    Specify the interpolation and extrapolation method.
-xlabel [label] Label for the x axis of the table.
-xunits [unit]  Unit for the x axis of the table.
-flabel [label] Label for the table data.
-funits [unit]  Unit for the table data.

Optional Parameters:
-title [name]   Specify the title of the table.
-ylabel [label] Label for the y axis of the table.
-yunits [unit]  Unit for the y axis of the table.
-colname [name] Adds a column to the table with the specified
                name. This parameter can be repeated for each
                named column in the table. If colname is omitted
                then default names will be auto generated.
-firstrowcolvalue Specifies that the first row of the input CSV
                file will be used as y values for each column.

EXAMPLE:
vs_table_tool -file FILE.CSV -title "Table Example"
              -type 2D_LINEAR -xlabel Station -xunits m -ylabel Lateral -yunits m
              -flabel Elevation -funits mm -colname XLabel -colname Coll
              -colname Col2 -colname Col3 -firstrowcolvalue
```

Figure 1: Screenshot of VS Table Tool Interface invoking the -help flag

The VS Table Tools is a command line utility. To use the VS Table Tool, navigate to the directory containing the VS Table Tool executable file and then run the executable with a number of input parameters. The available parameters are listed when the user executes the VS Table Tool executable using the `-help` parameter. Output from the `-help` parameter is shown in Figure 1. Proper syntax for using the VS Table Tool will require the path to the input CSV file, the table type, and labels and units for both the input parameter to the table and the table output. Optional arguments

may also be used, including adding a second input dimension and specifying how the data is arranged in the table.

One useful method for determining the appropriate axis labels, axis units, and table type, is to find the location in the VS Browser where the data is typically entered and then to view the Parsfile for that screen. Figure 2 shows an example of the Parsfile, and the relevant information for the axis labels and units, as well as the table type.

```

1 PARSFILE
2 #FullDataName Powertrain: Engine`125 kW Engine`43 kW - 125 kW Engines
3 #RingCtrl0 CARPET
4 #CheckBox0 0
5 OPT_THROTTLE_DELAY 0
6
7 *3D_XLabel Throttle (-)
8 *3D_YLabel Engine speed (rpm)
9 *3D_ZLabel Engine torque (N-m)
10
11 #DiagramTwo0
12 *3D_DATA 10, 22 ! columns x rows
13 MENGINE_CARPET 2D_LINEAR 5
14 0, 0, 0.1, 0.15, 0.2, 0.35, 0.5, 0.7, 0.85, 0.95, 1
15 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

```

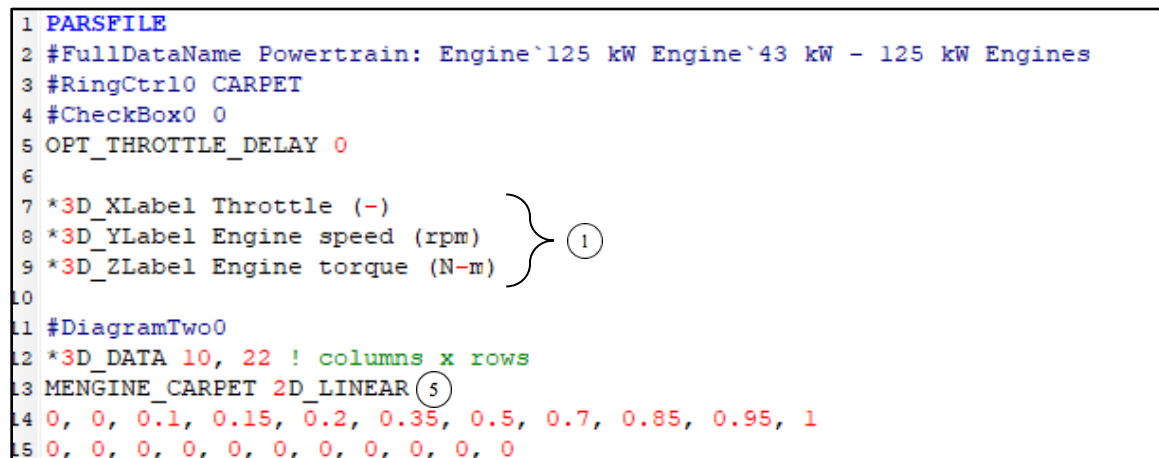


Figure 2: Parsfile within VS Browser, indicating input table characteristics

In Figure 2, the x-label, y-label, and z-label are *Throttle*, *Engine speed*, and *Engine torque*, respectively, while the corresponding units are '-', *rpm*, and *N-m* (1).

While most configurable functions are able to handle imported table data, there are some limitations. In particular, the Longitudinal Force table for one of the vehicle or trailer tires. Configurable functions which support some legacy data formats are currently unsupported by the VS Table Tool.

Example Specifics

This section will cover two examples involving the use of the VS Table Tool to create binary tables used in a VS simulation. The first example involves replacing an engine torque vs. speed map, while the second involves replacing road roughness data.

VS Table Engine Map: Torque vs. Speed

One configurable function that can access imported table data is the engine torque vs. engine speed map. This first example will walk through the process of moving the engine map data used with the C-Class Hatchback from the VS Browser to an imported binary table using the VS Table Tool.

Original VS Browser Data

Engine torque as a function of engine speed and driver load signal is stored in a two-dimensional map in the Powertrain: Engine library. For this example, the data used are for the C-Class

Hatchback 125 kW Engine Dataset, shown in (Figure 3). The configurable function specifies the table type as *2D linear interpolation & extrapolation* (2).

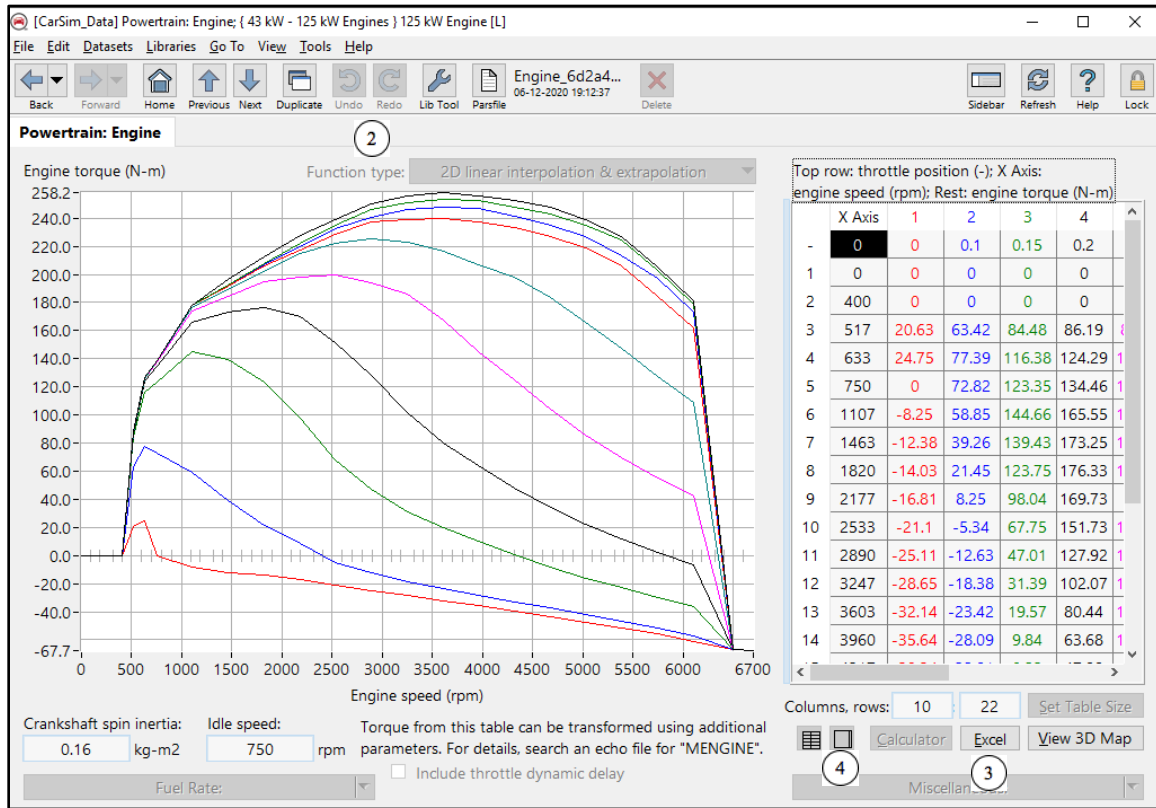


Figure 3: VS Browser map data for a 125 kW engine

One method to extract configurable function data from the VS Browser for use with the VS Table Tool is to click the *Excel* button located below the table input area in the user interface (3). This will automatically open Microsoft Excel with the data populated into a .csv file (Figure 4). Another way to extract the data is to select the *Scrollable Text Field* button (4), copy the comma separated data, and paste it into a spreadsheet or other text editor, then save that file in the .csv format.

	A	B	C	D	E	F	G	H	I	J	K	L
1	6	0	0	0.1	0.15	0.2	0.35	0.5	0.7	0.85	0.95	1
2		0	0	0	0	0	0	0	0	0	0	0
3		400	0	0	0	0	0	0	0	0	0	0
4		517	20.63	63.42	84.48	86.19	87.17	87.65	87.82	87.86	87.88	87.86
5		633	24.75	77.39	116.38	124.29	125.56	126.07	126.25	126.27	126.27	126.27
6		750	0	72.82	123.35	134.46	137.46	138.54	138.94	138.98	138.97	138.96
7		1107	-8.25	58.85	144.66	165.55	173.84	176.68	177.74	177.84	177.81	177.74
8		1463	-12.38	39.26	139.43	173.25	184.29	189.29	191.74	192.31	192.68	196.35
9		1820	-14.03	21.45	123.75	176.33	194.74	201.9	205.74	206.78	207.55	212.85
10		2177	-16.81	8.25	98.04	169.73	198	214.5	216.98	219.45	221.93	227.4
11		2533	-21.1	-5.34	67.75	151.73	199.65	222.5	228.53	232.7	235.13	239.25
12		2890	-25.11	-12.63	47.01	127.92	194.27	224.92	237.69	240.9	245.85	249.98
13		3247	-28.65	-18.38	31.39	102.07	186.21	222.75	239.25	245.85	250.8	255.75
14		3603	-32.14	-23.42	19.57	80.44	167.53	216.15	240.08	247.5	253.28	258.23
15		3960	-35.64	-28.09	9.84	63.68	145.29	207.19	237.6	246.68	252.45	255.75
16		4317	-39.34	-32.81	0.23	47.88	125.2	197.75	233.48	241.73	247.5	252.4
17		4673	-43.19	-37.51	-8.31	34.33	104.22	183.59	226.88	235.13	243.38	247.5
18		5030	-47.28	-42.14	-15.9	22.49	85.54	165.8	218.63	226.88	235.13	239.25
19		5387	-51.39	-46.87	-22.94	12.04	69.6	147.15	205.7	213.68	224.7	227.07
20		5743	-55.99	-51.8	-29.76	2.35	55.16	127.73	184.51	197.1	203.01	205.34
21		6100	-61.01	-57.01	-36.57	-6.8	41.98	108.88	161.55	173.27	178.77	180.88
22		6500	-66.83	-66.83	-66.83	-66.83	-66.83	-66.83	-66.83	-66.83	-66.83	-66.83
23		6700	-67.65	-67.65	-67.65	-67.65	-67.65	-67.65	-67.65	-67.65	-67.65	-67.65
24												
25												

Figure 4: Engine map data populated into a Microsoft Excel .csv file

Use of the VS Table Tool

Once the .csv file with the 125 kW engine map data is saved, the VS Table Tool can be used to create the binary and header files used by the VS Solver. For more information about how to use the VS Table Tool, please see the VS Table Tool documentation. The *Creating a VS Table* section of the [VS Table Tool Introduction](#) help document, highlights the steps used for this example. A sample command line entry that could work for this is shown below:

```
vs_table_tool.exe -file EngineTable_125kw.csv -title "Engine
Powertrain" -type 2D_LINEAR -xlabel Throttle -xunits - -ylabel
Engine Speed -yunits rpm -flabel Engine Torque -funits N-m -
firstrowcolvalue
```

Note The units here match those specified in Figure 2 (1), the type is 2D_Linear (5), and the -firstrowcolvalue option is selected, since the first row of the .csv file in Figure 4, represents the throttle values. In this case (6) is used for the Y axis values for each column. An explanation for the Y Value can be found in the [VSOutput VSTable API](#) document.

Loading VS Table data

After the VS Table Tool runs successfully, the next step is to create a new Dataset which will be used to load the external table information. For the '*Internal engine model*' link in the *Powertrain: Front-Wheel Drive* library, clicking the dropdown arrow and select the '*Link to New Dataset*' (7) option (Figure 5). Click the newly created link to enter the new dataset.

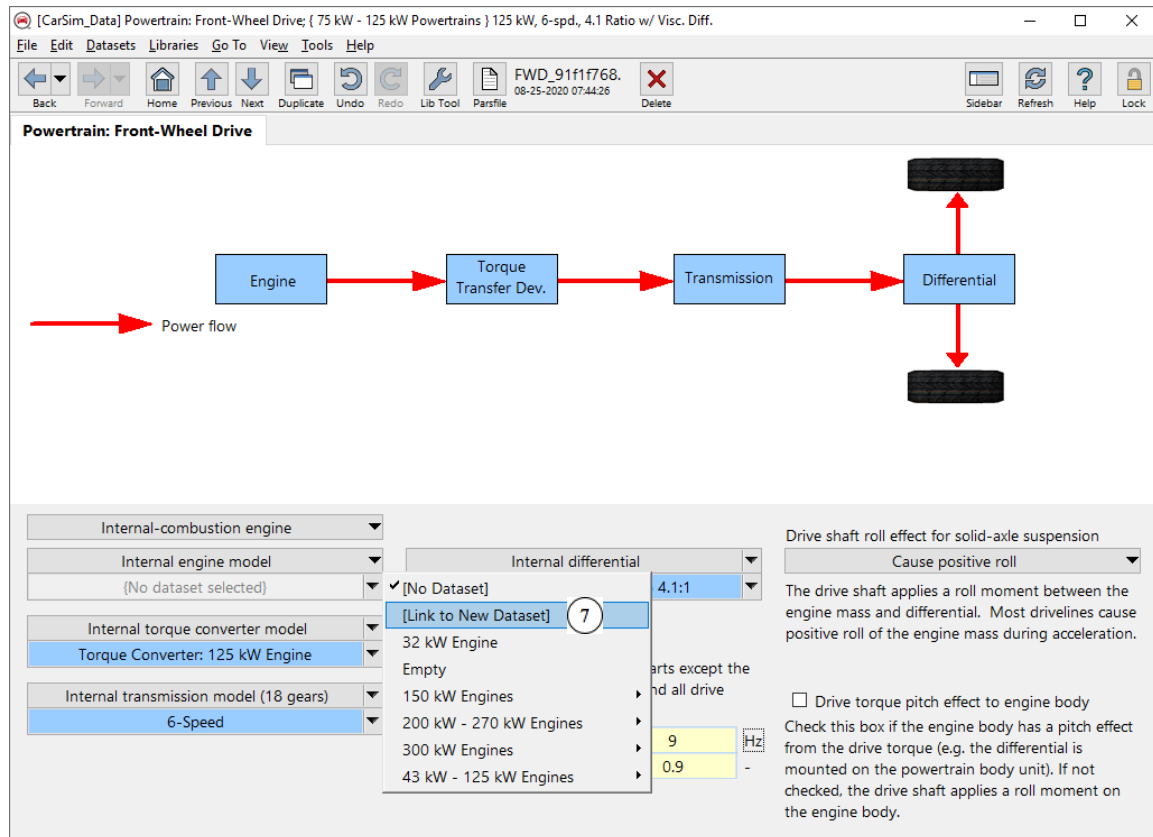


Figure 5: New Dataset creation within the '*Powertrain: Front-Wheel Drive* screen'

The Powertrain: Engine library has a *Miscellaneous* link (8) in the lower right-hand corner. Use that link to select the (9) Generic VS Commands library. A Generic VS Commands dataset will be used to load the newly created VS Table data (Figure 6). Link to a new Generic VS Commands dataset.

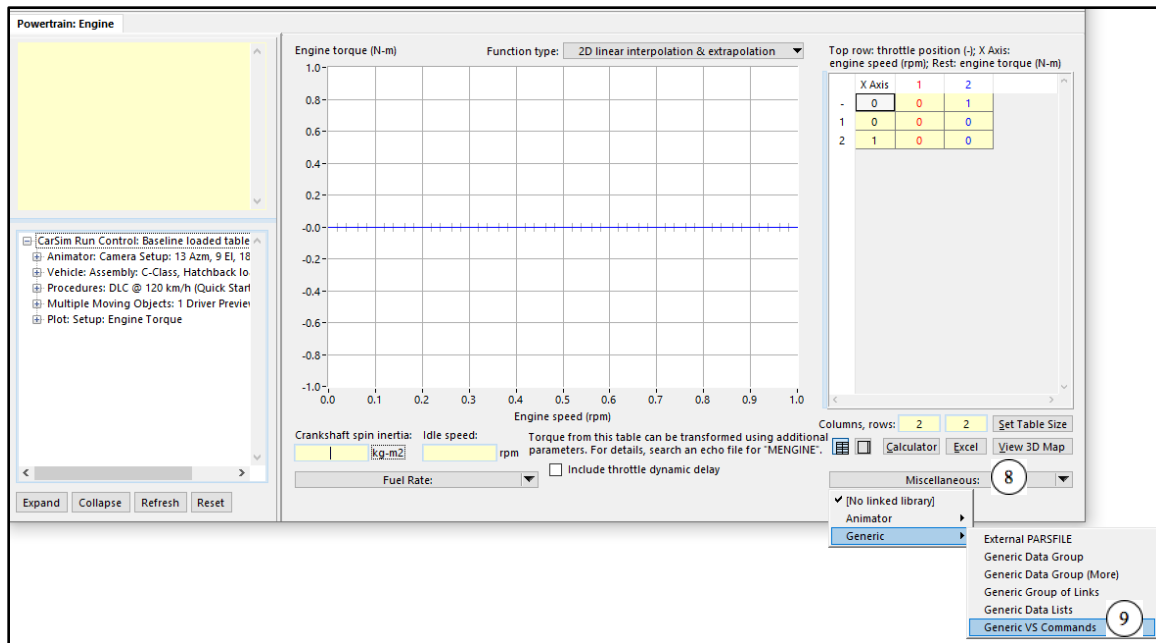


Figure 6: VS Commands Dataset selection on the Powertrain: Engine screen

Creating VS Commands

In the new Generic VS Commands Dataset, enter the VS Commands seen in Figure 7 to include the VS Table data in the simulation.

Generic VS Commands

Dataset label or code (optional):

Use the field below for data or commands.

```

! Load 125 kW Engine Data for 4.1 Ratio, 6-speed, w/visc. diff.

! Binary file from VS Table Tool is loaded to memory
LOAD_TABLE_FILE Extensions\VsTableTool\EngineTable_125kw.vstb 10

! Create data carpet from loaded VS Table data
FILE_TO_CARPET MENGINE_CARPET 11

! Load VS Table (carpet) data in the Echo file
ECHO_LOADED_TABLE MENGINE_CARPET 12

! Used to make files copy to Consolidated Pars File Export
*CPAR_INCLUDE Extensions\VsTableTool\EngineTable_125kw.vstb
*CPAR_INCLUDE Extensions\VsTableTable\EngineTable_125kw.vsb 13

!Engine idle speed
AV_ENG_IDLE = 750; 14

!Crankshaft spin inertia
IENG = 0.16; 15

```

Figure 7: VS Commands to load engine map table data generated by the VS Table Tool

The following steps are used in Figure 7 to load the VS Table data

1. The `LOAD_TABLE_FILE` command, used with the relative path from the root of the database to the VS Table header (.vstb) file will read both the header and binary file into memory, making the information available to the VS Solver (10).
2. The `FILE_TO_CARPET` command instructs the solver to use the data in the previously indicated file as a 2-D Carpet for the indicated keyword (11). This keyword can either be the name of a new User-Defined configurable function or an existing configurable function such as `MENGINE_CARPET`. Note that if the keyword is for a 1-D table, the command would be `FILE_TO_TABLE`.
3. The `ECHO_LOADED_TABLE` VS Command, along with the configurable function keyword will cause the loaded data to be written into the echo file (12). Loaded tables are not included in the echo file by default to improve readability for cases where the loaded data is extremely large.
4. Because the VS Table data is not embedded directly within the Parsfile, the `*CPAR_INCLUDE` VS Command is used with the path and filename for both the header and binary files that were created with the VS Table Tool (13) to instruct the browser to include both files into a Consolidated Parsfile.
5. With this particular VS Table data only representing the Engine Map, additional representative values for the Engine Idle Speed `AV_ENG_IDLE` (14), and the Crankshaft Spin Inertia (`IENG` (15), need to be entered as well into the VS Commands Dataset.

Note More detail on the various VS Commands that can be used to load and process VS Table data or VS Generic Tables can be found in the [Generic Table](#) Help document in the Generic Data Screens Help menu.

Also, more detail about the inclusion of support files such as VS Table data into Consolidated Parsfiles can be found in the [VS Commands](#) reference manual

Running the Example with VS Table Data

After entering the VS Commands to use the data from the VS Table file, return to the Run Control Screen, and click the *Run Math Model* button (16), to run the example with the newly loaded VS Table Engine Map data.

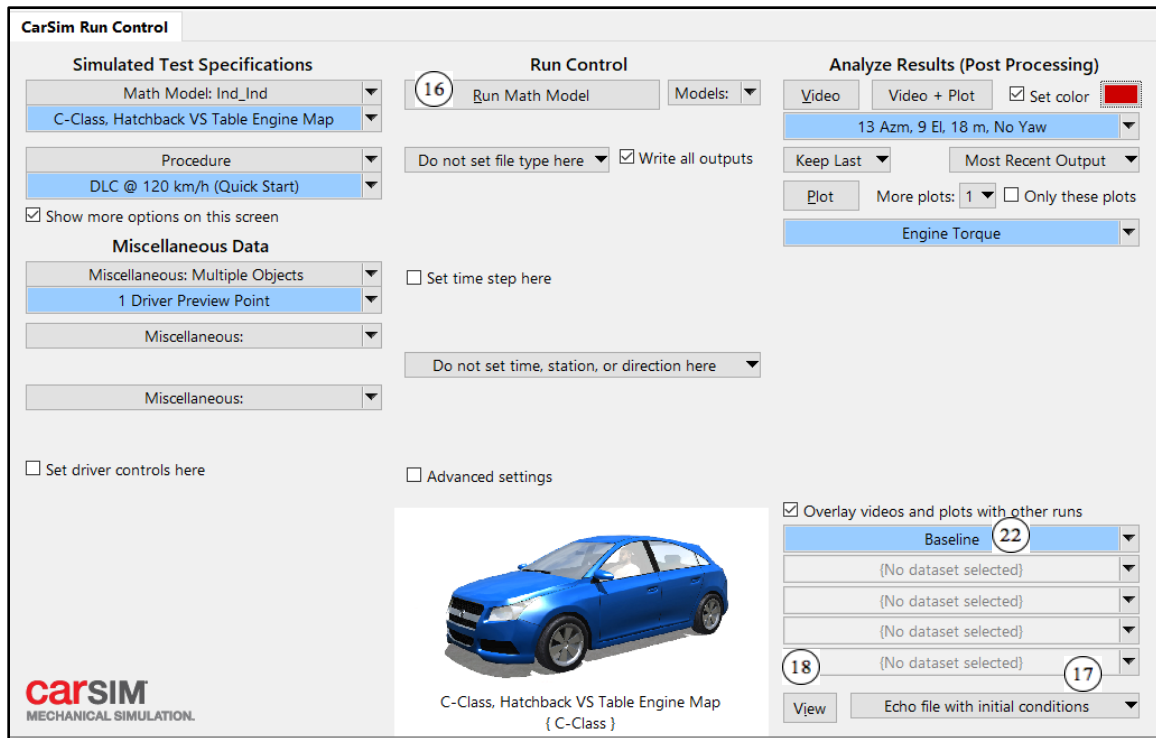


Figure 8: Run Control Screen with the Baseline DLC example including VS Table Engine Map data loaded with the VS Table Tool

The example used for this demonstration is the Baseline vehicle from the Quick Start Guide, running the double lane change (DLC) maneuver. This example in the CarSim shipped Datasets also has a 125 kW engine with data that match the newly created Engine Map data using the VS Table tool. To make sure the VS Table Engine Map data have correctly been loaded and run, it is useful to check the Echo file (17), by clicking the *View* button with the *Echo file with initial conditions* menu option selected (18). When using the *Find* feature and entering the keyword for the configurable function which uses the loaded data (MENGINE_CARPET in this case), the VS Table data should appear as expected, matching both the data from the .csv file used to create the VS Table data, and the echo file for the Quick Start Guide Baseline (19) (Figure 9).

```

3665 ! MENGINE: Engine torque applied to crankshaft. Engine torque can be a nonlinear
3666 ! CARPET function of normalized throttle and engine speed or a function of engine
3667 ! speed multiplied by a function MENGINE_THROTTLE of normalized throttle (CONSTANT,
3668 ! COEFFICIENT, or TABLE). Alternatively, a custom equation can be defined at runtime.
3669 ! Engine torque from the calculation can be adjusted with MENGINE_GAIN and
3670 ! MENGINE_OFFSET. Engine speed used in the calculation can be adjusted with
3671 ! SPIN_SCALE_MENGINE and SPIN_START_MENGINE. Normalized throttle used in the
3672 ! calculation can be adjusted with THROTTLE_SCALE_MENGINE and THROTTLE_START_MENGINE.
3673 LOAD_ERD_FILE Extensions\VsTableTool\EngineTable_l25kw.vstb ;
3674 ERD TO CARPET MENGINE_CARPET
3675 ECHO_LOADED_TABLE MENGINE_CARPET
3676
3677 ! 2D table: row 1 has "0" (place holder) followed by 11 values of normalized
3678 ! throttle (-). Other rows have engine speed (rpm) followed by 11 values of engine
3679 ! torque (N-m).
3680 MENGINE_CARPET 2D_LINEAR (19)
3681 0, 0, 0, 0.1, 0.15, 0.2, 0.35, 0.5, 0.7, 0.85, 0.95, 1
3682 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
3683 400, 41.88790205, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
3684 517, 54.1401134, 20.63, 63.42, 84.48, 86.19, 87.17, 87.65, 87.82, 87.86, 87.88, 87.86
3685 633, 66.28760499, 24.75, 77.39, 116.38, 124.29, 125.56, 126.07, 126.25, 126.27, 126.27, 126.27
3686 750, 78.53981634, 0, 72.82, 123.35, 134.46, 137.46, 138.54, 138.94, 138.98, 138.97, 138.96
3687 1107, 115.9247689, -8.25, 58.85, 144.66, 165.55, 173.84, 176.68, 177.74, 177.84, 177.81, 177.74
3688 1463, 153.2050017, -12.38, 39.26, 139.43, 173.25, 184.29, 189.29, 191.74, 192.31, 192.68, 196.35
3689 1820, 190.5899543, -14.03, 21.45, 123.75, 176.33, 194.74, 201.9, 205.74, 206.78, 207.55, 212.85
3690 2177, 227.9749069, -16.81, 8.25, 98.04, 169.73, 198, 214.5, 216.98, 219.45, 221.93, 227.4
3691 2533, 265.2551397, -21.1, -5.34, 67.75, 151.73, 199.65, 222.5, 228.53, 232.7, 235.13, 239.25
3692 2890, 302.6400923, -25.11, -12.63, 47.01, 127.92, 194.27, 224.92, 237.69, 240.9, 245.85, 249.98
3693 3247, 340.0250449, -28.65, -18.38, 31.39, 102.07, 186.21, 222.75, 239.25, 245.85, 250.8, 255.75
3694 3603, 377.3052777, -32.14, -23.42, 19.57, 80.44, 167.53, 216.15, 240.08, 247.5, 253.28, 258.23
3695 3960, 414.6902303, -35.64, -28.09, 9.84, 63.68, 145.29, 207.19, 237.6, 246.68, 252.45, 255.75
3696 4317, 452.0751829, -39.34, -32.81, 0.23, 47.88, 125.2, 197.75, 233.48, 241.73, 247.5, 252.4
3697 4673, 489.3554157, -43.19, -37.51, -8.31, 34.33, 104.22, 183.59, 226.88, 235.13, 243.38, 247.5
3698 5030, 526.7403683, -47.28, -42.14, -15.9, 22.49, 85.54, 165.8, 218.63, 226.88, 235.13, 239.25
3699 5387, 564.1253208, -51.39, -46.87, -22.94, 12.04, 69.6, 147.15, 205.7, 213.68, 224.7, 227.07
3700 5743, 601.4055537, -55.99, -51.8, -29.76, 2.35, 55.16, 127.73, 184.51, 197.1, 203.01, 205.34
3701 6100, 638.7905062, -61.01, -57.01, -36.57, -6.8, 41.98, 108.88, 161.55, 173.27, 178.77, 180.88
3702 6500, 680.6784083, -66.83, -66.83, -66.83, -66.83, -66.83, -66.83, -66.83, -66.83, -66.83, -66.83
3703 6700, 701.6223593, -67.65, -67.65, -67.65, -67.65, -67.65, -67.65, -67.65, -67.65, -67.65, -67.65
3704 ENDTABLE

```

Figure 9: Confirmation of the VS Table Engine Map data (19) in the Echo file

The same search process using the *Find* feature in the Echo file can be used to assure that the Crankshaft Spin Inertia, and the Engine Idle Speed are as expected (Figure 10).

```

561 !-----
562 ! ENGINE
563 !-----
564 INSTALL_ENGINE          ! VS Command to install an engine
565
566 OPT_ENGINE_INTERNAL 1 ! Engine model: 1 -> internal, 0 -> external [I]
567 OPT_ENGINE_RUNNING 1 ! [D] Is engine running? 1 -> yes, 0 -> no
568 OPT_THROTTLE_DELAY 0 ! Use throttle time constants: 0 -> no, 1 -> yes
569 AV_ENG_IDLE          750 ; rpm ! Engine idle speed [I]
570 ENGINE_ESC_PG         5 ; 1/s ! [D] P gain of ESC engine torque control
571 ENGINE_ESC_IG         0.5 ; 1/s^2 ! [D] I gain of ESC engine torque control
572 ENGINE_STALL_DAMP     0.2 ; N-m-s/deg ! [D] Damping rate of the stalled engine
573 IENG                  0.16 ; kg-m^2 ! Spin inertia of engine crankshaft [I] (20)
574 ITC_INPUT_SHAFT 0.015 ; kg-m^2 ! Spin inertia of input shaft of torque converter (21)

```

Figure 10: Confirmation of the Engine Crankshaft Spin Inertia (20) and Engine Idle Speed (21) within the Echo file

VS Browser vs. VS Table Output Comparison

Once the VS Table Engine Map data, Crankshaft Spin Inertia, and Engine Idle Speed have been confirmed, the output Animation and Plots of the Baseline example with VS Table data can be seen and compared with the standard Baseline example from the shipped data (22) (Figure 11)

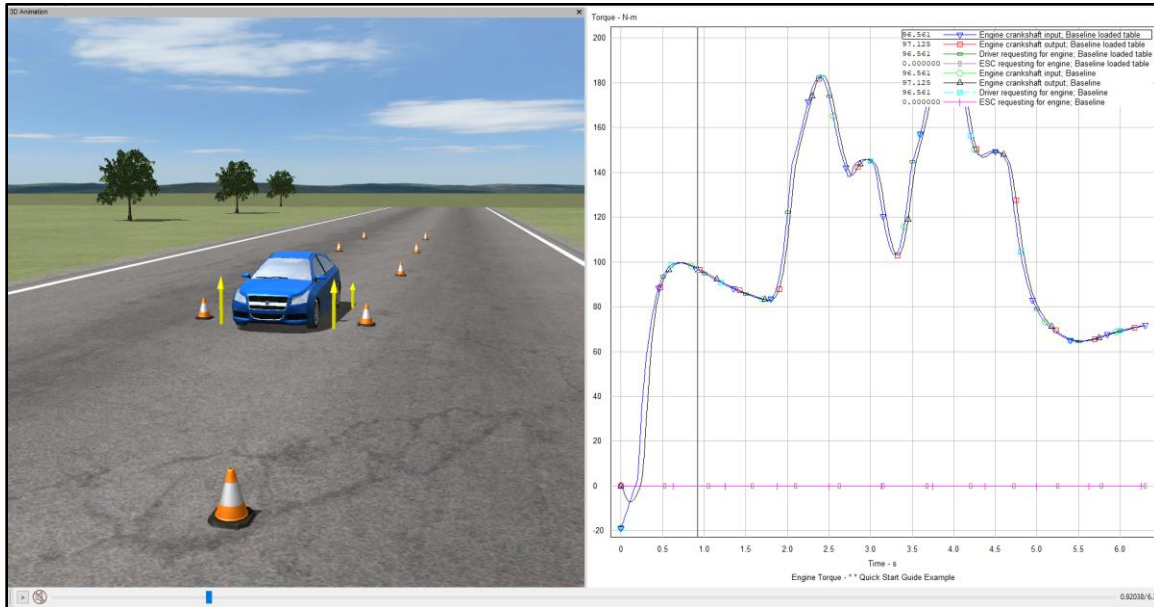


Figure 11: Comparison of the Baseline example with VS Table data and standard shipped data

This plot output comparison of the VS Browser Engine Map versus the VS Table Engine Map data in Figure 11 is the final confirmation to assure that the Engine Map data in this example were correctly converted from .csv data to VS Table data to be used in the CarSim simulation example.

VS Table Road Roughness Example

In CarSim and TruckSim there is another example included which uses VS Table data, only this application uses it for the Road: Surface Roughness profile. This example Dataset can be found in the **Road Networks: 3D Geometry and Terrain** Category, with the title **Baseline (loaded road profile)**.

Original VS Browser Data

The original dZ road roughness data are used in a separate example that utilizes the Road: Surface Roughness dataset in the VS Browser. That example is found in the **Road Networks: 3D Geometry and Terrain** Category, with Dataset name **Baseline (nominal profile)** and the loaded Roughness Profile (23) (Figure 13) is found in the **Road: 3D Surface (All Properties)** library located in the **Path and Road Surfaces** category (Figure 12).

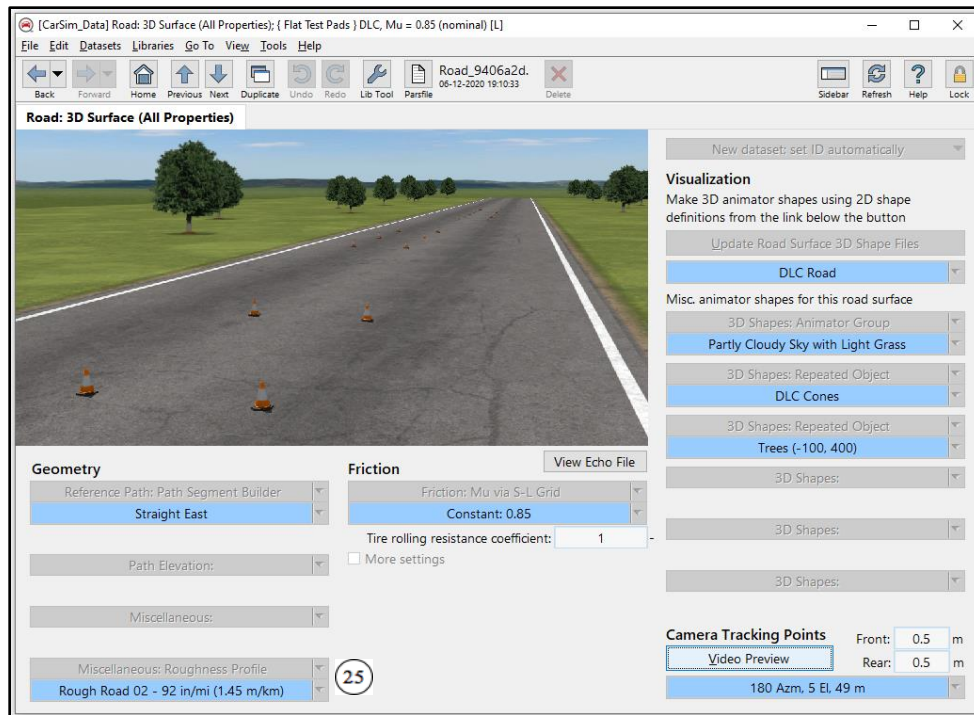


Figure 12: Road screen where Road Roughness VS Browser data are imported

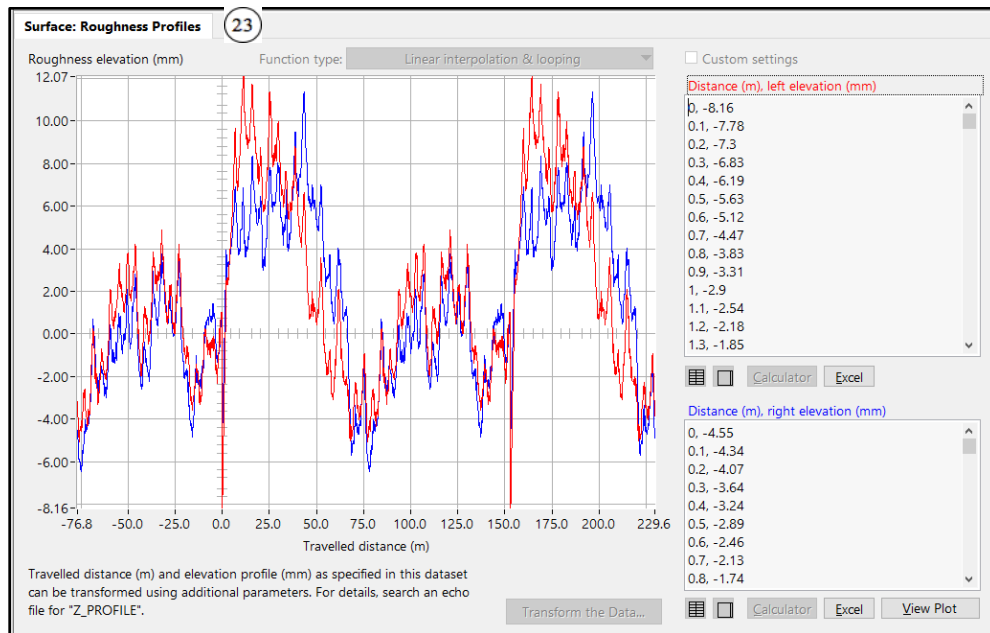


Figure 13: Road Roughness data used in the VS Browser for left and right sides

Loading VS Table Data

In this example the VS Table data are also imported into **Road: 3D Surface (All Properties)** screen found in the **Path and Road Surfaces Library** (Figure 14) (24).

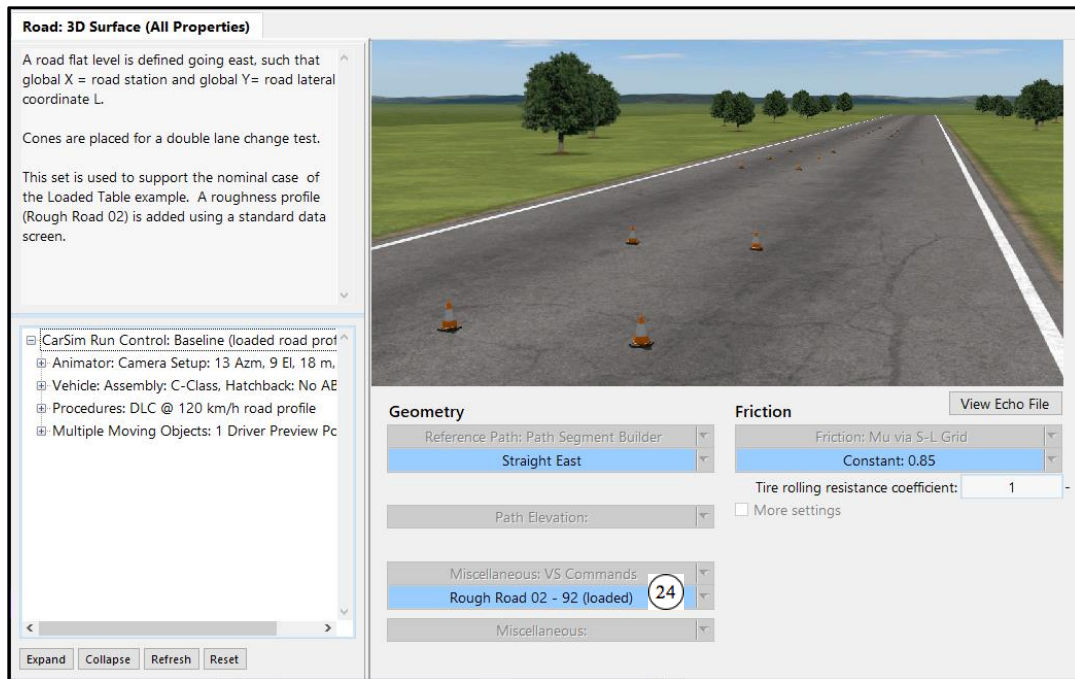


Figure 14: Road screen were Road Roughness VS Table data are imported

Like the Engine Map data from the **VS Table Engine Map: Torque vs. Speed** example, the surface roughness VS Table data are loaded using VS Commands in a Generic VS Commands Dataset (25) (Figure 15).

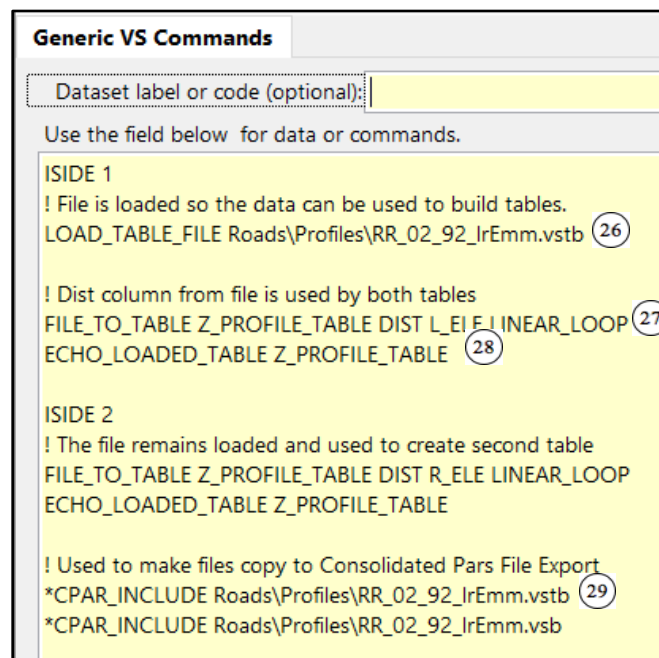


Figure 15: VS Commands Dataset with a Road: Surface Roughness profile loaded from VS Table data

Like many places in the simulation, the context for the loaded roughness profile matters. The road context (IROAD) is set by the Road: 3D Surface (All Properties) dataset which is the parent for the Generic VS Commands dataset which loads the roughness profile. The surface roughness profile is a wandering profile that is specified per side of the vehicle, so the vehicle side (1 = Left, 2 = Right) context must be specified in the VS Commands dataset that loads the profile information. This is done with the ISIDE keyword. The front and rear axles are handled automatically by the solver, with the rear tires encountering the data one wheelbase behind the front tires.

Note More detail regarding VS Roads, road roughness profiles, parameters, configurable functions, and keywords, can be found in the [Paths and Road Surfaces](#) Help document in the Paths, Road Surfaces, and Scenes Help menu.

Figure 15 details the VS Commands used to load the 1-D profile information. The VS Commands used are detailed below:

1. Set the profile context to the left-side tires using the keyword `ISIDE` and giving it a value of `1`.
2. The `LOAD_TABLE_FILE` command uses the relative path information to the VS Table header file and instructs the solver to load the header and binary table data into memory and make it accessible to the VS Solver (26).
3. The `FILE_TO_TABLE` VS Command instructs the solver to take the previously loaded table data and use it as a 1-D table for the configurable function with the given keyword; in this case: `Z_PROFILE_TABLE` (27). Unlike the `FILE_TO_CARPET` function used earlier, the `FILE_TO_TABLE` command supports extracting multiple channels from a single loaded table. The parameters for the X and Y data in this case, the name of the x-channel `DIST`, and the name of the y-channel `L_ELE`, are needed in addition to the interpolation/extrapolation type. The type for a surface roughness profile is: `LINEAR_LOOP`.

Note The VS Table Tool input to make this dZ road roughness example, uses `-flabel Elevation, funits mm, and xlabel DIST, xunits 'm', with -colname XLabel -colname L_ELE -colname R_ELE`

4. After the table file has been created, assure that the new loaded data appear in the Echo file by using the `ECHO_LOADED_TABLE` VS Command with the same configurable function keyword (28) that was used in 3.
5. Set the profile context to the right-side tires using the keyword `ISIDE` and giving it a value of `2`.
6. The commands from items 3 and 4 are repeated, only this time the channel `R_ELE` is used to obtain the information for the right-hand side.

Note It is not a problem that the same table name Z_PROFILE_TABLE is used for the left and right side data. It is important, however, that the sides of the vehicle were indicated with ISIDE 1 and ISIDE 2 before any table data were loaded. This indicates to the VS Solver that different data are loaded for each side of the vehicle

- As was done for the **VS Table Engine Map: Torque vs. Speed** example, the *CPAR_INCLUDE VS Command is used with the path and filenames of the VS Table Tool header and binary files, to include the table data files in a consolidated Parsfile generated from the run (29).

VS Browser vs. VS Table Output Comparison

The **Baseline (loaded road profile)** example with surface roughness is now ready for execution from the Run Control screen (30) (Figure 16).

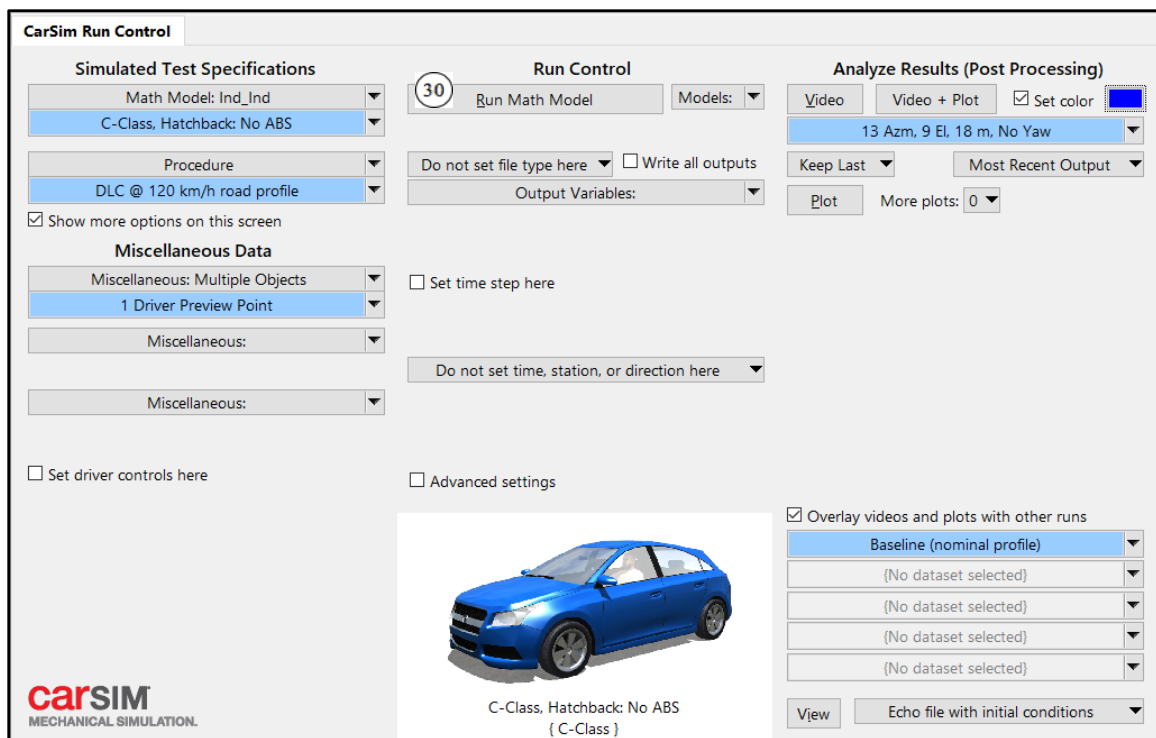


Figure 16: Run Control Screen for the Baseline (loaded profile)

The results of the **Baseline (loaded road profile)** example can be directly compared to the **Baseline (nominal)** example, and it is clear that the (vertical force) outputs are identical (Figure 15).

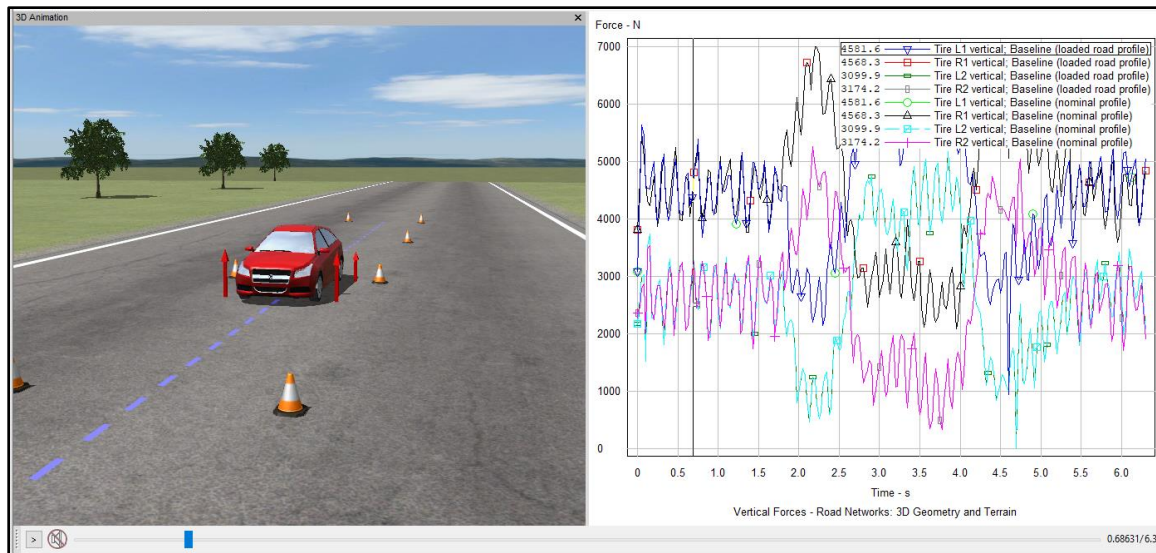


Figure 17: Output plot comparison between the VS Browser and VS Table versions of the Baseline DLC with dZ Road Roughness profile attached

Similar to the VS Table Engine Map: Torque vs. Speed example, for this VS Table Road Roughness Example VS Table data can be confirmed by comparing the Echo file table data from the **Baseline (loaded road profile)** to that of the **Baseline (nominal profile)** (31). Using the Compare tool found in the ConTEXT editor (default when clicking the (32) View button of the Run Control screen), shows the VS Table data (left) in the **Baseline (loaded road profile)** Echo file, and the **Baseline (nominal profile)** Echo (right) (Figure 18)

<pre> 7145: ENDTABLE 7146: Z_PROFILE_GAIN(1) 1 ! Gain multiplied with calculated value to get elevation 7147: ! increment 7148: Z_PROFILE_OFFSET(1) 0 ; mm ! Offset added (after gain) to get elevation increment 7149: SSTART_Z_PROFILE(1) 0 ; m ! Offset subtracted from distance travelled 7150: SSCALE_Z_PROFILE(1) 1 ! Scale factor divided into (distance travelled - 7151: ! SSTART_Z_PROFILE) 7152: LOAD_ECHO_FILE Roads\Profile\VR D2_R2_12mm_vesh : 7153: ERD_TO_TABLE Z_PROFILE_TABLE(2) DIST_R_ELE_LINEAR_LOOP 7154: ECHO_LOADED_TABLE Z_PROFILE_TABLE(2) 7155: 7156: ! [D] ID table: col 1 = distance travelled (m), col 2 = elevation increment (mm) 7157: Z_PROFILE_TABLE(2) LINEAR_LOOP ! linear interpolation, repeat in loop 7158: 0, -4.550000191 7159: 0.1000000015, -4.340000153 7160: 0.2000000003, -4.070000172 7161: 0.3000000119, -3.640000105 7162: 0.4000000006, -3.240000001 7163: 0.5, -2.890000105 7164: 0.6000000230, -2.4600000030 7165: 0.6999999801, -2.130000114 7166: 0.8000000119, -1.740000001 7167: 0.8999999762, -1.370000005 7168: 1, -0.8799999952 7169: 1.1000000024, -0.4499999801 7170: 1.2000000040, 0 7171: 1.2999999952, 0.3400000036 7172: 1.3999999976, 0.6600000022 7173: 1.5, 0.5499999881 7174: 1.6000000024, 1.3300000043 7175: 1.7000000040, 1.8500000024 7176: 1.7999999952, 2.2200000029 7177: 1.8999999976, 2.4700000029 7178: 2, 2.759999999 7179: 2.0999999905, 3.2999999952 7180: 2.2000000040, 3.9500000040 7181: 2.2999999952, 3.5899999914 7182: 2.4000000095, 3.0499999952 </pre>	<pre> 7142: ENDTABLE 7143: Z_PROFILE_GAIN(1) 1 ! Gain multiplied with calculated value to get elevation 7144: ! increment 7145: Z_PROFILE_OFFSET(1) 0 ; mm ! Offset added (after gain) to get elevation increment 7146: SSTART_Z_PROFILE(1) 0 ; m ! Offset subtracted from distance travelled 7147: SSCALE_Z_PROFILE(1) 1 ! Scale factor divided into (distance travelled - 7148: ! SSTART_Z_PROFILE) 7149: 7150: ! ID table: col 1 = distance travelled (m), col 2 = elevation increment (mm) 7151: Z_PROFILE_TABLE(2) LINEAR_LOOP ! linear interpolation, repeat in loop 7152: 0, -4.55 7153: 0.1, -4.34 7154: 0.2, -4.07 7155: 0.3, -3.64 7156: 0.4, -3.24 7157: 0.5, -2.89 7158: 0.6, -2.46 7159: 0.7, -2.13 7160: 0.8, -1.74 7161: 0.9, -1.37 7162: 1, -0.88 7163: 1.1, -0.45 7164: 1.2, 0 7165: 1.3, 0.34 7166: 1.4, 0.66 7167: 1.5, 0.95 7168: 1.6, 1.33 7169: 1.7, 1.85 7170: 1.8, 2.22 7171: 1.9, 2.47 7172: 2, 2.76 7173: 2.1, 3.3 7174: 2.2, 3.85 7175: 2.3, 3.59 7176: 2.4, 3.05 7177: 2.5, 2.47 7178: 2.6, 3.14 7179: 2.7, 3.17 </pre>
--	---

Figure 18: Comparison of the dZ road roughness data in the Echo files from the VS Table data (left), and the VS Browser data (right)

Note	There is a floating point precision difference between the VS Table data (left), and VS Browser data (right) in Figure 18. This comes from the VS Table data containing floating point doubles, whereas the VS Browser data contains ASCII numerical text in a table. This difference can be understood more with the explanation by (Goldberg, 1991)
-------------	---

Summary

In this Technical Memo, the VS Table Tool was highlighted as an executable command line utility that can convert .csv data from vehicle and environmental components into a pair of header and binary files that can be loaded by the VS Solver. Two examples were provided that explained how to create the header and binary files from .csv data using the VS Table Tool. These header and binary files were then included in a CarSim simulation using VS Commands.

Both engine map data and surface roughness data were shown in the examples within this Technical Memo. When the two examples' simulation output performance was compared between loaded VS Table data versus VS Browser data, Echo file and VS Visualizer plots indicated that the performance was identical. Applications of the ability to load external data created by the VS Table Tool have been noted; one particular usage is with automation of several different versions of one vehicle or environmental component represented by VS Table data.

References

Goldberg, D. (1991). What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys (CSUR)*, 23(1), 5-48.