# The Benefits of Scripting for CAE Engineers: How a little can go a long

Gavin Newlands/Miles Thornton

Arup

## 1 Introduction

The pressure on a CAE engineer can be great at times, with tight schedules and demands for results. Sometimes the exact tools you require are not available – but what if you could create them yourself? Writing a script or tool to speed up a process may seem daunting, or not worth the effort. Why would you spend a day writing a script to do something when it will take you a couple of hours to actually do the task? The answer is often it will not be the only time you will do the "something". It might be once, or it may be 10's or 100's of times.
This is where scripting can really help you. But it is not confined to "speeding things up". Scripting can be used in a number of areas to aid the CAE engineer:

- Custom tools to do exactly what you want.
- Custom checks specific to company guidelines to make sure your model is up to scratch.
- Model manipulation tools.
- Specify a process for others to follow (can reduce errors).
- And many more…

This paper looks at how scripting can have a part to play in the everyday work of a CAE engineer. Using the Oasys LS-DYNA Environment software we will demonstrate the benefits of scripting through a number of examples. We will also show that it is actually very quick and easy to learn and to write useful scripts.

## 2 Introduction to scripting – tools and resources

Today, most pre and post-processers used by CAE engineers have some form of scripting capability. These are included for various reasons:

- Engineers would like to access model data in different ways to suit their needs.
- To enhance or introduce new tools on top of the native tools in the software.
- Add abilities to read/write data not natively supported by the software.
- Add company specific tools/processes.
- Check models and results in particular ways.
- Speed up tasks through automation.

Any there are many more as well.

One issue with scripting is it can be quite daunting when you first start out. You can start reading a chapter in a users‘ manual on scripting and quickly lose interest and motivation. So how do you get over that hurdle?

The first thing to do is to convince yourself that it is worth the time to learn this new skill. Yes, it may take up some of your time now, but the benefits in the future (in terms of time saving) could be substantial.

When you know you want to dedicate some time to scripting, where do you start? Most scripting languages in pre and post-processers are based on a widely available language, for example Python or Javascript. So the temptation could be to jump in to buying a book, or finding an online course on that language and starting to learn. As a first step this is probably not recommended. You could spend

a lot of time learning things that are not applicable to the software you are using, or to the applications and tools you want to develop.
One good place to start is with tutorials and worked examples provided with the software you use. This way you can start straight away modifying and creating scripts. Once you are used to the scripting language somewhat you can start extending your knowledge in the core language by using reference books and online resources.
One thing that is advisable when creating a new script is to always have a pool of example scripts to use as a guide – these are generally included with the software you are working in, or with tutorials for the software. This makes it much easier to create a script. Add one line at a time, run the script and make sure it works then continue. Don’t try to write 100’s of lines of code before running or you could spend a long time tracking down problems.
When debugging scripts, often writing print statements into your code helps to track issues, or sometimes debuggers are available to allow you to investigate problems.
What you should find is that a few days of leaning, perhaps a couple of hours here and there, should be enough to gain knowledge that will allow you to do many things with scripting.

## 3 Pre-Processing Examples

The following are a series of examples for how scripting can be used during pre-processing of LS-DYNA models:

- Model assembly/multiple models.
- Reading/writing external data.
- Custom checking.
- Automation.

### 3.1 Model Assembly/Multiple models

Building models, and in-particular building multiple models is an obvious use of scripting. Yes, you can use pre-processors such as PRIMER to build multiple models using their inbuilt tools, but you may want to do something a bit different, or maybe something specific to the company you work in, or tailored towards the information you have to hand.
Let‘s take the example of information in a spreadsheet. You may have a series of values and parameters you wish to use in your model – the example below shows a speadsheet containing one row per analysis:

| Point | PARAMA | PARAMB | PARAMC | PARAMD | PARAME | PARAMF | PARAMG | PARAMH | PARAMI | PARAMJ | PARAMK |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 12 | -1.2 | 0 | 100 | 201 | 303 | 4.667 | 9.5 | 401 | 37 | 50 |
| 2 | 6 | -3.6 | 7 | 106 | 201 | 304 | 2.533 | 9.5 | 401 | 33 | 52 |
| 3 | 2 | 3 | 6 | 100 | 202 | 303 | 3.067 | 9.5 | 403 | 33 | 48 |
| 4 | 12 | 3 | 3 | 110 | 203 | 304 | 3.333 | 11.5 | 402 | 39 | 52 |
| 5 | 6 | 0.6 | -2 | 112 | 201 | 301 | 5.733 | 11.5 | 403 | 37 | 50 |
| 6 | 4 | -3 | 2 | 102 | 202 | 304 | 5.467 | 9.5 | 404 | 33 | 48 |
| 7 | 8 | 1.8 | -5 | 196 | 201 | 301 | 4.667 | 9.5 | 404 | 35 | 50 |
| 8 | 12 | -2.4 | 4 | 198 | 201 | 304 | 4.4 | 10 | 404 | 35 | 52 |
| 9 | 10 | -1.2 | -2 | 104 | 203 | 302 | 6 | 10 | 401 | 35 | 52 |
| 10 | 6 | -2.4 | 6 | 110 | 202 | 304 | 2 | 10 | 403 | 37 | 50 |

What you would like is to apply the data in the spreadsheet to a base model to create multiple models, one per row of data. As you can see, there are a variety of values you could want to set here. Initial velocity, time-to-fire of airbags and belts, failures in material models etc. The values may also form part of an include file name. For example, an include file name could contain the termination time mentioned in the control card held within.
So in this example, for each row of data in the spreadsheet, we would want to take the values and apply them to the model in some way. This could be substituting text in a master file (the example of the include file with a value in the name). Or it could be inserting a value into a particular field of a card. Or it could be creating or modifying parameters that already exist in the model:

Examples from the table above:

PARAMK – This is a velocity. The Initial velocity card can be modified directly using this value here. Any vector components and units conversion can be taken into account.

PARAME – This is a reference to an include file version. For example, this could be a BIW, and you want to use different versions in different runs. Text substitution can be used here.

PARAMH – This is failure information for a particular material card. Values can be varied to see the effect on overall results.

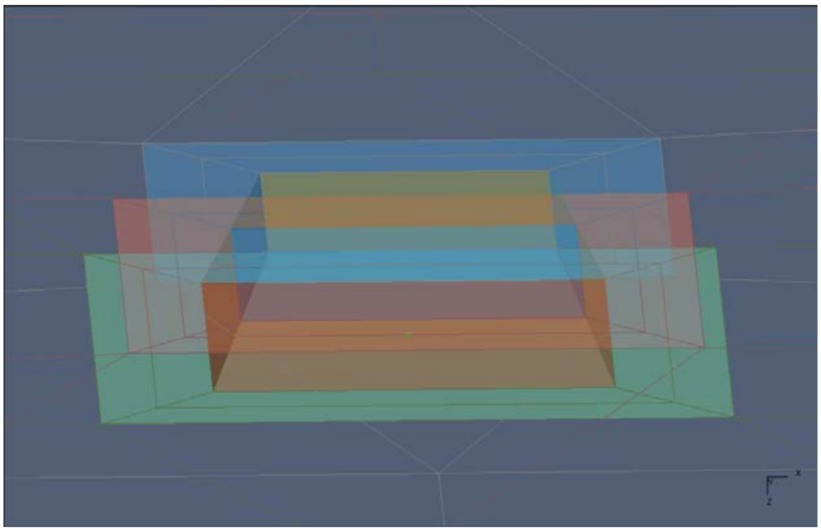
All the above is all very easy to do via a simple script. Depending on how your base model is setup, you can easily read the model into the software, read the spreadsheet data, apply the data to the model (in a number of ways) and write multiple models out. At a very basic level, you could just do some text substitution into your input decks to build multiple models very easily and quickly.
A script that does the above turns something that previously could take a long time, and has potential to introduce errors, into something that is robust and very quick. The benefit of scripting in this case is the ability to manipulate data in more complex ways than just assigning a field a particular value.

## 3.2 Reading/writing external data

Reading and writing information from and to external files is another example of how scripting can be beneficial. Pre-processers tend to do a good job of reading file formats, but there may be particular file formats that it does not read, or you want to interpret the data in a different way.
One example of this is connection information. This can be defined in a number of different ways, and can be in a file format not natively read by the pre-processer. It is fairly straight forward, with a few scripting skills, to create a tool that parses a file containing connection information, then creates connections in your model according to your modelling methodology.
The following example is a connections XML format that may not be supported by the pre-processer.

```
<definition type="hex spot">
  <id>100</id>
  <coord x="-4181.817871" y="-862.289124" z="605.012268" />
  <diam>5.0</diam>
  <material>10023</material>
  <panel>100345</panel>
  <panel>100676</panel>
  <panel>100113</panel>
  <haz>1 ring</haz>
  <haz diam>8.0</haz diam>
</definition>
<definition type="hex spot">
  <id>101</id>
  <coord x="-4341.792480" y="-841.940857" z="608.532227" />
  <diam>5.0</diam>
  <material>10023</material>
  <panel>100244</panel>
  <panel>100245</panel>
  <haz>1 ring</haz>
  <haz diam>8.0</haz diam>
</definition>
```

A fairly simple script can scan through the data in this file and create connections from the data.
One extension to this is that ability to modify models in more complex ways using fairly simple scripts. An example of this is assigning properties of connections. The following shows a spotweld modelled with solid elements.



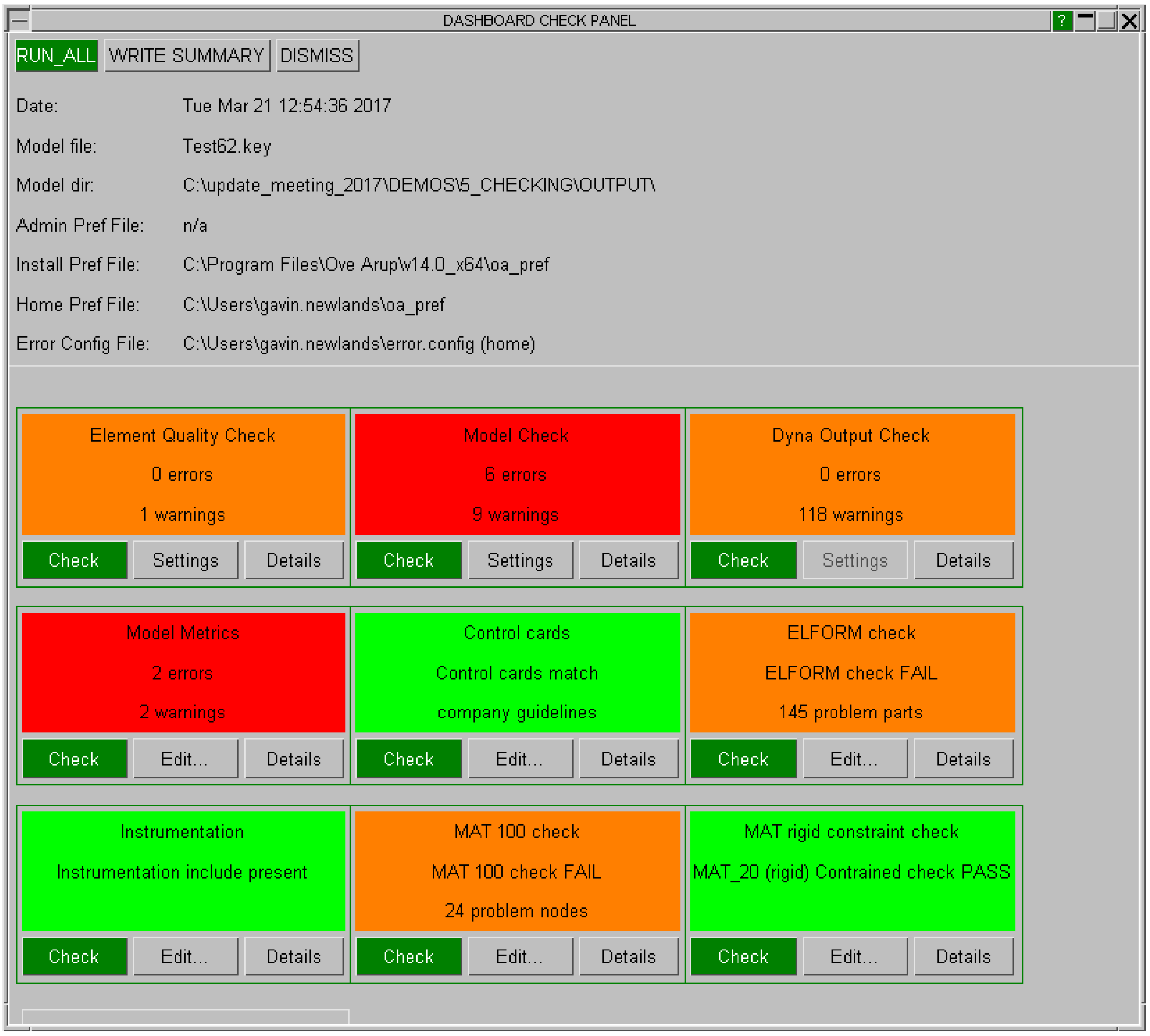
We have rings of elements on the panels to represent heat affected zones. These heat affected zone elements will have different properties to the base sheet metal. These properties could be set in a number of different ways. Perhaps your modelling methodology is that your HAZ part ID's have a label that is a particular label offset to the base panel part ID. Perhaps you have a series of HAZ parts in your model, and you need to choose one to use based on the diameter of the spotweld, or the number of panels being joined together, or the grade of the material being joined together – or perhaps a combination of the above. A script will allow you to loop through the connections in your model, and also the HAZ's and set the PID's according to any rule you want. Again, this is an example of being able to use and apply company specific modelling methodologies fairly easily with the addition of a script.

## 3.3 Custom checking

Model checking is an important part of the model setup. Checking a model can be a valuable time saver when trying to get a model to run in LS-DYNA. But why would you customise the checking instead of just relying on the pre-processers built in checks?

- Highlighting important errors and warnings.
- Introducing new checks not covered by the pre-processer.
- Company specific checking – make sure people are following company specific rules.
- Extracting checking information from different sources – output files as well as input.

The following is an example of a "dashboard" which pulls together checking information from a variety of sources. Some of the sources are scripting based.



| | |
|---|---|
| Date: | Tue Mar 21 12:54:36 2017 |
| Model file: | Test62.key |
| Model dir: | C:\update_meeting_2017\DEMOS\5_CHECKING\OUTPUT\ |
| Admin Pref File: | n/a |
| Install Pref File: | C:\Program Files\Ove Arup\v14.0_x64\oa_pref |
| Home Pref File: | C:\Users\gavin.newlands\oa_pref |
| Error Config File: | C:\Users\gavin.newlands\error.config (home) |

| | | |
|---|---|---|
| Element Quality Check<br>0 errors<br>1 warnings | Model Check<br>6 errors<br>9 warnings | Dyna Output Check<br>0 errors<br>118 warnings |
| Check Settings Details | Check Settings Details | Check Settings Details |
| Model Metrics<br>2 errors<br>2 warnings | Control cards<br>Control cards match<br>company guidelines | ELFORM check<br>ELFORM check FAIL<br>145 problem parts |
| Check Edit... Details | Check Edit... Details | Check Edit... Details |
| Instrumentation<br>Instrumentation include present | MAT 100 check<br>MAT 100 check FAIL<br>24 problem nodes | MAT rigid constraint check<br>MAT_20 (rigid) Contrained check PASS |
| Check Edit... Details | Check Edit... Details | Check Edit... Details |

The example shows various checks and whether the model passes the check or not. The scripting based ones are as follows:

Model Metrics – checks model mass and timestep/added mass information against company guidelines.

Control cards – Check if the control cards being used, or the values on the control cards follow your company guidelines.

ELFORM check – checks to see if there are any type 16 parts with <5 integration points.

Instrumentation – You may have an include file that always has to be included in your model that contains instrumentation information. This check looks to see if that include is present or not.

MAT 100 check – check to see if we have any nodes on MAT 100 parts not in a tied contact.

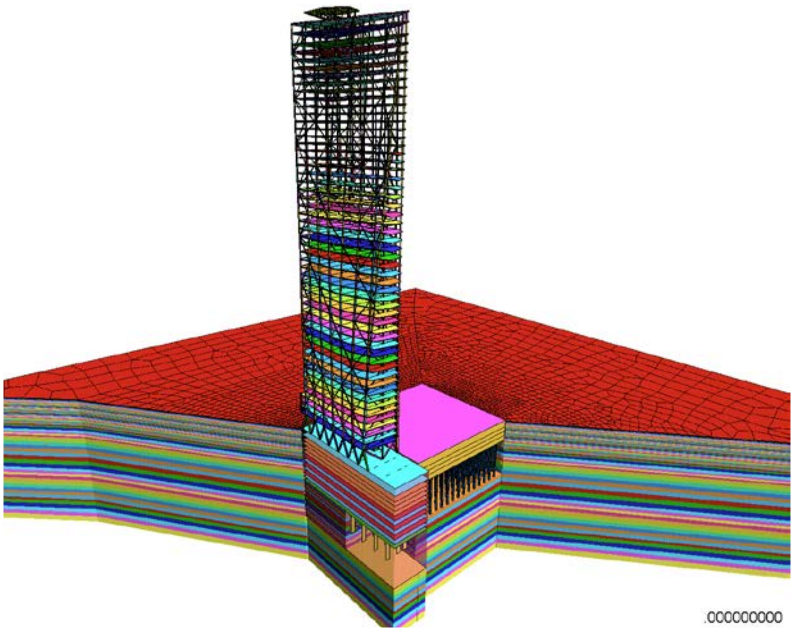
MAT rigid constraint checks – checks to see if we have any MAT 100 cards with rigid constraint values set.

As you can see with the above, they are not things that would cause LS-DYNA not to run, so may not be things that are picked up generally by a pre-processers built in checks, but they are useful tools that can be used to ensure consistency in your output.

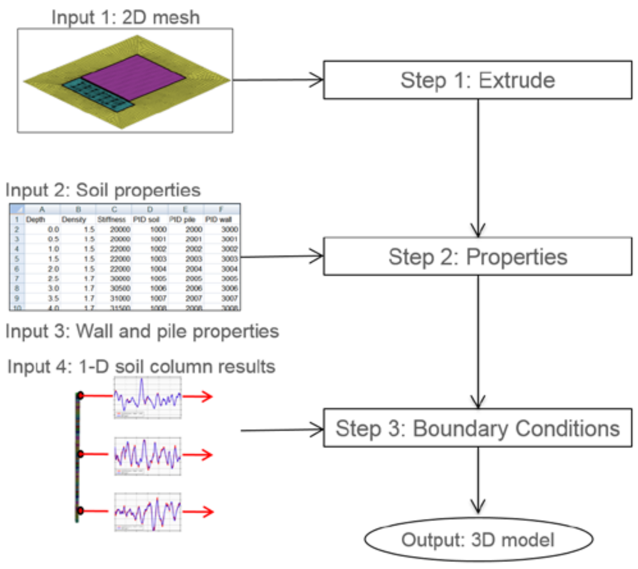
## 3.4 Automation

Automation is an increasingly important part of the CAE process. Shortened timescales and the need to improve consistency mean automation tools can be invaluable.

Scripting is a useful way of creating automation tools for your own purposes. One example of this is model setup. Here we have an example of a soil structure interaction LS-DYNA model for seismic analysis:



The soil is modelled in a series of layers going down from the surface. The thicknesses of these layers vary, as do the properties, and the source of the information is a spreadsheet from a geotechnical survey. It would be a long process to extrude the soil layers manually, and assign material properties, and the likelihood of introducing an error is quite high.
In this case a simple script can be written to read the information from the spreadsheet and automatically create the nodes, elements, parts and materials/loadcurves to match the data from the geotechnical information. An extension to this particular script was to apply the boundary and seismic loading conditions automatically.


Input 1: 2D mesh
Step 1: Extrude
Input 2: Soil properties
Step 2: Properties
Input 3: Wall and pile properties
Input 4: 1-D soil column results
Step 3: Boundary Conditions
Output: 3D model

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Depth | Density | Stiffness | PID soil | PID pile | PID wall |
| 2 | 0.0 | 1.5 | 20000 | 1000 | 2000 | 3000 |
| 3 | 0.5 | 1.5 | 20000 | 1001 | 2001 | 3001 |
| 4 | 1.0 | 1.5 | 22000 | 1002 | 2002 | 3002 |
| 5 | 1.5 | 1.5 | 22000 | 1003 | 2003 | 3003 |
| 6 | 2.0 | 1.5 | 22000 | 1004 | 2004 | 3004 |
| 7 | 2.5 | 1.7 | 30000 | 1005 | 2005 | 3005 |
| 8 | 3.0 | 1.7 | 30500 | 1006 | 2006 | 3006 |
| 9 | 3.5 | 1.7 | 31000 | 1007 | 2007 | 3007 |
| 10 | 4.0 | 1.7 | 31500 | 1008 | 2008 | 3008 |

The time saving benefit is clear, and the improvements in consistency of model build are important. With this particular example it has been applied to several projects, where the models have to be created again and again (for example when you get new geotechnical information) so the time saving can be significant.
An automotive example is assembling a vehicle crash model, and applying boundary conditions.

## 4 Post-Processing Examples

Here we will look at examples of using scripts within the post-processing environment. This will relate to:

- Interrogating and combining LS-DYNA output to produce user defined components that are applicable to your situation.
- Automatic post processing of results.

### 4.1 User defined results

Generally when looking at LS-DYNA output, you will start by looking at the data components that LS-DYNA writes out. You may then modify the data in some way to interpret the results in such a way that tells you what is going on in your model. This may be taking time history data, filtering it and combining it. It could be reading test data and performing a comparison with your LS-DYNA results. You may also want to create contour plots of your own user defined components, which are created by combining result components, or integrating with external data.
This is something that scripting can help with. Simple modification of result values (division, addition for example) can easily be handled by the post processer, but if you want to read external data and combine this with normal data components, writing a simple script to do this could save a lot of time. Images and detailed examples are not available at the time of writing this paper.

### 4.2 Automatic post-processing of results

Scripting can play a big role in automatically post-processing results. Templates provided with the post-processer for particular loadcases can get you some of what you want, but quite often you will want to modify the template somewhat. Scripting is quite useful in aspects of this. Let's take the example of a vehicle crash model. There are standard inputs you may want to retrieve, for example occupant injury results to calculate a star rating. This type of template is usually provided in post processing software. You may want to expand this according to various things you are interested in the model, to produce a bespoke report for your needs. Examples include:

- Carrying out a load path analysis, where you combine together output from various database cross sections to analyse how load is transferred through the vehicle body. You may also want to compare this to section properties (which may live is a separate text file).
- Produce particular images showing failure times of spotwelds and adhesive.
- Look at timings of peak values of occupant injuries and relate these to deformation and failures within the vehicle body.

This kind of bespoke post-processing is more relevant where standard loadcase templates are not available. For the previous seismic LS-DYNA analyses example, being able to automatically produce reports, combining data to tell the engineer all the relevant information was heavily reliant on scripting.

## 5 Summary

In this paper, I have shown that investing a small amount of time in learning scripting languages, tools and techniques can be beneficial to both the CAE engineer and the company they work for. For pre-processing tasks we can see that having the ability to create bespoke tools that allow you to carry out an array of different tasks and processes, in a minimal amount of time, can greatly improve the speed at which models are produced, and well as the quality and consistency. For post-processing, having the ability to automatically post-process results, and analyse and display the results in a way that is applicable to the loadcase and model frees up the time of the engineer to actually consider the results and progress the design.