# **Performance Simulation Generator Roadmap**

## Functionality & How to Use

The performance simulation generator works to remove manual input when setting up performance files that are run on 3DExperience. The code extracts and manipulates coordinates, meshes and locations of the bottle and rigid bodies (if present), as well as the mapping and scaling operations required to go from an SBM file to a performance file. This is complemented by a GUI, removing any need to interact with the code editor.

The app/GUI works as follows:

1. Import SBM data and Meshed bottle (from 3Dx). The files will be displayed for clarity.
2. Click the “Visualise Bottles” button to check bottle orientations and coordinates.
   1. Firstly, re-orientate the SBM data if required (change from z to y). Check the plot again to ensure the bottles are in the same orientation (it is assumed that the meshed bottle already has a primary axis of z).
   2. Secondly, if the SBM data requires reflection about a certain axis, click the “Mirror” checkbox. Check the plot again and choose the correct axis about which to reflect the coordinates.
3. Enter the bottle mass, in grams.
4. From the load cases provided, choose the desired option.
   1. If rigid bodies are present in the chosen case, translations will be required for these. Enter the xyz coordinates that the rigid bodies must move from the origin (0, 0, 0) to align with the meshed bottle. Labels are provided to guide the translation inputs.
5. Once steps 1-4 have been completed, the full assembly can be visualised. Click “Generate 3D Plot” to verify that the bottle and rigid bodies lie up with each other and that it matches with the chosen load case.
   1. If the assembly is not aligned correctly, adjust the rigid body translations and/or meshed bottle.
6. To generate the simulation file, click “Generate Simulation”. This will take roughly 1 minute, and the file will be outputted into the *simulation\_inps* directory.

## Scaling and/or Adapting

The following methods require interaction with the code editor. No GUI is provided.

**Adding rigid bodies:**

1. Under the *part\_creation\_inp* directory, open the *convert\_to\_inp.py* file.
2. Place the .inp file (with the new rigid body part in it) in that same directory.
3. At the bottom of the file, under \_\_name\_\_, paste the name of the .inp file into the *file* variable. Also, update the *kword* variable to match the name of the rigid body to be extracted.
4. When running, a visualisation will be shown.
   1. If the part looks correct, and the reference point is at the desired location, enter ‘Y’/’y’ into the terminal.
   2. If not, toggle settings available in lines 157, 159 and 160. Also, in line 186, adjust the index (1 = x, 2 = y, 3 = z) to align the reference point to the origin.
5. The finished .inp file will be located under the *part\_library* directory, named *rigid\_body\_{kword}.inp*.

**Adding load cases (backend):**

1. A new template must be created to add a new load case.
   1. Current templates are located under *template\_files*. These will give an idea for the outline required. It is important that tildes (~) follow the same format as other template files. Also, surfaces may need to be renamed in order for the script to pick up on appropriate parts and attributes.
2. If rigid bodies are required in the new load case, they must be added in.
   1. In the *get\_simulation\_components.py* (class: PopulateSimulation) file, a dictionary (line 59) is set up containing the load cases and their respective rigid bodies. Append the new load case and rigid bodies in this dictionary.
3. New attributes must be created to populate the simulation file.
   1. In the *extract\_component\_instances.py* (class: InstanceInp) file, a list of attributes has been set up. Attributes are extracted depending on the part – bottle; body groups, base ring, mapped coordinates, rigid bodies; reference nodes.
   2. Attributes can be added and updated within this file. Attribute naming is important due to the large number of variables and sets within assemblies, so aim for a specific name.
   3. If new rigid bodies or attributes are needed, *write\_to\_inp.py* (class: WriteToInp) must be updated to account for these changes.
      1. *\_extract\_kwords\_from\_line* works to automatically identify what needs to be pasted into the .inp file depending on the keyword in the template file. This function also involves checks for unique write outs (different write out format, unique attributes) to the .inp file.

**Adding load cases (GUI):**

1. Further load cases are added into the GUI as an additional radio button.
2. Labels for the translation matrix must also be updated.
   1. *get\_rb\_labels* contains a dictionary matching the load cases to the labels. The new load case and its respective labels should be appended to this dictionary.
      1. If no rigid bodies/labels are involved in the new load case, None should be used to denote the label.
   2. *load\_case\_radio* also has a dictionary containing load cases. Add the new load case to this dictionary too, following the key-value format.
3. The load case is fed in as a keyword for PopulateSimulation so it’s important that the names are identical, otherwise the template file won’t be picked out.

## Future Work

* The GUI is functional, however aesthetically it is not great.
* Currently, the process is set up to follow a certain set of parameters (in simulation and in mapping operations). It would be useful to implement a pop-out box for the user to adjust these parameters.
* The process takes roughly 1 minute, which is predominantly due to the mapping operations. Multiple approaches are used for mapping at BMT (LMcG, RB, DMcK) so aim to make this section more efficient. The current mapping operations bounce between numerous functions and files so a self-contained class configuration may be beneficial.
* In the GUI, when “Generate 3D Plot” is pressed it only highlights the locations of the assembly parts. A worthwhile addition to this would be showing the thickness and/or modulus map of the bottle.
  + This will require mapping before showing the visualisation which will add time, however it will greatly benefit the user when determining if the mapping has been carried out correctly.