FLAC3D – Reporting Automation

This workflow represents an integrated pipeline for flac3d modelling design and documentation. It combines automated data extraction from FLAC3D with a parametric Grasshopper routine that interprets geometry and dimensions pillars, all wrapped up in a reporting step that standardises and packages the outputs for decision-making.

The process begins inside FLAC3D, where displacement fields, maximum and minimum principal stresses, vertical effective stress (σzz), and zone state plots are generated. Rather than relying on a manual, time-consuming export of individual sections, the Python automation systematically sweeps through the model at defined slice intervals along the X, Y and Z axes. For vertical stress interpretation in particular, additional scripts allow top-view plan slices at user-specified elevations, capturing crown and floor loading conditions at critical levels. These images are produced in high resolution, named consistently, and stored in a structured directory, so that data from different runs or model variants can be compared directly.

The second stage takes these raw exports and compiles them into a professional report format. Images are converted and compressed, then laid out in a Word document with uniform headings and captions. Each slice is documented in sequence, grouped by displacement, principal stresses, zone state and vertical stress. This creates a figure booklet that can be immediately included in design submissions or used for internal review. By standardising this process, the same structure and appearance are maintained across multiple models and projects, which improves clarity and traceability.

Python Code Breakdown

- data-export-automated.py
  - Automates batch export of FLAC3D plots. Generates displacement, max/min principal effective stress, vertical effective stress (σzz), and zone state slices along X, Y, and Z planes. Ensures outputs are consistently named, stored in structured directories, and suitable for comparison between model runs.
- zz_slice_topview_batch.py
  - Produces top-down (plan view) σzz stress maps at user-specified elevations (RLs). Allows control of contour limits, intervals, and scaling to capture crown and floor loading conditions at critical levels. Optionally exports raw data files alongside images.
- create-docx-summary.py
  - Post-processes exported images and compiles them into a structured Word report. Converts and compresses BMPs to manageable size, inserts figures with captions and headings, and produces a clean figure booklet grouped by displacement, stresses, and zone state.
- pillar-stress.py
  - Focused on identifying and analysing pillar behaviour. Processes stress data to support design and support requirements of the pillars, complementing the Grasshopper workflow.

The coding effort was accelerated through the use of AI-assisted development, which helped generate the initial structure of the automation scripts quickly. Most of the refinement and debugging, particularly when working with the Itasca Python API inside FLAC3D was then carried out manually, ensuring the final tools were both functional and reliable in practice.

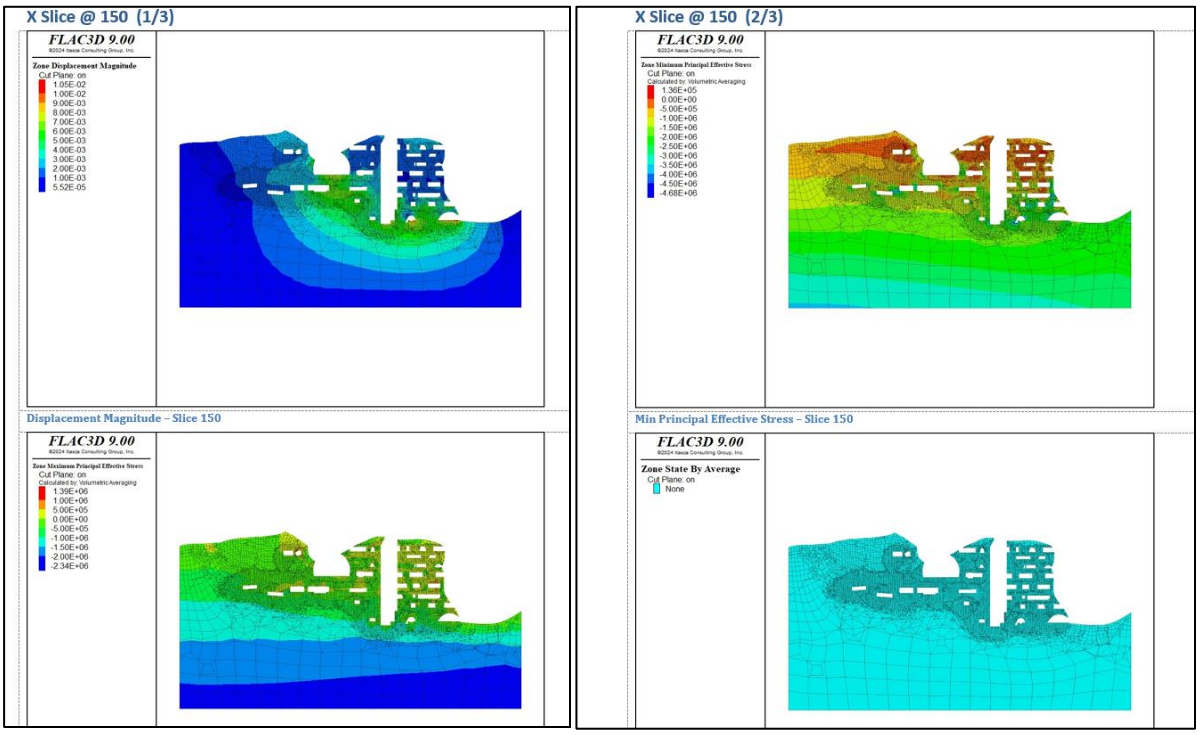


Figure 1 – automated reporting extract (create-docx-summary.py)

In parallel, the Grasshopper definition provides a parametric approach to identifying and sizing pillars. It accepts the geometric inputs that represent the model cross-sections or excavation layouts, then filters, merges and simplifies them to isolate pillars. Threshold values ensure only features above a minimum size are retained. Once filtered, the script calculates dimensions for each pillar (length, width and height) and assigns IDs, and outputs these as structured text as well as a CSV output. Because this is parametric, changes to floor level, excavation offset, or section geometry propagate instantly through the definition, giving updated pillar dimensions in real time. This makes it especially useful during optioneering, design iteration and sensitivity checks.

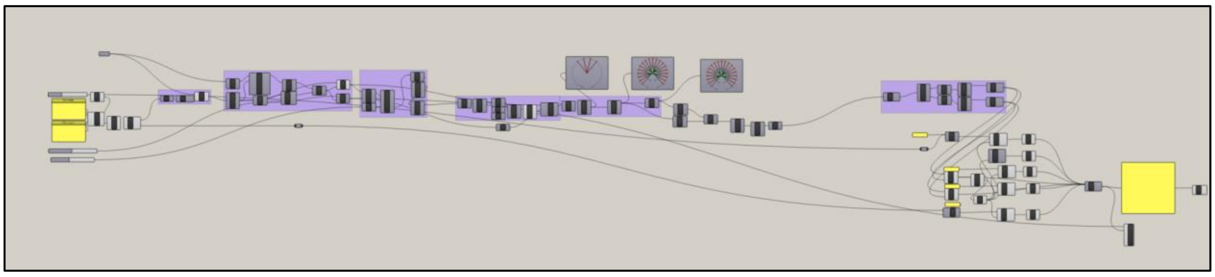
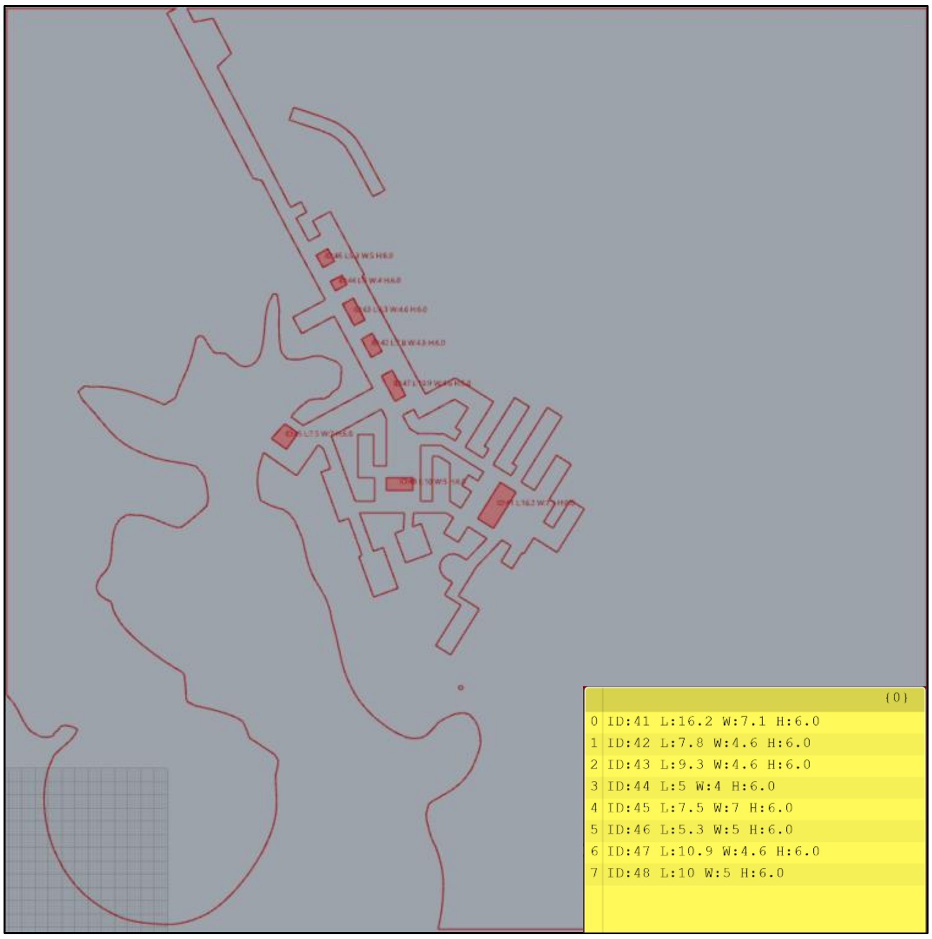


Figure 2- Rhino/Grasshopper workflow



```
{0}
0 ID:41 L:16.2 W:7.1 H:6.0
1 ID:42 L:7.8 W:4.6 H:6.0
2 ID:43 L:9.3 W:4.6 H:6.0
3 ID:44 L:5 W:4 H:6.0
4 ID:45 L:7.5 W:7 H:6.0
5 ID:46 L:5.3 W:5 H:6.0
6 ID:47 L:10.9 W:4.6 H:6.0
7 ID:48 L:10 W:5 H:6.0
```

Figure 3 - Rhino/Grasshopper Pillar Plan output

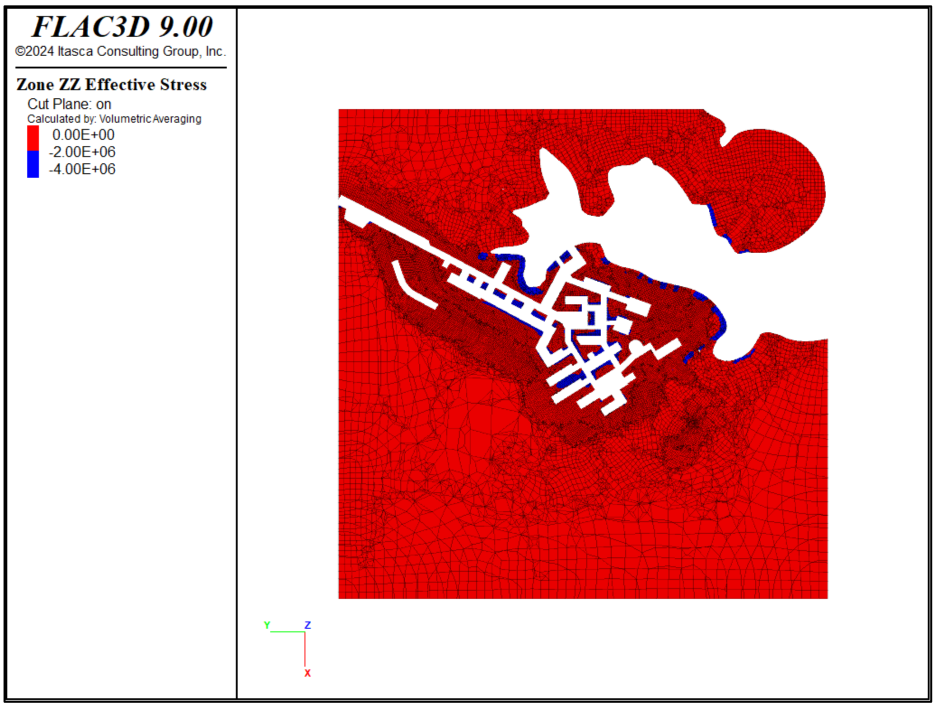


Figure 4 – FLAC3d Stress Plan

The combination of these elements results in a continuous pipeline. Numerical modelling establishes the stress and displacement environment, automated export scripts capture the data without manual effort, reporting scripts package the information for engineering communication, and the Grasshopper tool interprets geometry to provide quantitative pillar data. Together they reduce turnaround time from days to hours, ensure outputs are consistent and comparable, and directly link model results with the geometric and dimensional data needed for design. In practice, this means engineers can move quickly from simulation to design validation, with confidence that the pillars being proposed are aligned with the most current stress conditions and documented in a clear, reproducible format.

Link to the source code:

https://github.com/nmdevgit/itasca-flac3d-reporting