ESA Sysnova System Studies - Lunar Caves

Documentation Voxeliser scripts IGMAS modelling

Voxeling_Main script documentation

Author
Frank de Veld

June 4, 2020

Introduction

This documentation is regarding a series of MatLab scripts based on a main file called 'Voxeling_Main' used to convert cavity model files into .model files usable by IGMAS gravity modelling software, as well as functionalities for data handling. This is all in the context of the ESa Sysnova System Studies - Lunar Caves project, specifically Idea I-2019-01976 canadensys-CA-BU28802-1. For comments, questions or suggestions, please email the author at fdv@queens.ca or fdeveld@student.tudelft.nl.

Step 0

First, the script loads a file, where the file directory can be set beforehand. The most common files to read are either .csv files, for which the standard MatLab function `importdata` is used, or .stl files, for which the function `stlread` is used, another standard MatLab function. The sole goal of this is to obtain an n times 3 matrix of coordinates defining the cavity.

After this, the number of coordinates is reduced with the function `Reduce_coordinates`, made by the author. The reason is that files with more than 100.000 points defining the cavity take a very long time to process, and might be readable by IGMAS. The level of detail in these files is often unnecessary. The reduction is done by taking every xth cavity coordinate, with x a parameter `Reduce_factor`, which can be defined by the user. This can be changed as desired.

A number of optional other operations can be done, currently including a total sizing and a horizontal stretching. The former increases every coordinate by a factor `Sizing_Factor`, effectively making the cavity larger. The latter stretches the cavity horizontally without changing the height coordinate. This does not preserve the shape of the cavity. Tests have showed these methods to be working as expected. Later options should include shape-preserving stretching, vertical stretching, stretching until a certain average/maximum radius or length is reached, and local rather than global changes.

Step 1

This step has three goals; define the important parameters of the model, to create the grid, and to export a basic .model file. The parameters which can be define din this version are the density of the rock in grams per cubic centimeter in the `Rock_density` variable, the vertical distance between the top of the cavity and the stations in meter in the `Cavity_depth` variable, the number of stations in the `Station_resolution` variable, the number of voxels in the `Voxel_resolution` variable and the sizing factor to prevent edge effects in the `Sizing_factor` variable. The interpretation of the sizing factor is that at after the x- and y-range of the cavity are determined, a 'block' of rock is placed at either side of the cavity with the length equal to the sizing factor times the respective range. Thus, the model is $2 \cdot \text{Sizing\_Factor} + 1$ larger than the cavity. This is to prevent side effects and boundary effects in the gravity signal above the cavity. It is advised not to set this variable below 0.5, and preferably above 1. Lastly, the name of the model is defined. For the specific parts (voxel files, model files, station file), a separate prefix is added.

Next, based on the extent of the coordinates of the imported cavity, the extent of the model is defined as explained earlier for the x- and y-coordinates. For the z-coordinates, it is ensured that the cavity is indeed a depth `Cavity_depth` below the (flat) surface. The space under the cavity is of lesser importance; it is advised to have little material here, as it just adds voxels without importance and makes

the model bigger.

Based on the chosen voxel resolution, a grid spacing is defined. It is important to note that IGMAS always works with evenly spaced grids. Thus, in every direction the grid spacing should be equal. The calculated resolutions are the number of meter per voxel. In the current modulo-based implementation, this should be at least 1, and it is rounded afterwards. The maximum resolution is chosen, meaning that the voxel resolution only applies to one direction; in the other dimensions have the number of voxels is less or equal. another important note to make is that IGMAS can't support more than 30 million voxels in total, placing a limit to the voxel resolution. With this number of voxels, the total processing loading times are also in the order of tens of minutes per model.

With the grid spacing determined, it must be ensured that the model range is adapted to the size of the grid, so that the length in each direction is divisible by the grid spacing. This grid spacing and the number of voxels per direction are stored for later use.

Lastly, a basic .model file is written and exported with the function `Write_basic_model_file`. This .model file defines the basic properties of the model, namely the sizing and the bodies involved plus their densities. These bodies are a reference body, always present, the rock with the specified density, and a vacuum cavity with density 0. Note that in this model, the cavity is not yet defined; this happens later with the voxels. However, the voxels can only be imported and used when the model is available. The model has a number of sections equal to the number of voxels in the y-direction plus one, as the voxels are defined according the their middle points. The sections could also be parallel to the x-direction, as this choice only matters for visualisation.

Step 2

This step covers the creation of the voxel grid; it fills the grid with either a voxel with a density of the rock, predefined, or with a vacuum cavity, according to the cavity coordinates. For this, a main function called `Voxel_writing` is used for both the voxel model with cavity and the one without; the difference is indicated by a boolean.

In this main function `Voxel_writing`, first the range and length in all directions is defined. A starting grid is made with the function `Starting_grid` with the number of voxels as defined earlier and with each voxel being filled with rock. The majority of the voxels will be filled with rock, and now only the voxels near the cavity will be checked whether it is inside the cavity or not. To the limit vector, a value `Grid_spacing/2` is added to make the voxels be nicely aligned with the basic .model file.

Now, only if the voxel file with cavity is considered, a procedure is started to investigate which voxels need to be filled with a cavity. If the voxel file without cavity is considered, the grid already is the final voxel file. If the cavity is considered, first the cavity coordinates are rounded to the grid. Approximations are done here, but it makes it much faster to go through the voxels to check whether they are inside or outside the cavity. The function `Grid_cavity_coordinates` is used for this. in this function, for every cavity coordinate, its new position on the grid is determined. An efficient way of doing this is to divide the coordinate by the grid spacing, round this to the nearest integer, and multiply with the grid spacing again. The result is that the coordinate is rounded to the nearest multiple of the grid spacing. Now, if also the minimum absolute value of the respective coordinate is added, the coordinates end up on the grid. Otherwise, the coordinates might be spaced according to the grid spacing, but might not be aligned to the grid.

This procedure can lead to coordinates be in the grid more than once; if coordinates are very close, they can get rounded of to the same grid point. These double coordinates are then eliminated from the cavity coordinates vector with the standard MatLab function `unique`, based on the rows, representing coordinates. For later purposes, it is useful to sort the coordinates, in this case based on the y-coordinates. This is a matter of preference; the following procedure can be based on the x-coordinate as well, with identical results. Next, a silly fix is implemented for the fact that the `nonzero` functionality is used sometimes in the script, so there are problems when a (rounded) coordinate is indeed zero. The fix is to increase the zero by a tiny amount and reverse this change later on. Now, some basic operations are done to prepare for a loop through unique y-coordinates of the gridded cavity coordinates. After this, the function `Reshaper` is used, which despite its complex appearance has a simple goal; to reshape the gridded cavity coordinate matrix to a cell where the cavity coordinates of unique y-coordinates are stored. In this way, it is possible to loop through the cell and obtain the relevant cavity coordinates for that y-coordinate. The coordinates are re-arranged so that this is possible, and here the `nonzero` functionality is used to shape the coordinates accordingly.

Now, a loop can be started going through the number of unique y-coordinates, to fill in the grid step by step. The information of the cavity is retrieved from the `Coordinate_cell` variable, for which a function called `call2mat` must be used. Coordinates which previously were zero, can now be zero again. The approach chosen now is to find the rectangle such that all gridded cavity points of that section are either on the boundary or inside the rectangle. With the information of the grid spacing, the number of grid points in or on this rectangle can be obtained. For all these grid points, the required density is determined with the function `Calculate_density`. There are multiple ways of doing this, but the fastest with the current approach is to use the function `inpolygon`, which takes a set of coordinates to check and a set of coordinates defining a polygon (in this case, a slice of the gridded cavity). This function counts the occurrence of coordinates to check either in or on the polygon. Here, only one coordinate needs to be checked at a time, and if it is either in or on the cavity, the density of this voxel is changed to zero. Back in the `Voxel_writing` function, this coordinate is than sought for in the starting grid with `intersect` and the density is replaced with zero. By doing this for all cavity slices, all voxels that need to be changed are changed in a quite efficient way. Lastly, the result is written to a .vxo file.

Step 3

This step is for creating a .station file. While this can also be done in IGMAS itself, it is also useful to create this in the same process as the model is created, and to adapt the stations to the cavity. The .stations file is created with a function called `Export_stations_file`, requiring the limit vector for the boundaries of the stations, the station resolution parameter indicating the number of stations per direction and the model name, as the .stations file will have a name similar to the model name with the prefix `Vox_Stations_`.

The stations are placed in a regular grid, with a grid spacing equal to the dimension of the model divided by the station resolution. As IGMAS can also work with irregularly spaced stations, we are currently looking into the inclusion of other station patterns, such as circles, spirals, Gaussian distributed or uniformly distributed stations.

Step 4

This step is currently not automated yet. It requires to run the first three steps for the basic .model files, the .vxo files with and without cavity and the .station file. Then, in IGMAS the basic .model file should be imported, as well as the .vxo file without cavity and the .station file. After this, the user should calculate the anomalies he/she wants, wait for the calculation to finish and export the resulting .stations file with a clear name. the same process must be repeated for the .vxo file with a cavity. When both .stations result files are obtained, data analysing can start.

Step 5

This step is present for interpretation of data, if results file in the .stations format are available. As it currently isn't possible to run the whole script in one take, often the boolean `Calculation_bool` is set to false while running the script for the first time, and this part is ran later when results are ready.

A functionality called `xml2struct` is used, developed by Wouter Falkena. This makes it possible to use the xml file created by IGMAS in MatLab without modifications. in a for-loop, properties like the coordinates of stations as well as the measured values are extracted and stored in a separate matrix for both the file without cavity and the file with cavity. The dot indexing of the read .station results file makes it possible to go through all attributed of the variable. This structure forces the user to use 1,x-indexing after 'property', i.e. `geodata.vertex{1,a}.property{1,1}.Attributes.value` for the first value, if there are multiple, and omit this indexing if there is only a single measured variable. In general, the G_z value is always measured, and often the G_{zz} and H_{gz} variables as well.

Four plots are considered for the data visualisation. The difference in signal is considered the most interesting to plot, as it shows the effect of the cavity directly. The `scatter3` function is the easiest way of plotting and provides a quick overview of your results, though details are difficult to see. with `stem3`, bars are plotted in 3d as well, making it easier to judge the value at different coordinates. However, the information is visible the easiest in the `ContourPlot` and `trisurf` functionalities. The former is self-made, the latter is a combination of MatLab functionalities. With these visualisations, a colour bar clearly shows how the gravity value varies, which makes it possible to see the results in a 2D format. With the `trisurf` option, one can combine 3D and 2D views in a clear way.

General remarks

The time required to run the script in its entirety, without data handling, ranges from a few seconds to about ten minutes in general, with maxima of about fifteen minutes. Larger models (meaning; more voxels) require more time. The features requiring the most time are the rounding of the coordinates to grid points and the checking of the density of the coordinates, as well as looking up coordinates in the `Starting_grid` variable for replacing of densities. While the former two features are deemed difficult to make faster, the latter feature can be improved such that not the whole matrix needs to be investigated for this search. including the IGMAS part and data visualisation, a process of 2 minutes to 20 minutes can be expected.

This script is merely the bare basis for further features, and these will be added for the coming months. Efficiency is something that would be nice to improve, as well as data handling methods and introducing of variations in the cavity, random density anomalies in the surrounding rock, noise sources, terrain elevation and ways of different station patterns, as well as step-wise gravity signal interpretation. The goal is to end up with a way of inverse-modelling; so far this has all been regarding forward modelling.

Inverse modelling will be regarding much simples shapes than the complex cavities which can be used for forward modelling.

Suggestions from the reader for further improvements, additional features, bug reports or general questions are always welcome, also after this ESA project, lasting till October 2020, has ended. For this, contact the author on GitHub or via the email addresses in the introduction.

Common errors

Before consulting this section, make sure that you downloaded all necessary .m files from the GitHub page, that they are all in the same folder, and that the directories for the import of files are correct. Error: `"Error using zeros.  NaN and Inf not allowed."`. This occurs when your voxel resolution is much higher than the number of meters in either direction of the total model (sizing factor included). The result is that the grid spacing is rounded to zero, giving problems. Later versions of this script should allow grid spacings which are not integers, but for now it's advised to use a lower voxel resolution or increase the size of your cavity.

A problem which the author encountered is with the ^{3}-symbol. When a script is used once, everything works fine, but when it is run a second time afterwards, the ^{3}-symbol in the output files is replaced for the symbol `[U+FFFD]`, which IGMAS can't read. A quick fix is to shut down MatLab and run it again, or change the symbols manually, but this bug is very annoying and needs to be addressed, as it makes automation difficult.

Multiple errors can occur when the voxel resolution is so low that there are zero voxels in the z-direction. Place additional rock under the cave or increase the resolution if that happens.

On a similar note, if the cavity coordinates are reduced too much, the program still works, but the approximation is is large that the volume of the cavity is reduced tremendously.

Appendix

Effect of reducing the resolution of the model:

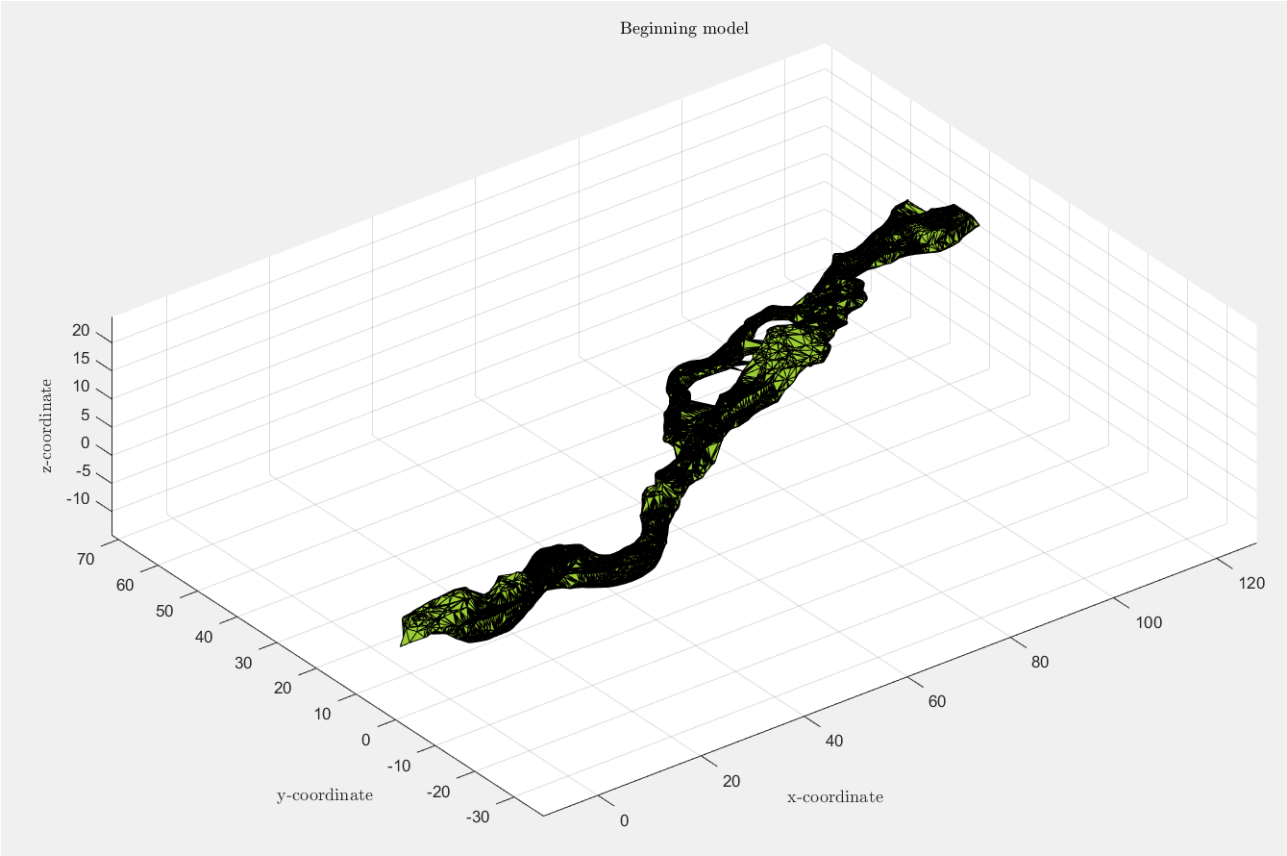

Figure 1: Beginning model

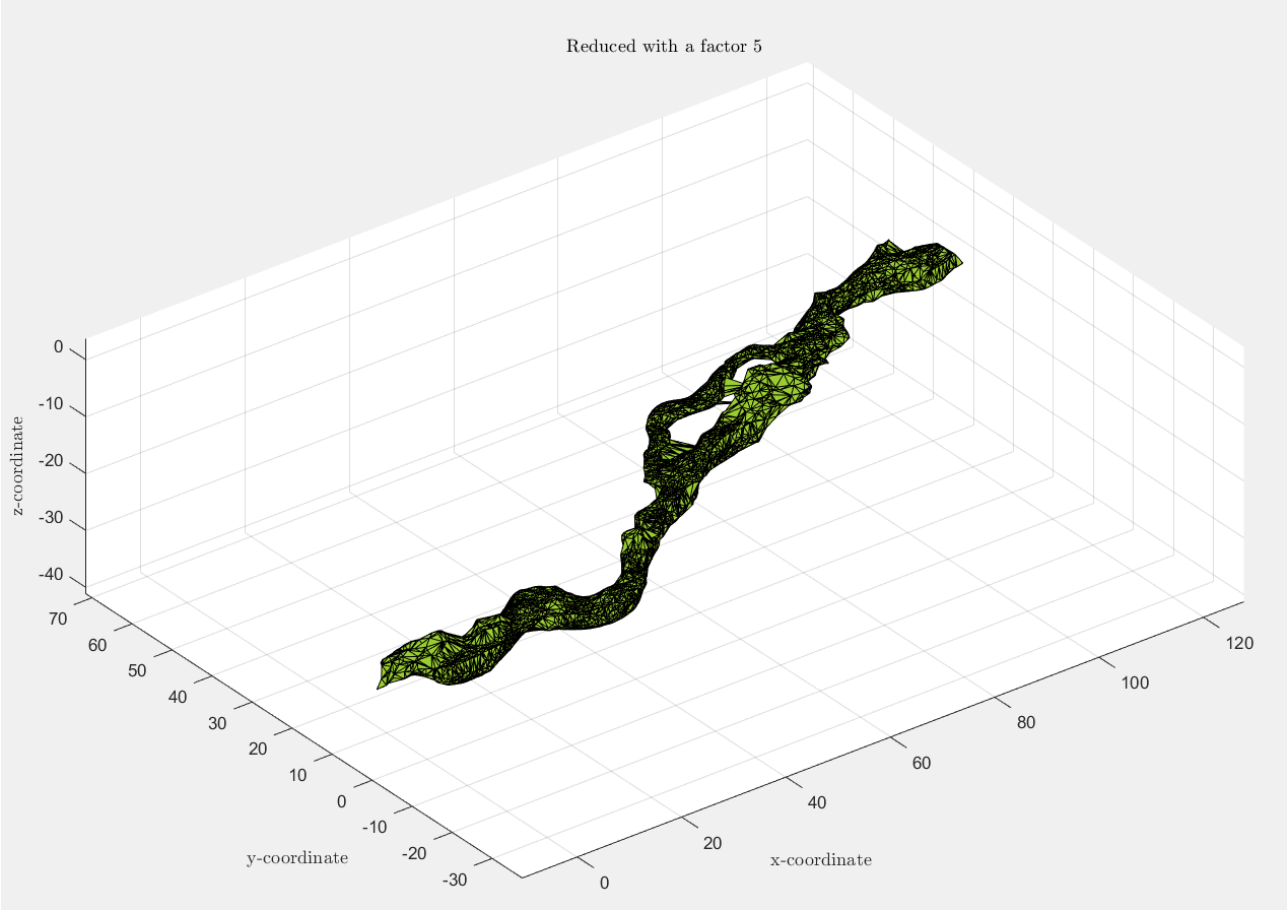

Figure 2: Coordinates reduced by a factor 5

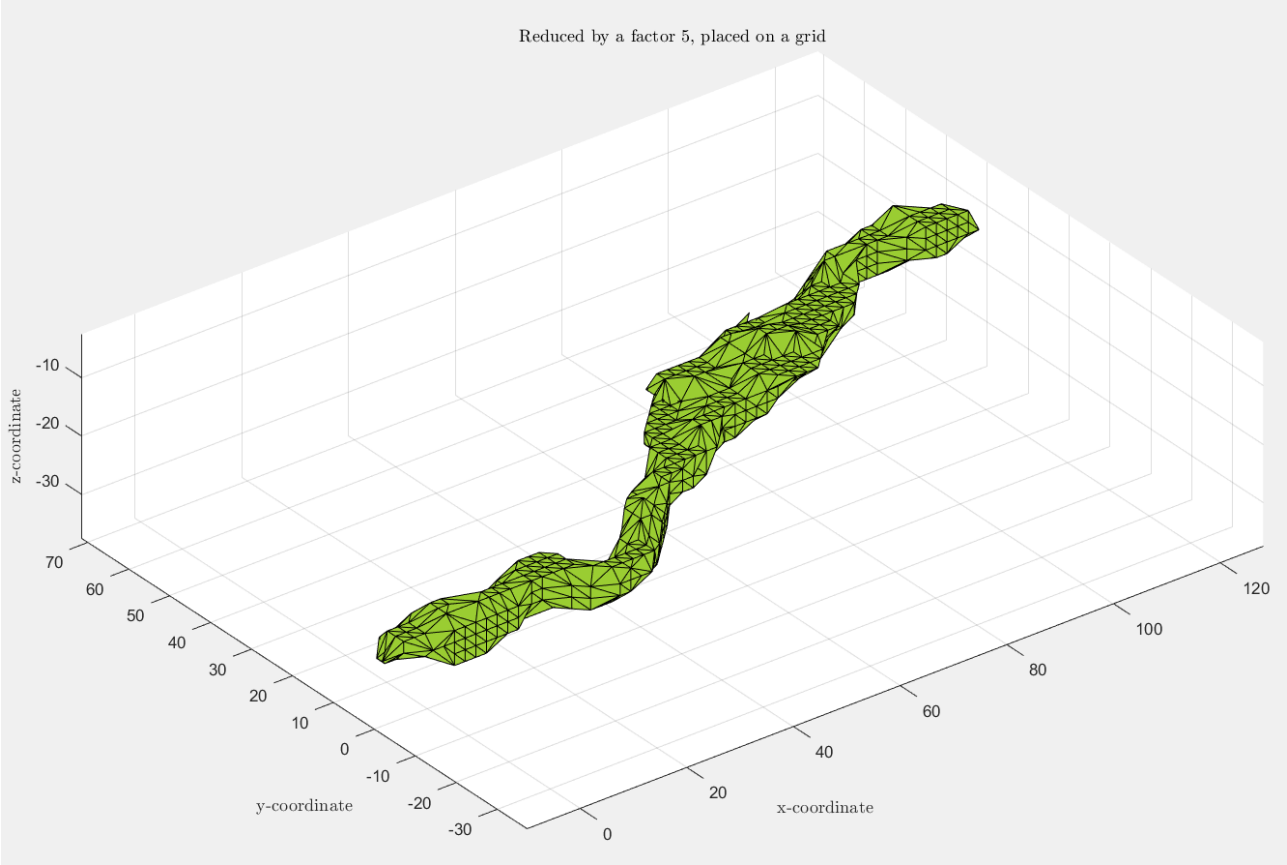

Figure 3: Coordinates reduced by a factor 5, placed on a grid