DIAPHRAGM SHEAR AND MOMENT DIAGRAMS - USER GUIDE

Posted by Srikrishnan Madhavan

A common use case prior to design of diaphragms is generating diaphragm shear and moment diagrams based on equivalent beam model.

In this post we'll look at how we can use Python to automate generating these diagrams for two different use cases. At the heart of it, there are two key inputs. The lengths (distances) to each line of lateral resistance and the forces at each of those points.

Use case A - Single Diaphragm - single load case/combo (for quick checks)

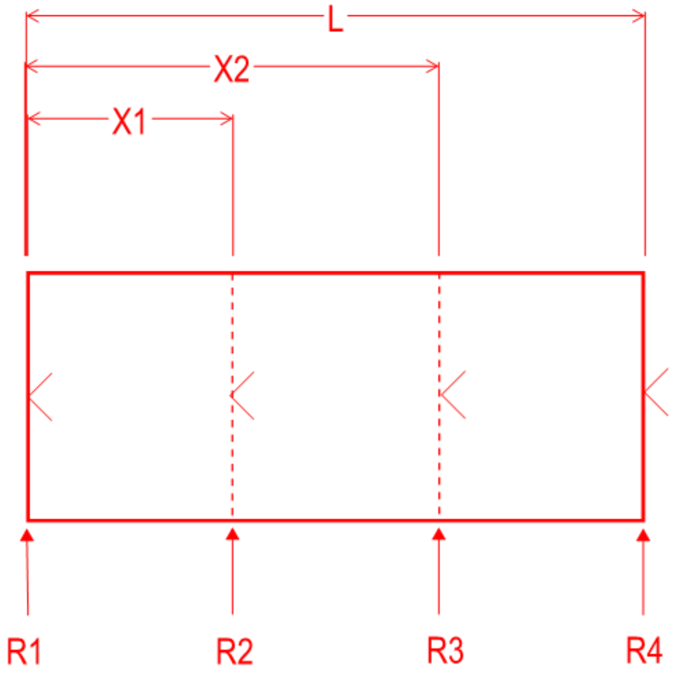
A1. Open up *"diaphragm_shear_diagrams_caseA.py"* in Spyder (or your IDE of choice/command line).
There are two ways to enter input,

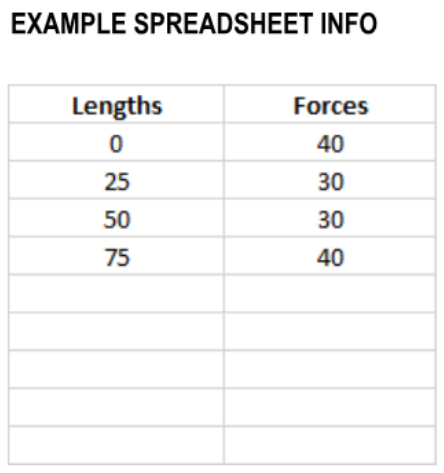
i. either manually by uncommenting the *lengths* and *forces* variables, or
ii. Via spreadsheet - see template *"diaphragm_template_input_caseA.xlsx"* attached. Enter the *filepath* of the spreadsheet, in the script.
iii. Run the script once inputs are ready. You'll see two plots one for shear and other for moment diagrams. You can save the plots in most figure formats.

<u>Note:</u> The lengths are **cumulative lengths** running from 0 to L (length of the diaphragm). The forces are **reactions** from lateral system at each point of resistance. (The total distributed load to the diaphragm is automatically calculated as sum of reactions.) You can have any number of lengths and forces. For cantilevered ends, include the lengths and just leave the forces as 0. This would have to be run in each direction separately. See schematic below.

LENGTHS = [0, X1, X2, L]

FORCES = [R1, R2, R3, R4]

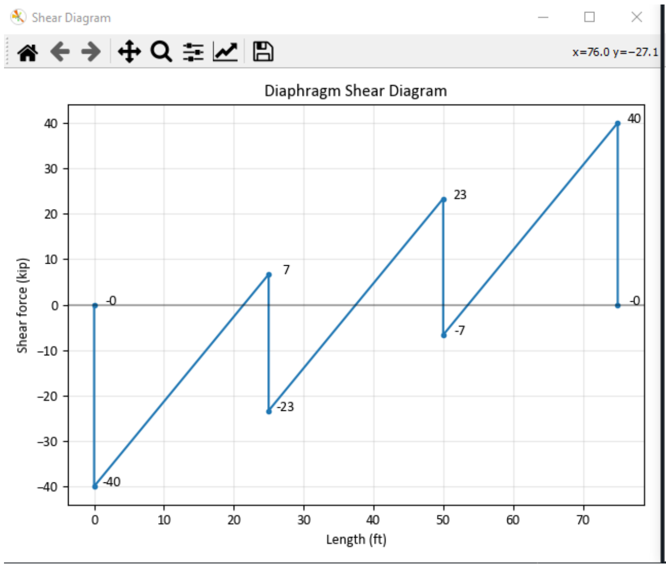


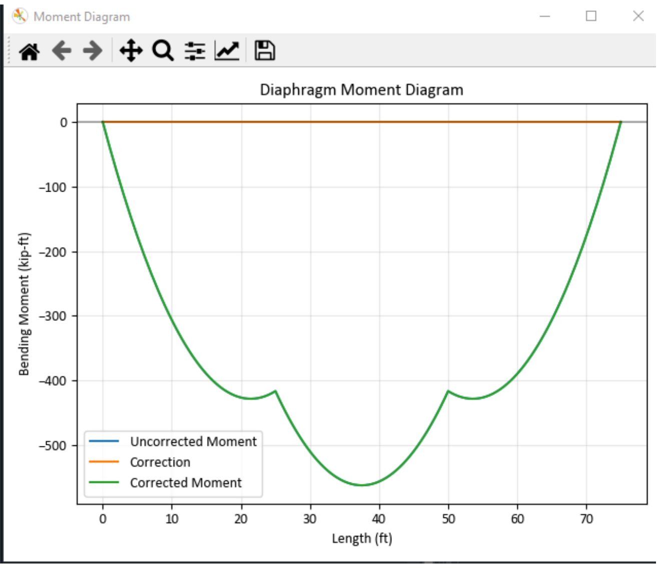


EXAMPLE SPREADSHEET INFO

Lengths	Forces
0	40
25	30
50	30
75	40

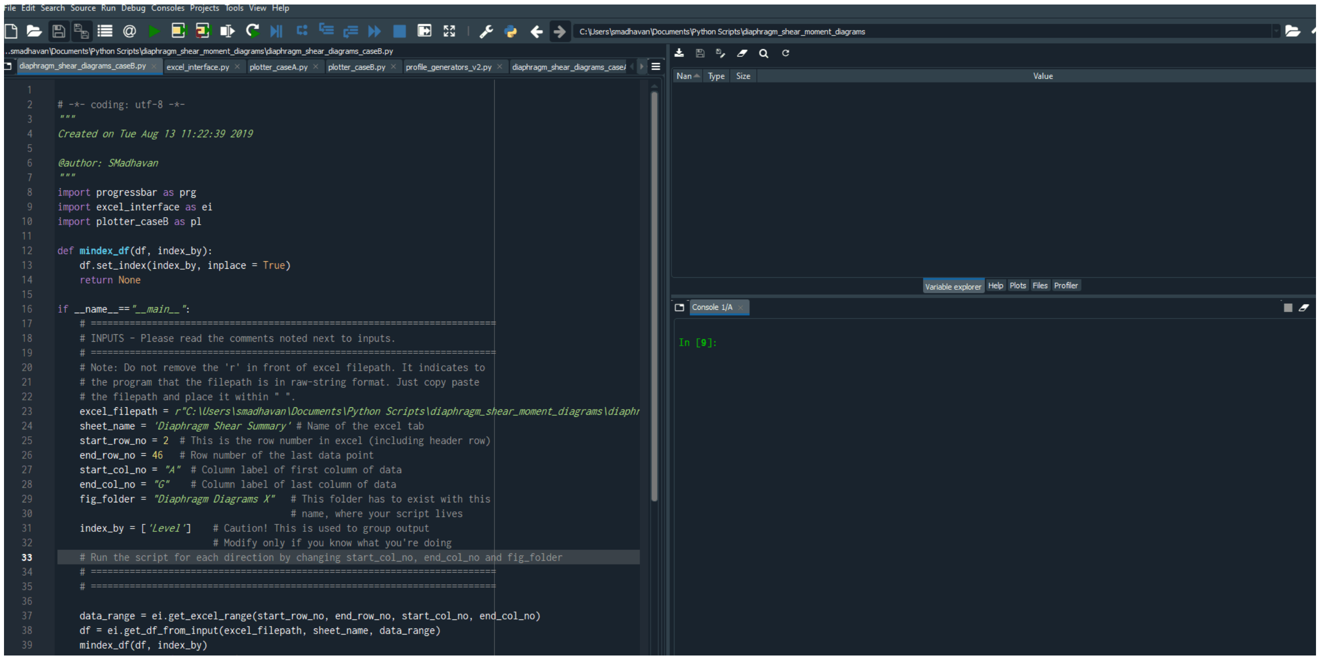
For the example template input file, you should see two diagrams like this:





Use Case B - Multiple diaphragm levels with three different ELF load cases

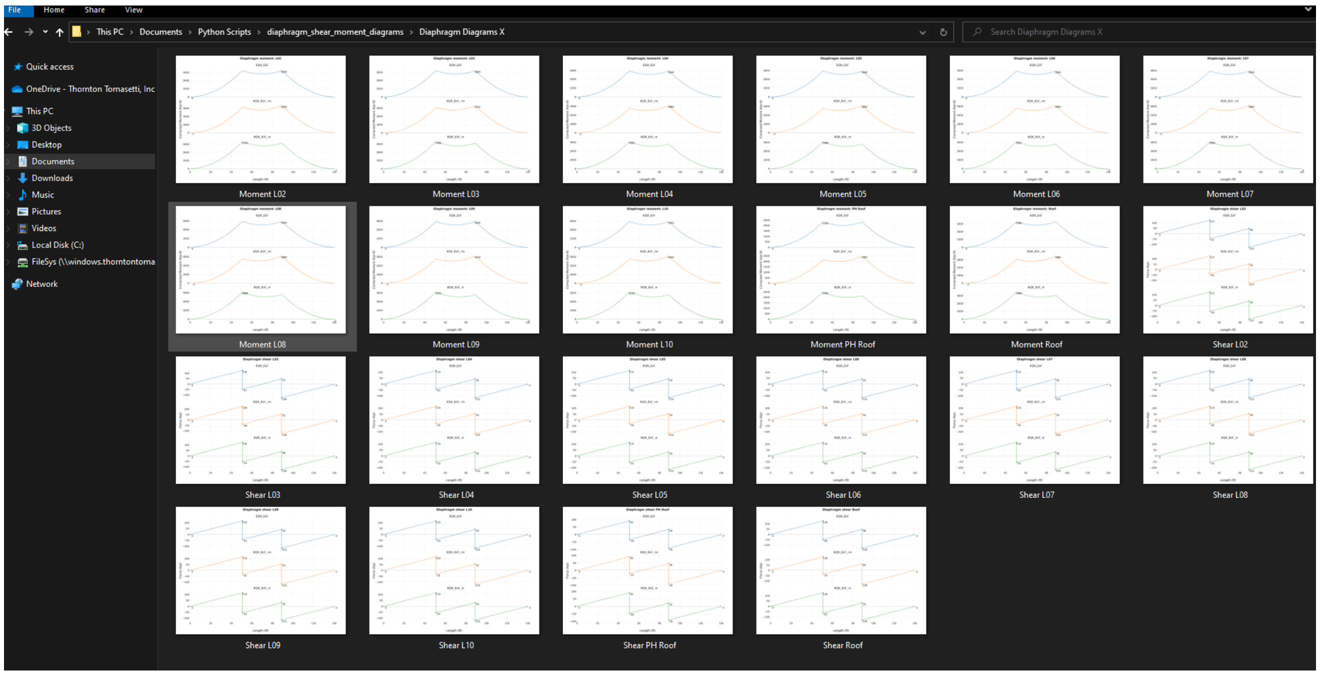
B1. For this case, setup your demands in a format shown in the template file provided: "*diaphragm_template_input_caseB.xlsx". You'd also need to create two folders where you scripts are located, to store the output plots. Unlike case A, the plots for case B are automatically saved into the folder.
B2. Open up *"diaphragm_shear_diagrams_caseB.py"* in Spyder (or your IDE of choice/command line). Enter your inputs. (The script has comments explaining each input) In general, you'd have to enter the excel filepath, the sheet name, start and end rows and columns where your data is located and the names of the folders you'd want to output the figures to. The current inputs in the script corroborate with that of the template file.
B3. Run this script once for each direction, with the appropriate inputs changed. The run would look something like this:
(The script opens up your spreadsheet, reads the data and outputs the plots to the folder.)



```

# -*- coding: utf-8 -*-
"""
Created on Tue Aug 13 11:22:39 2019

@author: SMadhavan
"""
import progressbar as prg
import excel_interface as ei
import plotter_caseB as pl

def mindex_df(df, index_by):
    df.set_index(index_by, inplace = True)
    return None

if __name__=="__main__":
    # ==========================================================================
    # INPUTS - Please read the comments noted next to inputs.
    # ==========================================================================
    # Note: Do not remove the 'r' in front of excel filepath. It indicates to
    # the program that the filepath is in raw-string format. Just copy paste
    # the filepath and place it within " ".
    excel_filepath = r"C:\Users\smadhavan\Documents\Python Scripts\diaphragm_shear_moment_diagrams\diaphr
    sheet_name = 'Diaphragm Shear Summary' # Name of the excel tab
    start_row_no = 2  # This is the row number in excel (including header row)
    end_row_no = 46   # Row number of the last data point
    start_col_no = "A"  # Column label of first column of data
    end_col_no = "G"    # Column label of last column of data
    fig_folder = "Diaphragm Diagrams X"   # This folder has to exist with this
                                          # name, where your script lives
    index_by = ['Level']    # Caution! This is used to group output
                            # Modify only if you know what you're doing
    # Run the script for each direction by changing start_col_no, end_col_no and fig_folder
    # ==========================================================================
    # ==========================================================================

    data_range = ei.get_excel_range(start_row_no, end_row_no, start_col_no, end_col_no)
    df = ei.get_df_from_input(excel_filepath, sheet_name, data_range)
    mindex_df(df, index_by)
```

```
In [9]:
```

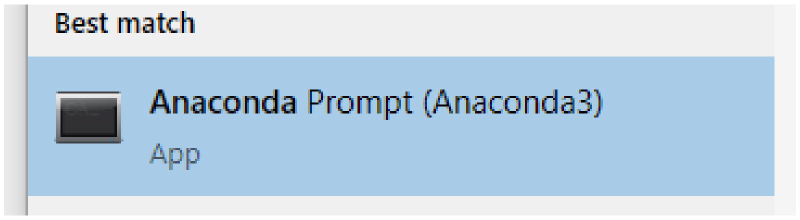
B4. Once it has finished running (you get an output of "Success! ..." on your console), you'd see the figure folder populated with your diaphragm shear and moment diagrams. For the template file, it'd look like the snapshot shown below:



Optional: Having a progressbar during script runs

If you look at caseB, you'd see a progress bar indicated each level's output plot as it's being processed. By default that's turned off, so you wont be able to see the interim progress; just the final output message. To enable progress bar you'd have to do two things:

1. Uncomment lines 8, 41 and 51 of *"diaphragm_shear_diagrams_caseB.py".*



2. Open up anaconda prompt and type "conda install -c conda-forge progressbar2" in order to install the progressbar module. You should then see the progress as it happens.

Notes and usual caveats:

1. As with any tool, verify all results to see if they make sense. While this would provide the demand diagrams for idealized diaphragms, local discontinuities such as openings would have to be investigated separately (for load path and resulting force distributions).
2. Also be wary of floor plates and analyses where these idealizations don't apply (for e.g. complex geometry, combination of materials where shear flow isn't uniform etc.).

PS: An astute reader can realize that idealized diaphragm shear diagrams and idealized collector-line force diagrams with points of resistance (such as at the nodes of braced frames and moment frames) are inherently similar in that they form the same type of demand distribution, where there is a uniform load resisted by discrete points.
This implies that it's a trivial extension of the above automated plots to also generate collector force diagrams.