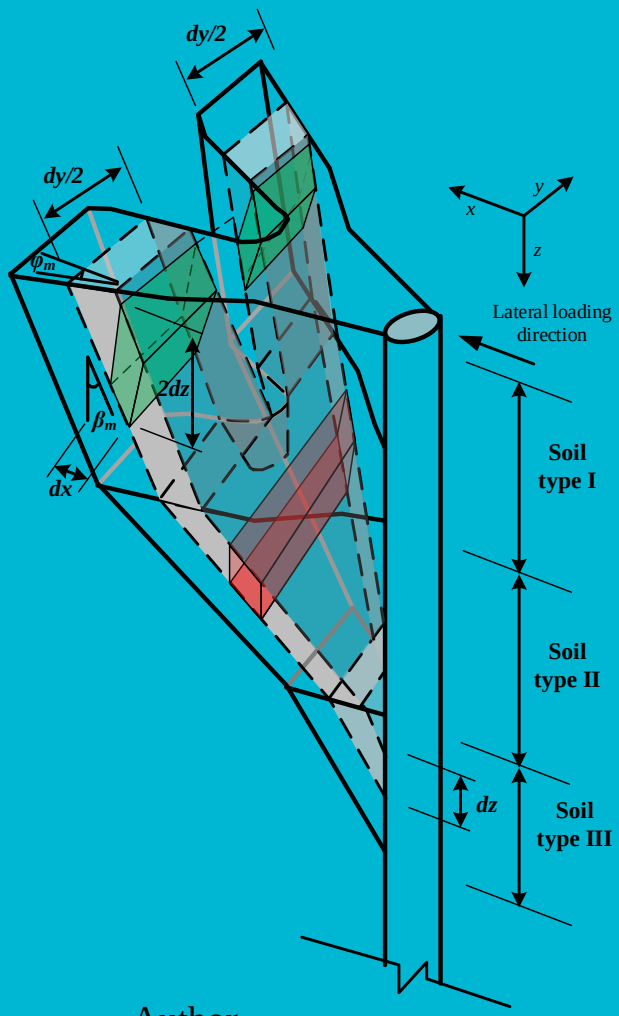
MANUAL
Analytical quay wall model

A computational model in MATLAB



Author
Mart-Jan Hemel
Version: 10/10/2023

TU Delft
Delft University of Technology

Contents

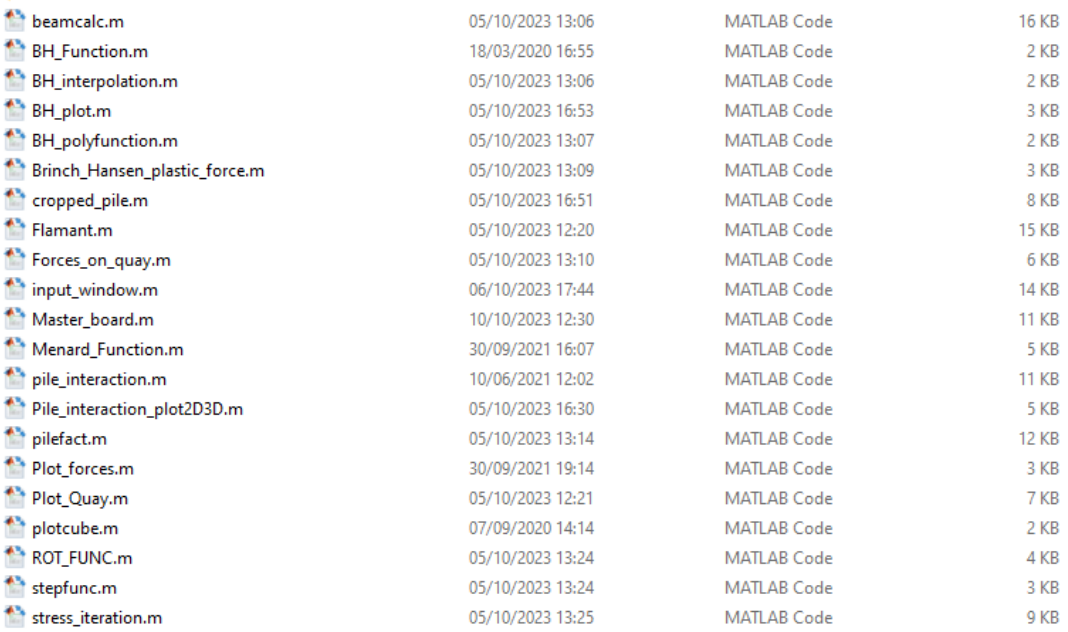
1 INTRODUCTION

As an integral component of the doctoral dissertation titled "Amsterdam Quay Walls Under Pressure – Modeling and Testing of Historic Canal Walls" authored by Mart-Jan Hemel, an analytical quay wall program was meticulously developed using MATLAB. This specialized program served various purposes, including the replication of experiments conducted at the Overamstel test site, conducting sensitivity and probability studies, and facilitating forensic research into the Grimburgwal's collapse. Notably, this model demonstrates substantial potential in efficiently and accurately modelling historical quay walls.

The primary objective of this manual is to provide a concise overview of how the program can be effectively utilized and an introduction to its constituent components. However, it's important to note that this manual does not delve into the in-depth methodologies that underpin the model. Detailed explanations of these methodologies can be found in the referenced dissertation.

2 MODULES

The program for modelling quay walls consists of 20 files. Three of them are scripts, and 17 files are referred to as functions. The entire program is controlled by the **Master board**, where the various functions and scripts are called upon to perform the quay wall calculation.



beamcalc.m	05/10/2023 13:06	MATLAB Code	16 KB
BH_Function.m	18/03/2020 16:55	MATLAB Code	2 KB
BH_interpolation.m	05/10/2023 13:06	MATLAB Code	2 KB
BH_plot.m	05/10/2023 16:53	MATLAB Code	3 KB
BH_polyfunction.m	05/10/2023 13:07	MATLAB Code	2 KB
Brinch_Hansen_plastic_force.m	05/10/2023 13:09	MATLAB Code	3 KB
cropped_pile.m	05/10/2023 16:51	MATLAB Code	8 KB
Flamant.m	05/10/2023 12:20	MATLAB Code	15 KB
Forces_on_quay.m	05/10/2023 13:10	MATLAB Code	6 KB
input_window.m	06/10/2023 17:44	MATLAB Code	14 KB
Master_board.m	10/10/2023 12:30	MATLAB Code	11 KB
Menard_Function.m	30/09/2021 16:07	MATLAB Code	5 KB
pile_interaction.m	10/06/2021 12:02	MATLAB Code	11 KB
Pile_interaction_plot2D3D.m	05/10/2023 16:30	MATLAB Code	5 KB
pilefact.m	05/10/2023 13:14	MATLAB Code	12 KB
Plot_forces.m	30/09/2021 19:14	MATLAB Code	3 KB
Plot_Quay.m	05/10/2023 12:21	MATLAB Code	7 KB
plotcube.m	07/09/2020 14:14	MATLAB Code	2 KB
ROT_FUNC.m	05/10/2023 13:24	MATLAB Code	4 KB
stepfunc.m	05/10/2023 13:24	MATLAB Code	3 KB
stress_iteration.m	05/10/2023 13:25	MATLAB Code	9 KB

Figure 2.1, Files within the MATLAB program.

2.1 Master_board – Global flow of the program

The master board contains of sections A – L. Within each section, a separate process is evaluated/computed. The sections are briefly listed.

A. Input of quay wall parameters
B. Pile wedge (no interaction and flat surface)
C. Pile wedge interaction correction (interaction + slope)
D. Brinch Hansen plastic force
E. Menard springs
F. Flamant soil stress model
G. Compute loads on quay wall
H. Iteration file creator
I. 1st iteration, fully elastic computation
J. nth iteration, elastic - plastic computation.
K. Storage of output (just a selection)
L. OUTPUT; W, M, V, Phi, Sigma

2.2 Run the program – errors or issues.

Before reading this manual, download the zip file and unzip all files into the one folder. Please verify whether the code runs successfully and generates the desired output.

To start the computation for a base case quay wall analysis, simply click the "RUN" 

button on the master board. This computation represents one of the quay wall experiments (segment A.I) conducted at the Overamstel test site as part of the PhD dissertation titled "Amsterdam Quay Walls Under Pressure – Modeling and Testing of Historic Canal Walls". If everything works properly, the output discussed in SECTION L should appear. If you encounter any **issues or errors**, it is highly likely that you may not have the proper version of MATLAB (2021a) installed or the necessary toolboxes required for the computation. To ensure smooth operation, it is recommended to install all the toolboxes when downloading MATLAB. By following these steps, you can enhance the likelihood of a successful execution of your MATLAB code for the quay wall analysis. Now the separate sections within the Master_board are discussed.

2.3 SECTION A: Input of quay wall parameters

In this section, the majority of the model input for the quay wall computation can be inserted. These are the quay wall geometry, structural properties, geotechnical properties and other conditions. The dimensions and type of input parameter is defined in the code itself.

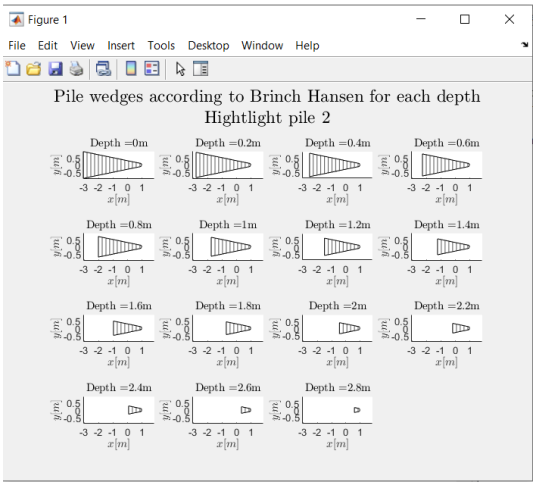
2.4 SECTION B: Pile wedge (no interaction and flat surface)

In this section, the focus is on the geometric development of three-dimensional pile wedges without considering slopes or neighboring piles. The construction of these wedges is accomplished through the utilization of the "pile_fact.m" file (Pile factory). A crucial factor in the determination of the pile wedge is the wedge fanning angle. This angle is explained in (Hemel, 2022) and can by found via: Lateral Loaded piles . The program is set so that granular layers have a wedge fanning angle equal to the angle of internal friction (line 52) and for cohesive layers that the wedge fanning angle is equal to 15 degrees (line 55). (See red box below)

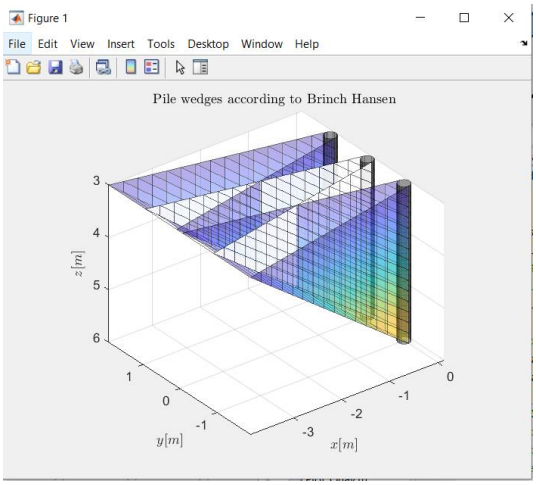
```
%% Wedge fan angel!!%%%%%%%%%%%%%%%%%%%%%%%%%%%% very important for the pile-soil-pile interaction
    f_phimo = 1;  %f_phimob * phi = alpha
    alpha = phi*f_phimo;                                    %horizontal wedge angle
    %cohesive layers do have a mobilized angle greater than 0 deg. (Gabr et al., 1990
    alpha(alpha==0) = 15 ; % EXAMPLE OF 15 DEGREES IS USED!!!!
    beta = 45 + alpha./2;                                   %vertical wedge angle
```

For visualizing the pile wedges, you have two available options. These options allow you to select the most suitable visualization method based on your preferences and requirements.

1) 2D Depth Sections: You can generate visualizations in 2D depth sections, progressing from top to bottom. To activate this option, set "PLT.TDwedgeplot" to 1. See an example of the plot below. To display a specific pile, you can do so by specifying the parameter "PLT.pileOI2D = 2".



2) 3D Visualization: Alternatively, you can choose to visualize the pile wedges in a three-dimensional format. To activate this option, set "PLT.D3Wedgeplot" to 1. See an example of the plot below. To display a specific pile, you can do so by specifying the parameter "PLT.pileOI2D = 2".



Important note: When generating plots for the soil wedges, it's crucial to deactivate all code starting from line 46 (identified as 'SECTION D Brinch Hansen plastic force') down to the last line by adding a '%' character at the beginning of each of these lines.

Furthermore, you should manually specify the desired plastic depth for the soil wedges, typically within the range of 2 to 3 meters. You can easily achieve this by entering the depth value through the input window in line 50 -> **See red box below.**

```
49    %% FOR PLOTS OF WEDGES ENTER HERE THE PLASTIC HEIGHTS PER PILE
50 -  plastic_zones = [3 3 3 3]; %EXAMPLE FOR PLOTS
```

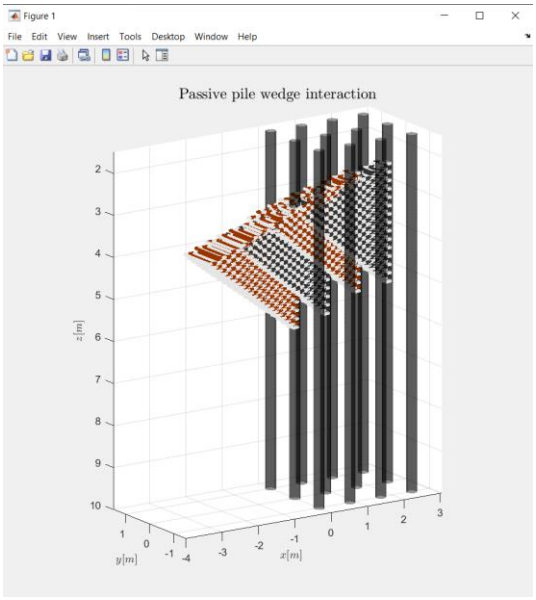
For a streamlined workflow, we recommend conducting the full calculation first and subsequently inputting the plastic depth values manually for the plots.

2.5 SECTION C: Pile wedge interaction correction (interaction + slope)

The calculation of wedge interactions is conducted in Section C. To determine these interactions, we employ the versatile "polyshape" tool, a fundamental component of MATLAB's toolbox. Utilizing this tool, we can calculate the reduction in pile wedge at each discretized depth. It's worth noting that there are multiple approaches available for these geometric calculations, allowing program designers the flexibility to choose the most suitable method.

Within the "pile_interaction.m" file, we undertake a comparison between the cropped pile wedges and a freely developed pile wedge. This comparison enables us to compute the Brinch Hansen correction factors, which account for both soil weight and soil cohesion. These correction factors are essential components in the analysis of lateral-loaded piles, as described in the context of the Lateral Loaded piles method.

You have the option to visualize the corrected and cropped pile wedges in a three-dimensional format. To enable this feature, simply set "PLT.cropwedgeplot" to 1 in line 39 of the master_board.m code. An illustrative example of such a plot is provided below for your reference.

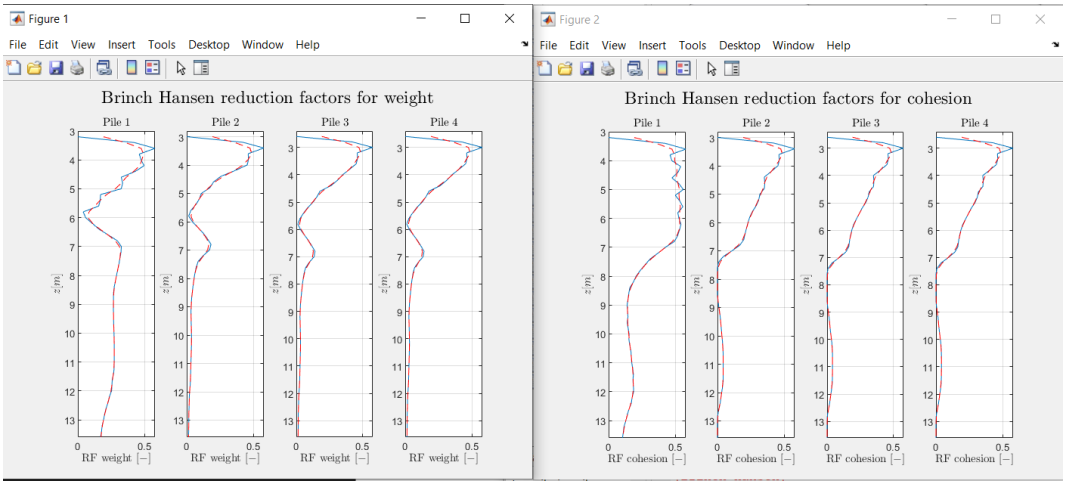


2.6 SECTION D: Brinch Hansen plastic force

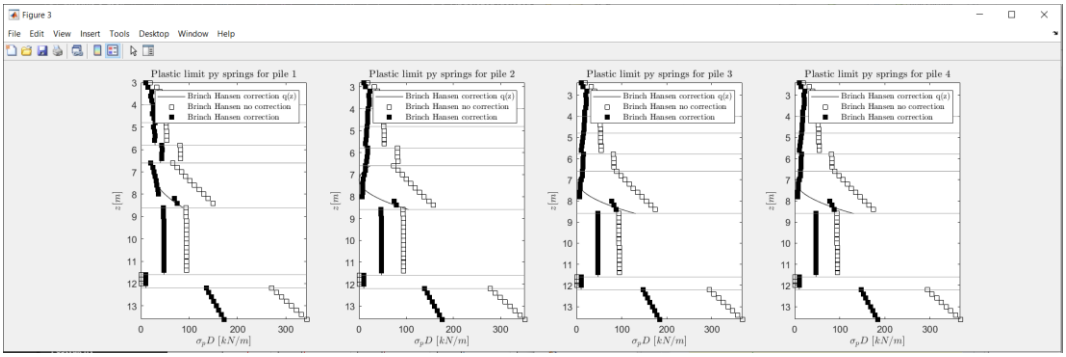
In this section, you will find two computational functions and two plotting functions. In the "Brinch_Hansen_plastic_force" function, we determine the ultimate soil resistance at various depths based on the pile dimensions and geotechnical data, following Brinch Hanssen's theory. Within the same

function, we also apply the Brinch Hanssen correction factors, which are obtained in Section C. Since we have now computed the Brinch Hanssen factors for each discrete depth, the "BH_interpolation" function performs interpolation on these discrete values to obtain continuous functions.

If you wish to visualize the Brinch Hanssen reduction factors for cohesion and specific weight, you can set "PLT.BHfactorplot = 1" in line 64. See example figure below, red dashed is interpolated. Furthermore, if you'd like to create plots for the corrected Brinch Hanssen ultimate strength, set "PLT.BH_forceplot = 1" in line 65. This will provide a clear visual representation of these important factors and strengths.



Brinch Hanssen reduction factors



Corrected Brinch Hanssen ultimate strength

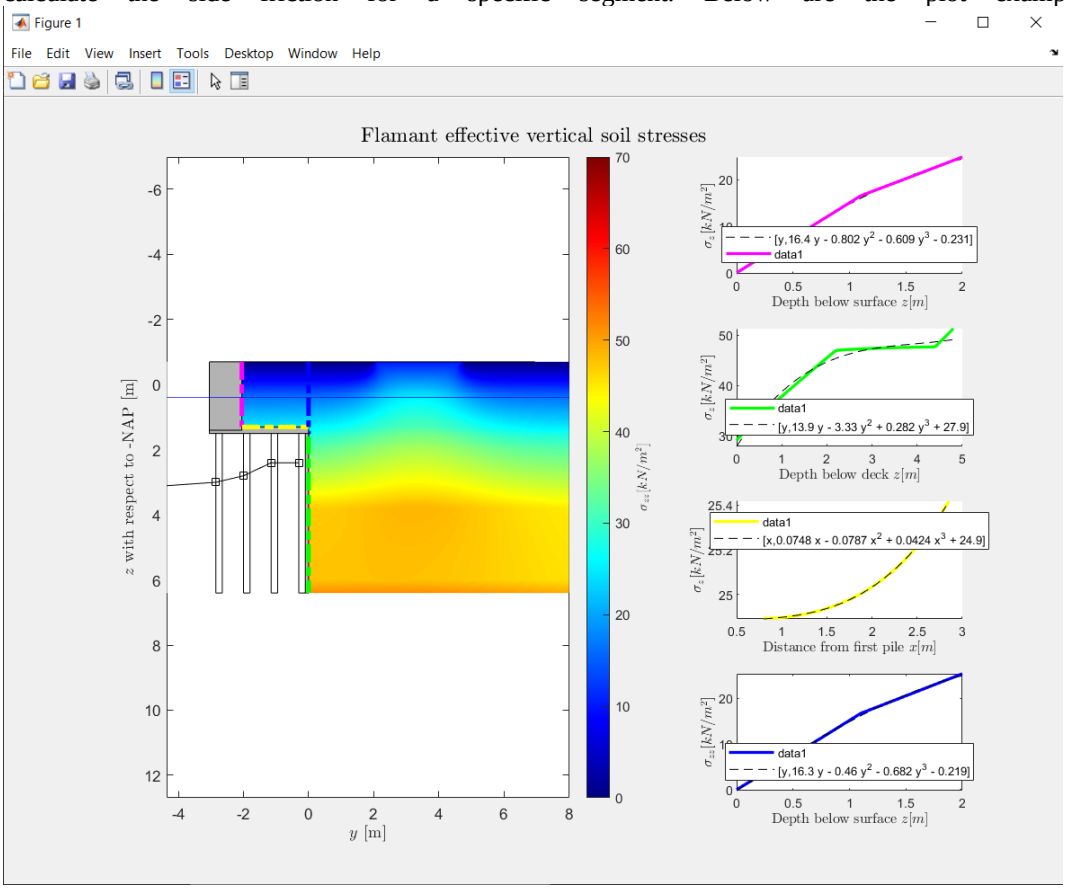
2.7 SECTION E: Menard springs

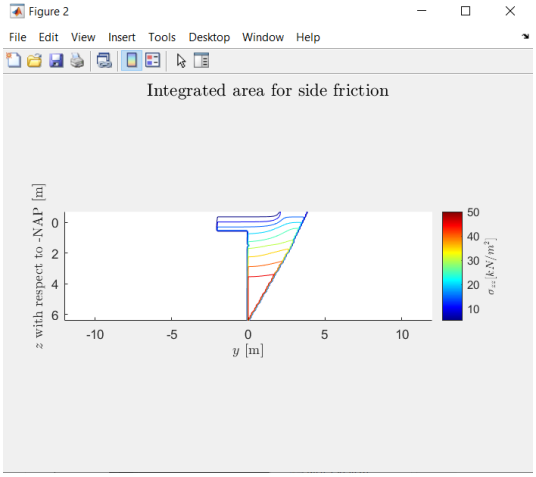
Using the "menard_function.m," we calculate the elastic soil response for each designated soil layer. It's important to note that this response remains constant within each specific soil layer.

2.8 SECTION F: Flamant soil stress model

In this module, we employ the theory of Flamant to compute the superposition of stress fields, resulting in the total effective vertical stress field within the active quay domain. The computations are carried out within the "Flamant.m" function, where you'll also find the relevant plotting code. To visualize the Flamant stress field, simply set "onoff_flamant = 1" in line 77.

Two plots will be generated when you set "onoff_flamant = 1" in line 77. One of these plots will display the effective vertical soil stress field, while the other will illustrate the domain of the stress field used to calculate the side friction for a specific segment. Below are the plot examples.





2.9 SECTION G: Compute loads on quay wall

In this module, we calculate the forces acting on the quay by integrating the stress field and multiplying it by the appropriate K factors (translation of vertical to horizontal soil stress). All these computations are carried out within the "Forces_on_quay.m" function.

2.10 SECTION H: Iteration file creator

This section is responsible for constructing an iteration MATLAB structure file that plays a crucial role in the iterative process where the elastic beam framework seeks equilibrium with external forces acting upon it. This iteration structure is dynamically updated throughout the various iterations performed. It serves as an input for the 'beamcal.m' function, which will be elaborated upon in the subsequent section, Section I.

2.11 SECTION I: 1st iteration, fully elastic computation

In this section, the first computation is performed, which is a fully elastic computation. No plastic spring limit is used. The provided MATLAB function, **beamcalc**, performs calculations related to a beam model representing the quay structure. Here is a breakdown of what this function does:

1. **Initialization**: The function initializes various parameters and variables, including defining symbolic variables and constants for calculations.
2. **Pile and Deck Parameters**: It imports parameters related to piles and decks, including their properties like spring constants, geometry, and loads.
3. **Connection Forces**: The function calculates and assigns connection forces and moments between the piles and the deck.
4. **Equation Generation**: It generates equations for beam deflections, rotations, bending moments, and shear forces for both piles and deck beams.
5. **Boundary Conditions**: The function sets boundary conditions at various points, including interfaces between soil layers, connections between piles and deck beams, and boundary conditions at the free end.
6. **Equation Assembly**: It assembles all generated equations into a single equation array.
7. **Solving the System**: The equation array is solved to determine the integration constants (C) for piles and deck beams.
8. **Substituting Constants**: The constants obtained from the solution are substituted back into the equations to obtain the final results for beam deflections, rotations, bending moments, and shear forces.
9. **Result Processing**: The function processes and returns the calculated results, including deflections, rotations, moments, shear forces, and other relevant variables.
10. **Additional Calculations**: Some additional calculations are performed, including the calculation of normal forces, pile head rotations, and tangential shear forces.
11. **Output**: The function returns the calculated results as output variables, including deflections (**w_d** and **w_p**), rotations (**phi_d** and **phi_p**), bending moments (**M_d** and **M_p**), shear forces (**V_d** and **V_p**), coordinates (**X** and **Y**), normal forces (**Normal_force**), shear forces (**Shear_force**), pile head rotations (**Phi_pile**), and tangential shear forces (**V_tangent**).

Overall, this function is used to analyze the structural behavior of a quay structure, considering the interaction between piles, soil, and deck beams. It calculates and provides various mechanical properties and forces at different points within the structure.

2.12 SECTION J: nth iteration, elastic - plastic computation.

In this section, the elastic-plastic computations are performed. Based on the output of the first iteration, a new input iteration file is created and used in a new beam_calc calculation. An iterative process is used for the structural analysis. Here's a breakdown of what this code is doing:

1. **Initialization**: The code begins by initializing the maximum number of iterations (**max_number_of_itterations**) to 10.
2. **Iteration Loop**: It then enters a loop that iterates from 1 to the maximum number of iterations (**max_number_of_itterations**).
3. **Display Iteration Number**: Within each iteration, it displays the current iteration number using the **disp** function.
4. **Stress Iteration**: It calls the **stress_iteration** function, passing various inputs, and updates the **iteration** data structure.
5. **Beam Calculation**: It calls the **beamcalc** function, passing the updated **iteration**, and other input data (**pile_input**, **loads**, and **quay_input**) to calculate various beam-related parameters.
6. **Checking for Stability**: The code checks for the stability of each pile by comparing matrices from the current iteration with matrices from previous iterations. If a pile's behavior has stabilized (meaning the matrices are the same as in previous iterations), it marks that pile as stable and increments a counter (**paalteller**). If all piles are stable or the maximum number of iterations is reached without stability, it breaks out of the iteration loop.
7. **Completion Message**: If all piles are stable, it displays a message indicating that the computation has been completed and specifies the number of iterations required to achieve stability.

2.13 SECTION K: Storage of output (just a selection)

In this section, we calculate and record key structural parameters, including deflection, bending moment, shear force, and phi, for both headstocks and piles. The computed values are organized into output arrays. It's important to note that this section provides a subset of the available output data, and you can extract any specific results of interest from the comprehensive calculations performed in the beamcalc function.

3 OUTPUT - SECTION L: OUTPUT; W, M, V, Phi, Sigma

The last section of the master board is a plotting section. This code section sets variables to control whether various types of plots are generated or not. Here's what each variable does:

- **WPLOT**: If set to 1, it indicates that deflection (**W**) plots should be generated.
- **MPLOT**: If set to 1, it indicates that bending moment (**M**) plots should be generated.
- **VPLOT**: If set to 1, it indicates that shear force (**V**) plots should be generated.
- **PHIPLOT**: If set to 1, it indicates that rotation (**Phi**) plots should be generated.
- **SIGMAPLOT**: If set to 1, it indicates that stress (**Sigma**) plots should be generated.

After setting these variables, the code calls a plotting program or script (**Plot_Quay**) to generate the plots based on the settings of these variables. **Below, each plot is visualized for illustrative purposes.**

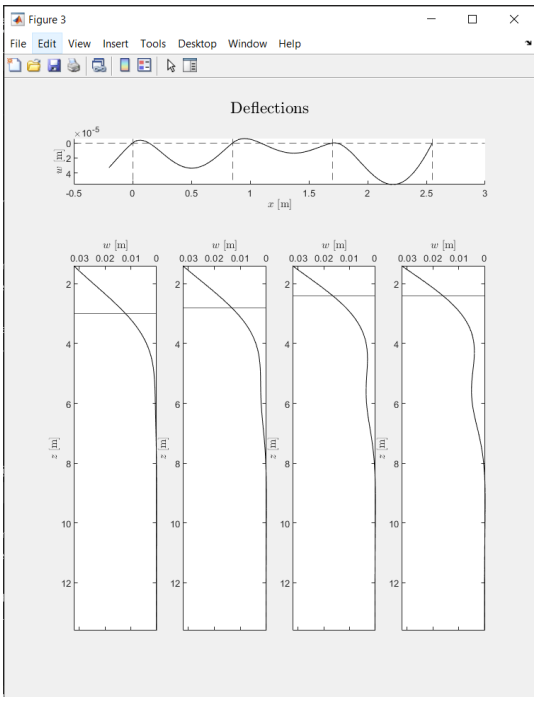


Figure 3.1 Deflection of the headstock and piles

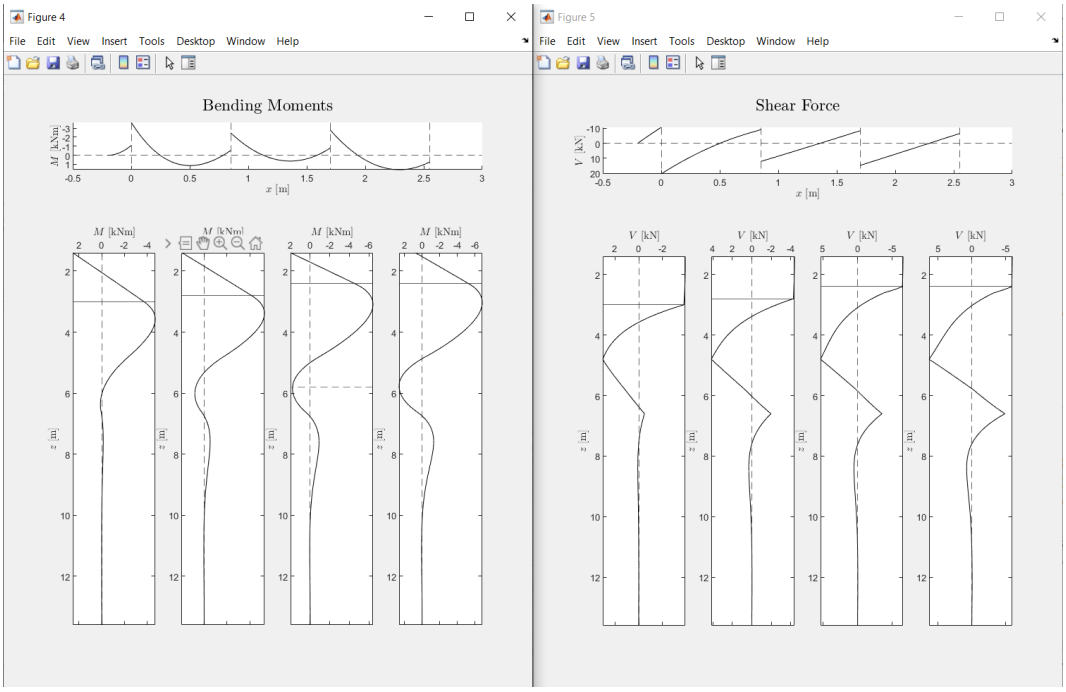


Figure 3.2, Moment of the headstock and piles LEFT. Shear force of the headstock and piles RIGHT.

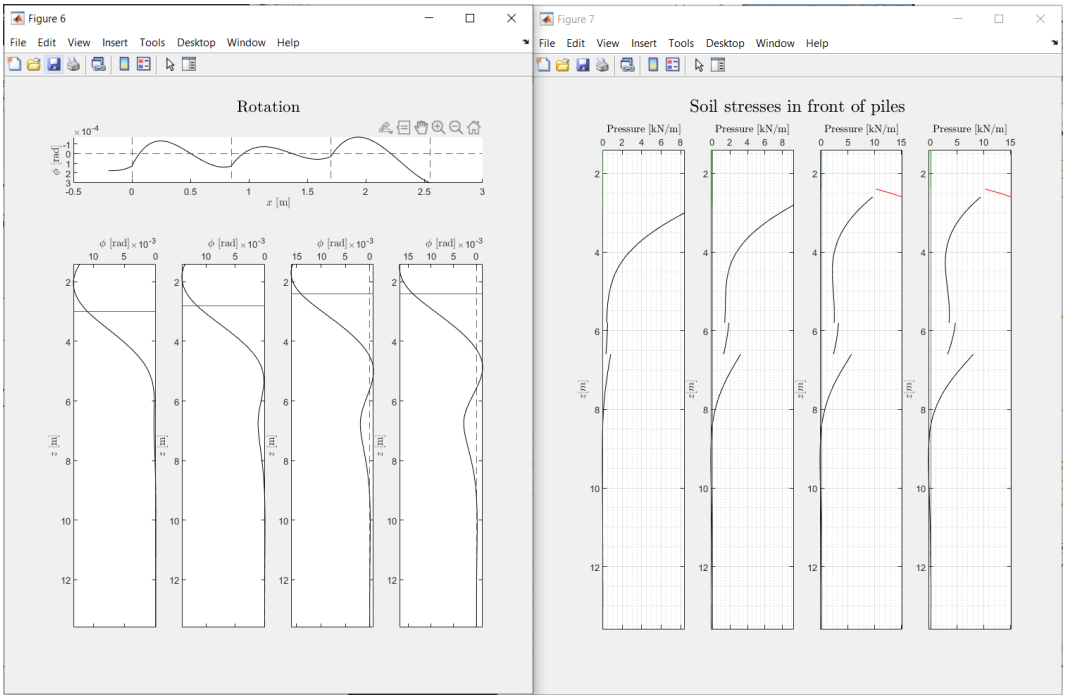


Figure 3.3, Rotation of the headstock and piles LEFT. Soil stresses in front of piles RIGHT.

4 PERFORMING MULTIPLE QUAY WALL COMPUTATIONS

This program describes a single quay wall calculation using the master board. However, when you need to perform multiple calculations rapidly, it's not practical to execute each one separately. To streamline the process, it is recommended to transform the master board into a function with defined input and output parameters. By doing this, the master board can be easily called upon to calculate quay wall forces or displacements based on various quay wall inputs.

4.1 Creating a function for the masterboard and sampling example

1) First, consider converting the master board into a function. This can be done by defining the necessary input parameters and specifying the desired output. An example could be:

```
function    [maxMoment,deflection]    =    Masterboard(Top_load,Number_of_Piles,
Diameter )

        SECTIONS A-L (THE PROGRAM)!!

        maxMoment = xxxx
        deflection = xxxx

end
```

This function calculates the maximum moment and deflection of a quay wall based on input parameters including, top load, number of piles, and diameter. It provides a versatile tool for performing multiple computations by calling the 'Masterboard' function with varying input values.

2) Ensure that any input data you intend to use in the masterboard is also passed correctly to the input window function. To achieve this, you can simply add more input parameters to the input function.

An example is given:

Example MATLAB CODE

```
% Define input parameters
topLoad = [2, 4, 6, 8];
numPiles = [3, 4];
diameter = [0.24, 0.26]; % Ensure diameter is specified as a 1x2 vector

% Initialize arrays to store results
maxMoment = zeros(length(topLoad), length(numPiles), length(diameter));
deflection = cell(length(topLoad), length(numPiles), length(diameter)); % Use a cell
array

% Loop through each combination of input parameters
for j = 1:length(topLoad)
    for k = 1:length(numPiles)
        for l = 1:length(diameter)
            [maxMoment(j, k, l), deflection{j, k, l}] = Masterboard(topLoad(j),
numPiles(k), diameter(l));
        end
    end
end

% Analyze and display results if needed
% You can access the results using maxMoment and deflection cell array

% Example 1: Display maximum moments for different top loads
figure;
for j = 1:length(topLoad)
    subplot(length(topLoad), 1, j);
    imagesc(squeeze(maxMoment(j, :, :))); % Use 1 for the first diameter
    title(['Top Load = ', num2str(topLoad(j))]);
    colorbar;
end
xlabel('Number of Piles Index');

% Example 2: Display deflection for different diameters
figure;
for l = 1:length(diameter)
    subplot(1, length(diameter), l);
    % Check if deflection is numeric before using surf
    if isnumeric(deflection{1, 1, l})
        surf(cell2mat(squeeze(deflection(:, :, l))));
        title(['Diameter = ', num2str(diameter(l))]);
        shading interp;
        colorbar;
    else
        disp(['Deflection data for Diameter = ', num2str(diameter(l)), ' is not
numeric.']);
    end
end
xlabel('Top Load Index');
```

4.2 Reduction in computational time for sampling computations

To reduce computational time, do the following:

Larger dz Grid Size: Increase the grid size (dz) that is used in the calculations. A larger dz value means that the calculations will be performed with larger intervals along the depth or position, which can reduce the number of computations required. This can be beneficial for speeding up calculations, but it may also result in less detailed or accurate results, depending on the specific analysis.
Turn Off All Plots: Set all the plot-related variables (WPLOT, MPLOT, VPLOT, PHIPLOT, SIGMAPLOT) to 0. This means that no plots will be generated during the computation. Also turn illustrative plots (flamant, brinch hanssen, wedge plots off). Generating plots can consume computational resources and time, so turning them off can significantly speed up the overall analysis.
Use Fewer Soil Layers: Consider using a reduced number of soil layers, especially if layers below NAP-7 in Amsterdam are not relevant for the deflection of the piles. Simplifying to one layer for these lower depths can further enhance computational efficiency.

----------END OF MANUAL----------