

**Report by:** *Ana Clara R. Gouveia*

05/13/2015

22.520 NUMERICAL METHODS FOR PDE’S

Instructor: David J. Willis

Circular Foundations Deflection Analysis

Finite Element Method

Term Project

Table of Contents

[Project Summary 4](#_Toc419285645)

[Physical Problem 4](#_Toc419285646)

[Numerical Method 5](#_Toc419285647)

[Results 7](#_Toc419285648)

[Validation 10](#_Toc419285649)

[Error Consideration 12](#_Toc419285650)

[References 13](#_Toc419285651)

[Appendix A: Matlab Files 14](#_Toc419285652)

[Circular Discretization: 14](#_Toc419285653)

[Deflection Solver: 14](#_Toc419285654)

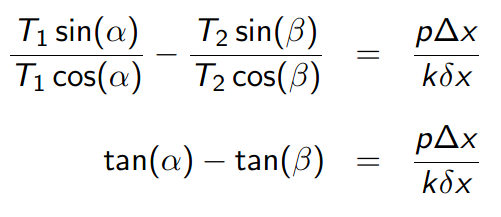
# Project Summary

Circular plates are used in several kinds of structural engineering applications. In particular, the use of circular foundations is common practice within specialty construction such as wind turbines and nuclear power plants footings. Structural Engineers must go across several steps in order to efficiently analyze and design shallow foundations. In general the approach taken correlates soil bearing pressure to the foundation materials strength capacity and its supported load.

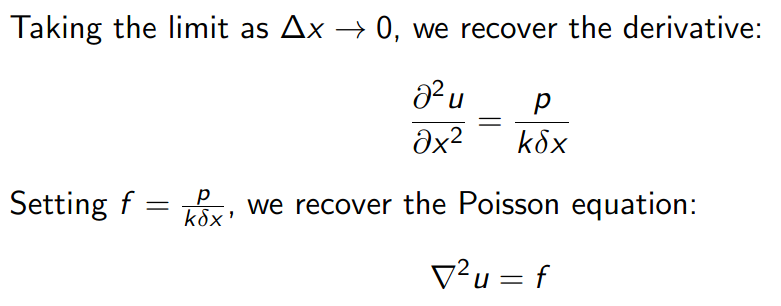
On the other hand, Geotechnical Engineers must assure that these foundations are safe against overall shear failure in the soil that supports them, and that they do not undergo excessive displacement, i.e. soil settlement. Within the field, the load per unit area of a foundation at which shear failure occurs is named ultimate bearing capacity.

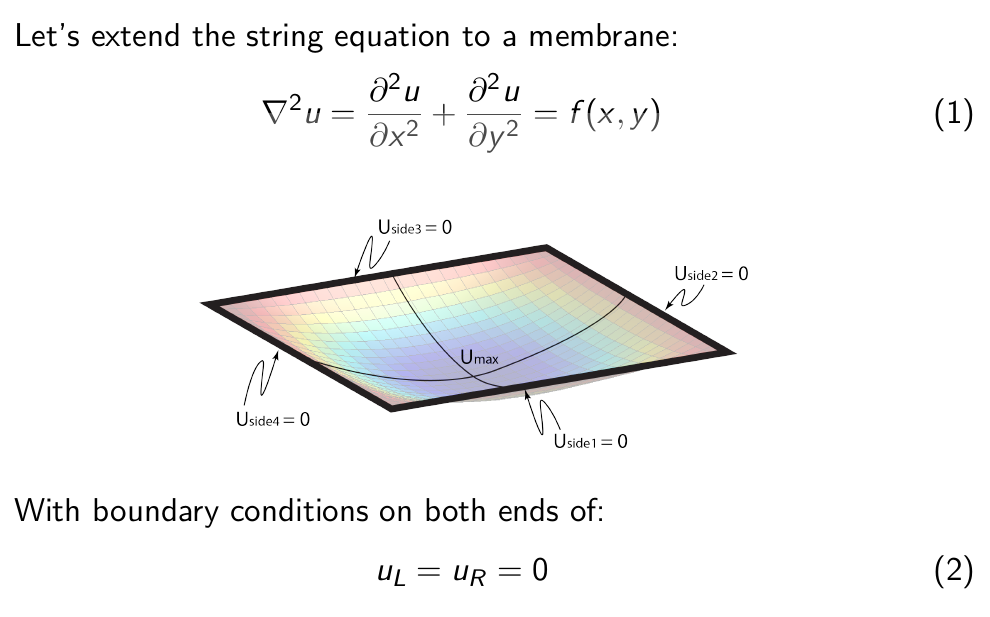
This project aims to develop a comprehensive numerical model to determine the deflection of a circular footing foundation, further correlating results obtained with the method to current analytical solutions. In order to satisfy this project’s requirements the numerical model developed for this project uses of polar coordinates to obtain the deflection of the structure in question.

# Physical Problem



EQ (1)





In this project we’ll solve the Laplace equation = f = qu, as the ultimate bearing capacity load caused by the footing.

In this problem it is assumed that the ends of this circular plate is clamped, thus boundary conditions to the problem are the traditional u = 0 at all boundary points.

# Numerical Method

In the case of a circular foundation, the numerical approach taken in this project will consist of the development of a polar coordinates method for the problem solution. In a nutshell, this program would use of finite elements for structural analysis. The purpose of the program would consist of solving a 2-D FEM problem to determine the structural deformation of the structure, therefore solving the problem in 3-D. The non-Euclidean nature of the proposed mathematical approach had potentially present itself as a challenge in this project’s development, however, the stress distributions obtained with the solution, see Figures 3 and 4, seem to agree with physical expectations.

Analytically, an alternative to determine the maximum deflection1 occurring at r = 0 is then:



EQ (2)

Where,

P = load

a = area radius

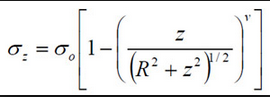
D = Eh3 / [12(1 – v2)]

E = elastic modulus

h = plate thickness

v = Poisson’s ratio

In this report, one of the numerical model validations is done using Boussinesq’s Equation for Vertical Stress:



EQ (3)

Where,

 = load

 = area radius

z = depth below soil surface

R = foundation radius

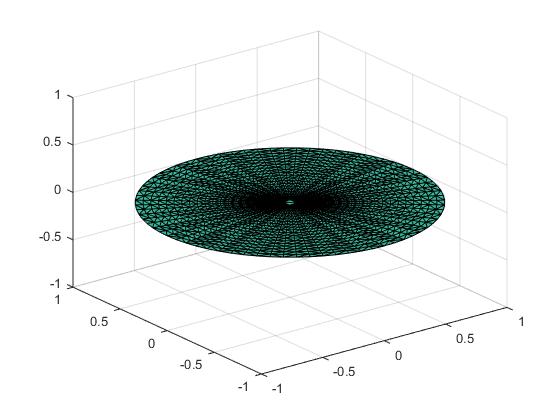
Since this project’s analysis is done in the elastic range, the equation used to obtain the soil settlement uses of the stress obtained with Boussinesq’s (EQ. 3) as seen in EQ. (4).

# Results

In order to design this project’s numerical method the following steps were taken. Files associated to each one of these parts are attached to Appendix A of this project and in the virtually shared Matlab folder.

1. **Circular Discretization of Geometry:**

In this step, a for loop with r and theta in small increments was used. Using the Matlab special algorithm Delaunay, allowed for a creation of a fine mesh. Figure 1 represents the resulting meshed structure.

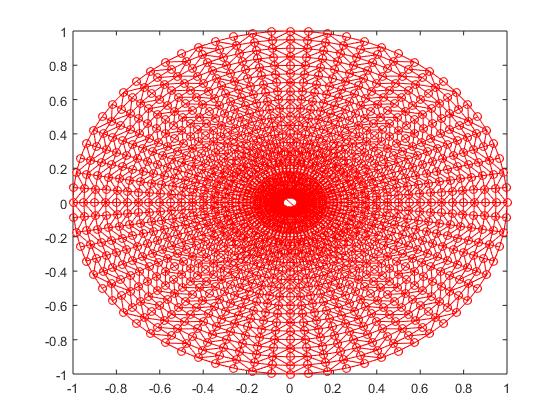


***Figure 1.*** Meshed circular structure

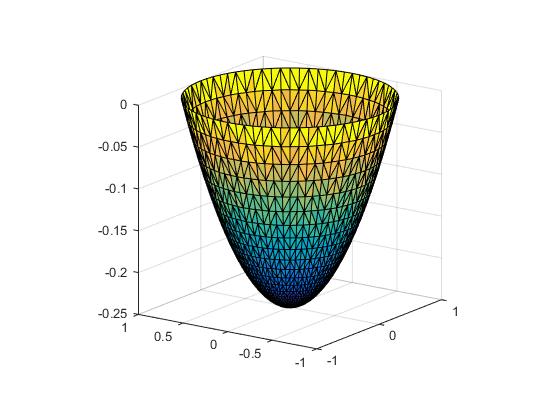
1. **Finite Element Model of the circle**

In this model, instead of using an assumed bearing capacity, qu, the value of force on the domain used is equal to q = f = -1. The purpose of this value is to generate a unit gravitational force representation that can potentially further extrapolate this project results to specific applications, such as q = qu. The solution of the problem yielded a maximum deflection equal to z = -0.2494.

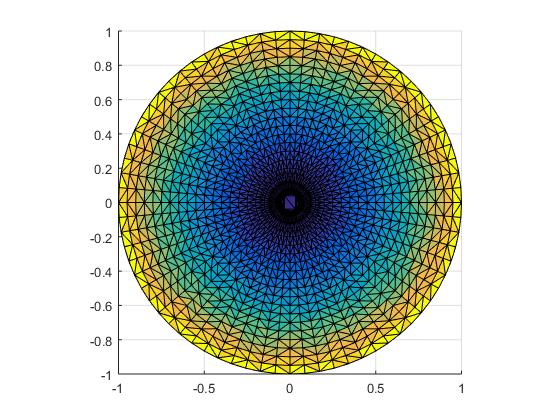
The m.file associated to this part of the project can be found in Appendix A and shared Matlab folder. Figure 2 represent the finite element model of this membrane, whereas Figure 3 portrays the deflected shape of the membrane. Figure 4 show the plan view of the deflected structure.



***Figure 2.*** Finite Element Model Representation



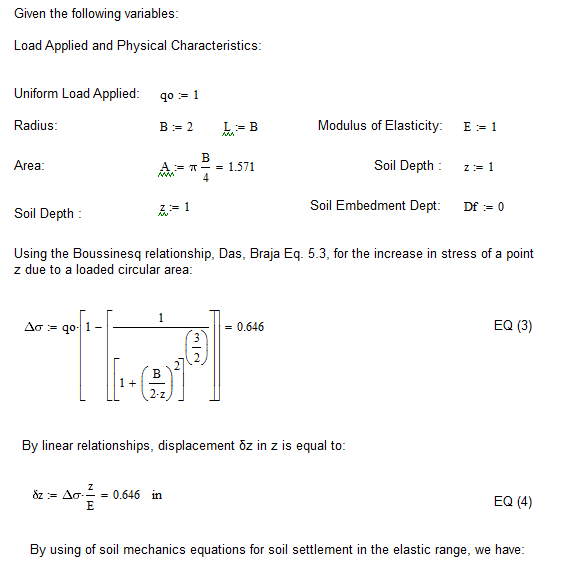
***Figure 3.*** 3-D view of the deflected structure

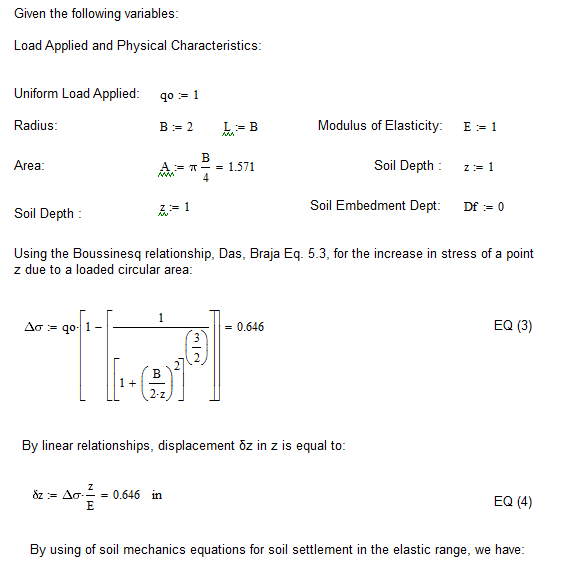


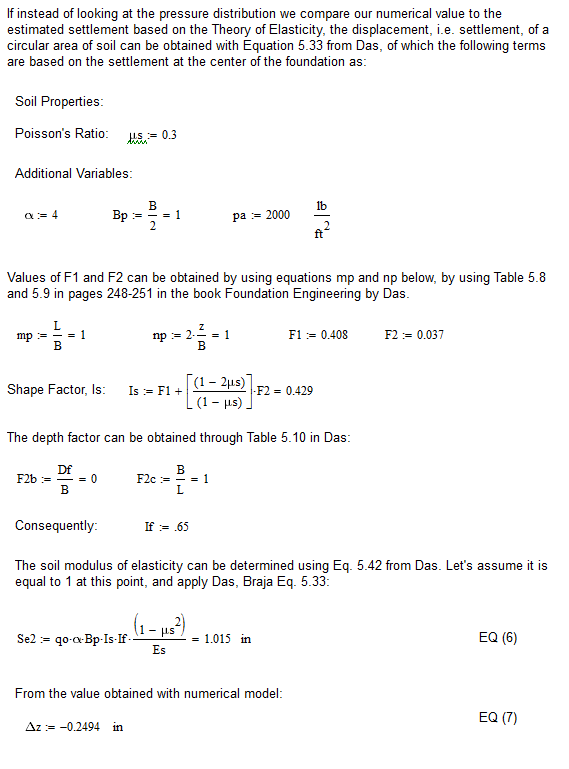
***Figure 4.*** Plan view of stresses distribution in the structure

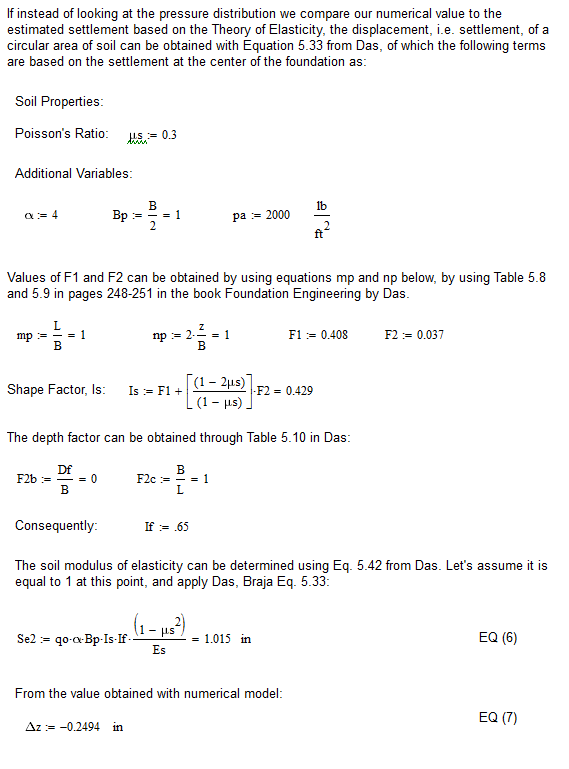
# Validation

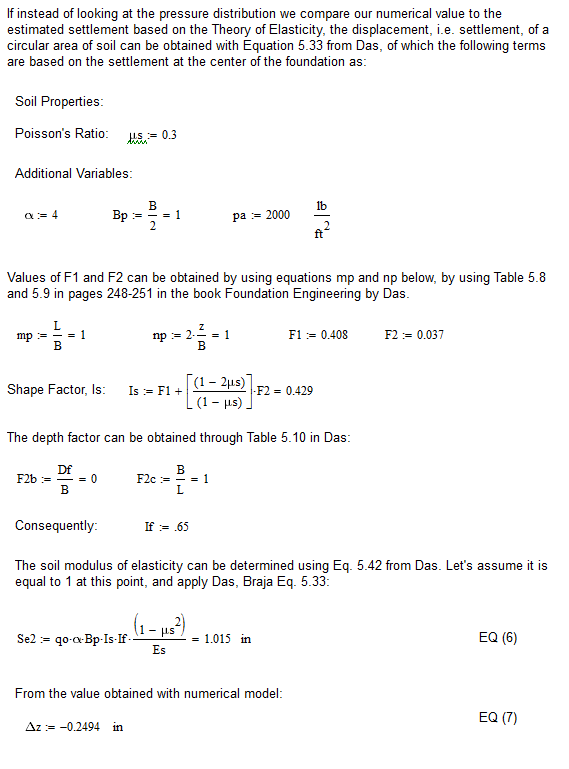
Current analytical solutions will be used in order to validate the numerical model developed.

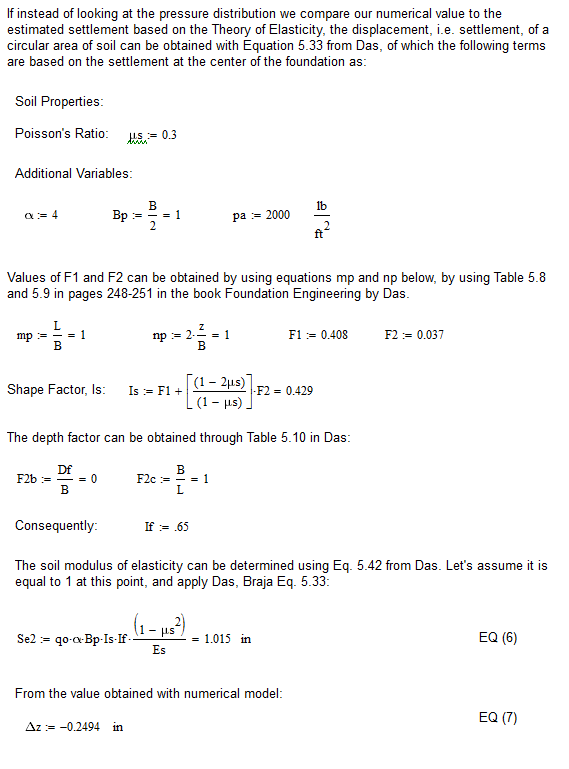








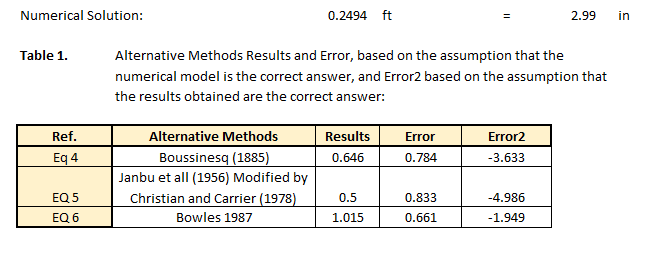






# Error Consideration

The values for deflection and soil settlement obtained with equations 4 through 6 in this report were further compared to this project’s results, yielding the errors in Table 1.



It is necessary to note that the following list of assumptions may have caused the difference between solutions:

* Soil Modulus of Elasticity equal to 1.
* Possible agreement and understanding of units used.
* Need to expand numerical model to accommodate appropriate energy losses, hence developing a more adequate model with Poisson’s equation.

# References

Das, Braja. *Principles of Foundation Engineering*. Cengage Learning, 2011.

Faraji, Susan and Jerome Connor. *Principles of Structural Engineering*. New York: Springer, 2013.

Lindfield, George and John Penny. *Numerical Methods Using Matlab*. Upper Saddle River: Prentice Hall, 1999.

# Appendix A: Matlab Files

## Circular Discretization:

%CircularDiscretization:

clear all

counter = 1;

for(r = 0.05:0.05:1)

for(theta = 0:5:355)

x(counter) = r\*cosd(theta);

y(counter) = r\*sind(theta);

counter = counter + 1;

end

end

Connectivity = delaunay(x,y);

Vertices = [x',y']

figure

trisurf(Connectivity, x,y,x\*0)

## Deflection Solver:

% Finite elements on circle

% discretizing the domain

Vertices

Connectivity

NNodes = length(Vertices);

NElem = length(Connectivity);

% Initialize A and f

A = zeros(NNodes, NNodes);%spalloc(NNodes, NNodes,6\*NNodes);

F = zeros(1, NNodes)

% Setup the A-matrix & RHS

for(i=1:NElem)

% Figure out what the triangle is!!!!!

ConnectedNodes = Connectivity(i,:);

V1 = Vertices(ConnectedNodes(1),:);

V2 = Vertices(ConnectedNodes(2),:);

V3 = Vertices(ConnectedNodes(3),:);

plot([V1(1) V2(1) V3(1) V1(1)], [V1(2) V2(2) V3(2) V1(2)],'r-o')

%pause

hold on

% First calculate the area -- Oh No!!!!!!!!!!!!!!!!!!!!!

L1 = [V2-V1,0];

L2 = [V3-V1,0];

Area = norm(1/2\*cross( L1, L2));

% Solve the matrix of basis function nodes

Plane\_Mat = [V1(1) V1(2) 1; V2(1) V2(2) 1; V3(1) V3(2) 1];

RHS1 = [1 0 0]';

RHS2 = [0 1 0]';

RHS3 = [0 0 1]';

abc1 = Plane\_Mat\RHS1;

abc2 = Plane\_Mat\RHS2;

abc3 = Plane\_Mat\RHS3;

a1 = abc1(1);

b1 = abc1(2);

c1 = abc1(3);

a2 = abc2(1);

b2 = abc2(2);

c2 = abc2(3);

a3 = abc3(1);

b3 = abc3(2);

c3 = abc3(3);

% Interactions between basis functions (gradient \* gradient)

ElementMatrix = Area\*[a1\*a1 + b1\*b1 a2\*a1 + b2\*b1 a3\*a1 + b3\*b1;

a1\*a2 + b1\*b2 a2\*a2 + b2\*b2 a3\*a2 + b3\*b2;

a1\*a3 + b1\*b3 a2\*a3 + b2\*b3 a3\*a3 + b3\*b3];

CN = ConnectedNodes;

A(CN(1),CN(1)) = A(CN(1),CN(1)) + ElementMatrix(1,1);

A(CN(1),CN(2)) = A(CN(1),CN(2)) + ElementMatrix(1,2);

A(CN(1),CN(3)) = A(CN(1),CN(3)) + ElementMatrix(1,3);

A(CN(2),CN(1)) = A(CN(2),CN(1)) + ElementMatrix(2,1);

A(CN(2),CN(2)) = A(CN(2),CN(2)) + ElementMatrix(2,2);

A(CN(2),CN(3)) = A(CN(2),CN(3)) + ElementMatrix(2,3);

A(CN(3),CN(1)) = A(CN(3),CN(1)) + ElementMatrix(3,1);

A(CN(3),CN(2)) = A(CN(3),CN(2)) + ElementMatrix(3,2);

A(CN(3),CN(3)) = A(CN(3),CN(3)) + ElementMatrix(3,3);

F(CN(1)) = F(CN(1))+Area/3;

F(CN(2)) = F(CN(2))+Area/3;

F(CN(3)) = F(CN(3))+Area/3;

end

% Boundary conditions

for(i = 1:NNodes)

V1 = Vertices(i,1);

V2 = Vertices(i,2);

if( (V1^2 + V2^2)>.99)

A(i,:) = 0;

A(i,i) = 1;

F(i) = 0;

end

end

Solution = A\-F';

figure

trisurf(Connectivity, Vertices(:,1), Vertices(:,2), Solution)