Finite difference analysis of plate structures

By Chulong Li

To finish the additional master project as part of master course of Science at the Delft University of Technology

Student:	Chulong Li	
Project duration	Sept 3, 2020 – Oct 31, 2020	TU Delft
Assessment committee	Dr.ir. P.C.J. (Pierre) Hoogenboom	TU Delft
	Dr.ir. M.A.N. (Max) Hendriks	

Summary

The finite difference method (FDM) is widely used in solving plate problems. However, the traditional application of FDM on plate problems involves higher-order differential approximation and a number of edge and corner molecules. It makes those programs contain larger numbers of code and therefore programming errors are probable. One way to avoid that is by decomposing higher-order differential approximation into a number of first-order differential equations. Solving these first-order differential equations simultaneously can be programmed shortly, however, it requires much computer time and memory. Since computer capacity increases every year, this method seems to come within reach.

The research question of this report is: Can plate problems be solved by considering first-order differential equations only?

To this end, a 600-line Python program has been built which is able to solve 11 plate equations simultaneously only by first-order finite-difference approximation. Analyzed were three isotropic rectangular plates with various loadings and edge conditions. The major results (displacement, bending moment, and shear force) are generally 10 to 20% smaller than the analytical solutions. The difference can be caused by the grid size but also by a programming error.

The conclusion is that the proposed method works and is within reach of modern computation capacity.

Content

List of figures

List of tables

List of equations

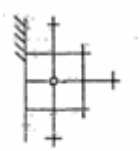
Introduction

1.1. Motivation for research

In the classical finite difference method (FDM) for perpendicularly loaded plates, the biharmonic equation is solved by a fourth-order differential approximation. However, to including various boundary conditions, classical FDM always involves a large number of edge and corner molecules (see Equation 2).

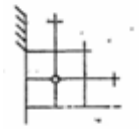
$$\frac{\partial^4 w}{\partial x^4}+2\frac{\partial^4 w}{\partial x^2\partial y^2}+\frac{\partial^4 w}{\partial y^4}=\frac{q}{D}$$

Equation 1: Biharmonic equation



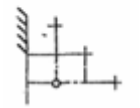
Point adjacent to fixed edge

$$21w_0-8\left(w_N+w_E+w_S\right)+2\left(w_{NE}+w_{SE}\right)+w_{NN}+w_{SS}+w_{EE}=\frac{P_0\lambda^4}{D}$$



Point adjacent to fixed-free corner

$$20w_0-8\left(w_N+w_E\right)+\left(-6+2\mu\right)w_S+2w_{NE}+\left(2-\mu\right)w_{SE}+w_{NN}+w_{EE}=\frac{P_0\lambda^4}{D}$$



Point adjacent to fixed corner on free edge

$$\left(17-8\mu+5\mu^2\right)w_0+\left(-12+4\right)w_N+\left(-8+4\mu+4\mu^2\right)w_E+\left(4-2\mu\right)w_{NE}+2w_{NN}$$
$$+\left(1-\mu^2\right)w_{EE}=\frac{P_0\lambda^4}{D}+\frac{2\lambda^3}{D}\left(R_y\right)_0$$

Equation 2: Edge and corner molecules adopted from (Doshi, 1964)

$$\frac{\partial^4 f}{\partial x^2\partial y^2}\approx\frac{1}{\left(\Delta x\Delta y\right)^2}\left[4f_{m,n}-2\left(f_{m+1,n}+f_{m-1,n}+f_{m,n+1}+f_{m,n-1}\right)+f_{m+1.n+1}+f_{m+1.n-1}+f_{m-1.n+1}+f_{m-1.n-1}\right]\equiv L^4{}_{xy}\left(f\right)$$

Equation 3: Central difference approximation of fourth-order derivatives adopted from (Reddy and Gera, 1979)

This report investigates a newly proposed finite difference program, in which only first order differential equations are solved, which greatly reduces the number of molecules. For the bending of thin elastic plates, the exact plate theory includes three constitutive equations and eight first-order differential equations. The new program could solve 11 equations simultaneously without higher-order differential approximation and edge and corner molecules.

$$u\left(x\right)=u_0+x\left(\frac{\partial u}{\partial x}\right)+\frac{x^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)+\frac{x^3}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)+\frac{x^4}{24}\left(\frac{\partial^4 u}{\partial x^4}\right)$$

$$u\left(x\right)\approx a+b\cdot x+c\cdot x^2$$

$$\left\{\begin{matrix}u_0=u\left(-\Delta x\right)\\u_1=u\left(0\right)\\u_2=u\left(\Delta x\right)\end{matrix}\right\}\Rightarrow\frac{\partial u}{\partial x}\approx\left\{b=-\frac{u_0-u_2}{2\Delta x}\right\}$$

Equation 4: Central difference approximation of first order derivative

Compared to the fourth-order derivatives in classical FDM, the difference approximation of first order derivatives is simpler which leads to a smaller size of the proposed program. Moreover, it has potential to be extended to shell structures, for which the classical molecules are too large to write down. However, in order to have reasonable accuracy, the enormous size of the matrixes causes this program usually to take a substantial amount of computing time and computer memory.

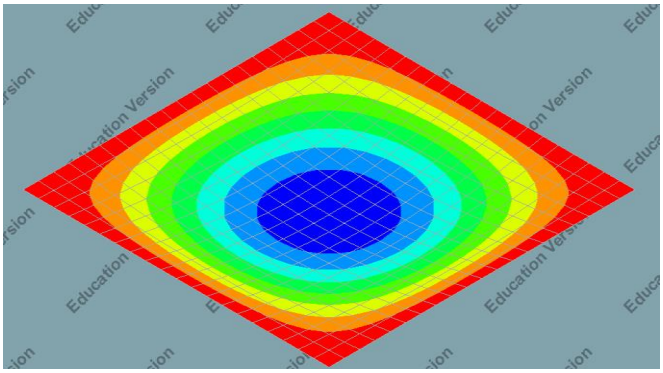


Figure 1: Deflection of perpendicularly loaded plates

1.2 Why finite differences

Finite element and finite volume methods are industry standard nowadays due the greater generality and sophistication of those methods. Various FEM packages are intensively used by structural engineers for plate and shell analysis. However, experience shows that often engineers do not understand the software that they are using. Compared to FEM, the advantage of finite differences is its simplicity. Finite difference programs usually have simpler structure from both mathematical and coding perspective. Especially for education purpose, some simple models are frequently used a lot for teaching and in research. It is always good for engineers to understand from scratch with easy-to-verify computer codes when they are solving the model problems.

1.3 Research questions

The idea of solving plate equations by first-order finite differential equations was taken from the previous code done by Dr. P.C.J. Hoogenboom. He has attempted to use the finite difference method for solving the 21 Sanders-Koiter equations for any shell with orthogonal parameterization. However, the development of his method was not successful yet.

For the purpose of examining the validity of this method, a simplified version of this code is created as newly proposed finite difference program aiming to solve 11 thin plate bending equations simultaneously. At start of research, the validity of this new program remains unknowns. So, the main question is: “Does the proposed method work?”

If the newly proposed program does not work, the potential causes behind its failure will be discussed. The next research question will be “What are the fundamental reasons behind its failure?” If the newly proposed program does preform reasonably, it has to be validated on its accuracy of the results. So, the next research question is “what is level of its accuracy compared to classical FDM?”.

Based on the experience gained in this project, the final research question “Can the proposed method be extended to shell structures?” will not be answered. It will be recommended for future research.

1.4 Research workflow

Based on the research questions, the workflow of research is organized as follows:

1. Comprehend the structure and format of the code developed by Dr. P.C.J. Hoogenboom
2. Follow the same idea used, write a new code which can evaluate equations of thin plate bending for analysis (orthogonal parameterization required)
3. Obtain analysis results from new code (if this code does works) and compares it with results from classical FDM.

4. Discuss the reason behind the failure of code (if this code does not work).
5. Based on gained experience in above, discuss whether this method can be extended to shell structures.

Literature review

2.1 A short review on plate theories

A short overview on the development of plate theory is given here to show the understanding for plate theories behind the code application.

A certain character of plate and shell structures is that unlike other elastic bodies they can still remain in elastic phase with small strains while undergoing large deformation. The classical theories can only treat such structures where the displacements and their derivatives are considerably small. Therefore, it is necessary to develop a theory that describes the nonlinear behavior of such elastic bodies within small strains.

There are two fundamental methods for solving such a problem in the classical theory of plates. The first method is to describe this problem under the general theory of elasticity. The ideal is that the deformation in the neighbourhood of each point still can be described linearly. Cauchy given the expression of displacements and stresses as a power series of the distance z from the middle surface (1828). Poisson then successfully solve the Germain-Lagrange plate equation under static loading condition (Poisson, 1829). The second method is developed by Kirchhoff in which he introduced physical meaning into the theory of plates by the famous "Kirchhoff's hypotheses" (1850).

Kirchhoff's hypotheses are a series of fundamental assumptions used in thin plate bending theory in which the deflection of plate is assumed to be small, linear and elastic. Restated assumptions (Ventsel and Krauthammer, 2001) are list as below:

- The elastic, homogenous, and isotropic material.
- Initially flat plate.
- Small vertical deflection of the midplane compared with the thickness of the plate.
- "Needle hypotheses": The normal lines of the middle plane remain straight and normal to the middle surface during the deformation. Thickness remains constant. Negligible vertical shear strains (γ_{xy}, γ_{yz}) and normal strain (ε_z).
- Negligible normal stress σ_z
- Middle surface remains unstrained

Under the above assumptions, the governing equation of classic plate bending theory (small deflection only) can be derived as:

$$D\left[\frac{\partial^4 w}{\partial x^4}+2\frac{\partial^4 w}{\partial x^2 \partial y^2}+\frac{\partial^4 w}{\partial y^4}\right]=\rho h\frac{\partial^2 w}{\partial t^2}$$

Equation 5: Governing equation of classic plate bending theory

where D is the flexural rigidity of the plate, w is the deflection of the plate, h is the plate's thickness, and ρ is the density.

In his method, Kirchhoff declared that the Germain-Lagrange equation can be equivalent to the Euler equation under certain conditions and there are only two types of boundary conditions for plate edges: dynamic & kinematic boundary conditions.

2.2 Analytical solutions by Timoshenko's text book

One of the most significant contribution in development of plate theory was made by Timoshenko in earlier 1910s. He figured out the solutions of large deflections in circular plate problems and developed the theory of elastic stability (1913). The text book he wrote with Woinowsky-Krieger (1987) is still fundamental material for any beginner of learning plate theories. In the chapter 5 of this text book, the analytical solutions on deflection, shear force and bending moment regarding to simply supported rectangular plates are given. They are list as following:

Uniformly loaded simply supported rectangular plates (Squared plates, length=width= a)	
Maximum deflection	$w_{\max} = 0.00406 \cdot \frac{qa^4}{D}$
Maximum bending moment	$m_{xx} = m_{yy} = 0.0479 \cdot qa^2$
Maximum shear force	$v_{xx} = v_{yy} = 0.338 \cdot qa$

where D is the flexural rigidity and q is an applied uniformly distributed load. The equation that describes the flexural rigidity of a plate is given as below:

$$D = \frac{Eh^3}{12\left(1-\nu^2\right)}$$

2.3 Analytical solutions of square plate with two-way sine load

A square plate with simply supported edges and under distributed load is described in the following form:

$$p = \hat{p}\sin\left(\frac{\pi x}{a}\right)\cos\left(\frac{\pi x}{a}\right)$$

The particular solutions that satisfies boundary and load conditions of this plate is given in a the plate theory book by the Blaauwendraad (2014). And they are listed as below:

Square plate with two-way sine load	
Maximum deflection	$w_{\max} = \frac{qa^4}{4\pi^2 D}$
Maximum bending moment	$m_{xx} = m_{yy} = \frac{(1+\nu)}{4\pi^2} qa^2$

Given formulas of bending moments are:

$$m_{xx} = \frac{1+\nu}{4\pi^2} qa^2 \sin\left(\frac{\pi x}{a}\right)\cos\left(\frac{\pi y}{a}\right);\ m_{yy} = \frac{1+\nu}{4\pi^2} qa^2 \cos\left(\frac{\pi x}{a}\right)\sin\left(\frac{\pi y}{a}\right);\ m_{xy} = -\frac{1-\nu}{4\pi^2} qa^2 \cos\left(\frac{\pi x}{a}\right)\cos\left(\frac{\pi y}{a}\right)$$

And the shear force can be derived from the moments above as shown in below:

$$v_x = \frac{\partial m_{xx}}{\partial x} + \frac{\partial m_{yx}}{\partial y} = \frac{\pi}{a}\frac{1+\nu}{4\pi^2} qa^2 \cos\left(\frac{\pi x}{a}\right)\left(\cos\left(\frac{\pi y}{a}\right) + \sin\left(\frac{\pi y}{a}\right)\right)$$

$$v_y = \frac{\partial m_{yy}}{\partial x} + \frac{\partial m_{xy}}{\partial y} = \frac{\pi}{a}\frac{1+\nu}{4\pi^2} qa^2 \cos\left(\frac{\pi y}{a}\right)\left(\sin\left(\frac{\pi x}{a}\right) + \cos\left(\frac{\pi x}{a}\right)\right)$$

Maximum shear force $$v_{xx} = v_{yy} = \frac{\sqrt{2}(1+\nu)}{4\pi} qa$$

2.4 Sanders-Koiter equations

Sanders-Koiter equations is designation of equations individually developed by Sanders (Sanders, 1963) and Koiter (1966) for refined nonlinear theory of shells. The development of Sanders-Koiter equations can be sourced back to Reissner-Mindlin plate theory

In Reissner-Mindlin plate theory, the transverse shear strains were introduced under the assumption of constant shear angle through the thickness. However, in this way transverse shear boundary conditions at the top and bottom surfaces dose not be satisfied and shear correction factors are required to reach the equilibrium.

Sanders and Koiter found a better way to solve it by introducing in-plane displacement kinematics. The fourth order terms are used to describe the shear deformation through the thickness. The shear boundary conditions of shear stress at top and bottom surfaces are satisfied without shear correction factors. In the Sanders-Koiter theory, equations of motion include all three displacements and variation of curvature and torsion are described linearly. The accuracy of Sanders-Koiter theory on calculating larger vibration amplitudes has been proved (Amabili, 2003).

2.5 A short review on finite different method

The ideal of finite difference method is studying the continuous process by applying mathematical discretization first. By dividing the process into a finite number of sufficiently small parts, the function of that process is possible to be approximated by linear expressions. The results of derivative over a continuous domain can be approximated as the summation of a weight function multiplied with results of discrete points.

The general procedure of applying different finite difference schemes for the numerical solution of partial differential equation is outlined as below:

- ✧ Convert the continuous process variables into a discrete set of points.
- ✧ Approximate partial derivatives using finite difference approximations.
- ✧ Solve the resulting finite difference equation.

For example, domain of variable x of the continuous function $f(x)$ is interval AB. The interval AB starts at point A (a, 0) and ends at B (b, 0) which is divided into equal division $\Delta x = h$. Assume the $f(x)$ is linear continuous function with expression: $f(x) = a + b \cdot x$

The first order derivative of $f(x)$ is given by below calculation:

$$\left\{\begin{array}{l} f_{i-1} = f(x-h) \\ f_i = f(x) \\ f_{i+1} = f(x+h) \end{array}\right\} \Rightarrow \left\{ b = -\frac{f_{i-1} - f_{i+1}}{2h} \right\} \Rightarrow \frac{\partial f(x)}{\partial x} = b = \frac{f_{i-1} - f_{i+1}}{2h}$$

And by definition of derivative, the first order derivative can also be calculated as below. The first order derivative is the slope at a point of function $f(x)$ calculated based on values of adjacent points

$$f' = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} = \lim_{h \to 0} \frac{f(x) - f(x-h)}{h} = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

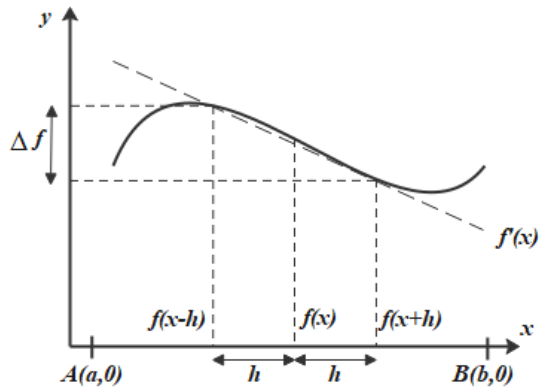


Figure 2: First order derivative by finite difference method

For one-sided finite differences of first order derivative:

$$\left\{\begin{array}{l} f_i = f(x) \\ f_{i+1} = f(x+h) \\ f_{i+2} = f(x+2h) \end{array}\right\} \Rightarrow \left\{ b = -\frac{3 \cdot f_i - 4 \cdot f_{i+1} + f_{i+2}}{2h} \right\}$$

$$\Rightarrow \frac{\partial f(x)}{\partial x} = b = -\frac{3 \cdot f_i - 4 \cdot f_{i+1} + f_{i+2}}{2h}$$

For calculating second order derivative, Taylor series expansion of function $f(x)$ is used and given by below expression:

$$f(x)=\sum_{n=0}^{\infty}\frac{(x-x_i)^n}{n!}\left(\frac{\partial^n f}{\partial x^n}\right)_i$$

$$T_1: f_{i+1}=f_i+h\left(\frac{\partial f}{\partial x}\right)_i+\frac{h^2}{2}\left(\frac{\partial^2 f}{\partial x^2}\right)_i+\frac{h^3}{6}\left(\frac{\partial^3 f}{\partial x^3}\right)_i+\cdots$$

$$T_2: f_{i-1}=f_i-h\left(\frac{\partial f}{\partial x}\right)_i+\frac{h^2}{2}\left(\frac{\partial^2 f}{\partial x^2}\right)_i-\frac{h^3}{6}\left(\frac{\partial^3 f}{\partial x^3}\right)_i+\cdots$$

$$T_1+T_2 \Rightarrow f_{i+1}+f_{i-1}=2f_i+h^2\left(\frac{\partial^2 f}{\partial x^2}\right)_i$$

$$\Rightarrow \frac{\partial^2 f}{\partial x^2}=\frac{f_{i+1}+f_{i-1}-2f_i}{h^2}$$

$$\Rightarrow f''=\frac{f(x+h)-2f(x)+f(x-h)}{h}$$

where f'' is central difference of second order derivative.

2.6 A short overview of application of finite difference method on plate theory

The analytical solution of the most equations in plate theories are usually expressed in terms of infinite trigonometric series since they are partial differential equations. A short analytical solution can only be found in limited simple conditions. In earlier years, the further numerical evaluation of those analytical solutions is performed through the human effort with limited accuracy and a substantial amount of computing time. As described by Rudolph Szilard (1974), since the advancement of electronic computers, researchers start to study the plate bending problems with help of approximate methods.

Traditionally, the application of finite difference method on plate theory starts with the biharmonic equation from the classic plate theory. Solving such fourth-order differential equation with approximation usually results in inaccurate bending moments. To cope with such difficult, Marcus (1932) spilited biharmonic equation into three equations: two second-order deflection equations and one normal moment equation. However, despite it is mathematically correct, this method only gives good results for plates with simple supported edges.

Another alternate approximation was proposed by Reddy and Gera (1979) in which fourth-order differential equation is replaced with three second-order differential equations, as shown in Equation 6. For various boundary condition, those equations can produce bending moment analysis for rectangular plates with conventional finite-difference.

$$M_x=-D\left(\frac{\partial^2 w}{\partial x^2}+\nu\frac{\partial^2 w}{\partial y^2}\right)$$

$$M_y=-D\left(\frac{\partial^2 w}{\partial x^2}+\nu\frac{\partial^2 w}{\partial y^2}\right)$$

$$-P=\frac{\partial^2 M_x}{\partial x^2}-2D(1-\nu)\frac{\partial^2 w}{\partial x^2\partial y^2}+\frac{\partial^2 M_y}{\partial x^2}$$

Equation 6: Three second-order differential equations used in finite difference approximation

$$M_x=-D\left(\frac{\partial\beta_x}{\partial x}+\nu\frac{\partial\beta_y}{\partial y}\right)\quad Q_x=-\frac{\pi^2}{12}Gh\left(\frac{\partial w}{\partial x}-\beta_x\right)$$

$$M_y=-D\left(\frac{\partial\beta_y}{\partial y}+\nu\frac{\partial\beta_x}{\partial x}\right)\quad Q_y=-\frac{\pi^2}{12}Gh\left(\frac{\partial w}{\partial y}-\beta_y\right)$$

$$M_{xy}=-(1-\nu)\frac{D}{2}\left(\frac{\partial\beta_y}{\partial x}+\frac{\partial\beta_x}{\partial y}\right)$$

$$\beta_x=\frac{\partial w}{\partial x}-\gamma_{zx}\quad \beta_y=\frac{\partial w}{\partial y}-\gamma_{zy}$$

Equation 7:Six first-order differential equations used in finite difference approximation (h = plate thickness)

Instead of three second-order equations, Assadi-Lamouki and Krauthammer (1989) formulate an explicit finite difference method to study plate vibration based on six first-order equations from Mindlin plate theory and Kirchhoff plate theory (see in equation 8). Compared to the classical plate theory, the truncation errors of vibration are negligible for short duration short and severe dynamic loads. By comparing the work done by those previous researches, it can be found that the accuracy of finite difference approximation is improved when the order of equations used is reduced.

2.7 Thin plate bending equations

The thin bending equations used here is modified version of Sanders-Koiter equations where the in-plane displacement kinematics including shear deformation through the thickness and torsion are omitted. The 21 first-order Sanders-Koiter equations are reduced to 11 equations for describing the thin plate bending. Below are equations would be applied in the second code including 11 equations of plate theory and 8 equation describing the boundary conditions for an edge.

equilibrium equations	$\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}+p_z=0$	1
	$\frac{\partial m_{xx}}{\partial x}+\frac{\partial m_{xy}}{\partial y}-v_x=0$	2
	$\frac{\partial m_{yy}}{\partial y}+\frac{\partial m_{xy}}{\partial x}-v_y=0$	3
constitutive equations	$m_{xx}=\frac{E t^3}{12(1-\nu^2)}(\kappa_{xx}+\nu\kappa_{yy})$	4
	$m_{yy}=\frac{E t^3}{12(1-\nu^2)}(\kappa_{yy}+\nu\kappa_{xx})$	5
	$m_{xy}=\frac{E t^3}{24(1+\nu)}\rho_{xy}$	6
kinematic equations	$\varphi_x=-\frac{\partial u_z}{\partial x}$	7
	$\varphi_y=-\frac{\partial u_z}{\partial y}$	8
	$\kappa_{xx}=\frac{\partial \varphi_x}{\partial x}$	9
	$\kappa_{yy}=\frac{\partial \varphi_y}{\partial y}$	10
	$\rho_{xy}=\frac{\partial \varphi_x}{\partial y}+\frac{\partial \varphi_y}{\partial x}$	11

Table 1: Equilibrium, constitutive and kinematic equations for plates

Boundary conditions for thin plate is defined as following:

Kinematic (K)	Dynamic (D)	
Impose displacement u_z	or apply line load $q_z=v_y+\frac{\partial V}{\partial x}$.	1
Impose rotation $-\varphi_y$	or apply line moment $-m_{yy}$.	2

Table 2: Boundary conditions for an edge in the x direction and the y axis pointing outwards

Impose displacement u_z	or apply line load $q_z=-v_y-\frac{\partial V}{\partial x}$.	3
Impose rotation $-\varphi_y$	or apply line moment m_{yy}.	4

Table 3: Boundary conditions for an edge in the x direction and the y axis pointing inwards

Impose displacement u_z	or apply line load $q_z=v_x+\frac{\partial V}{\partial y}$.	5

Impose rotation φ_x or apply line moment m_{xx}.	6

Table 4: Boundary conditions for an edge in the y direction and the x axis pointing outwards

Impose displacement u_z or apply line load $q_z = -v_x - \frac{\partial V}{\partial y}$.	7
Impose rotation φ_x or apply line moment $-m_{xx}$.	8

Table 5: Boundary conditions for an edge in the y direction and the x axis pointing inwards

For thin plate loaded perpendicularly, the deformation due to shear and membrane force is negligibly small. Plate theory mainly described its bending deformation under the perpendicular load as constrained by boundary conditions. And for the corner, the support reaction V is calculated as the sum of torsion moments m_{xy} and m_{yx}.

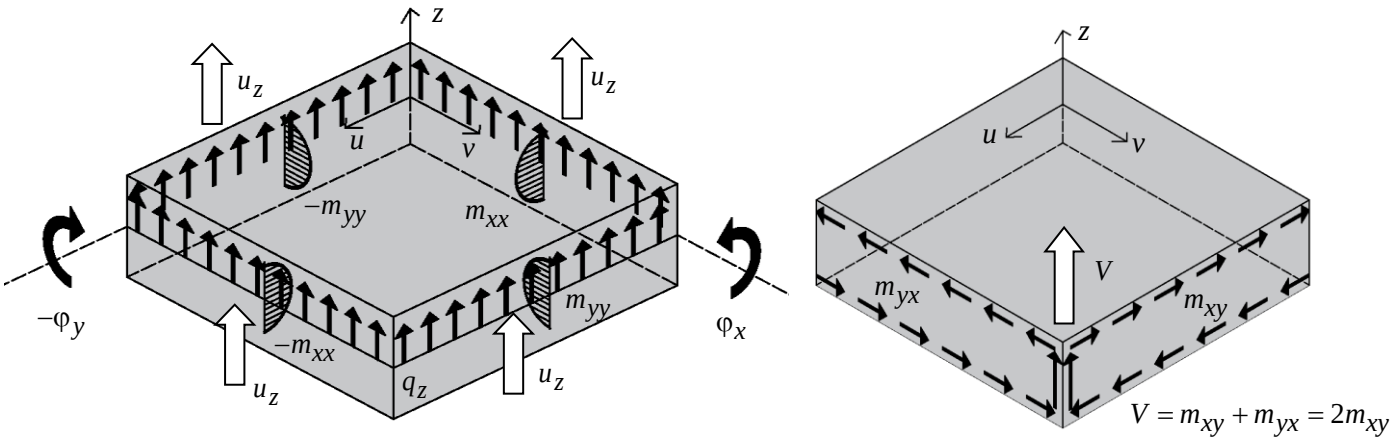


Figure 3: Boundary conditions and corner conditions of square plate

The example Python code for solving Sanders-Koiter equations

In this work, an example Python code was used for solving Sanders-Koiter equations. This program was proposed and developed by Dr. P.C.J. Hoogenboom for validating the idea of solving first-order difference plate equations by applying the finite difference method. The selected plate theory is Sanders-Koiter equations which can be expressed in the forms of 21 first-order difference equations. Although it did not prove validity of this idea, it is important to understand the mechanism and structure of this code.

3.1 General information

The work procedure of main code can be categorized as 8 major steps by their execution order and they are list as following:

Step 1: Inputs of material parameters, geometry and boundary condition type
Step 2: Define load components
Step 3: Define differential equations in x and y direction by finite difference methods
Step 4: Create empty matrixes for stiffness [M], motions [u] and load [f]
Step 5: Add Sanders-Koiter equations to stiffness matrix [M]
Step 6: Add load components to load matrix [f] while define the boundary conditions
Step 7: Solve [u] from [M]*[u] = [f] by least square method
Step 8: Postprocessing and display results of solved [u]

The studied geometry in this program is canopy defined by the geometry parameters $k_{xx}=1$ and $k_{yy}=-\frac{1}{a^2}$ where a is radius of curved surface. The general purpose of this program is to solve the displacement matrix [u] from [M]* [u] = [f] while stiffness matrix [M] and load matrix [f] are defined by Sanders-Koiter equations, boundary and corner conditions.

3.2 Formation of matrixes

First, empty matrixes are created waiting to be filled. Size of matrixes are determined by number of nodes of model. For the matrix [M], the column number is $21\cdot mn$ (m: number of nodes in u-direction, n: number of nodes in v-direction) and row number is $21\cdot mn+8\cdot m+8\cdot n+10$ ($8\cdot m+8\cdot n$: 4 boundary conditions per edge, 10: 5 corner conditions for per corner).

As defined in Sanders-Koiter equations, 21 unknown quantities are assigned to each node. which are u_x, u_y, u_z, ε_{xx}, ε_{yy}, γ_{xy}, φ_x, φ_y, φ_z, κ_{xx}, κ_{yy}, ρ_{xy}, n_{xx}, n_{yy}, n_{xy}, n_{yx}, v_x, v_y, m_{xx}, m_{yy} and m_{xy}. During the tests of code, they are assigned with an integral value from 0 to 21 indicating their proposed location in a solved [u], so that the results can be extracted accordingly.

For example, the Sanders-Koiter equation 1 is added to the matrix [M] by the blow code. For adding one value to matrix, the process can be described as $M[\text{Row number}][K\cdot m\cdot n+j\cdot m+i]$ = Parameter of unkown where K represents the location value.

$$\underset{①}{k_{xx}n_{xx}}+\underset{②}{k_{xy}\left(n_{xy}+n_{yx}\right)}+\underset{③}{k_{yy}n_{yy}}+\frac{\partial q_x}{\partial x}+\frac{\partial q_y}{\partial y}+k_y q_x+k_x q_y+p_z=0$$

Sanders-Koiter equation 1

```
1: M[row][nxx*m*n+j*m+i]=kxx(i/(m-1),j/(n-1))
2: M[row][nxy*m*n+j*m+i]=kxy(i/(m-1),j/(n-1))
3: M[row][nyx*m*n+j*m+i]=kxy(i/(m-1),j/(n-1))
```

Adding process

And the similar procedure is repeated for other quantities in the equation. The function of adding equation to matrixes is divided into two parts (Loop ① & Loop ②). The parameters of unknowns are added into matrixes and their starting point is determined by the assigned location value. Loop ① repeats the adding process for every node at one line along u-direction. Loop ② repeats Loop ① until all lines are fulfilled, so that unknown quantities are added to every node.

```
row=-1
for j in range(n): # Add Sanders-Koiter equation 1 to the matrix ----------------------------
   for i in range(m):
      row=row+1
      M[row][nxx*m*n+j*m+i]=kxx(i/(m-1),j/(n-1))
      M[row][nxy*m*n+j*m+i]=kxy(i/(m-1),j/(n-1))
      M[row][nyx*m*n+j*m+i]=kxy(i/(m-1),j/(n-1))
      M[row][nyy*m*n+j*m+i]=kyy(i/(m-1),j/(n-1))
      Dx(vx,1.0)
      Dy(vy,1.0)
      M[row][vx*m*n+j*m+i]=ky(i/(m-1),j/(n-1))
      M[row][vy*m*n+j*m+i]=kx(i/(m-1),j/(n-1))
      f[row]=-pz(i/(m-1),j/(n-1))
```

Loop _ ① *Loop _ ②*

Code 1: Add Sanders-Koiter equation 1 to the matrix [M]

The column number of starting points for unknown quantity K is $K \cdot m \cdot n$. For each adding process in Loop ①, the column number and row number increased accordingly for m times. Then Loop ② repeats Loop ① for n times. Meanwhile for every Loop ①, the row number is incremented by one. By adding one equation, $m \cdot n$ rows of matrix have been generated.

Below figure shows the pattern of non-zero values in matrix [M] during this adding procedure where the values are diagonally distributed. The same adding procedure is also utilized for adding boundary conditions and corner conditions.

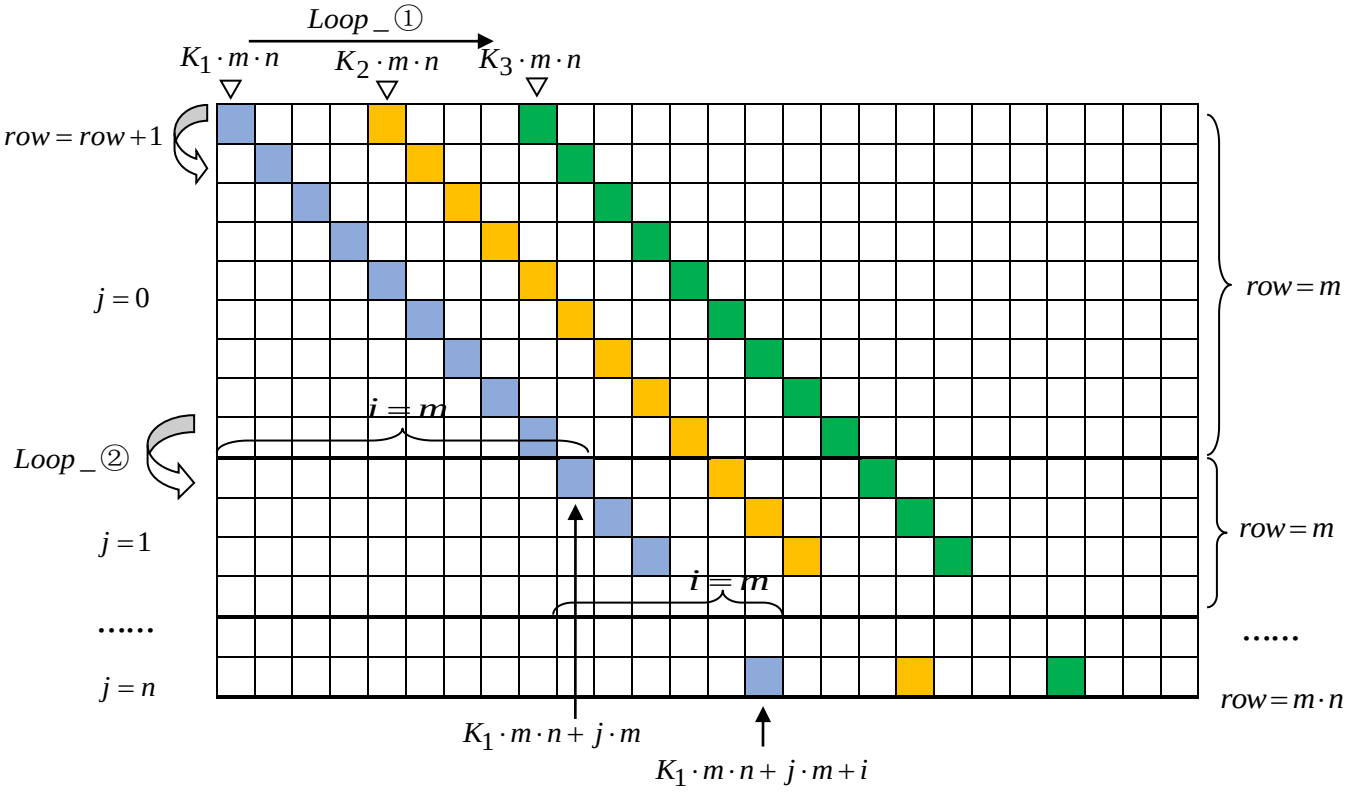


Figure 4: Add equations to [M]

3.3 Differentiation approximated by FDM

$$k_{xx}n_{xx}+k_{xy}\left(n_{xy}+n_{yx}\right)+k_{yy}n_{yy}+\underset{①}{\frac{\partial v_x}{\partial x}}+\underset{②}{\frac{\partial v_y}{\partial y}}+k_y q_x+k_x q_y+p_z=0$$

Sanders-Koiter equation 1

……

```
1: Dx(vx,1.0)
2: Dy(vy,1.0)
```

Differentiation

……

As discussed in section 3.4, first order derivative of $f(x)$ is given as $\frac{\partial f(x)}{\partial x}=\frac{f_{i-1}-f_{i+1}}{2h}$ and the one-sided finite differences of first order derivative is $\frac{\partial f(x)}{\partial x}=-\frac{3\cdot f_i-4\cdot f_{i+1}+f_{i+2}}{2h}$ where h represents the distance between nodes. And h here is replaced with Lamé parameters (α_x, α_y). The differentiation of one quantity ($\frac{\partial}{\partial x},\frac{\partial}{\partial y}$)in the program is approximated by finite difference method ($D_x(k,g)$, $D_x(k,g)$) defined in below where k is the location value of the unknown quantity while g works as positive/negative sign of the value (g = -1/1).

Inside grids: $D_x(k,g)=\frac{g}{2\alpha_x}(f_{i-1}-f_{i+1})$ $D_y(k,g)=\frac{g}{2\alpha_y}(f_{i+1}-f_{i-1})$

At edges $D_x(k,g)=\frac{g}{2\alpha_x}(3\cdot f_i-4\cdot f_{i+1}+f_{i+2})$ $D_y(k,g)=\frac{g}{2\alpha_y}(3\cdot f_i-4\cdot f_{i+1}+f_{i+2})$

$D_x(k,g)=-\frac{g}{2\alpha_x}(3\cdot f_i-4\cdot f_{i+1}+f_{i+2})$ $D_y(k,g)=-\frac{g}{2\alpha_y}(3\cdot f_i-4\cdot f_{i+1}+f_{i+2})$

```
def Dx(k,g):                                                                    Location:
   if i==0:                                                                     (at left edge)
      M[row][k*m*n+j*m+i+2]=-1*g/(2*alphax(i/(m-1),j/(n-1))/(m-1))               k3 = k*m*n+j*m+i+2
      M[row][k*m*n+j*m+i+1]= 4*g/(2*alphax(i/(m-1),j/(n-1))/(m-1))               k2 = k*m*n+j*m+i+1
      M[row][k*m*n+j*m+i  ]=-3*g/(2*alphax(i/(m-1),j/(n-1))/(m-1))               k1 = k*m*n+j*m+i
   elif i==m-1:                                                                 (at right edge)
      M[row][k*m*n+j*m+i  ]= 3*g/(2*alphax(i/(m-1),j/(n-1))/(m-1))               k3 = k*m*n+j*m+i
      M[row][k*m*n+j*m+i-1]=-4*g/(2*alphax(i/(m-1),j/(n-1))/(m-1))               k2 = k*m*n+j*m+i-1
      M[row][k*m*n+j*m+i-2]= 1*g/(2*alphax(i/(m-1),j/(n-1))/(m-1))               k1 = k*m*n+j*m+i-2
   else:                                                                        (inside grids)
      M[row][k*m*n+j*m+i+1]= 1*g/(2*alphax(i/(m-1),j/(n-1))/(m-1))               k2 = k*m*n+j*m+i+1
      M[row][k*m*n+j*m+i-1]=-1*g/(2*alphax(i/(m-1),j/(n-1))/(m-1))               k1 = k*m*n+j*m+i-1
   Return
```

Code 2: Finite difference approximation in x direction

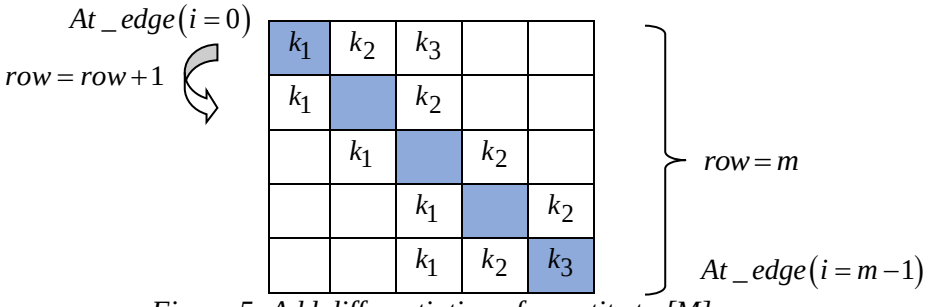


Figure 5: Add differentiation of quantity to [M]

3.4 Definition of boundary condition and corner condition

This model has three free edges and one fixed edge. For each edge, four boundary conditions are defined They can be type D or type K (D = dynamic boundary condition, K= kinematic boundary condition). For example, the edge 1 is an edge in the x direction and y axis point outwards. For four boundary conditions from BC1 to BC4, type D and type K are both defined.

Type K	Type D	
Impose displacement u_x	or line load $q_x = n_{yx} - k_{xx}V$	BC1
Impose displacement u_y	or line load $q_y = n_{yy} - k_{xy}V$	BC2
Impose displacement u_z	or line load $q_z = v_y + \frac{\partial V}{\partial x}$	BC3
Impose displacement $-\varphi_y$	or line load $-m_{yy}$	BC4

Table 6: Boundary conditions for an edge in the x direction and the y axis pointing inwards

For example, the boundary condition BC1 is defined as below:

```
BC1='D';  BC1Value=0
BC2='D';  BC2Value=0
BC3='D';  BC3Value=0
BC4='D';  BC4Value=0
……
BC13='K'; BC13Value=0
BC14='K'; BC14Value=0
BC15='K'; BC15Value=0
BC16='K'; BC16Value=0
```

```
if BC1=='K':
   for i in range(1,m):
      row=row+1
      M[row][ux*m*n+j*m+i]=1
      f[row]=BC1Value
else:
   for i in range(1,m):
      row=row+1
      M[row][nyx*m*n+j*m+i]=1
      M[row][mxy*m*n+j*m+i]=-kxx(i/(m-1),j/(n-1))
      f[row]=BC1Value
```

Code 3: Definition of boundary condition for BC1

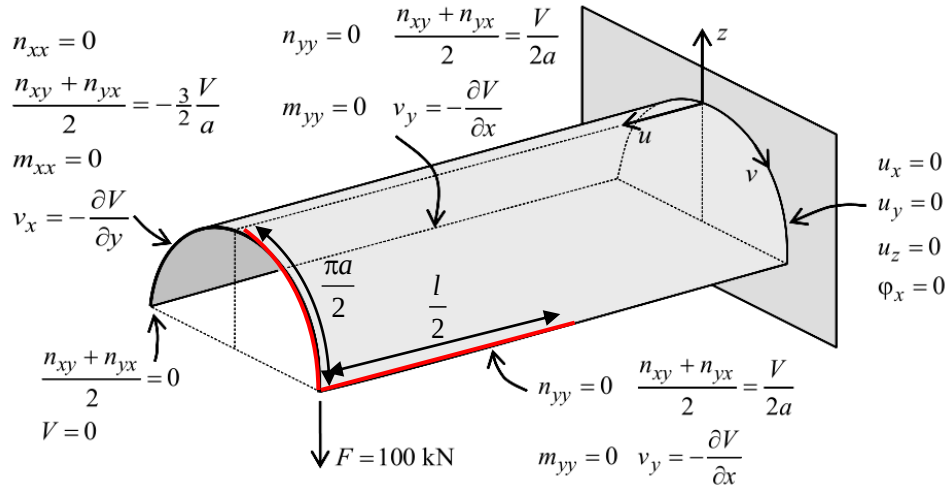


Figure 6: Shell boundary conditions and corner conditions of the canopy

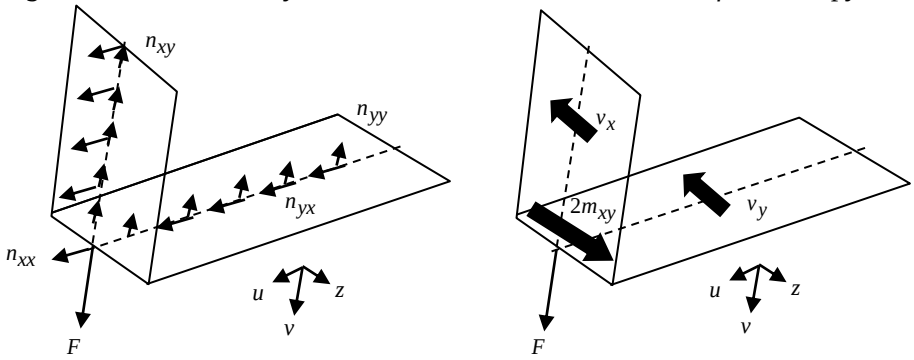


Figure 7: Corner condition of loaded corner of the canopy

The condition of corners on the free curved edge is defined in the following. For each corner, 5 conditions are defined. Half of the free curved edge and half of length edge are involved for the equilibrium at the loaded corner.

Free corner $n_{yy} - n_{xy} = 0\,,\ n_{xx} - n_{yy} = 0\,,\ m_{xy} = 0\,,\ v_x = 0\,,\ v_y = 0$

Loaded corner $\frac{l}{2} n_{yy} + \frac{\pi a}{2} n_{xy} = F\,,\ \frac{l}{2} n_{yx} + \frac{\pi a}{2} n_{xx} = 0\,,\ 2m_{xy} - \frac{l}{2} v_y - \frac{\pi \cdot a}{2} v_x = 0\,,\ m_{yy} = 0\,,\ v_x = 0$

In the code, they are defined as below:

```
i=m-1 # 5 corner conditions
j=0
row=row+1; M[row][nyy*m*n+j*m+i]=1; M[row][nxy*m*n+j*m+i]=-1; f[row]=0
row=row+1; M[row][nxx*m*n+j*m+i]=1; M[row][nyx*m*n+j*m+i]=-1; f[row]=0
row=row+1; M[row][mxy*m*n+j*m+i]=1; f[row]=0
row=row+1; M[row][vx*m*n+j*m+i]=1; f[row]=0
row=row+1; M[row][vy*m*n+j*m+i]=1; f[row]=0

i=m-1
j=n-1
row=row+1; M[row][nyy*m*n+j*m+i]=1/2/(m-1); M[row][nxy*m*n+j*m+i]=3.1415*a/2/(n-1); f[row]=F
row=row+1; M[row][nxx*m*n+j*m+i]=3.1415*a/2/(n-1); M[row][nyx*m*n+j*m+i]=1/2/(m-1); f[row]=0
row=row+1; M[row][mxy*m*n+j*m+i]=2; M[row][vy*m*n+j*m+i]=-1/2/(m-1); M[row][vx*m*n+j*m+i]=-
3.1415*a/2/(n-1); f[row]=0
row=row+1; M[row][myy*m*n+j*m+i]=1; f[row]=0
row=row+1; M[row][vx*m*n+j*m+i]=1;  f[row]=0
```

Code 4: Two corner conditions for free corner and loaded corner

The Python code for solving thin plate bending

In this chapter a program is developed based on the example shown in above section where the major difference is the applied plate theory. Instead of 21 Sanders-Koiter equations, 11 equations for describing thin plate bending are used which are discussed in section 2.7. A simpler plate theory used means smaller size of matrixes and less computational time. It helps to debugger and prove the validity of method with higher effectiveness.

4.1 General information

The major steps of work procedure keep same with that of example where the studied geometry is replaced with flat square plate. The general purpose of this program is to solve the displacement matrix [u] from [M]* [u] = [f] while stiffness matrix [M] and load matrix [f] are defined by 11 equations and boundary conditions.

The material parameters used in this program are defined as following:

Young's modulus of steel:	$E = 21 \cdot 10^7$ kN/m^2
Poisson's ratio of steel:	$nu = 0.3$
Length of plate:	$L = 1$ m
Thickness of plate:	$t = L / 25 = 0.04$ m
Number of nodes in x direction:	m
Number of nodes in y direction:	n

For above configuration, two types of boundary condition are defined. The first one is simply supported conditions for all edges (two-way slab). The second one is two simply supported edges with two free edges (one-way slab). They are set for testing the accuracy of numerical results of the models by compared to the corresponding analytical solution results.

The two load cases used in this program are defined as following

Distributed load: $q = 10\ \text{kN/m}^2$

Two-way sine load $q_s = q \cdot \sin\left(\frac{\pi x}{L}\right) \cdot \cos\left(\frac{\pi y}{L}\right)$

There are three models defined in code whose load and boundary conditions are listed as following:

Model	Load case	Boundary condition
Model 1	Distributed load	two-way slab
Model 2	Two-way sine load	two-way slab
Model 3	Distributed load	one-way slab

Table 7: Load & boundary conditions of models

After each running, 11 unknown quantities are plot, and the maximum value of each plot is found. For the propose of mesh refinement study, the execution of code is repeated for each model with different node number m and n which is set as 10, 20, 30.

Linear least squares method is used to solve [u] from the overdetermined system where matrix [M] is the "Coefficient" matrix and force vector [f] is the "dependent variable". It returns the least-squares solution [u] to a linear matrix equation [M]*[u]=[f].

4.2 Formation of matrixes

First, empty matrixes are created waiting to be filled. For the matrix [M], the column number is $11 \cdot mn$ (m: number of nodes in u-direction, n: number of nodes in v-direction) and row number is $11 \cdot mn + 4 \cdot m + 4 \cdot n$ ($4 \cdot m + 4 \cdot n$: 2 boundary conditions per edge).

As defined in equations, 11 unknown quantities are assigned to each node. which are u_z, φ_x, φ_y , κ_{xx}, κ_{yy}, ρ_{xy}, v_x, v_y, m_{xx}, m_{yy} and m_{xy}. They are assigned with an integral value from 0 to 11 indicating their proposed location in a solved [u], so that the results can be extracted accordingly.

For example, the equation 1 is added to the matrix [M] by the blow code. And the similar procedure is repeated for other quantities in the equation.

$$\underset{①}{\frac{\partial v_x}{\partial x}} + \underset{②}{\frac{\partial v_y}{\partial y}} + \underset{③}{p_z} = 0$$

Thin plate equation 1

```
row=-1
for j in range(n): # TODO Add plate equation 1 to the matrix
      row=row+1
      D1x(vx,1/Delta_x(0,0))  ①
      D1y(vy,1/Delta_y(0,0))  ②
      f[row]=-pz(i/m,j/n)  ③
```

Adding process

The same adding procedure is also utilized for adding boundary conditions. Below figure shows the pattern of non-zero values in matrix [M] after formation of matrix [M] is finished. As predicted, those values are diagonally distributed.

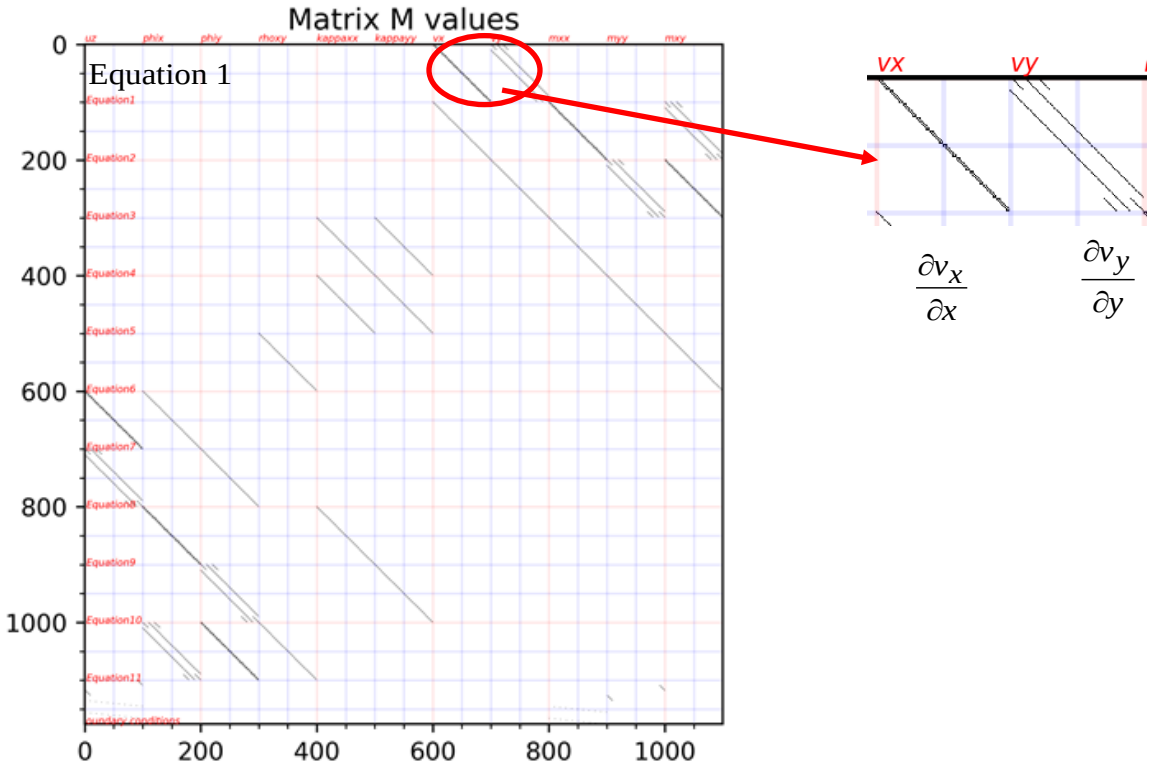


Figure 8: Distribution pattern of non-zero values in finished matrix [M]

4.3 Differentiation approximated by FDM

The same finite difference approximation is used in this program

Inside grids:

$$D_x(k,g)=\frac{g}{2\alpha_x}\left(f_{i-1}-f_{i+1}\right) \qquad D_y(k,g)=\frac{g}{2\alpha_y}\left(f_{i+1}-f_{i-1}\right)$$

At edges

$$D_x(k,g)=\frac{g}{2\alpha_x}\left(3\cdot f_i-4\cdot f_{i+1}+f_{i+2}\right) \qquad D_y(k,g)=\frac{g}{2\alpha_y}\left(3\cdot f_i-4\cdot f_{i+1}+f_{i+2}\right)$$

$$D_x(k,g)=-\frac{g}{2\alpha_x}\left(3\cdot f_i-4\cdot f_{i+1}+f_{i+2}\right) \qquad D_y(k,g)=-\frac{g}{2\alpha_y}\left(3\cdot f_i-4\cdot f_{i+1}+f_{i+2}\right)$$

Result

5.1 Analytical solutions

Based on the analytical solutions given in literature review section, the analytical solution of the maximum value of three major quantities (displacement, bending moment and shear force) used to describing plate behaviours are calculated as shown in following:

Flexural rigidity $D = \frac{E \cdot t^3}{12\left(1 - nu^2\right)} = 1230.8 \text{ kN/m}^2$

Model 1

Load & boundary conditions: Uniformly load, two-way slab

Maximum deflection:	$w_{\max} = 0.00406 \cdot \frac{qL^4}{D} = 3.3 \cdot 10^{-5}$ m	$3.299 \cdot 10^{-5}$ m
Maximum bending moment:	$m_{xx} = m_{yy} = 0.0479 \cdot qL^2 = 0.479$ kNm/m	0.479 kNm/m
Maximum shear force:	$v_{xx} = v_{yy} = 0.338 \cdot qL = 3.38$ kN/m	3.38 kN/m

Model 2

Load & boundary conditions: Two-way sine load, two-way slab

Maximum deflection:	$w_{\max} = \frac{qL^4}{4\pi D} = 2.1 \cdot 10^{-5}$ m	$2.085 \cdot 10^{-5}$ m
Maximum bending moment:	$m_{xx} = m_{yy} = \frac{(1+\nu)}{4\pi^2} qL^2 = 0.329$ kNm/m	0.329 kNm/m
Maximum shear force:	$v_{xx} = \frac{\sqrt{2}(1+\nu)}{4\pi} qL = 1.463$ kN/m	1.463 kN/m

Model 3

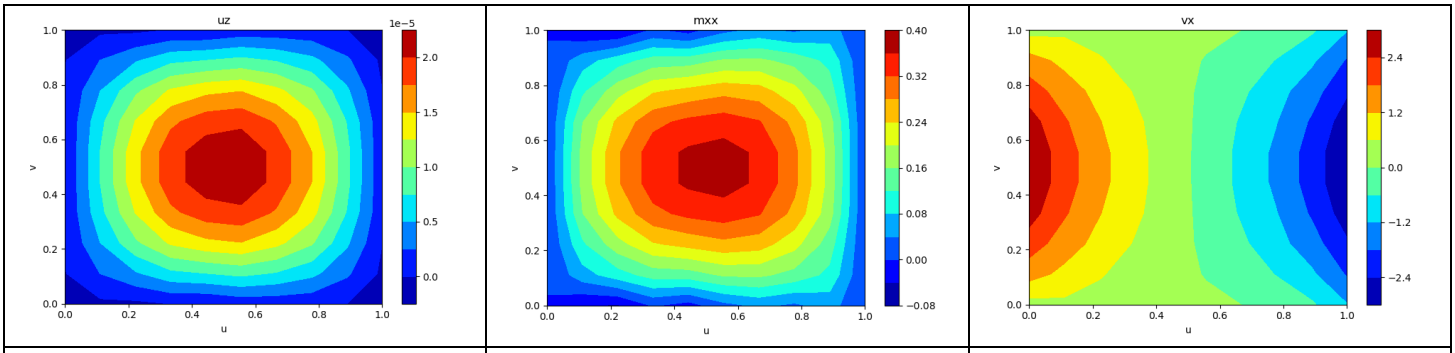
Load & boundary conditions: Uniformly load, one-way slab

Maximum deflection:	$w_{\max} = \frac{5}{384} \frac{qL^5}{EI} = 1.163 \cdot 10^{-4}$ m	$1.163 \cdot 10^{-4}$ m
Maximum bending moment:	$m_{yy} = \frac{1}{8} \cdot qL^2 = 2.5$ kNm/m	1.25 kNm/m
Maximum shear force:	$v_{yy} = \frac{1}{2} \cdot qL = 5$ kN/m	5 kN/m

5.2 Code solutions

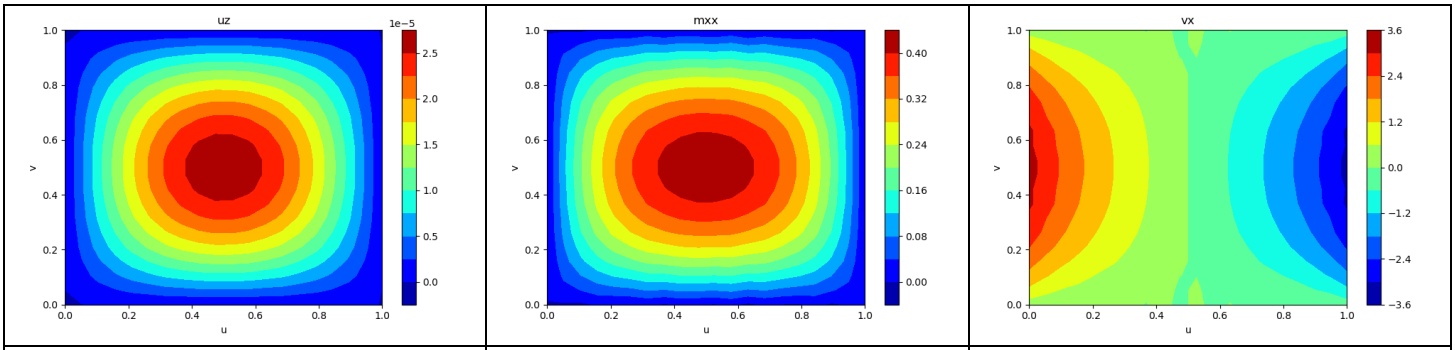
The generated plots and maximum value results are collected from the execution results of the python code. For each model, the plots of displacement u_z, bending moment m_{xx} and shear force v_x are shown with their maximum value.

Plots of displacement u_z, bending moment m_{xx} and shear force v_x with their maximum value



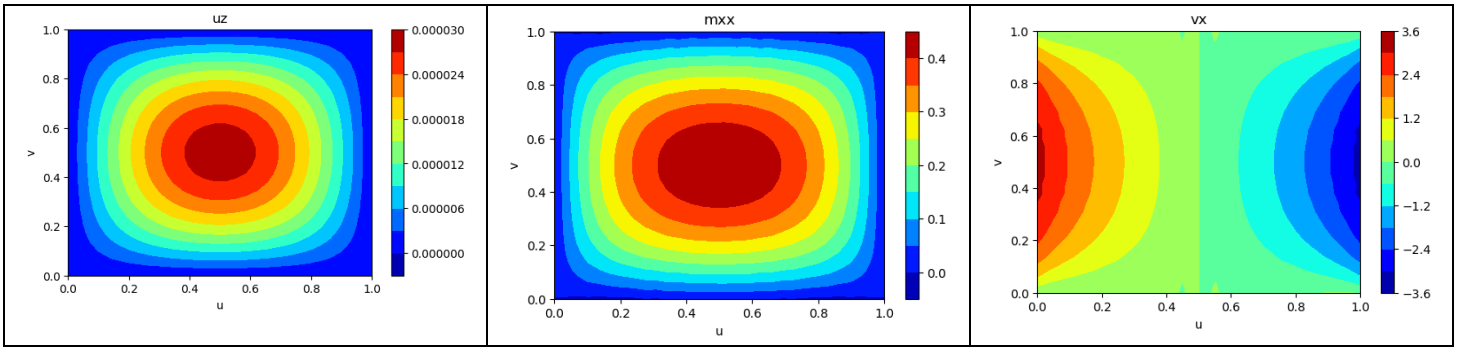
a) Displacement (max = 0.000022)	b) Bending moment (max = 0.375784)	c) Shear force (max = 2.958091)

*Figure 9: Result plot of model 1 with the maximum values (10*10)*



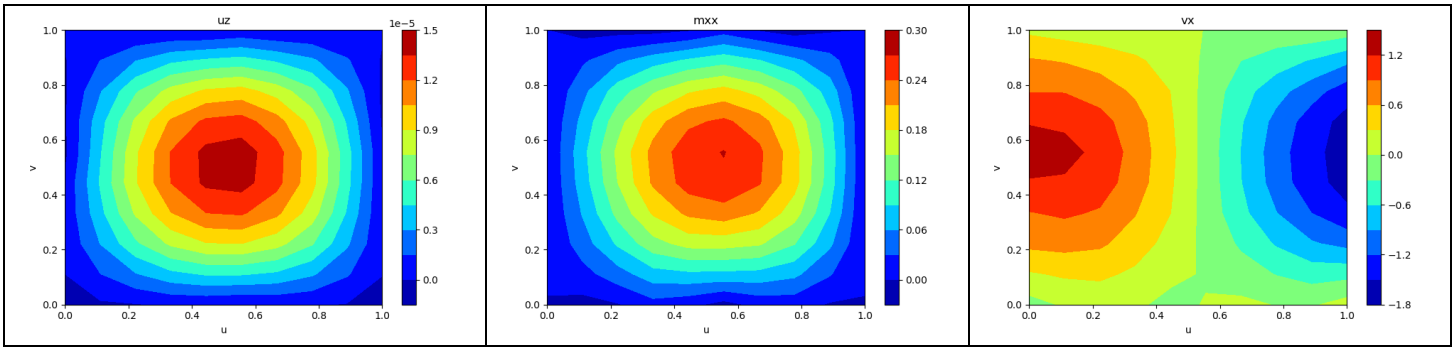
a) Displacement (max = 0.000027)	c) Bending moment (max = 0.427163)	c) Shear force (max = 3.216659)

*Figure 10: Result plot of model 1 with the maximum values (20*20)*



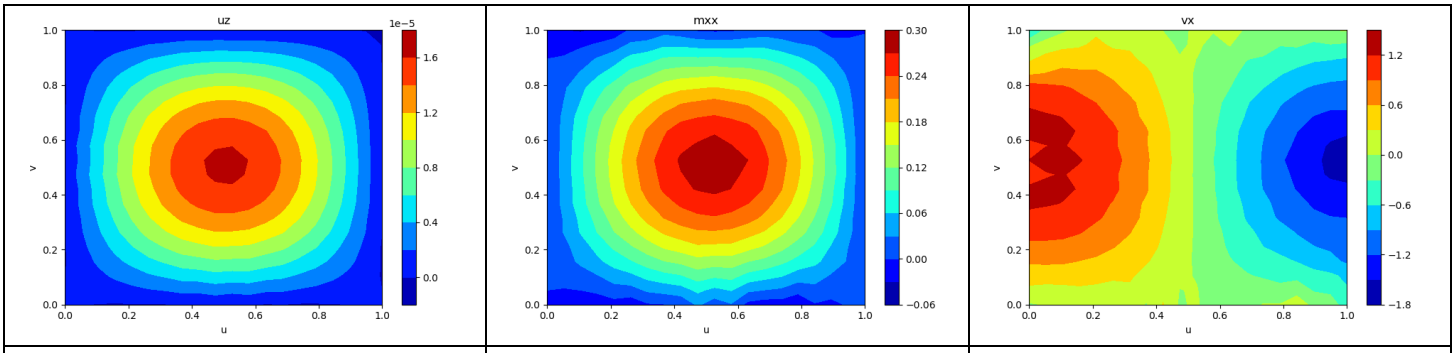
a) Displacement (max = 0.000029)	d) Bending moment (max = 0.444856)	c) Shear force (max = 3.212631)

*Figure 11: Result plot of model 1 with the maximum values (30*30)*



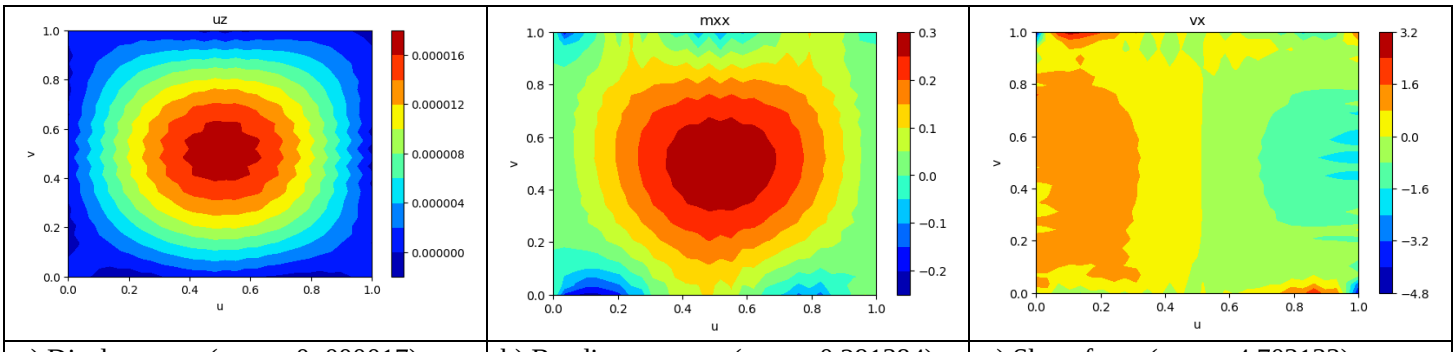
a) Displacement (max = 0.000014)	b) Bending moment (max = 0.271283)	c) Shear force (max = 1.648043)

*Figure 12: Result plot of model 2 with the maximum values (10*10)*



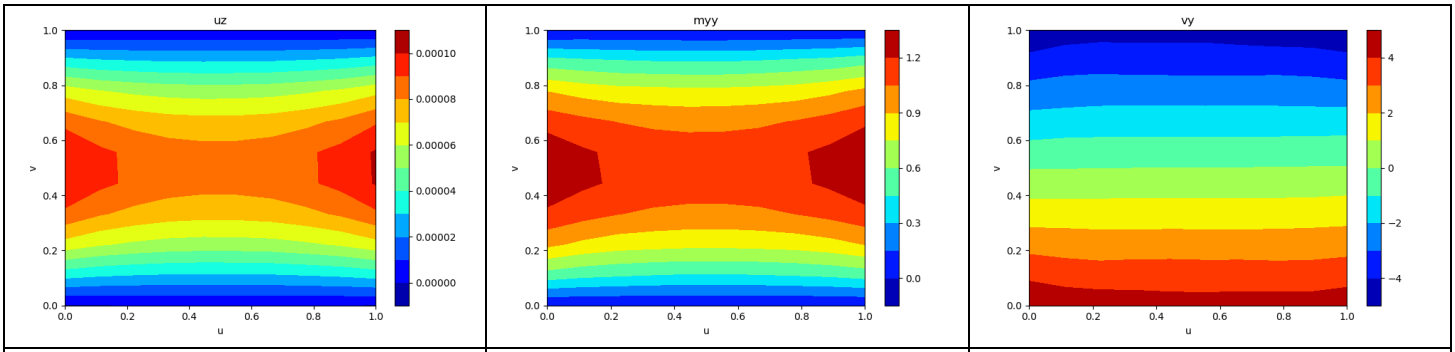
a) Displacement (max = 0. 000016)	b) Bending moment (max = 0.288521)	c) Shear force (max = 1.704489)

*Figure 13: Result plot of model 2 with the maximum values (20*20)*



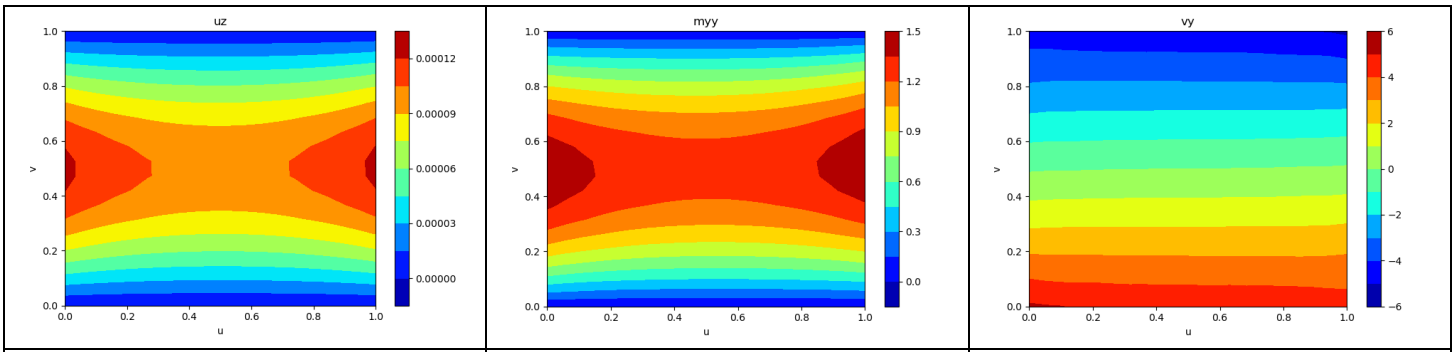
a) Displacement (max = 0. 000017)	b) Bending moment (max = 0.291384)	c) Shear force (max = 4.703133)

*Figure 14: Result plot of model 2 with the maximum values (30*30)*



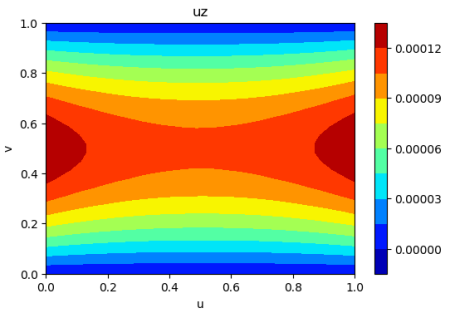
a) Displacement (max = 0.000101)	b) Bending moment (max = 1.312363)	c) Shear force (max = 4.990231)

*Figure 3: Result plot of model 3 with the maximum values (10*10)*

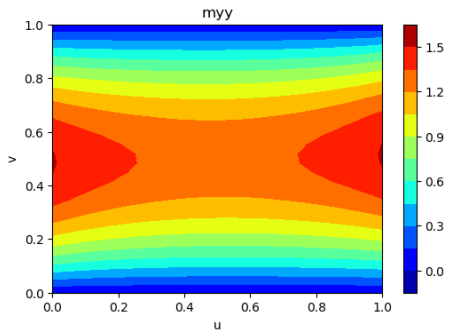


a) Displacement (max = 0.000123)	b) Bending moment (max = 1.458606)	c) Shear force (max = 5.217669)

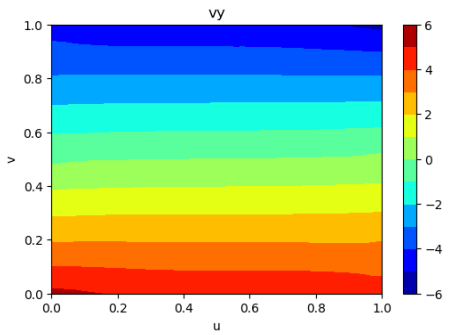
*Figure 15: Result plot of model 3 with the maximum values (20*20)*



a) Displacement (max = 0. 000132)



b) Bending moment (max = 1.510968)



c) Shear force (max = 5.313728)

*Figure 16: Result plot of model 3 with the maximum values (30*30)*

Based on the observation of those contour plots, except for model 2, contour lines become more smoothly with increased node number. It indicates the better interpolation of values between nodes by introducing smaller element size. However, for model 2, while under two-way sine load, the plots become heavily distorted with increased node number. Especially for the shear force plot with 30*30 nodes, the maximum value and contour lines dramatically deviate from previous results. And for the rest of model 2 plots, jagged patterns appear more frequently as node numbers increased.

Meanwhile, the maximum values per model are summarized as below:

Node number	Maximum values	**Model 1**	**Model 2**	**Model 3**
10*10	Displacement	2.200E-05	1.400E-05	1.010E-04
	Bending moment	3.758E-01	2.713E-01	1.312E+00
	Shear force	2.958E+00	1.648E+00	4.990E+00
20*20	Displacement	2.700E-05	1.600E-05	1.230E-04
	Bending moment	4.272E-01	2.885E-01	1.459E+00
	Shear force	3.217E+00	1.704E+00	5.218E+00
30*30	Displacement	2.900E-05	1.700E-05	1.320E-04
	Bending moment	4.449E-01	2.914E-01	1.511E+00
	Shear force	3.213E+00	4.703E+00	5.314E+00
Analytical solution	Displacement	3.299E-05	2.085E-05	1.163E-05
	Bending moment	4.79E-01	0.329E-01	1.25E+00
	Shear force	3.38E+00	1.463E+00	5E+00

Table 8: Maximum values python code results

By comparing the code result with the analytical solutions, the rate of difference in percentage is calculated for each quantity as shown in below figures:

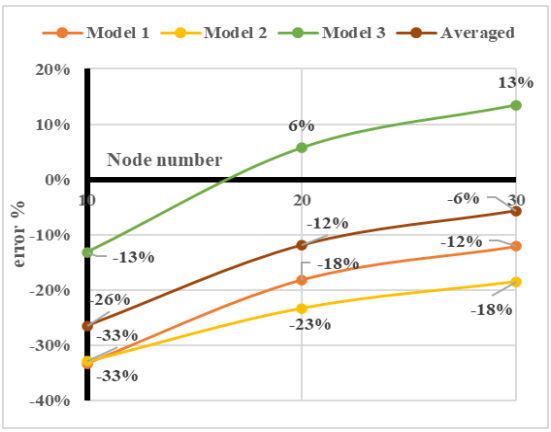


Figure 17: Difference (%) between analytical solution and code results (displacement)

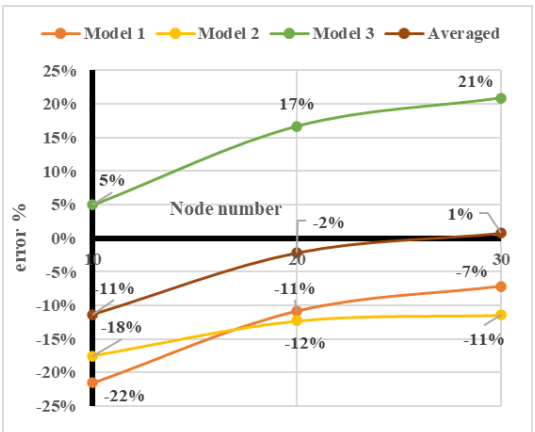


Figure 18: Difference (%) between analytical solution and code results (bending moment)

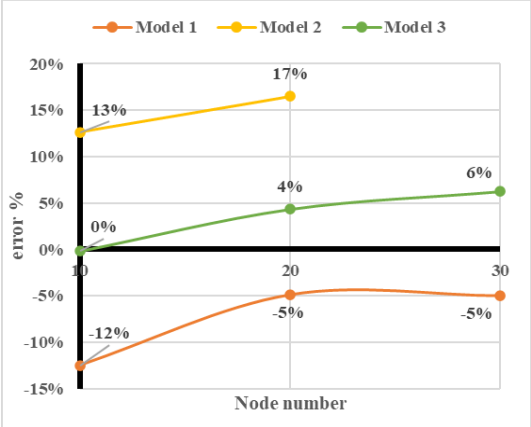


Figure 19: Difference (%) between analytical solution and code results (shear force)

As shown in the above figures, most results are lower compared to the analytical solution. With increased number of nodes, the difference in percentage is generally reduced but the extent of reduction deviates between models and quantities. For displacement and bending moment results, the difference drops faster where the difference is reduced by averaging 20% and 10% respectively. For shear force results, the dropping of difference is less than 10% regardless of models. When the node number is 10*10, the shear force results have an averaged smaller difference rate. Contrary to other models, the all results of model 3 and shear force results of model 2 are generally higher than its analytical solution. As the node number increased, the difference is even increased. The maximum value of shear force result of model 2 with 30*30 node is four times its analytical solution so that it is not marked in figure 20.

Although some result shows good approximation, for example the shear force of model 3, the most results are far from being accurate compared to the analytical solutions.

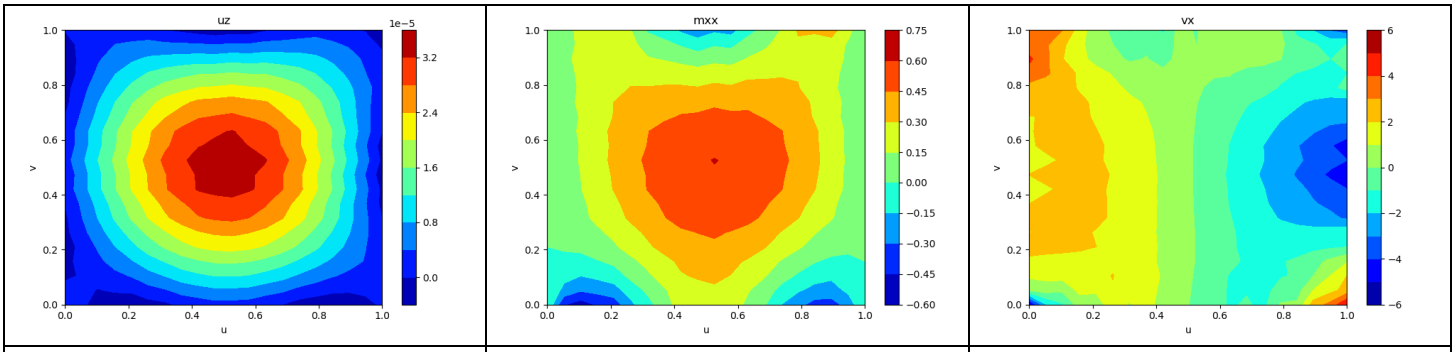
Discussion

From figure 13 to figure 15, the distortion of contour lines becomes more severe as the node number increased. It is can be explained as an expression of the disadvantage of the linear least squared method which is used in this program to solve matrixes. For such a sparse matrix, it can be difficult to find a precisely linear solution that complies with data points. And the solution given by least square method is significantly sensitive to existence of data points with extreme values.

In the practice, the only non-zero values in the load vector [F] are in equilibrium with shear force distribution (thin plate bending equation 1) which are the extreme values in the vector that dominate the process of finding the optimal estimation when using the least squares method to solve the matrix. This is why the error in shear force results are significantly smaller compared to displacement and bending moment results.

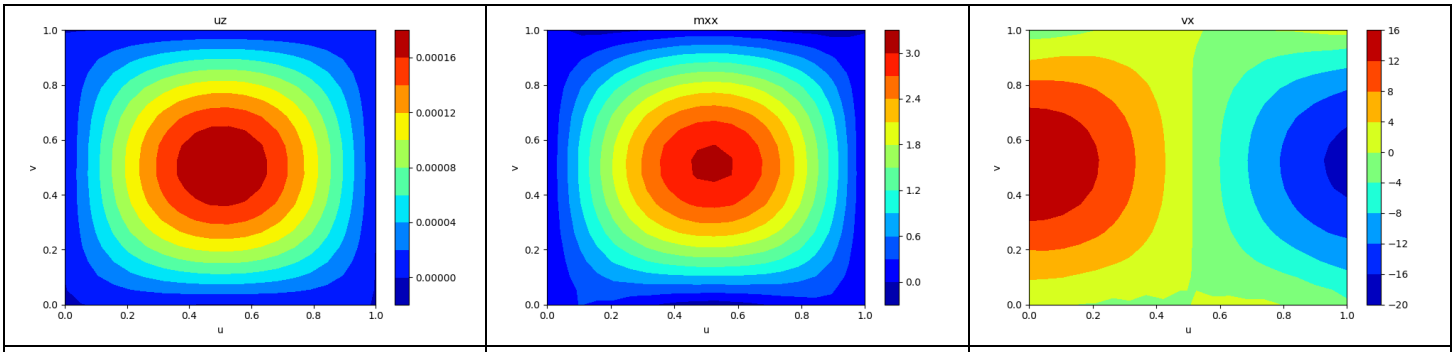
Another possible source of error is the noise of the data point. Since there is no weight function ever considered, extrapolating the results might be misleading as the linear model takes error from each data equally. When directly adopting this data for the optimal fitting, the noise of data in different position could have amplification or reduction effect, which leads to fitting errors.

The distortion of contour lines in model 2 results might be caused by data noise. To verify such assumption, the load value is altered from 10kN/m to 100kN/m and 1000kN/m for model 2 with node number of 20*20. By using large value of load, the data error will become less significant.



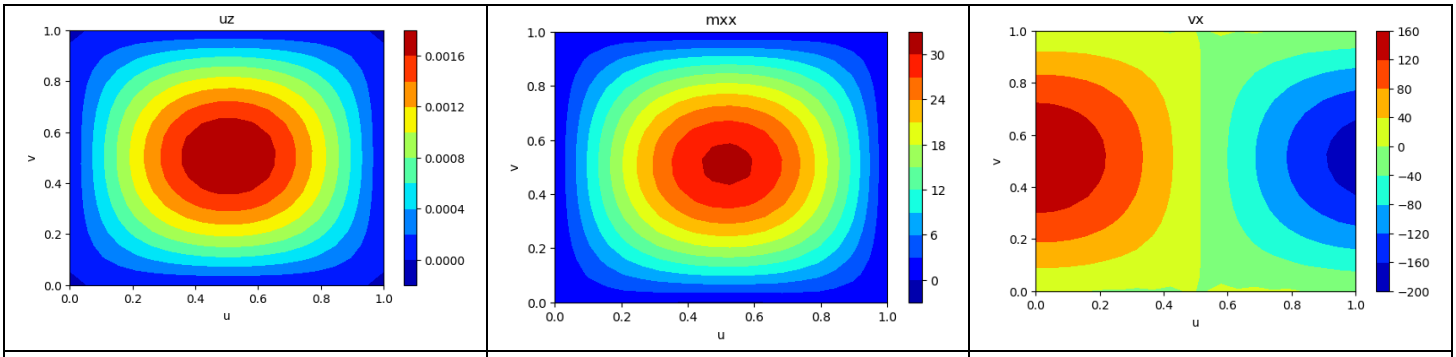
a) Displacement (max = 0.000035)	b) Bending moment (max = 0.604248)	c) Shear force (max =5.345979)

*Figure 20: Result plot of model 2 with the maximum values (higher load q=20kN/m, 20*20)*



a) Displacement (max = 0.000178)	b) Bending moment (max = 3.064449)	c) Shear force (max =17.255760)

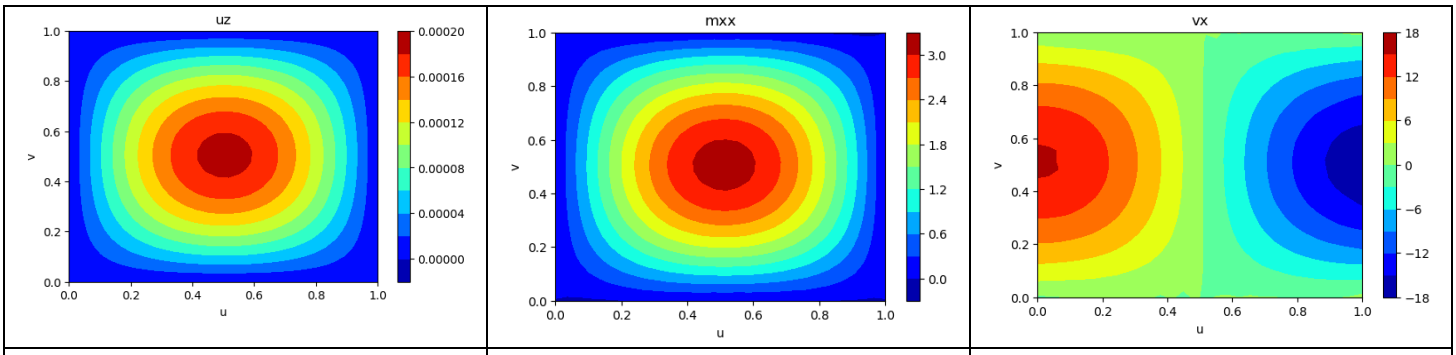
*Figure 21: Result plot of model 2 with the maximum values (higher load q=100kN/m, 20*20)*



a) Displacement (max = 0.001790)	b) Bending moment (max = 30.874547)	c) Shear force (max =180.396483)

*Figure 22: Result plot of model 2 with the maximum values (higher load q=1000kN/m, 20*20)*

As shown in above figures, the extent of distortion of contour lines has been decreased as the using the higher load. To comparing to the figure 15 (q=10kN/m, 30*30). the load is altered to 100 kN/m with same load number is used. The plot is shown in following. The severe distortion in figure 15 is dramatically reduced as a higher load value is used.



a) Displacement (max = 0.000187)	b) Bending moment (max = 3.133083)	c) Shear force (max =17.122023)

*Figure 23: Result plot of model 2 with the maximum values (higher load q=100kN/m, 30*30)*

By observing figures 18, 19, and 20, there is a clear trend of converging of results shown as the node number increased. However, all results of model 3 and shear force results of model 2 intend to converge to some values higher than their analytical solutions. It indicates that this program could underestimate the distribution of shear force on the plate. Based on model 3 results, the stiffness of the plate in terms of bending and deformation is also underestimated. To compensate for such underestimation, a higher node number is recommended to use. However, the node number that gives results with a satisfying accuracy level will require enormous computational time, which basically could make this program unpractical to use.

Conclusion

The method of solving plate equations by only applying first-order finite-difference approximation has been proved to be workable by the python code newly proposed by this report. Several models with various loadings and edge conditions are analyzed and have produced manfully results. The plots of displacement, shear force, and bending moments can be viewed as only approximations compared to the analytical as the produced results are usually 10 to 20% lower. Meanwhile, the matrix will be enormous if the denser grid is applied to minimize computational error. In that case, the consumed computational time will basically make this program unpractical to use. A possible solution to this is using Scipy's sparse matrices where the only non-zero element is stored to save memory usage. For a small program whose effective lines of code is merely around 600, while being capable to produce solve 11 plate equations per node, its simplicity gives its advantage in terms of educational purpose. This program has been proved as an easy-to-verify computer code that can solve model problems. As a feasible and simple method to solve plate problems, further research could focus on its application with more sophisticated plate theory like Sanders-Koiter equations to prove its capability.

Reference list

Amabili, M. (2003) ‘A comparison of shell theories for large-amplitude vibrations of circular cylindrical shells: Lagrangian approach’, *Journal of Sound and Vibration TA - TT -*, 264(5), pp. 1091–1125. doi: 10.1016/S0022-460X(02)01385-8 LK - https://tudelft.on.worldcat.org/oclc/4924705977.

Assadi-Lamouki, A. and Krauthammer, T. (1989) ‘An explicit finite difference approach for the Mindlin plate analysis’, *Computers & structures*. Elsevier, 31(4), pp. 487–494.

Blaauwendraad, J. (2014) *Plates and FEM, Surprises and pitfall*. Volume 171, *Springer Science*. Volume 171. Edited by G.M.L. GLADWELL. Dordrecht Heidelberg: Springer Science. doi: 10.1007/978-90-481-3596-7.

Cauchy, A. L. (1828) *Exercices de Mathematiques*. Kessinger Publishing. Available at: https://books.google.nl/books?id=5lbgRwAACAAJ.

Doshi, A. D. (1964) *ACCURACY OF SEVERAL FINITE DIFFERENCE METHODS FOR PLATE PROBLEMS*. RICE UNIVERSITY.

Kirchhoff, G. (1850) *Über das Gleichgewicht und die Bewegung einer elastischen Scheibe*.

Koiter, W. T. and Delft University of Technology, D. of M. E. L. of E. M. (1966) *Purpose and achievements of research in elastic stability., Report Department of Mechanical Engineering, Delft University of Technology ; 363 TA - TT -*. Delft SE - 29 blz. ; .. cm.: Delft University of Technology. Available at: https://tudelft.on.worldcat.org/oclc/841152853.

Marcus, H. (1932) *Die Theorie elastischer Gewebe und ihre Anwendung auf die Berechnung biegsamer Platten*. Springer.

Reddy, J. N. and Gera, R. (1979) ‘An improved finite-difference analysis of bending of thin rectangular elastic plates’, *Computers and Structures*, 10(3), pp. 431–438. doi: 10.1016/0045-7949(79)90018-X.

Sanders, J. L. (1963) ‘Nonlinear theories for thin shells’, *Quarterly of Applied Mathematics*, 21(1), pp. 21–36. doi: 10.1090/qam/147023.

Szilard, R. T. A.-T. T.- (1974) ‘Theories and Applications of Plate Analysis : Classical Numerical and Engineering Methods’. Hoboken: Wiley [Imprint]. Available at: http://onlinelibrary.wiley.com/book/10.1002/9780470172872.

Timoshenko, S. (1913) *Sur la stabilité des systèmes élastiques*. A. Dumas.

Timoshenko, S. and Woinowsky-Krieger, S. (1987) *Theory of plates and shells*. 2nd ed., *Engineering societies monographs TA - TT -*. 2nd ed. New York SE - 580 blz. ; .. cm.: McGraw-Hill. Available at: https://tudelft.on.worldcat.org/oclc/840267388.

Ventsel, E. and Krauthammer, T. T. A.-T. T.- (2001) ‘Thin plates and shells : theory, analysis, and applications’. New York: Marcel Dekker. doi: 10.1201/9780203908723 LK - https://tudelft.on.worldcat.org/oclc/54351771.