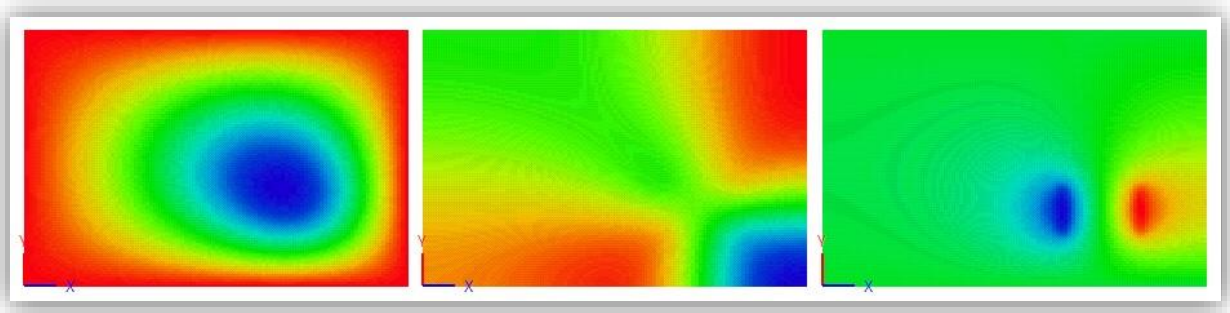
BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS

FACULTY OF CIVIL ENGINEERING

COMPUTER PROGRAM FOR THE CALCULATION OF MINDLIN PLATES

Scientific Student's Association Thesis



BENCE BALOGH

Student, MSc in Computational Structural Engineering, BUTE

Supervisor

Dr. Imre Bojtár

Department of Structural Mechanics, BUTE

Budapest, 2013.

***To my grandmother**.*

PREFACE

This thesis is about the making of a computer program capable of calculating rectangular Mindlin plates resting on elastic Winkler foundation. I devote the program to be **helpful to the students** and be **a useful tool in the preliminary design of simple structures**. To achieve these goals, my expectations towards the software are **high calculation speed**, **self-explanatory graphical user interface** and the **ability to visualize** the results in a clear manner. It is important that the program is free and it will be available for download at the web site of the Department of Structural Mechanics (http://www.me.bme.hu).

The program is intended to provide solution of four section types, so in the **first chapter** I present the common mechanical models of these types of plates. The first type deals with isotropic homogeneous plates. In this context, I will detail first the classical Kirchoff and later introduce the Mindlin theory as an extension using first-order shear theory. The second type treats sandwich constructions. The third involves composite material plates including anisotropy and symmetrically laminated considerations while the last deals with plates including voids.

After the theoretical introduction, in the **second chapter** I introduce the original solution of Professor Ernest Hinton* for rectangular isotropic plates based on Mindlin plate theory. The original code went beyond an overhaul, and this final version will be detailed. The program itself is constructed in FORTRAN computational language and provides solution for simply supported plates.

The **third section** covers the making of the graphical user interface (GUI). For this task I have chosen the fully object oriented VISUAL BASIC .NET 4.5 as a programming environment. This part proved to have the biggest extent, so the reader can meet in more details with the different solution techniques and tricks. At the end of this chapter the numerical and graphical results of the program will be verified by finite element calculations.

In the **last chapter** there is a short comparison of the created program with the leading inland civil engineering software, namely AXIS and FEM-Design. It is important to emphasize, that is was not between the goals of creating this program to produce a rival of these software, but to provide an **easy way to get results very quickly** in case of some simple plated structures.

The standard educational routine **does not include programming courses**, therefore I find it to be a very important result, that through the thesis I managed to learn two different programming language from the bottom and this knowledge could be a useful tool in the future.

BEVEZETŐ

Jelen munka tárgya egy olyan számítógépes program megírása, mely lehetővé teszi négyzet alaprajzú, csuklósan megtámasztott, rugalmasan megtámasztott lemezek számítását a Mindlin lemezelmélet szerint. A program készítésének célja, hogy **hasznos eszköz legyen mind a hallgatóság, mind a gyakorló mérnökök kezében** akár az **előtervezés folyamatában**. Hogy ezeket a célokat elérjem, előzetes elvárásaim a **gyors futási idő**, **könnyen kezelhető grafikus felület** és a **látványos eredmény feldolgozás** voltak. A program ingyenes, elkészültével letölthető lesz a Tartószerkezetek Mechanikája Tanszék honlapjáról (http://www.me.bme.hu).

A program négy keresztmetszet típust támogat, az **első fejezetben** ezeknek a lemezeknek a szokásos mechanikai modelljeit mutatom be. Az első típus a homogén lemez, aminek kapcsán bemutatásra kerülnek a klasszikus Kirchoff, majd a Mindlin – Reissner lemezelmélet az előbbi kiterjesztéseként. A második típus a szendvics keresztmetszetű lemez. A harmadik szimmetrikusan laminált lemezekkel foglalkozik, az utolsó pedig az egyik irányban üreges lemezek számításának módját mutatja be.

Az elméleti áttekintés után, a **második fejezetben** bemutatom Ernest Hinton megoldását. Az eredeti kód FORTRAN77 nyelven íródott, azonban a jelentős mértékű átalakítás miatt csak a végső, javított verziót közlöm.

A **harmadik fejezet** bemutatja az elkészült grafikus felhasználói felületet. Ehhez a munkához az immár teljesen objektum orientált VISUAL BASIC .NET 4.5 programnyelvet választottam. Ez a munka bizonyult a legterjedelmesebbnek, ezért részletesebben bemutatásra kerülnek majd a különböző megoldások. A fejezet végén mind a numerikus, mind a grafikai eredmények verifikálásra kerülnek végeselemes számolások révén.

Az **utolsó fejezetben** ismertetek egy szubjektív összehasonlítást az elkészült program és az ANSYS, valamint a vezető hazai építőmérnöki statikai méretező programok (Axis, FEM-Design) között a számítási idő tekintetében. Hangsúlyozom, hogy a program céljai között nem szerepelt, hogy felvegye a versenyt az említett szoftverekkel, ugyanakkor bizonyos **egyszerű esetekben gyors és megbízható eredményre vezet**.

Végül megemlítem, hogy az építőmérnöki kari tanrend nem tartalmaz programozási kurzusokat, így a fent említett két programozási nyelvet önállóan tanultam meg, a megszerzett tudást pedig mindenféleképpen hasznosnak tartom a továbbiakban.

Table of Contents

References

The references of equations, figures, tables and diagrams are hyperlinks in the text, the reader can easily jump to the referenced object by clicking them.

TERMINOLOGY

Greek letter scalars

ε_{ij}	normal strain
γ_{ij}	shear strain
κ_x	curvature with respect to axis *x*
κ_y	curvature with respect to axis *y*
κ_{xy}	warping
σ_{ij}	normal stress
τ_{ij}	shear stress
ν_{ij}	Poisson's ratio
ν_f	Poisson's ratio of the fiber (1.3)
ν_f	Poisson's ratio of the facing (1.8)
ν_c	Poisson's ratio of the core (1.8)
ν_m	Poisson's ratio of the matrix (1.3)
ϑ_x	rotation of a normal with respect to axis *x*
ϑ_y	rotation of a normal with respect to axis *y*
θ	angle between the material and problem coordinate system
ξ	*x* coordinate of the center of the load
η	*y* coordinate of the center of the load
Π	strain energy

Scalars

t_f	thickness of the facing (1.8, 1.9)
t_c	thickness of the core (1.8)
t_w	thickness of the web (1.9)
a	side length of the plate parallel to axis *x*
b	side length of the plate parallel to axis *y*
w	width of a void in a voided plate
h	height of a void in a voided plate
k	bedding constant of Winkler foundation
m	running parameter of the Fourier summation
n	running parameter of the Fourier summation
u_0	displacement of the middle plane of the plate in *x* direction
v_0	displacement of the middle plane of the plate in *x* direction
w_0	displacement of the middle plane of the plate in *x* direction
u	displacement of a point in *x* direction
u	side length of the load patch parallel to axis *x* (2.1)
v	displacement of a point in *y* direction
v	side length of the load patch parallel to axis *y* (2.1)
w	displacement of a point in *z* direction
z	distance from the middle plane of the plate
t	thickness of the plate
E	Young's modulus (without subscripts)

E_{ij} nonlinear strain term (with subscripts)

E_f Young's modulus of the fiber (1.3)

E_f Young's modulus of the facing (1.8)

E_c Young's modulus of the core (1.8)

E_m Young's modulus of the matrix material (1.3)

G shear modulus

G_f shear modulus of the facing (1.8)

G_c shear modulus of the core (1.8)

$\bar{A}$ membrane stiffness of the plate

$\overline{D}$ bending stiffness of the plate

M_x bending moment with respect to axis *x*

M_y bending moment with respect to axis *y*

M_{xy} twisting moment

Q_x vertical shear force of an infinitesimal volume, on the side whith normal vector parallel with axis x

Q_y vertical shear force of an infinitesimal volume, on the side whit normal vector parallel with axis y

q vertical load

$p(x, y)$ vertical load

K_s shear modification factor

l_{ij} direction cosine

v_f fiber volume friction (1.3)

v_m matrix volume friction (1.3)

N number of laminas in a laminate

k index of the actual lamina in a laminate

A_{mn} Fourier coefficient

B_{mn} Fourier coefficient

C_{mn} Fourier coefficient

q_{mn} Fourier coefficient

P load intensity

Vectors

$\{\varepsilon\}$ normal strain vector

$\{\varepsilon_0\}$ strain vector of the middle plane

$\{\kappa\}$ curvature vector

$\{\sigma\}$ stress vector

$\{N\}$ vector of normal forces

$\{M\}$ vector of moments

$(\hat{e}_\iota)_m$ orthonormal basis vectors in the material coordinate system

$(\hat{e}_\iota)_p$ orthonormal basis vectors in the problem coordinate system

Matrices

$[Q]$ material stiffness matrix of the plate

$[A]$ sub matrix of $[Q]$

$[B]$ sub matrix of $[Q]$

$[D]$ sub matrix of $[Q]$

$[L]$ transformation matrix

Tensors

C_{ijkl} fourth order stiffness tensor of material parameters

S_{ijkl} fourth order compliance tensor of material parameters

$[\sigma]_m$	stress tensor with respect to the material coordinate system
$[\sigma]_p$	stress tensor with respect to the problem coordinate system

FORTRAN variables

A()	angle of principal laminate direction
C	core thickness
E	Elastic modulus
E11, E22	elastic modulus in the first and in the second principal directions
EF	elastic modulus of the facing
GC	shear modulus of the core
G12	in plane shear modulus
HD	center to center distance between flanges
WD	center to center distance between webs
H1()	z coordinate of boundary of laminate
IQQ	= 1 for homogeneous plates = 2 for sandwich plates = 3 for voided plates = 4 for laminated plates
M	number of laminates in a laminated plate
T	plate thickness
TF	thickness of the flanges
TW	thickness of the web
VNU	Poisson's ratio
V12	in plane Poisson's ratio
VF	Poisson's ratio of the facing material
AA, BB	in plane dimensions of the plate in *x* and *y* directions respectively
ETA, ETB	*x* and *y* coordinate of the center of the load
G1	shear modification factor
GFS	foundation modulus
IQ	= 1 for uniform load = 2 for concentrated load = 3 for rectangular patch load = 4 for pyramid patch load
IT	= 1 for non-symmetric loads = 2 for symmetric loads
NPON	number of output points
NUM	number of Fourier terms
PZ	load intensity
U, V	side length of patch loads parallel to axis *x* and *y* respectively
PI	π
W()	deflection
WUX(),WUY()	rotations of the cross sections in case of the kinematic assumptions of the Mindlin – Reissner theory

`CX(),CY(),CXY()`	curvatures with respect to axis x and y and the mixed curvature parameter, called warping
`BMX(),BMY(),BMXY()`	bending moments M_x, M_y and twisting moment M_{xy}
`QX(),QY()`	– shear forces Q_x and Q_y

1. PLATE THEORY

In continuum mechanics, plate theories are mathematical descriptions of mechanics for calculating deformations and stresses in flat plates. A plate is defined as a planar structural element, whose thickness is small with respect to its other dimensions. This property is utilized to reduce the three-dimensional problem to a two-dimensional one.

There are numerous plate theories that have been developed. The most widely used are the classical Kirchoff-Love and the Mindlin-Reissner theories, these will be introduced in the next pages. Although the numerical solution is concerned only with the Mindlin – Reissner theory, I find it necessary for the possible users to have a theoretical summary.

1.1. Kirchoff - Love plate theory

The theory of thin plates was developed in 1888 by Love[1] using assumptions proposed by Kirchoff[2]. Although this theory is not related to the main objective of the thesis, it worth mentioning for comparison and as a fundamental basis for further thoughts.

A plate can be considered to be thin, if its thickness is typically less than a 1/20 of the smallest side length.

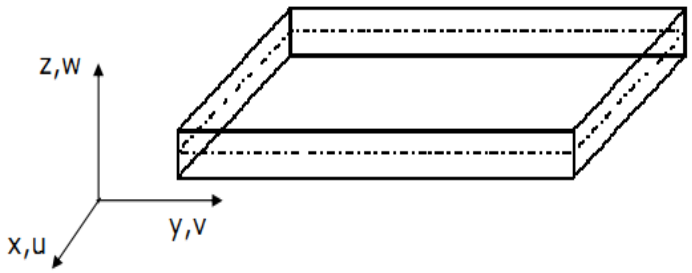


Figure 1 – Coordinate axes and the corresponding displacements of a simple plate (Airoldi).

Assumptions

The classical theory of plates holds for thin plates, and uses the next assumptions:

- There is no deformation in the middle plane of the plate. This plane remains neutral during bending.

  If there are external forces acting in the middle plane of the plate, this assumption does not hold and the effect of in-plane stresses on bending should be taken into account.

  The criteria can completely satisfied only, if the plate is bent into a developable surface. In other cases, strains arise in the middle plane. If the deflections are small, the corresponding stresses are negligible.

[1] Augustus Edward Hough Love (17 April 1863 – 5 June 1940). British mathematician, famous for his work on the mathematical theory of elasticity and wave propagation (Wikipedia).
[2] Gustav Kirchoff (12 March 1824 – 17 October 1887). German physicist who contributed to the fundamental understanding of electrical circuits, spectroscopy, and the emission of black-body radiation by heated objects (Wikipedia).

- Straight lines normal to the mid-surface remain normal to the mid-surface after deformation.

  This assumption is equivalent to the disregard of the effect of shear forces on the deflection. In most cases it is satisfactory, but there are situations when the effect of shear becomes important and corrections should be made in the theory.

- The deflection of the mid-plane is small with respect to the plate thickness.

  It means, that the theory is developed under small strain and rotation assumptions. Therefore equilibrium can be referred to the undeformed configuration and the small strain tensor can be used.

- Normal stresses acting on planes that are parallel to the faces can be neglected.

Using these four assumptions, all stress components can be expressed in the terms of the deflection *w*, which is a function of the coordinates *x* and *y*. The deflection function has to satisfy a linear partial differential equation, which, together with the boundary conditions, totally defines *w*. After we obtained the deflection, we can calculate the stresses at any point of the plate.

Strains and displacements

According to the third assumption, the elements of the strain tensor are:

$$\varepsilon_{xx} = \frac{\partial u}{\partial x};\ \varepsilon_{yy} = \frac{\partial v}{\partial y};\ \varepsilon_{zz} = \frac{\partial w}{\partial z};\tag{1.1}$$

$$\gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y};\ \gamma_{zx} = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z};\ \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}.\tag{1.2}$$

The normality condition is a constraint for the motion of the plate. It leads to express the in-plane displacements of an arbitrary point as a function of the derivative of the vertical displacement w_0, and the distance of the point from the mid-plane.

$$u_{bending} = -z\frac{\partial w_0}{\partial x},\tag{1.3}$$

$$v_{bending} = -z\frac{\partial w_0}{\partial y}.\tag{1.4}$$

If we add the displacements of the mid-plane, the following relations are obtained:

$$u = u_0 - z\frac{\partial w_0}{\partial x},\tag{1.5}$$

$$v = v_0 - z\frac{\partial w_0}{\partial y}.\tag{1.6}$$

Again, as a consequence of the third condition, the vertical displacement of any point is identical

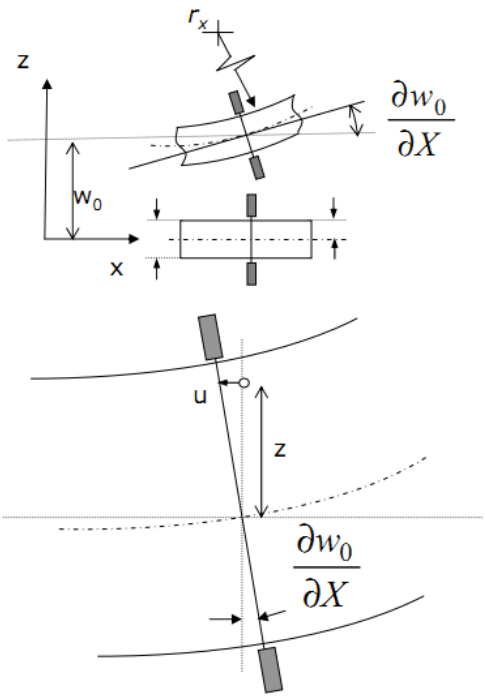


Figure 2 – Displacement of the middle surface (Airoldi).

to the vertical displacement of the mid-plane. Thus the displacement field of a general point can be obtained:

$$u(x_0, y_0, z) = u_0(x_0, y_0) - z\frac{\partial w_0}{\partial x}, \tag{1.7}$$

$$v(x_0, y_0, z) = v_0(x_0, y_0) - z\frac{\partial w_0}{\partial y}, \tag{1.8}$$

$$w(x_0, y_0, z) \cong w_0(x_0, y_0). \tag{1.9}$$

The expressions for the displacements can be substituted back into the definition of the strain terms, so the strain state of the plate can be expressed as:

$$\varepsilon_{xx} = \frac{\partial u_0}{\partial x} - z\frac{\partial^2 w_0}{\partial x^2}, \tag{1.10}$$

$$\varepsilon_{yy} = \frac{\partial v_0}{\partial y} - z\frac{\partial^2 w_0}{\partial y^2}, \tag{1.12}$$

$$\varepsilon_{zz} = 0, \tag{1.11}$$

$$\gamma_{yz} = \frac{\partial}{\partial z}\left(v_0(x_0, y_0) - z\frac{\partial w_0}{\partial y}\right) + \frac{\partial w_0}{\partial y} = -\frac{\partial w_0}{\partial y} + \frac{\partial w_0}{\partial y} = 0, \tag{1.13}$$

$$\gamma_{zx} = \frac{\partial}{\partial z}\left(u_0(x_0, y_0) - z\frac{\partial w_0}{\partial x}\right) + \frac{\partial w_0}{\partial x} = -\frac{\partial w_0}{\partial x} + \frac{\partial w_0}{\partial x} = 0, \tag{1.14}$$

$$\gamma_{xy} = \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} - 2z\frac{\partial^2 w_0}{\partial x \partial y}. \tag{1.15}$$

These relations are in good agreement with the fourth assumption.

The strain state can be considered as a sum of strains due to membrane deformation and flexural deformation. Therefore

$$\varepsilon_{xx} = \varepsilon_{0_{xx}} - z\frac{\partial^2 w_0}{\partial x^2} = \varepsilon_{0_{xx}} - z\,\kappa_x, \tag{1.16}$$

$$\varepsilon_{yy} = \varepsilon_{0_{yy}} - z\frac{\partial^2 w_0}{\partial y^2} = \varepsilon_{0_{yy}} - z\,\kappa_y, \tag{1.17}$$

$$\gamma_{xy} = \gamma_{0_{xy}} - 2z\frac{\partial^2 w_0}{\partial x \partial y} = \gamma_{0_{xy}} - z\,\kappa_{xy}. \tag{1.18}$$

In the later relations κ_x and κ_y are curvatures with respect to axis *x* and *y*, while κ_{xy} is a mixed curvature parameter called warping.

The strain state can be expressed also in vector notation:

$$\{\varepsilon\} = \{\varepsilon_0\} + z\{\kappa\}. \tag{1.19}$$

Constitutive equations

We apply Hooke's model do describe the connection between stresses and strains. It is only a first order approximation of the real material behavior, but it holds, if the forces and deformations are small enough.

According to the assumptions, a thin plate can be studied only considering a plane stress state, therefore the form of the material model in matrix notation:

$$\begin{Bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \tau_{xy}\end{Bmatrix}=\begin{bmatrix}\frac{E}{1-\nu^2} & \frac{\nu E}{1-\nu^2} & 0\\ \frac{\nu E}{1-\nu^2} & \frac{E}{1-\nu^2} & 0\\ 0 & 0 & \frac{E}{2(1+\nu)}\end{bmatrix}\begin{Bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \gamma_{xy}\end{Bmatrix}=[Q]\{\varepsilon\}=[Q]\{\varepsilon_0\}+[Q]z\{\kappa\}. \quad (1.20)$$

The generalized force parameters are shown on Figure 3. These can be determined by

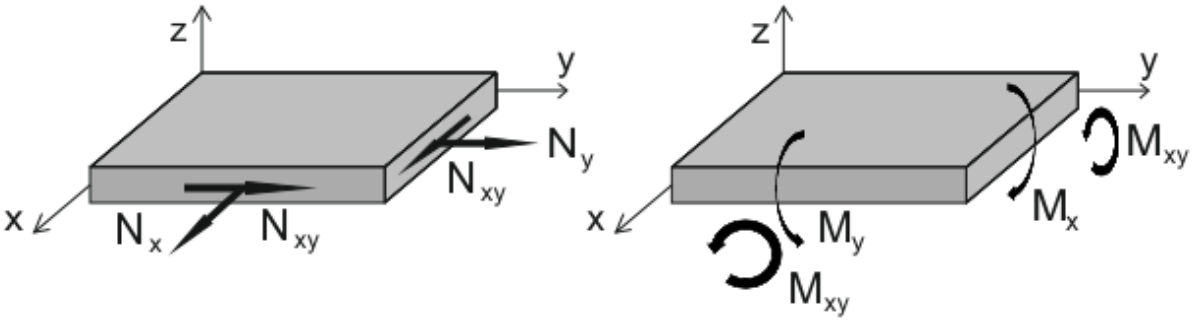


Figure 3 – Stress resultants.

integrating the stress components along the plate thickness, so we obtain the resultants and the moments per unit length.

$$\{N\}=\begin{Bmatrix}N_x\\ N_y\\ N_{xy}\end{Bmatrix}=\int_{-t/2}^{t/2}\{\sigma\}dz=\int_{-t/2}^{t/2}([Q]\{\varepsilon_0\}+[Q]z\{\kappa\})dz=[Q]t\{\varepsilon_0\}, \quad (1.21)$$

$$\begin{Bmatrix}N_x\\ N_y\\ N_{xy}\end{Bmatrix}=\underbrace{\begin{bmatrix}1 & \nu & 1\\ \nu & 1 & 0\\ 0 & 0 & (1-\nu)/2\end{bmatrix}\underbrace{\frac{Et}{(1-\nu^2)}}_{\tilde{A}}}_{[A]}\begin{Bmatrix}\varepsilon_{0_{xx}}\\ \varepsilon_{0_{yy}}\\ \gamma_{0_{xy}}\end{Bmatrix}. \quad (1.22)$$

$$\{M\}=\begin{Bmatrix}M_x\\ M_y\\ M_{xy}\end{Bmatrix}=\int_{-t/2}^{t/2}\{\sigma\}zdz=\int_{-t/2}^{t/2}([Q]z\{\varepsilon_0\}+[Q]z^2\{\kappa\})dz=[Q]\frac{t^2}{12}\{\kappa\}, \quad (1.23)$$

$$\begin{Bmatrix}M_x\\ M_y\\ M_{xy}\end{Bmatrix}=\underbrace{\begin{bmatrix}1 & \nu & 1\\ \nu & 1 & 0\\ 0 & 0 & (1-\nu)/2\end{bmatrix}\underbrace{\frac{Et^3}{12(1-\nu^2)}}_{\tilde{D}}}_{[D]}\begin{Bmatrix}\kappa_x\\ \kappa_y\\ \kappa_{xy}\end{Bmatrix}. \quad (1.24)$$

These equations can be arranged into a more compact form, by introducing the sub-matrices$[A]$,$[B]$,$[D]$, though this form is used for orthotropic laminates.

$$\begin{Bmatrix}\{N\}\\ \{M\}\end{Bmatrix}=\begin{bmatrix}[A] & [B]\\ [B]^T & [D]\end{bmatrix}\begin{Bmatrix}\{\varepsilon_0\}\\ \{\kappa\}\end{Bmatrix}. \quad (1.25)$$

The matrix constructed by such sub-matrices is the *material stiffness matrix* of the plate. Sub-matrix $[B]$ represents a *coupling between flexural and membrane deformations*, therefore it is a null matrix in the case of isotropic laminates. Note that it is null because the boundaries of integration differ only in sign. Later we will discuss laminated plates, where this condition is not valid.

In the above expressions, $\bar{A}$ and $\bar{D}$ characterizes the *membrane and bending stiffness*. The latter one is also called as the *flexural rigidity of plates*.

Bending equations of isotropic plates

Let us consider a plate of uniform thickness, equal to *t*, and be the *xy* plane the middle plane of the plate before loading. Let *x* and *y* axes coincide with one of the longitudinal edges, and let the positive direction of the vertical *z* axis be upward, as shown on Figure 4, as well as the free body diagram of an elementary volume, cut out of the plate by four planes parallel to the *xy* and *yz* planes.

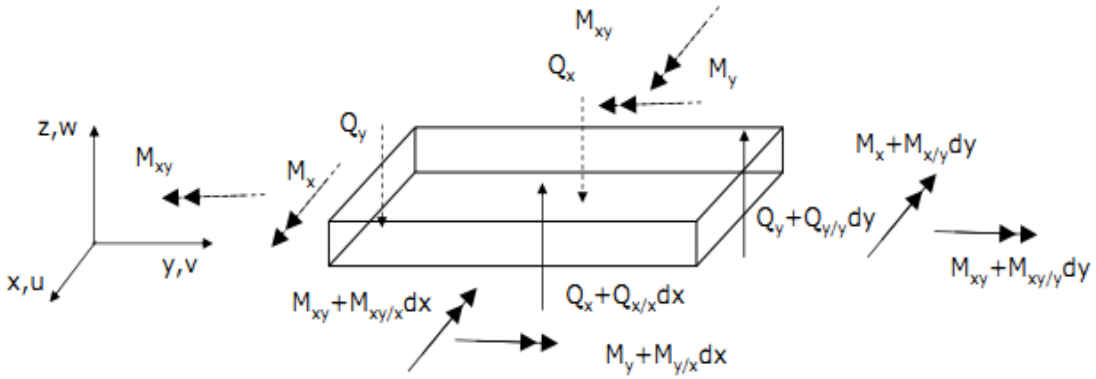


Figure 4 – Forces and moments acting on an infinitesimal volume. The '/' sign means partial differentiation with respect to the subsequent character(s) (Airoldi).

One can see internal forces of one element as bending moments M_x and M_y, twisting moments M_{xy} and M_{yx} and vertical shearing forces Q_x and Q_y. Though transverse shear stresses are neglected as a consequence of normality restrictions, transverse forces are applied to the plate. It means, that though shear stress exists, the shear stiffness of the plate is very high, let say infinite compared to the bending stiffness. Anyway, such shear forces causes bending and must be taken into account in equilibrium considerations. The connection of the internal forces and moments with the stresses are:

$$M_x = \int_{-t/2}^{t/2} \sigma_x \cdot z \, dz, \quad M_y = \int_{-\frac{t}{2}}^{\frac{t}{2}} \sigma_y \cdot z \, dz\,, \tag{1.26}$$

$$M_{xy} = \int_{-t/2}^{t/2} \tau_{xy} \, dz, \quad M_{yx} = \int_{-t/2}^{t/2} \tau_{yx} \, dz, \tag{1.28}$$

$$Q_x = \int_{-t/2}^{t/2} \tau_{xz} \, dz, \quad Q_y = \int_{-t/2}^{t/2} \tau_{yz} \, dz. \tag{1.27}$$

Formerly we restricted the loads to be normal to the surface. Denote the intensity of this load by q, so the load acting on the element is $q \cdot dx \cdot dy$. Since the stress component σ_z is neglected, we are not able to apply this load on the upper face of the plate. Instead, the transverse load appears as a discontinuity in the magnitude of the shear stresses, which vary according to the parabolic law through the thickness. If the effect of the surface load becomes of special interest, thick-plate theory has to be used.

If in addition to the former assumptions we suppose, that the *edges of the plate are free to move in the plane of the plate*, we can neglect any strain in the mid-plane of the plate during bending.

On the basis of the figures above, we can write the equilibrium equations:

$$\frac{\partial Q_x}{\partial x}\,dx\,dy + \frac{\partial Q_y}{\partial y}\,dy\,dx + q\,dx\,dy = 0, \tag{1.29}$$

$$\frac{\partial M_{xy}}{\partial x}\,dx\,dy - \frac{\partial M_y}{\partial y}\,dy\,dx + Q_y\,dx\,dy = 0, \tag{1.30}$$

$$\frac{\partial M_{yx}}{\partial y}\,dx\,dy + \frac{\partial M_x}{\partial x}\,dy\,dx - Q_x\,dx\,dy = 0. \tag{1.31}$$

As it can be seen, the moments due to the lateral load and the change of the shearing forces are neglected, because this members are small compared to those retained.

After simplification we obtain:

$$\frac{\partial Q_x}{\partial x} + \frac{\partial Q_y}{\partial y} + q = 0, \tag{1.32}$$

$$\frac{\partial M_{xy}}{\partial x} - \frac{\partial M_x}{\partial y} + Q_y = 0, \tag{1.33}$$

$$\frac{\partial M_{yx}}{\partial y} + \frac{\partial M_y}{\partial x} - Q_x = 0. \tag{1.34}$$

We can rule out the shearing forces by expressing them from the last two equations and substituting into the first:

$$\frac{\partial^2 M_x}{\partial x^2} + \frac{\partial^2 M_{yx}}{\partial x \partial y} + \frac{\partial^2 M_y}{\partial y^2} - \frac{\partial^2 M_{xy}}{\partial x \partial y} = -q. \tag{1.35}$$

Due to the fact that $\tau_{xy} = \tau_{yx}$ we can conclude, that $M_{xy} = -M_{yx}$, so finally we present the equation of equilibrium in the following form:

$$\frac{\partial^2 M_x}{\partial x^2} + \frac{\partial^2 M_y}{\partial y^2} - 2\frac{\partial^2 M_{xy}}{\partial x \partial y} = -q. \tag{1.36}$$

In matrix notation:

$$\left\{\frac{\partial^2}{\partial x^2} \quad \frac{\partial^2}{\partial y^2} \quad -2\frac{\partial^2}{\partial x \partial y}\right\}\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} = -q. \tag{1.37}$$

Here we can substitute the expressions for the moments from equation (1.24), so we obtain:

$$\left\{\frac{\partial^2}{\partial x^2} \quad \frac{\partial^2}{\partial y^2} \quad -2\frac{\partial^2}{\partial x \partial y}\right\}\begin{bmatrix} 1 & \nu & 1 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix}\overline{D}\begin{Bmatrix} -\dfrac{\partial^2 w}{\partial x^2} \\ -\dfrac{\partial^2 w}{\partial y^2} \\ -2\dfrac{\partial^2 w}{\partial x \partial y} \end{Bmatrix} = -q. \tag{1.38}$$

From this matrix form a fourth order differential equation can be obtained[3]:

(1.39)

[3] This equation was obtained first by Lagrange in 1811 (Wikipedia).

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

Then the solution of bending of plates simplifies to the integration of the latter equation. If for a particular case, a solution of this differential equation satisfies the prescribed boundary conditions, bending moments can be calculated by means of equations (1.22) and (1.24). Thereafter we determine the shearing forces from the rotational equilibrium equations (1.32)-(1.34), which lead to the stresses. Shear stresses τ_{xz} and τ_{yz} can be obtained by assuming, that they are distributed along the thickness of the plate according to the parabolic law. This assumption is called from the 'cylindrical bending of plates', where we state, that the distribution of transverse shear stress τ_{xz} can be found only considering the state of stress induced by M_y and Q_x, without any parameters along the *y* direction, and τ_{yz} similarly. Despite the transverse shear stress can only be approximately recovered, the underestimation of deflections and other parameters can be neglected in case of isotropic plates if side to thickness ratio remains over 20, otherwise higher order theory shall be used.

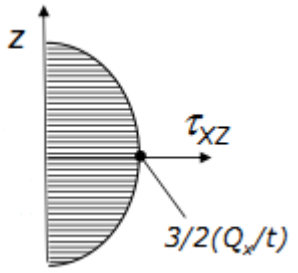


Figure 5 – Assumed distribution of the shear stresses (Airoldi).

1.2. Mindlin - Reissner plate theory

Two similar, but not identical theory were proposed by Raymond Mindlin[4] in 1951 and Eric Reissner[5] in 1945. Both of them are intended for the calculation of thick plates, where the normal to the mid-surface remains straight, but not necessarily perpendicular to the mid-surface after the deformations. The main difference is that Reissner's theory does not invoke the plane stress condition and the plate thickness may change during the deformation. The Mindlin – Reissner theory obeys the assumptions of Mindlin, thus it is more properly called Mindlin plate theory. This theory is also called the first-order shear deformation theory of plates, since it implies a **linear displacement variation through the thickness**. The Mindlin theory holds, even if the side to thickness ratio goes under 20.

Kinematic assumptions

Mindlin theory allows to take into account the effects of a constant transverse shear stress state by removing the normality conditions from the kinematic assumptions of classical plate theory. All other assumptions are kept.

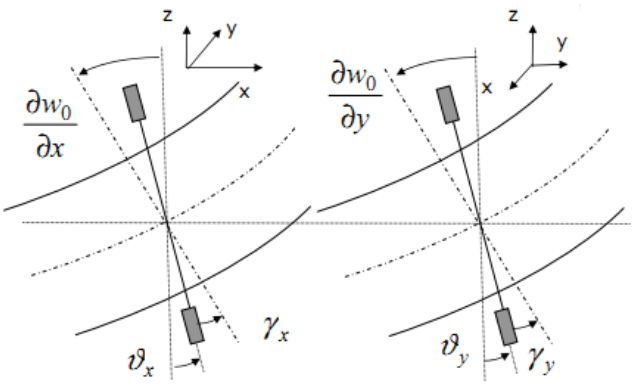


Figure 6 - Displacement of the middle surface (Airoldi).

[4] Raymond David Mindlin (17 September 1906 – 22 November 1987). American mechanical engineer who made seminal contributions to many branches of applied mechanics, applied physics, and engineering sciences (Wikipedia).

[5] Eric Reissner (5 January 1913 – 1 November 1996). German-born American engineer who dealt with the theory of elasticity, the theory of plates, shells and beams, dynamics of structures and turbulence, aerodynamics and wing theory (Wikipedia).

On Figure 6 ϑ_x and ϑ_y are the rotations of the sections that were originally normal to the mid-plane after the deformations. Thus the displacement field is:

$$u(x_0, y_0, z) = u_0(x_0, y_0) - z\vartheta_x, \tag{1.40}$$

$$v(x_0, y_0, z) = v_0(x_0, y_0) - z\vartheta_y, \tag{1.41}$$

$$w(x_0, y_0, z) \cong w_0(x_0, y_0). \tag{1.42}$$

Due to the new kinematic assumptions, we have to redefine the curvatures and the in-plane strain components.

$$\varepsilon_{xx} = \frac{\partial u_0}{\partial x} - z\frac{\partial \vartheta_x}{\partial x} = \varepsilon_{0_{xx}} - z\frac{\partial \vartheta_x}{\partial x} = \varepsilon_{0_{xx}} + z\,\kappa_x, \tag{1.43}$$

$$\varepsilon_{yy} = \frac{\partial v_0}{\partial y} - z\frac{\partial \vartheta_y}{\partial y} = \varepsilon_{0_{yy}} - z\frac{\partial \vartheta_y}{\partial y} = \varepsilon_{0_{yy}} + z\,\kappa_y, \tag{1.44}$$

$$\gamma_{xy} = \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} - z\frac{\partial \vartheta_x}{\partial y} - z\frac{\partial \vartheta_x}{\partial x} = \gamma_{0_{xy}} - z\frac{\partial \vartheta_x}{\partial y} - z\frac{\partial \vartheta_x}{\partial x} = \gamma_{0_{xy}} + z\,\kappa_{xy}. \tag{1.45}$$

For the transverse shear components the following relations hold:

$$\gamma_{zx} = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} = \frac{\partial w_0}{\partial x} + \frac{\partial}{\partial z}(u_0(x_0, y_0) - z\vartheta_x) = \frac{\partial w_0}{\partial x} - \vartheta_x = \gamma_x, \tag{1.46}$$

$$\gamma_{zy} = \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} = \frac{\partial w_0}{\partial y} + \frac{\partial}{\partial z}\left(v_0(x_0, y_0) - z\vartheta_y\right) = \frac{\partial w_0}{\partial y} - \vartheta_y = \gamma_y. \tag{1.47}$$

After all, plane deformation can be considered as a superimposition of a bending mode and a shear mode. If $\gamma = 0$, pure bending is obtained (A case), whereas if $\vartheta = 0$ a pure shear situation occurs (B case) as it is visualized on Figure 7.

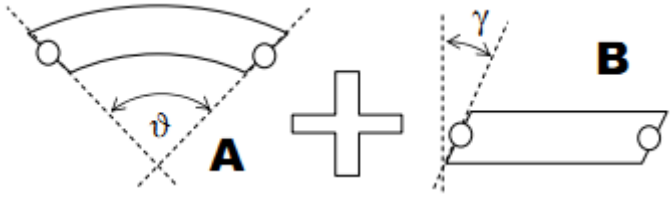


Figure 7 – Superposition of plane deformations (Airoldi).

Constitutive equations

Equations (1.20), (1.22) and (1.24) formulated in the previous chapter still hold with the redefined curvature tensor, but an additional constitutive relation has to be considered between the average shear strain and the shearing forces. As it was mentioned, in the classical plate theory the shear modulus is infinite. Now this statement is not valid, so it calls for further equations.

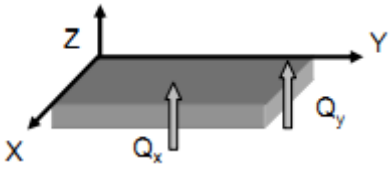


Figure 8 – Shear forces (Airoldi).

$$\begin{Bmatrix} Q_x \\ Q_y \end{Bmatrix} = \int_{-t/2}^{t/2} \begin{Bmatrix} \tau_{xz} \\ \tau_{yz} \end{Bmatrix} dt = \int_{-t/2}^{t/2} \begin{bmatrix} G & 0 \\ 0 & G \end{bmatrix} \begin{Bmatrix} \gamma_x \\ \gamma_y \end{Bmatrix} dt = \begin{bmatrix} Gt & 0 \\ 0 & Gt \end{bmatrix} \begin{Bmatrix} \gamma_x \\ \gamma_y \end{Bmatrix}. \tag{1.48}$$

Shear correction factor

As it was previously mentioned, Mindlin theory considers a constant transverse shear stress state. Such solution does not properly estimate the shear deformation work, a more realistic solution could be obtained by parabolic shear stress distribution. Therefore a correction factor is introduced to better approximate the contribution of the shear deformation work. The factor is based on the comparison of the strain energy per unit width corresponding to the constant and the parabolic shear stress distribution.

$$\begin{aligned}\Pi_{parabolic} &= \int_{-t/2}^{t/2} \frac{1}{2}\frac{{\tau_{xz}}^2}{G}dz \\ &= \int_{-t/2}^{t/2} \frac{1}{2}\frac{\left(Q_x\frac{3}{2t}\right)^2\left(1-\left(\frac{2z}{t}\right)^2\right)^2}{G}dz \\ &= \frac{1}{2}\left(Q_x\frac{3}{2t}\right)^2\frac{1}{G}\int_{-t/2}^{t/2}\left(1-2\left(\frac{2z}{t}\right)^2+\left(\frac{2z}{t}\right)^4\right)dz \\ &= \frac{1}{2}\left(Q_x\frac{3}{2t}\right)^2\frac{1}{G}\left|z-\frac{2}{3}\left(\frac{2z}{t}\right)^3\frac{t}{2}\right|_{-t/2}^{t/2} = \frac{3{Q_x}^2}{5Gt},\end{aligned} \tag{1.49}$$

$$\Pi_{constant} = \int_{-t/2}^{t/2}\frac{1}{2}\frac{{\tau_{xz}}^2}{G}dz = \int_{-t/2}^{t/2}\frac{1}{2}\frac{(Q_x/t)^2}{G}dz = \frac{1}{2}\frac{{Q_x}^2}{Gt}. \tag{1.50}$$

The ratio of the strain energies in this case is:

$$\frac{\Pi_{constant}}{\Pi_{parabolic}} = \frac{5}{6} = K_s. \tag{1.51}$$

If the shear modulus is corrected with this value, the shear energy correspond to one of parabolic shear stress distribution. Thus the corrected version of (1.48) is:

$$\begin{Bmatrix}Q_x\\Q_y\end{Bmatrix} = K_s\begin{bmatrix}Gt & 0\\0 & Gt\end{bmatrix}\begin{Bmatrix}\gamma_x\\\gamma_y\end{Bmatrix}. \tag{1.52}$$

Equilibrium equations

Equations of vertical equilibrium and rotations for Mindlin theory are identical to those formed in (1.32)-(1.34). By substituting the expressions for κ_x, κ_y, κ_{xy}, γ_x and γ_y from (1.43)-(1.45) into the constitutive relations (1.24) and (1.48), we obtain the following equations:

$$\begin{Bmatrix}M_x\\M_y\\M_{xy}\end{Bmatrix} = \underbrace{\begin{bmatrix}1 & \nu & 0\\\nu & 1 & 0\\0 & 0 & (1-\nu)/2\end{bmatrix}\underbrace{\frac{Et^3}{12(1-\nu^2)}}_{\bar{D}}}_{[D]}\begin{Bmatrix}-\frac{\partial\vartheta_x}{\partial x}\\-\frac{\partial\vartheta_y}{\partial y}\\-\frac{\partial\vartheta_x}{\partial y}-\frac{\partial\vartheta_x}{\partial x}\end{Bmatrix}, \tag{1.53}$$

$$\begin{Bmatrix} Q_x \\ Q_y \end{Bmatrix} = K_s \begin{bmatrix} Gt & 0 \\ 0 & Gt \end{bmatrix} \begin{Bmatrix} \frac{\partial w_0}{\partial x} - \vartheta_x \\ \frac{\partial w_0}{\partial y} - \vartheta_y \end{Bmatrix}. \tag{1.54}$$

If we substitute these into (1.32)-(1.34), than we obtain the three separate 2nd order differential equations of the Mindlin plate theory as follows.

$$K_s Gt \left(\frac{\partial^2 w}{\partial x^2} - \frac{\partial \vartheta_x}{\partial x} \right) + K_s Gt \left(\frac{\partial^2 w}{\partial y^2} - \frac{\partial \vartheta_y}{\partial y} \right) = -p(x,y), \tag{1.55}$$

$$K_s Gt \left(\frac{\partial w}{\partial x} - \vartheta_x \right) = -\overline{D} \frac{\partial^2 \vartheta_x}{\partial x^2} - \nu \overline{D} \frac{\partial^2 \vartheta_y}{\partial x \partial y} - \frac{1-\nu}{2} \overline{D} \left(\frac{\partial \vartheta_y}{\partial y \partial x} + \frac{\partial^2 \vartheta_x}{\partial y^2} \right), \tag{1.56}$$

$$K_s Gt \left(\frac{\partial w}{\partial y} - \vartheta_y \right) = -\overline{D} \frac{\partial^2 \vartheta_y}{\partial x^2} - \nu \overline{D} \frac{\partial^2 \vartheta_x}{\partial x \partial y} - \frac{1-\nu}{2} \overline{D} \left(\frac{\partial \vartheta_x}{\partial y \partial x} + \frac{\partial^2 \vartheta_y}{\partial y^2} \right). \tag{1.57}$$

1.3. Constitutive equations of laminated composite plates

Composite materials consist two or more materials which together produce beneficial properties that cannot be achieved with any of the constituents alone. In this point I introduce the stress-strain relations which are needed for the further investigation.

Generalized Hooke's Model

At first, let me take some comments on the Hooke's model, mentioned in the previous chapters. So far, the subjects of our investigations were homogeneous isotropic plates, which have the advantage of material symmetry in every directions, thus the number of independent material parameters could be reduced to 2. At more complicated materials we lose from the symmetry and a more general way of discussion is needed.

If the stress components are assumed to be linear functions of the strain components and we assume that the reference configuration is stress free, then the most general form of the linear constitutive equations with indicial notation:

$$\sigma_{ij} = C_{ijkl} \cdot \varepsilon_{kl}, \quad \varepsilon_{kl} = \varepsilon_{lk}, \tag{1.58}$$

where C_{ijkl} is the fourth order stiffness tensor of material parameters. This tensor has 81 scalar components which can be reduced by utilizing symmetry conditions, as discussed next.

The equation of angular momentum is a momentum equation written on an infinitesimal cube with unit size cut out from a solid body. This principle requires the stress tensor to be symmetric, so $\sigma_{ij} = \sigma_{ji}$. Then it follows, that C_{ijkl} must be symmetric in the first two subscripts and the number of independent material parameters reduces to 54. Since the strain tensor is symmetric by definition, then C_{ijkl} must be symmetric in the last two subscripts as well, further reducing the independent parameters to 36.

A material is said to be elastic, if the material behavior is only a function of the current state of deformation. In the special case, in which the work done by the stresses during a deformation is dependent only on the initial state and the current configuration, the material is called hyperelastic. Utilizing this property the independent material parameters can be reduced to 21, because $C_{ijkl} = C_{klij}$. To show this I express equation (1.58) with

single subscript notation. This notation is called engineering notation or Kelvin-Voigt notation.

Thus we have:

$$\sigma_1 = \sigma_{11},\ \sigma_2 = \sigma_{22},\ \sigma_3 = \sigma_{33},\ \sigma_4 = \sigma_{23},\ \sigma_5 = \sigma_{13},\ \sigma_6 = \sigma_{12}, \tag{1.59}$$

$$\varepsilon_1 = \varepsilon_{11},\ \varepsilon_2 = \varepsilon_{22},\ \varepsilon_3 = \varepsilon_{33},\ \varepsilon_4 = 2\varepsilon_{23},\ \varepsilon_5 = 2\varepsilon_{13},\ \varepsilon_6 = 2\varepsilon_{12}. \tag{1.60}$$

Equation (1.58) now takes the form

$$\sigma_i = C_{ij} \cdot \varepsilon_j, \quad \text{where i, j=1,...,6} \tag{1.61}$$

or in matrix notation

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{Bmatrix}. \tag{1.62}$$

Here the coefficients C_{ij} (i, j=1,…,6) are symmetric, so we have $6 + 5 + 4 + 3 + 2 + 1 = 21$ independent stiffness coefficients for the most general elastic material.

We assume that the stress-strain relations are invertible, thus

$$\varepsilon_i = S_{ij} \cdot \sigma_j, \tag{1.63}$$

$$\begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{Bmatrix} = \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} & S_{15} & S_{16} \\ S_{21} & S_{22} & S_{23} & S_{24} & S_{25} & S_{26} \\ S_{31} & S_{32} & S_{33} & S_{34} & S_{35} & S_{36} \\ S_{41} & S_{42} & S_{43} & S_{44} & S_{45} & S_{46} \\ S_{51} & S_{52} & S_{53} & S_{54} & S_{55} & S_{56} \\ S_{61} & S_{62} & S_{63} & S_{64} & S_{65} & S_{66} \end{bmatrix} \begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{Bmatrix}, \tag{1.64}$$

where S_{ij} are the material compliance parameters and $[S]$ is the compliance tensor.

Coordinate systems and transformation

First suppose that (x_1, x_2, x_3) denote the coordinate system with respect to which equations (1.61) and (1.62) are defined. We call it *material coordinate system*. The coordinate system (x, y, z) used to write the equations of motion and strain-displacement equations will be called the *problem coordinates* to distinguish them from the material coordinate system. Note that the phrase “material coordinates” of advanced mechanics **should not be confused** with the present term. Both of the mentioned coordinate systems are fixed in the body and the two systems are oriented with respect to each other. When elastic material parameters at a point have the same values for every pair of coordinate systems that are mirror images of each other in a certain plate, that plane is called a *material plane of symmetry*. It is important that symmetry

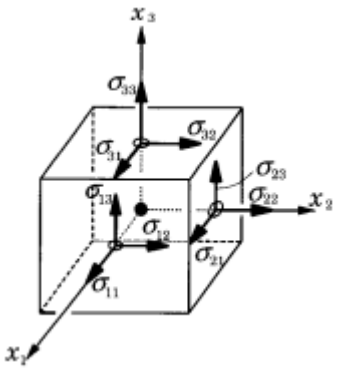


Figure 9 – Stresses acting on an elementary volume. (Reddy, 2004)

under discussion is a directional property, not a positional. Therefore a material can have certain elastic symmetry and different properties from point to point.

The details of coordinate transformation and the deduction of the below expressions can be found in the appendix of (Lengyel, 2012). Now shortly, vector $\{a\}$ has components a_i with respect to the rectangular Cartesian basis (e_1, e_2, e_3) and its components referred to another rectangular Cartesian basis $(\hat{e_1}, \hat{e_2}, \hat{e_3})$ are $\hat{a_i}$. The two sets of components are related according to

$$\hat{a_i} = l_{ij} a_j, \quad l_{ij} = \hat{e_i} e_j, \tag{1.65}$$

where l_{ij} are called the direction cosines. Similarly, the components of a second order tensor $[A]$ transform according to the rule

$$\widehat{A_{ij}} = l_{im} l_{ju} A_{mn} \text{ or } \widehat{[A]} = [L][A][L]^T. \tag{1.66}$$

Special case: Consider the case, when axes $\hat{z}$ and z as the out of plane axes of the plate in the two Cartesian coordinate systems are identical. The transformation corresponds to a rotation about the common axis by a certain angle α. The transformation matrix becomes

$$[L] = \begin{bmatrix} cos\alpha & sin\alpha & 0 \\ -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.67}$$

Realize, that this condition always holds in the case of flat plates with one or more layers of constant thickness.

Material Symmetry

Further reduction in the number of independent stiffness (or compliance) parameters comes from the material symmetry.

When three mutually orthogonal planes of material symmetry exist, the number of elastic coefficients is reduced to 9, and such materials are called orthotropic. The stress-strain relations for such a case:

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & & & \\ C_{21} & C_{22} & C_{23} & & [0] & \\ C_{31} & C_{32} & C_{33} & & & \\ & & & C_{44} & 0 & 0 \\ & [0] & & 0 & C_{55} & 0 \\ & & & 0 & 0 & C_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{Bmatrix}, \tag{1.68}$$

$$\begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{Bmatrix} = \begin{bmatrix} S_{11} & S_{12} & S_{13} & & & \\ S_{21} & S_{22} & S_{23} & & [0] & \\ S_{31} & S_{32} & S_{33} & & & \\ & & & S_{44} & 0 & 0 \\ & [0] & & 0 & S_{55} & 0 \\ & & & 0 & 0 & S_{66} \end{bmatrix} \begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{Bmatrix}. \tag{1.69}$$

More often, the material parameters are determined from laboratory experiments. Without the details, I enclose how to relate the stresses to the strains with engineering constants, in case of an orthotropic material.

$$\begin{Bmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_3\\ \varepsilon_4\\ \varepsilon_5\\ \varepsilon_6\end{Bmatrix} = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{21}}{E_2} & -\frac{\nu_{31}}{E_3} & & & \\ -\frac{\nu_{12}}{E_1} & \frac{1}{E_2} & -\frac{\nu_{32}}{E_3} & & [0] & \\ -\frac{\nu_{13}}{E_1} & -\frac{\nu_{23}}{E_2} & \frac{1}{E_3} & & & \\ & & & \frac{1}{G_{23}} & 0 & 0 \\ & [0] & & 0 & \frac{1}{G_{13}} & 0 \\ & & & 0 & 0 & \frac{1}{G_{12}} \end{bmatrix} \begin{Bmatrix}\sigma_1\\ \sigma_2\\ \sigma_3\\ \sigma_4\\ \sigma_5\\ \sigma_6\end{Bmatrix}. \tag{1.70}$$

In equation (1.70) E_1, E_2, E_3 are Young's moduli in 1,2 and 3 material directions, ν_{ij} is Poisson's ratio, defined as the ratio of transverse shear in the *j*th direction to the axial strain in the *i*th direction when stressed in the ith direction, and G_{23}, G_{13}, G_{12} are shear moduli in the 2-3, 1-3, and 1-2 planes, respectively. It is a symmetric matrix, therefore the 9 independent coefficients for an orthotropic material are:

$$E_1, E_2, E_3, G_{23}, G_{13}, G_{12}, \nu_{12}, \nu_{13}, \nu_{23}. \tag{1.71}$$

Because of the assumptions of the plate theories, we restrict our studies to *plane stress situations* which simplifies the above equations to:

$$\begin{Bmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_6\end{Bmatrix} = \begin{bmatrix} S_{11} & S_{12} & 0 \\ S_{12} & S_{22} & 0 \\ 0 & 0 & S_{66} \end{bmatrix} \begin{Bmatrix}\sigma_1\\ \sigma_2\\ \sigma_6\end{Bmatrix} = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{21}}{E_2} & 0 \\ -\frac{\nu_{12}}{E_1} & \frac{1}{E_2} & 0 \\ 0 & 0 & \frac{1}{G_{12}} \end{bmatrix} \begin{Bmatrix}\sigma_1\\ \sigma_2\\ \sigma_6\end{Bmatrix}, \tag{1.72}$$

and

$$\begin{Bmatrix}\sigma_1\\ \sigma_2\\ \sigma_6\end{Bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & 0 \\ Q_{12} & Q_{22} & 0 \\ 0 & 0 & Q_{66} \end{bmatrix} \begin{Bmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_6\end{Bmatrix}, \tag{1.73}$$

where Q_{ij} are the plane *stress-reduced stiffnesses*, given by inverting the compliance matrix

$$Q_{11} = \frac{S_{22}}{S_{11}S_{22} - {S_{12}}^2} = \frac{E_1}{1-\nu_{12}\nu_{21}}, \tag{1.74}$$

$$Q_{12} = \frac{S_{12}}{S_{11}S_{22} - {S_{12}}^2} = \frac{\nu_{21}E_1}{1-\nu_{12}\nu_{21}}, \tag{1.75}$$

$$Q_{22} = \frac{S_{11}}{S_{11}S_{22} - {S_{12}}^2} = \frac{E_2}{1-\nu_{12}\nu_{21}}, \tag{1.76}$$

$$Q_{66} = \frac{1}{S_{66}} = G_{12}. \tag{1.77}$$

It is to note, that the reduced stiffnesses have only four independent material constants, namely E_1, E_2, ν_{12} and G_{12}.

The transverse shear stresses are related to the transverse shear strains in an orthotropic material by the relations

$$\begin{Bmatrix}\sigma_4\\ \sigma_5\end{Bmatrix} = \begin{bmatrix}Q_{44} & 0\\ 0 & Q_{55}\end{bmatrix}\begin{Bmatrix}\varepsilon_4\\ \varepsilon_5\end{Bmatrix}, \quad Q_{44} = G_{23}, \quad Q_{55} = G_{13}. \tag{1.78}$$

Lamina and Laminates

A *lamina* or ply is a fundamental building block, a sheet of a composite material. Such a layer is often fiber-reinforced. The fibers can be continuous or discontinuous, woven, unidirectional, bidirectional or randomly distributed as seen on Figure 10.

A *laminate* is a collection of laminas stacked to produce the desired stiffness and thickness. The individual laminas can have the same or various orientations. The sequence of these various orientations is called *lamination scheme* or *stacking sequence*.

The chapter is devoted to the theoretical study of laminated structures, therefore manufacturing or design questions are omitted. In the remaining portion of the section we study the mechanical behavior of a single lamina, treating it as an orthotropic, linear elastic continuum.

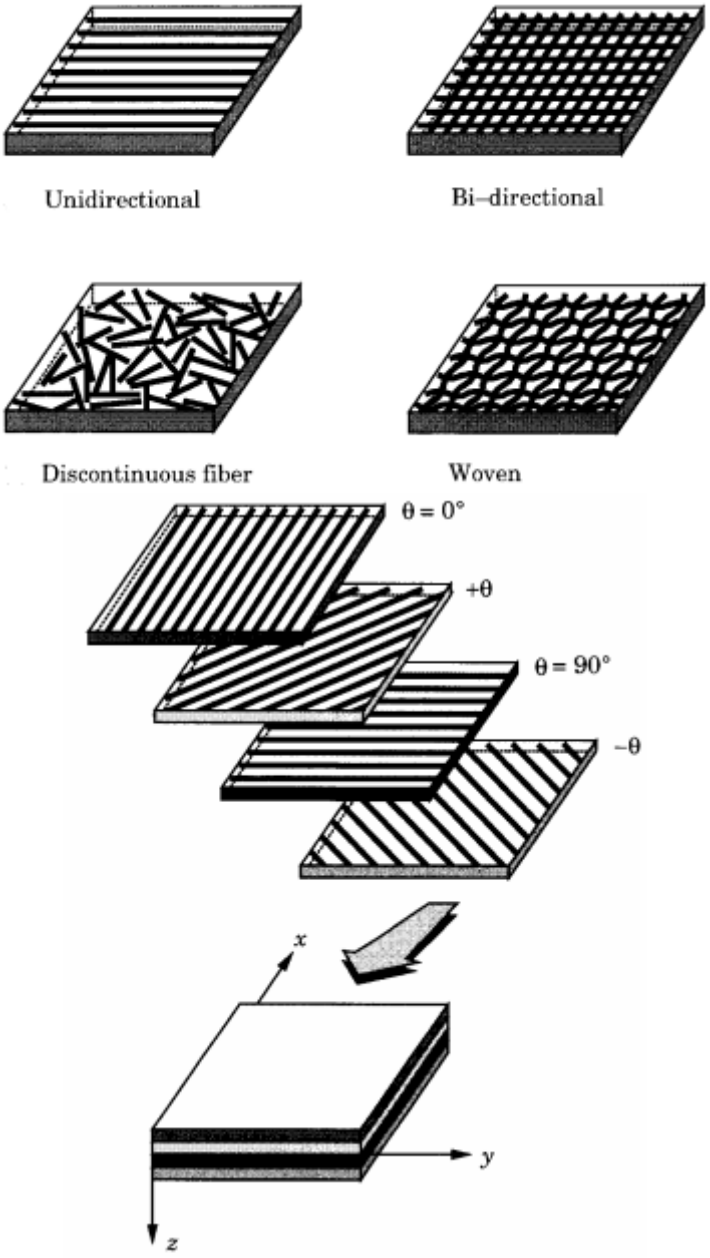


Figure 10 – Fiber oriantations and the development of a laminate. (Reddy, 2004)

Characterization of a unidirectional lamina

A unidirectional fiber-reinforced lamina is treated as an orthotropic material with symmetry planes parallel and transverse to the laminate fiber direction. The material coordinate axis x_1 is parallel to the fiber, x_2 is transverse to the fiber in the plane of the plate, and x_3 is perpendicular to the plane of the lamina, as seen on Figure 11. The orthotropic material parameters are obtained either by laboratory tests or theoretical approach.

The theoretical approach, or with other words *micromechanics approach,* determines the engineering constants based on the next assumptions:

- perfect bending exists between fibers and matrix,
- fibers are parallel and uniformly distributed throughout,
- the matrix is free of voids or microcracks and initially in a stress-free state,
- both fibers and matrix are isotropic and obeys Hooke's model,
- the applied loads are either parallel or perpendicular to the fiber direction.

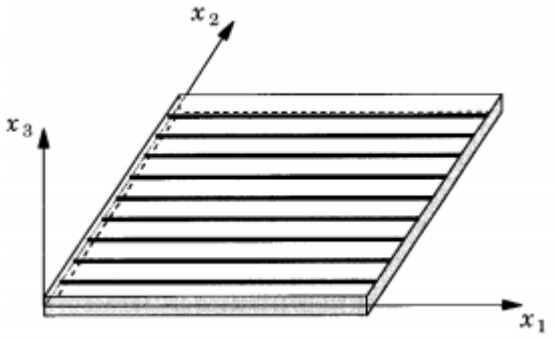


Figure 11 – Unidirectional composite lamina with the material coordinate system. (Reddy, 2004)

The elastic moduli and Poisson's ratio of a fiber-reinforced material can be expressed in terms of moduli, Poisson's ratios, and volume frictions of the constituents. To this let introduce

$E_f =$ modulus of the fiber; $E_m =$ modulus of the matrix;

$\nu_f =$ Poisson's ratio of the fiber; $\nu_m =$ Poisson's ratio of the matrix;

$v_f =$ fiber volume friction; $v_m =$ matrix volume friction.

Then the lamina engineering constants are given by

$$E_1 = E_f v_f + E_m v_m, \tag{1.79}$$

$$E_2 = \frac{E_f E_m}{E_f v_m + E_m v_f}, \tag{1.80}$$

$$\nu_{12} = \nu_f v_f + \nu_m v_m, \tag{1.81}$$

$$G_{12} = \frac{G_f G_m}{G_f v_m + G_m v_f}. \tag{1.82}$$

where E_1 is the longitudinal modulus, E_2 is transverse modulus, ν_{12} is the major Poisson's ratio, and G_{12} is the shear modulus, and

$$G_f = \frac{E_f}{2(1+\nu_f)}, \quad G_m = \frac{E_m}{2(1+\nu_m)}. \tag{1.83}$$

The 9 independent material parameters can also be determined experimentally using an appropriate test specimen made up from the material under consideration. One can found detailed instructions on the experimental arrangements and the execution method in the work of Reddy (2004). Here I enclose some values of several materials to service a basis for the later investigations.

***1. Table**: Values of engineering constants for some materials*.*

Material	E_1	E_2	G_{12}	G_{13}	G_{23}	ν_{12}
Aluminimum	10.60	10.60	3.38	3.38	3.38	0.33
Copper	18.00	18.00	6.39	6.39	6.39	0.33
Steel	30.00	30.00	11.24	11.24	11.24	0.29
Gr.-Ep (AS)	20.00	1.30	1.03	1.03	0.90	0.30
Gr.-Ep (T)	19.00	1.50	1.00	0.90	0.90	0.22
Gl.-Ep (1)	7.80	2.60	1.30	1.30	0.50	0.25
Gl.-Ep (2)	5.60	1.20	0.60	0.60	0.50	0.26
Br.-Ep	30.00	3.00	1.00	1.00	0.60	0.30

*Moduli are in msi = million psi; 1 psi= 6.894.76 N/m^2

*Gr.-Ep(AS) = graphite-epoxy (AS/3501); Gr.-Ep(T) = graphite-epoxy (T3000/934); Gl.-Ep = glass-epoxy; Br.-Ep = boron-epoxy

***2. Table**: Additional values of material constants*.*

Material	E_3	ν_{13}	ν_{23}	α_1	α_2
Aluminimum	10.60	0.33	0.33	13.10	13.10
Copper	18.00	0.33	0.33	18.00	18.00
Steel	30.00	0.29	0.29	10.00	10.00
Gr.-Ep (AS)	1.30	0.30	0.49	1.00	30.00
Gr.-Ep (T)	1.50	0.22	0.49	-0.17	15.60
Gl.-Ep (1)	2.60	0.25	0.34	3.50	11.40
Gl.-Ep (2)	1.30	0.26	0.34	4.80	12.30
Br.-Ep	3.00	0.25	0.25	2.50	8.00

*Value of E_3 is understood in msi, and α_1 and α_2 are 10^{-6} in./in./°F

Coordinate transformations

After discussing material symmetry, clarifying the definitions of a lamina and a laminate and the determination of its engineering constants, continue the train of thought about the coordinate systems and coordinate transformation.

The constitutive relations (1.68) and (1.69) were written with respect to the principal material coordinate system of the actual lamina, which in most of the situations does not coincide with the problem coordinate system. Furthermore, a laminate in general case is composed of a set of laminas with different directions of their principal material coordinate systems. Therefore there is a need to transform the relations among stresses and

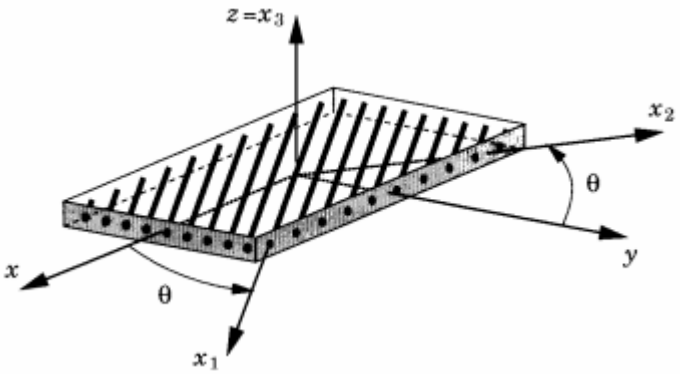


Figure 12 – A lamina with material and problem coordinate system. (Reddy, 2004)

strains for each layer into the problem coordinate system. As explained before, the x_3 axis of every lamina always coincides with the z axis of the problem coordinate system, and this causes the transformation matrix of expression (1.67). As a reminder I repeat this transformation matrix with notations of Figure 12:

$$[L] = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{1.84}$$

and the coordinates of a material point in the two coordinate systems are related as follows:

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = [L] \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}. \tag{1.85}$$

Transformation of stress components

Now we have to investigate the relationship between the stress components in the material and in the problem coordinate system. Denote the components of the stress tensor with respect to the material coordinate system with $\sigma_{11}, \sigma_{12}, \dots, \sigma_{33}$ and with $\sigma_{xx}, \sigma_{xy}, \dots, \sigma_{zz}$ in the problem coordinate system. In matrix form:

$$[\sigma]_m = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix}, \tag{1.86}$$

$$[\sigma]_p = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}. \tag{1.87}$$

According to equation (1.66) the matrices transform to each other with the next formula:

$$[\sigma]_m = [L][\sigma]_p[L]^T, \tag{1.88}$$

$$[\sigma]_p = [L]^T[\sigma]_p[L], \tag{1.89}$$

here matrix $[L]$ is the 3x3 matrix of direction cosines l_{ij}, where

$$l_{ij} = (\hat{e_i})_m \left(\hat{e_j}\right)_p. \tag{1.90}$$

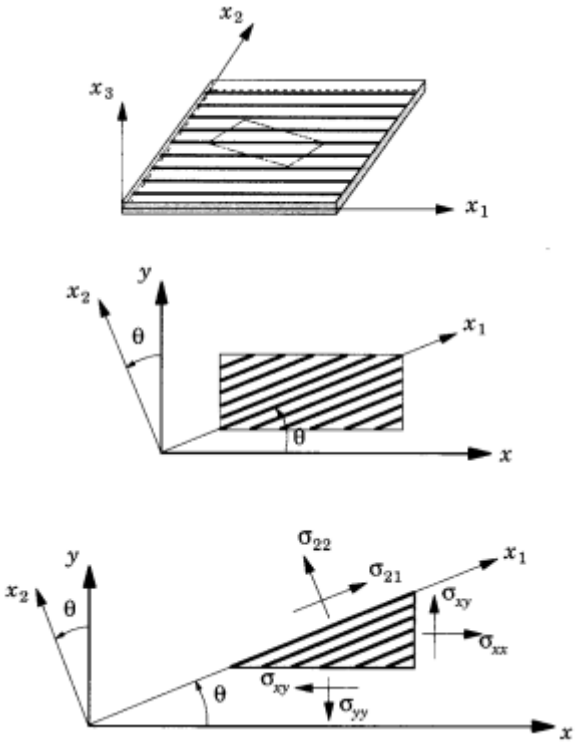


Figure 13 – A free body diagram with stress components in different coordinate systems. (Reddy, 2004)

In expression (1.90) $(\hat{e_i})_m$ and $\left(\hat{e_j}\right)_p$ are orthonormal basis vectors in the material and the problem coordinate systems respectively.

After performing the operations in expression (1.89) and rearranging to have the single subscript stress components we obtain

$$\begin{Bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \sigma_{zz}\\ \sigma_{yz}\\ \sigma_{xz}\\ \sigma_{xy}\end{Bmatrix} = \begin{bmatrix} cos^2\theta & sin^2\theta & 0 & 0 & 0 & -sin2\theta \\ sin^2\theta & cos^2\theta & 0 & 0 & 0 & sin2\theta \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & cos\theta & sin\theta & 0 \\ 0 & 0 & 0 & -sin\theta & cos\theta & 0 \\ sin\theta cos\theta & -sin\theta cos\theta & 0 & 0 & 0 & cos^2\theta - sin^2\theta \end{bmatrix} \begin{Bmatrix}\sigma_1\\ \sigma_2\\ \sigma_3\\ \sigma_4\\ \sigma_5\\ \sigma_6\end{Bmatrix}, \quad (1.91)$$

$$\{\sigma\}_p = [T]\{\sigma\}_m. \quad (1.92)$$

The inverse relationship between $\{\sigma\}_p$ and $\{\sigma\}_m$ can be obtained by substituting $-\theta$ in the place of θ in expression (1.91).

$$\begin{Bmatrix}\sigma_1\\ \sigma_2\\ \sigma_3\\ \sigma_4\\ \sigma_5\\ \sigma_6\end{Bmatrix} = \begin{bmatrix} cos^2\theta & sin^2\theta & 0 & 0 & 0 & sin2\theta \\ sin^2\theta & cos^2\theta & 0 & 0 & 0 & -sin2\theta \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & cos\theta & -sin\theta & 0 \\ 0 & 0 & 0 & sin\theta & cos\theta & 0 \\ -sin\theta cos\theta & sin\theta cos\theta & 0 & 0 & 0 & cos^2\theta - sin^2\theta \end{bmatrix} \begin{Bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \sigma_{zz}\\ \sigma_{yz}\\ \sigma_{xz}\\ \sigma_{xy}\end{Bmatrix}, \quad (1.93)$$

$$\{\sigma\}_m = [R]\{\sigma\}_p. \quad (1.94)$$

Transformation of strain components

Since the strain tensor is also second order, the expressions for the transformations formulated for the stress components are valid. Therefore

$$[\varepsilon]_m = [L][\varepsilon]_p[L]^T, \quad [\varepsilon]_p = [L]^T[\varepsilon]_p[L]. \quad (1.95)$$

Matrices $[T]$ and $[R]$ of expressions (1.92) and (1.94) are the same, so the single subscript versions can be formulated as follows

$$\begin{Bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \varepsilon_{zz}\\ 2\varepsilon_{yz}\\ 2\varepsilon_{xz}\\ 2\varepsilon_{xy}\end{Bmatrix} = \begin{bmatrix} cos^2\theta & sin^2\theta & 0 & 0 & 0 & -sin2\theta \\ sin^2\theta & cos^2\theta & 0 & 0 & 0 & sin2\theta \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & cos\theta & sin\theta & 0 \\ 0 & 0 & 0 & -sin\theta & cos\theta & 0 \\ sin\theta cos\theta & -sin\theta cos\theta & 0 & 0 & 0 & cos^2\theta - sin^2\theta \end{bmatrix} \begin{Bmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_3\\ \varepsilon_4\\ \varepsilon_5\\ \varepsilon_6\end{Bmatrix}, \quad (1.96)$$

$$\{\varepsilon\}_p = [T]\{\varepsilon\}_m, \quad (1.97)$$

and the inverse relations

$$\begin{Bmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_3\\ \varepsilon_4\\ \varepsilon_5\\ \varepsilon_6\end{Bmatrix} = \begin{bmatrix} cos^2\theta & sin^2\theta & 0 & 0 & 0 & sin2\theta \\ sin^2\theta & cos^2\theta & 0 & 0 & 0 & -sin2\theta \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & cos\theta & -sin\theta & 0 \\ 0 & 0 & 0 & sin\theta & cos\theta & 0 \\ -sin\theta cos\theta & sin\theta cos\theta & 0 & 0 & 0 & cos^2\theta - sin^2\theta \end{bmatrix} \begin{Bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \varepsilon_{zz}\\ 2\varepsilon_{yz}\\ 2\varepsilon_{xz}\\ 2\varepsilon_{xy}\end{Bmatrix}, \quad (1.98)$$

$$\{\varepsilon\}_m = [R]\{\varepsilon\}_p. \quad (1.99)$$

Transformation of material coefficients

After the stresses and strains are transformed, the only quantities left are the material coefficients which are components of a fourth order tensor. To obtain the transformation expressions we use the relations between the stresses and strains with the already known transformation formulas.

$$\{\sigma\}_p = [T]\{\sigma\}_m = [T][C]_m\{\varepsilon\}_m = [T][C]_m[T]^T\{\varepsilon\}_p = [C]_p\{\varepsilon\}_p. \quad (1.100)$$

Thus

$$[C]_p = [T][C]_m[T]^T, \quad (1.101)$$

where $[C]_p$ and $[C]_m$ are the material stiffness matrices in the "problem" and in the "material" coordinate systems. From now on I will denote the components of $[C]_m$ as C_{ij}, while $\overline{C_{ij}}$ will be the components of the problem stiffness matrix $[C]_p$. By carrying out the matrix operations of equation (1.101), the desired material coefficients can be obtained, so equations (1.73) and (1.78) turn to

$$\begin{Bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \sigma_{xy}\end{Bmatrix} = \begin{bmatrix}\bar{Q}_{11} & \bar{Q}_{12} & \bar{Q}_{16}\\ \bar{Q}_{12} & \bar{Q}_{22} & \bar{Q}_{26}\\ \bar{Q}_{16} & \bar{Q}_{26} & \bar{Q}_{66}\end{bmatrix}\begin{Bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \gamma_{xy}\end{Bmatrix}, \quad (1.102)$$

$$\begin{Bmatrix}\tau_{yz}\\ \tau_{xz}\end{Bmatrix} = \begin{bmatrix}\bar{Q}_{44} & \bar{Q}_{45}\\ \bar{Q}_{45} & \bar{Q}_{55}\end{bmatrix}\begin{Bmatrix}\gamma_{yz}\\ \gamma_{xz}\end{Bmatrix}, \quad (1.103)$$

where

$$\bar{Q}_{11} = Q_{11}cos^4\theta + 2(Q_{12} + 2Q_{66})sin^2\theta cos^2\theta + Q_{22}sin^4\theta, \quad (1.104)$$

$$\bar{Q}_{12} = (Q_{11} + Q_{22} - 4Q_{66})sin^2\theta cos^2\theta + Q_{12}(sin^4\theta + cos^4\theta), \quad (1.105)$$

$$\bar{Q}_{22} = Q_{11}sin^4\theta + 2(Q_{12} + 2Q_{66})sin^2\theta cos^2\theta + Q_{22}cos^4\theta, \quad (1.106)$$

$$\bar{Q}_{16} = (Q_{11} - Q_{12} - 2Q_{66})sin\theta cos^3\theta + (Q_{12} - Q_{22} + 2Q_{66})cos\theta sin^3\theta, \quad (1.107)$$

$$\bar{Q}_{26} = (Q_{11} - Q_{12} - 2Q_{66})cos\theta sin^3\theta + (Q_{12} - Q_{22} + 2Q_{66})sin\theta cos^3\theta, \quad (1.108)$$

$$\bar{Q}_{66} = (Q_{11} + Q_{22} - 2Q_{12} - 2Q_{66})sin^2\theta cos^2\theta + Q_{66}(sin^4\theta + cos^4\theta), \quad (1.109)$$

$$\bar{Q}_{44} = Q_{44}cos^2\theta + Q_{55}sin^2\theta, \quad (1.110)$$

$$\bar{Q}_{45} = (Q_{55} - Q_{44})\, sin\theta cos\theta, \quad (1.111)$$

$$\bar{Q}_{55} = Q_{55}cos^2\theta + Q_{44}sin^2\theta. \quad (1.112)$$

Plane stress constitutive relations

Most laminates are thin and has an approximately plane state of stress. For a lamina in the $x_1 - x_2$ plane, the transverse stress components are σ_{33}, σ_{13} and σ_{23}, as can be seen in Figure 14. Although these components are small compared to σ_{11}, σ_{22} and σ_{12}, they can lead to failures in laminated structures, hence they are weak in transverse direction. For this reason these components are not neglected in the shear theories of laminated plates. Even so, some equivalent single layer theories neglect σ_{33}, so the constitutive equations should be modified to match this fact. For homogeneous case this relations are formulated in expressions (1.72)-(1.78). These are valid for each separate lamina if we regard the indices with respect to the material coordinate system of the layer under consideration.

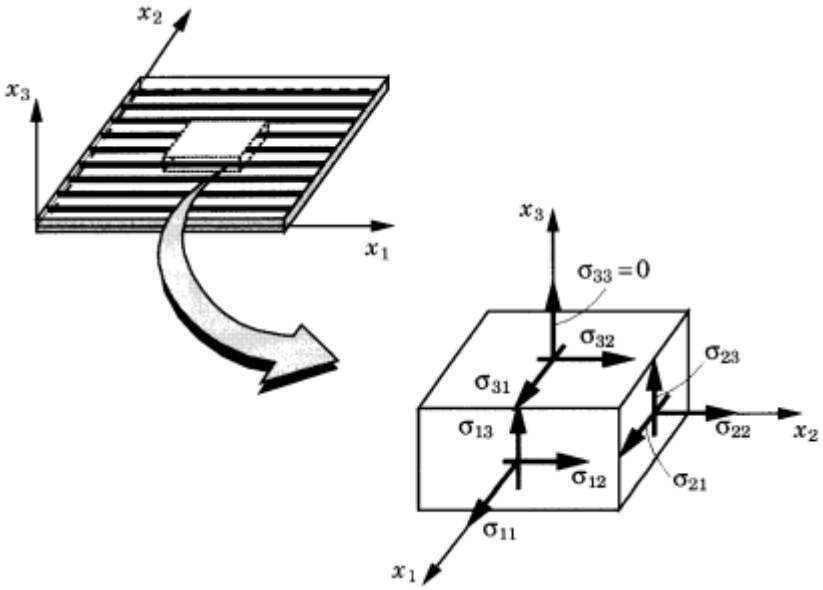


Figure 14 – The approximately plane stress state of an elementary volume in a lamina. (Reddy, 2004)

1.4. Classical theory of laminated composite plates with small strains and small rotations

In the past, analysis of composite plates have been performed on the basis of the following theories:

1) Equivalent single-layer theories (2-D)
   a) Classical laminated plate theory
   b) Shear deformation laminated plate theories
2) Three-dimensional elasticity theory (3-D)
   a) Traditional 3-D elasticity formulations
   b) Layerwise theories

The equivalent single layer plate theories (ESL) uses assumptions to decrease the dimension of the problem to 2-D. In these, a laminated plate is considered as a statically equivalent single layer having complex constitutive behavior. The simplest ESL theory is the *classical laminated plate theory*, which is an extension of the Kirchoff-Love theory, discussed in point 1.1.

Assumptions

The assumptions are identical to those discussed in point 1.1 at the Kirchoff-Love plate theory.

Displacements and strains

Let consider a laminated composite layer of total thickness h with N orthotropic layers. Each layer has its own material coordinate system in the principal direction, so the k^{th} lamina has coordinates $(x_1{}^k, x_2{}^k, x_3{}^k)$ and is oriented at angle θ_k to the laminate (problem) coordinate axis x. It is convenient to take the x-y plane in the mid-plane of the plate, furthermore let axis z to point downward. The k^{th} layer then is located vertically between $z = z^k$ and $z = z^{k+1}$.

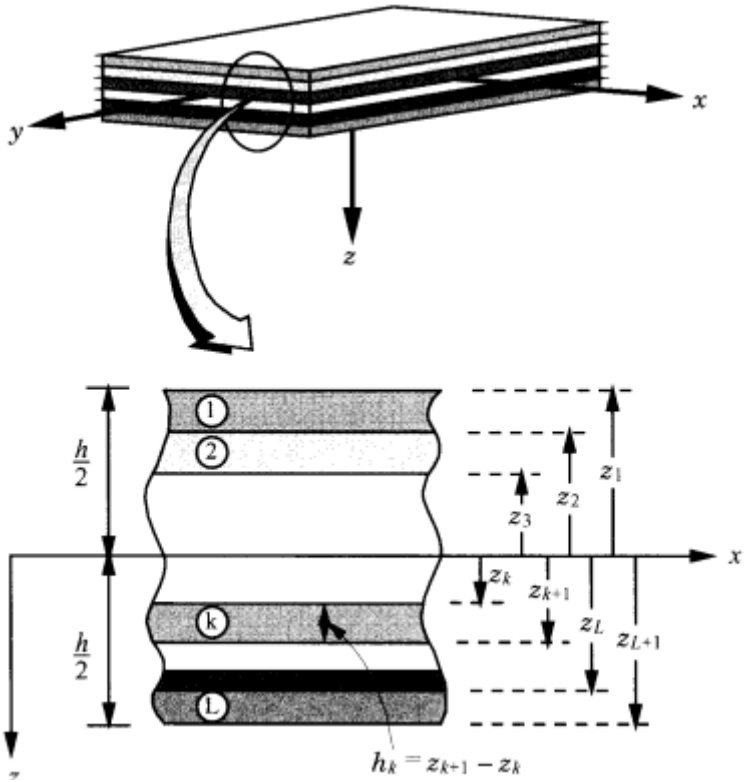


Figure 15 – Coordinate system and layer numbering of a laminate. (Reddy, 2004)

In formulating the theory we make some assumptions[6] (A) and restrictions (R) as stated below:

- The layers are perfectly bonded together (A).
- Each of the layers has a linearly elastic material behavior and has three planes of material symmetry (orthotropic) (R).
- Every layer is of uniform thickness (R).
- The strains and displacements are small compared to the total thickness of the laminate (R).
- The transverse shear stresses on the bottom and top faces of the laminate are zero (R).

The assumed displacement field is the same as before at expressions (1.7)-(1.9), so

$$u(x_0, y_0, z) = u_0(x_0, y_0) - z\frac{\partial w_0}{\partial x}, \tag{1.113}$$

$$v(x_0, y_0, z) = v_0(x_0, y_0) - z\frac{\partial w_0}{\partial y}, \tag{1.114}$$

[6] An assumption is that which is necessary to formulate a mathematical model, while a restriction is not a necessary condition for the same operation.

$$w(x_0, y_0, z) \cong w_0(x_0, y_0). \tag{1.115}$$

The strains are related to the displacement with the help of the infinitesimal strain tensor, which components are listed in the equations of expressions (1.10)-(1.15).

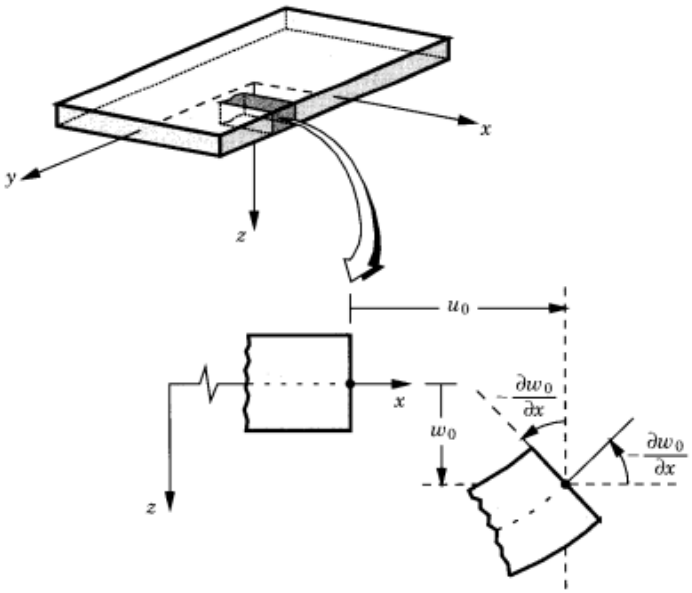


Figure 16 – Geometry of the middle plane based on Kirchoff assumptions. (Reddy, 2004)

Lamina constitutive relations

In the previous chapter I discussed in details the necessary considerations on the constitutive model of a lamina. We diagnosed, that at every lamina the constitutive relations of equations (1.72)-(1.78) should be transferred to the laminate (problem) coordinate system according to the transformation law, so in each lamina the stresses are related to the strains with equations by (1.102) and (1.103).

Note that in these expressions the coefficients $\bar{Q}_{ij}$ can vary from layer to layer, so the stress variation through the laminate thickness is not necessarily linear, even if the strain variation is linear. Typical stress and strain variations are shown in Figure 17.

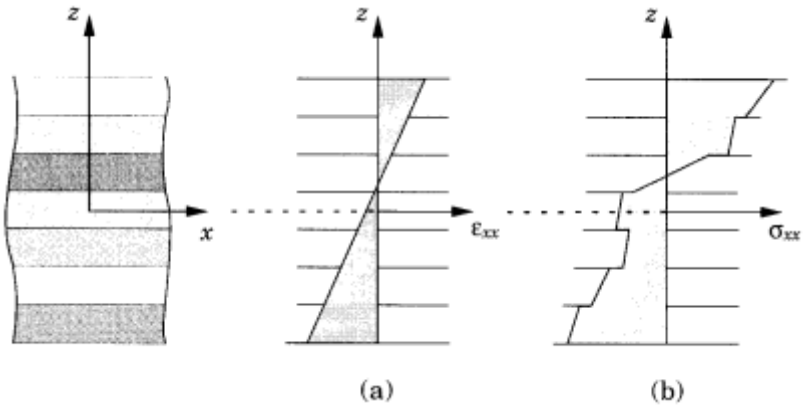


Figure 17 – Variation of stresses and strains through the layers. (Reddy, 2004)

Resultant laminate forces and moments

The calculation of the resultants can be done with expressions (1.21) and (1.23), so we have to integrate the stresses in each layer as follows

$$\{N\}_{(k)} = \begin{Bmatrix} N_x \\ N_y \\ N_{xy} \end{Bmatrix}_{(k)} = \int_{-t/2}^{t/2} \{\sigma\}_{(k)} dz, \quad \{M\}_{(k)} = \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix}_{(k)} = \int_{z_k}^{z_{k+1}} z\{\sigma\}_{(k)} dz. \tag{1.116}$$

The entire collection of force and moments resultants for an *N*-layer laminate can be obtained by summing these layer components, so

$$\{N\} = \begin{Bmatrix} N_x \\ N_y \\ N_{xy} \end{Bmatrix} = \int_{-t/2}^{t/2} \begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{Bmatrix} dz = \sum_{k=1}^{N} \int_{z_k}^{z_{k+1}} \begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{Bmatrix}_{(k)} dz, \tag{1.117}$$

$$\{M\} = \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} = \int_{-t/2}^{t/2} z \begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{Bmatrix} dz = \sum_{k=1}^{N} \int_{z_k}^{z_{k+1}} z \begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{Bmatrix}_{(k)} dz. \tag{1.118}$$

These expressions can be rearranged, if we take advantage of that the stiffness matrix is constant within a lamina. Utilizing this property and using equations (1.20), (1.102) and (1.103) we come to

$$\begin{Bmatrix} N_x \\ N_y \\ N_{xy} \end{Bmatrix} = \sum_{k=1}^{N} \begin{bmatrix} \bar{Q}_{11} & \bar{Q}_{12} & \bar{Q}_{13} \\ \bar{Q}_{12} & \bar{Q}_{22} & \bar{Q}_{23} \\ \bar{Q}_{13} & \bar{Q}_{23} & \bar{Q}_{33} \end{bmatrix}_{(k)} \left\{ \int_{z_k}^{z_{k+1}} \begin{Bmatrix} \varepsilon_{0_{xx}} \\ \varepsilon_{0_{yy}} \\ \gamma_{0_{xy}} \end{Bmatrix} dz + \int_{z_k}^{z_{k+1}} z \begin{Bmatrix} \kappa_x \\ \kappa_y \\ \kappa_{xy} \end{Bmatrix} dz \right\}, \tag{1.119}$$

$$\{N\} = \sum_{k=1}^{N} [\bar{Q}]_{(k)} \left\{ \int_{z_k}^{z_{k+1}} \{\varepsilon_0\} + z\{\kappa\}\, dz \right\}, \tag{1.120}$$

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} = \sum_{k=1}^{N} \begin{bmatrix} \bar{Q}_{11} & \bar{Q}_{12} & \bar{Q}_{13} \\ \bar{Q}_{12} & \bar{Q}_{22} & \bar{Q}_{23} \\ \bar{Q}_{13} & \bar{Q}_{23} & \bar{Q}_{33} \end{bmatrix}_{(k)} \left\{ \int_{z_k}^{z_{k+1}} z \begin{Bmatrix} \varepsilon_{0_{xx}} \\ \varepsilon_{0_{yy}} \\ \gamma_{0_{xy}} \end{Bmatrix} dz + \int_{z_k}^{z_{k+1}} z^2 \begin{Bmatrix} \kappa_x \\ \kappa_y \\ \kappa_{xy} \end{Bmatrix} dz \right\}, \tag{1.121}$$

$$\{M\} = \sum_{k=1}^{N} [\bar{Q}]_{(k)} \left\{ \int_{z_k}^{z_{k+1}} z\{\varepsilon_0\} + z^2\{\kappa\}\, dz \right\}. \tag{1.122}$$

We should now recall, that $\{\varepsilon_0\}$ and $\{\kappa\}$ are not dependent on coordinate z so they can be removed from within the summation signs. Thus the primer equation can be written as

$$\begin{Bmatrix} N_x \\ N_y \\ N_{xy} \end{Bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{16} \\ A_{12} & A_{22} & A_{26} \\ A_{16} & A_{26} & A_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_{0_{xx}} \\ \varepsilon_{0_{yy}} \\ \gamma_{0_{xy}} \end{Bmatrix} + \begin{bmatrix} B_{11} & B_{12} & B_{16} \\ B_{12} & B_{22} & B_{26} \\ B_{16} & B_{26} & B_{66} \end{bmatrix} \begin{Bmatrix} \kappa_x \\ \kappa_y \\ \kappa_{xy} \end{Bmatrix}, \tag{1.123}$$

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} = \begin{bmatrix} B_{11} & B_{12} & B_{16} \\ B_{12} & B_{22} & B_{26} \\ B_{16} & B_{26} & B_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_{0_{xx}} \\ \varepsilon_{0_{yy}} \\ \gamma_{0_{xy}} \end{Bmatrix} + \begin{bmatrix} D_{11} & D_{12} & D_{16} \\ D_{12} & D_{22} & D_{26} \\ D_{16} & D_{26} & D_{66} \end{bmatrix} \begin{Bmatrix} \kappa_x \\ \kappa_y \\ \kappa_{xy} \end{Bmatrix}. \tag{1.124}$$

In more compact form

$$\begin{Bmatrix} \{N\} \\ \{M\} \end{Bmatrix} = \begin{bmatrix} [A] & [B] \\ [B]^T & [D] \end{bmatrix} \begin{Bmatrix} \{\varepsilon_0\} \\ \{\kappa\} \end{Bmatrix}, \tag{1.125}$$

where

$$A_{ij} = \sum_{k=1}^{N} \left(\bar{Q}_{ij}\right)_{(k)} (z_{k+1} - z_k), \quad B_{ij} = \frac{1}{2} \sum_{k=1}^{N} \left(\bar{Q}_{ij}\right)_{(k)} ({z_{k+1}}^2 - {z_k}^2), \tag{1.126}$$

$$D_{ij} = \frac{1}{3} \sum_{k=1}^{N} \left(\bar{Q}_{ij}\right)_{(k)} ({z_{k+1}}^3 - {z_k}^3). \tag{1.127}$$

In expressions (1.23)-(1.27), A_{ij} are extensional stiffnesses, B_{ij} are bending-extensional stiffnesses and D_{ij} are bending stiffnesses. The presence of B_{ij} implies coupling between bending and extension of a laminate. It means that it is impossible to pull a laminate without bending and/or twisting it at the same time. Similarly, such a laminate cannot bent without suffering extension of the middle surface. This behavior is illustrated on Figure 18.

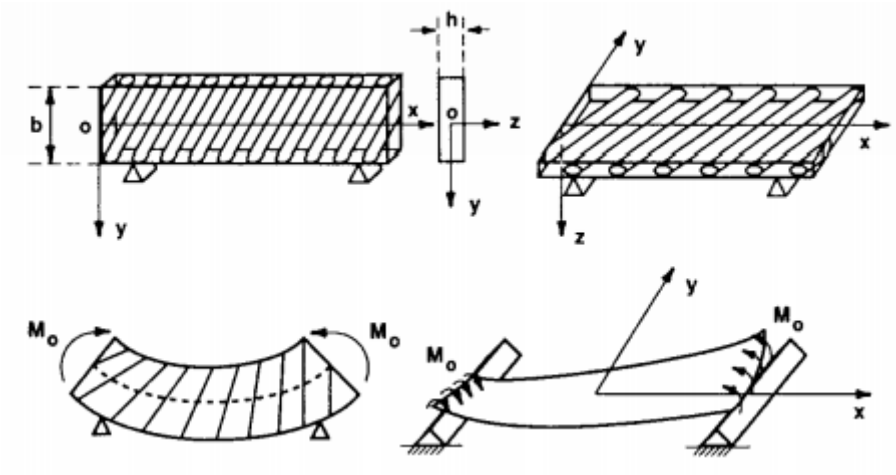


Figure 18 – Bend-twist coupling phenomenon. (Jones, 1999)

Observe how the behavior changes if the fibers are in the plate of the bending moment or opposite. On the left figure no twisting occurs when bending the beam, but if the cross section is rotated with 90°, the behavior changes.

The steps of determining the governing equations are identical to those discussed in point 1.1.

1.5. Classical theory of laminated composite plates with small strains and moderate rotations

In those cases when we count on moderate rotations (say 10°-15°), we have to modify the relations between strains and displacements. Now we can use for this purpose the nonlinear strains.

The explicit form of the six Cartesian components of the Green[7]-Lagrange[8] strain tensor are given by formulas (1.128)-(1.133). Between the second order terms I marked with red those, which can be neglected as a consequence of the small strain assumption. In case of moderate rotations some second order terms should be included in the strain-displacement relations and these are market with green color in the expressions. The first order terms were already taken into account in the Kirchoff model, these are left unchanged in color.

[7] George Green (14 July 1793 – 31 May 1841) was a British mathematical physicist. He was the first person to create a mathematical theory of electricity and magnetism, and he introduced several important concepts of today's principles in use (Wikipedia).

[8] Joseph-Louis Lagrange (25 January 1736 – 10 April 1813) was an Italian Enlightenment Era mathematician and astronomer. He made significant contributions to all fields of analysis, number theory, and both classical and celestial mechanics (Wikipedia).

$$E_{xx} = \frac{\partial u}{\partial x} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2\right], \tag{1.128}$$

$$E_{yy} = \frac{\partial v}{\partial y} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial y}\right)^2\right], \tag{1.129}$$

$$E_{zz} = \frac{\partial w}{\partial z} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial z}\right)^2 + \left(\frac{\partial v}{\partial z}\right)^2 + \left(\frac{\partial w}{\partial z}\right)^2\right], \tag{1.130}$$

$$E_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} + \frac{\partial u}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\frac{\partial v}{\partial y} + \frac{\partial w}{\partial x}\frac{\partial w}{\partial y}\right), \tag{1.131}$$

$$E_{xz} = \frac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} + \frac{\partial u}{\partial x}\frac{\partial u}{\partial z} + \frac{\partial v}{\partial x}\frac{\partial v}{\partial z} + \frac{\partial w}{\partial x}\frac{\partial w}{\partial z}\right), \tag{1.132}$$

$$E_{yz} = \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} + \frac{\partial u}{\partial y}\frac{\partial u}{\partial z} + \frac{\partial v}{\partial y}\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\frac{\partial w}{\partial z}\right). \tag{1.133}$$

Therefore for small strains and moderate rotations the strain-displacement relations take the form

$$\varepsilon_{xx} = \frac{\partial u}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2;\ \varepsilon_{yy} = \frac{\partial v}{\partial y} + \frac{1}{2}\left(\frac{\partial w}{\partial y}\right)^2;\ \varepsilon_{zz} = \frac{\partial w}{\partial z}; \tag{1.134}$$

$$\gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y};\ \gamma_{zx} = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z};\ \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} + \frac{\partial w}{\partial x}\frac{\partial w}{\partial y}. \tag{1.135}$$

Substituting here the displacements function we obtain

$$\varepsilon_{xx} = \frac{\partial u_0}{\partial x} + \frac{1}{2}\left(\frac{\partial w_0}{\partial x}\right)^2 - z\frac{\partial^2 w_0}{\partial x^2}, \tag{1.136}$$

$$\varepsilon_{yy} = \frac{\partial v_0}{\partial y} + \frac{1}{2}\left(\frac{\partial w_0}{\partial y}\right)^2 - z\frac{\partial^2 w_0}{\partial y^2}, \tag{1.137}$$

$$\varepsilon_{zz} = 0 \tag{1.138}$$

$$\gamma_{yz} = \frac{\partial}{\partial z}\left(v_0(x_0, y_0) - z\frac{\partial w_0}{\partial y}\right) + \frac{\partial w_0}{\partial y} = -\frac{\partial w_0}{\partial y} + \frac{\partial w_0}{\partial y} = 0, \tag{1.139}$$

$$\gamma_{zx} = \frac{\partial}{\partial z}\left(u_0(x_0, y_0) - z\frac{\partial w_0}{\partial x}\right) + \frac{\partial w_0}{\partial x} = -\frac{\partial w_0}{\partial x} + \frac{\partial w_0}{\partial x} = 0, \tag{1.140}$$

$$\gamma_{xy} = \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} + \frac{\partial w_0}{\partial x}\frac{\partial w_0}{\partial y} - 2z\frac{\partial^2 w_0}{\partial x \partial y}. \tag{1.141}$$

To distinguish the obtained expressions well from the Kirchoff's versions, I marked the additional terms in (1.134)-(1.141) again with green. These are present due to the not neglected (green) second order strain terms in expressions (1.128)-(1.133). In this way we can note, that the transverse strains are identically zero as there were in the classical plate theory.

The strains in (1.36)-(1.41) are the so called *von Kármán*[9] *strains*, and the associated plate theory is termed *von Kármán plate theory*.

The non-zero terms can be separated to *membrane strains* ε^0 and *flexural (bending) strains* ε^1, known as curvatures, ergo

$$\begin{Bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \gamma_{xy}\end{Bmatrix} = \begin{Bmatrix}\varepsilon_{xx}{}^{(0)}\\ \varepsilon_{yy}{}^{(0)}\\ \gamma_{xy}{}^{(0)}\end{Bmatrix} + z\begin{Bmatrix}\varepsilon_{xx}{}^{(1)}\\ \varepsilon_{yy}{}^{(1)}\\ \gamma_{xy}{}^{(1)}\end{Bmatrix} = \{\varepsilon^0\} + z\{\varepsilon^1\}, \tag{1.142}$$

$$\{\varepsilon^0\} = \begin{Bmatrix}\dfrac{\partial u_0}{\partial x} + \dfrac{1}{2}\left(\dfrac{\partial w_0}{\partial x}\right)^2\\ \dfrac{\partial v_0}{\partial y} + \dfrac{1}{2}\left(\dfrac{\partial w_0}{\partial y}\right)^2\\ \dfrac{\partial u_0}{\partial y} + \dfrac{\partial v_0}{\partial x} + \dfrac{\partial w_0}{\partial x}\dfrac{\partial w_0}{\partial y}\end{Bmatrix}, \quad \{\varepsilon^1\} = \begin{Bmatrix}-z\dfrac{\partial^2 w_0}{\partial x^2}\\ -z\dfrac{\partial^2 w_0}{\partial y^2}\\ -2z\dfrac{\partial^2 w_0}{\partial x \partial y}\end{Bmatrix}. \tag{1.143}$$

Notice that $\{\varepsilon^0\}$ and $\{\varepsilon^1\}$ are identical to the formerly used vectors $\{\varepsilon_0\}$ and $\{\kappa\}$.

The point of this theory were in the different consideration of the displacement-strain relationships. The governing equations can be obtained in the same manner as before, it can be found in the book of Reddy, see (Reddy, 2004).

1.6. The first-order laminated plate theory with small strains and small rotations

The first order laminated plate theory can be expanded the same way from the Reissner-Mindlin, as the classical laminated plate theory was expanded from the Kirchoff-Love version in point 1.4.

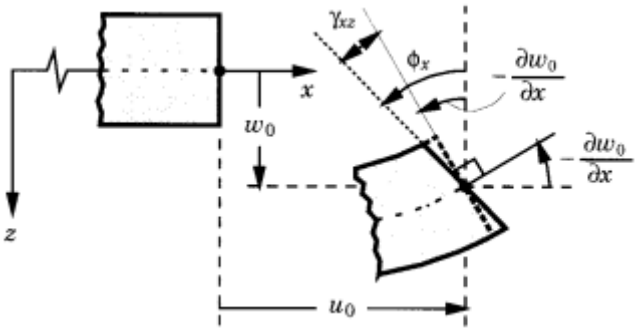


Figure 19 - Geometry of the middle plane based on the assumptions of the first-order plate theory. (Reddy, 2004)

For now it is assumed that the reader is familiar with the used coordinate systems and definitions which can be associated with a laminated composite plate. Furthermore this chapter uses the results and statements of earlier points 1.2 and 1.4, thus it is rather a summary of the already known results, than building a theory from the bottom up. I want to highlight, that this theory is from the family of single-layer theories.

Assumptions

[9] Theodore von Kármán (in Hungarian: Szőllőskislaki Kármán Tódor; May 11, 1881 – May 6, 1963). Hungarian-American mathematician, aerospace engineer and physicist who was active primarily in the fields of aeronautics and astronautics. He is regarded as the outstanding aerodynamic theoretician of the twentieth century (Wikipedia).

The assumptions are of point 1.2 supplemented with the next assumptions (A) and restrictions (R), as follows

- The layers are perfectly bonded together (A).
- Each of the layers has a linearly elastic material behavior and has three planes of material symmetry (orthotropic) (R).
- Every layer is of uniform thickness (R).
- The strains and displacements are small compared to the total thickness of the laminate (R).
- The transverse shear stresses on the bottom and top faces of the laminate are zero (R).

These additional conditions are identical to those, which were used in point 1.4.

Strains and displacements

The displacement field is the same as in point 1.2, so

$$u(x_0, y_0, z) = u_0(x_0, y_0) - z\vartheta_x, \tag{1.144}$$

$$v(x_0, y_0, z) = v_0(x_0, y_0) - z\vartheta_y, \tag{1.145}$$

$$w(x_0, y_0, z) \cong w_0(x_0, y_0). \tag{1.146}$$

Thus non-zero components of the infinitesimal strain tensor are:

$$\varepsilon_{xx} = \frac{\partial u_0}{\partial x} - z\frac{\partial \vartheta_x}{\partial x} = \varepsilon_{0_{xx}} - z\frac{\partial \vartheta_x}{\partial x} = \varepsilon_{0_{xx}} + z\,\kappa_x, \tag{1.147}$$

$$\varepsilon_{yy} = \frac{\partial v_0}{\partial y} - z\frac{\partial \vartheta_y}{\partial y} = \varepsilon_{0_{yy}} - z\frac{\partial \vartheta_y}{\partial y} = \varepsilon_{0_{yy}} + z\,\kappa_y, \tag{1.148}$$

$$\gamma_{xy} = \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} - z\frac{\partial \vartheta_x}{\partial y} - z\frac{\partial \vartheta_x}{\partial x} = \gamma_{0_{xy}} - z\frac{\partial \vartheta_x}{\partial y} - z\frac{\partial \vartheta_x}{\partial x} = \gamma_{0_{xy}} + z\,\kappa_{xy}, \tag{1.149}$$

$$\gamma_{zx} = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} = \frac{\partial w_0}{\partial x} + \frac{\partial}{\partial z}(u_0(x_0, y_0) - z\vartheta_x) = \frac{\partial w_0}{\partial x} - \vartheta_x = \gamma_x, \tag{1.150}$$

$$\gamma_{zy} = \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} = \frac{\partial w_0}{\partial y} + \frac{\partial}{\partial z}\left(v_0(x_0, y_0) - z\vartheta_y\right) = \frac{\partial w_0}{\partial y} - \vartheta_y = \gamma_y. \tag{1.151}$$

Constitutive equations

The plate constitutive equations were clarified in equations (1.25)-(1.27) and are also valid for the first-order theory, so

$$\begin{Bmatrix}\{N\}\\ \{M\}\end{Bmatrix} = \begin{bmatrix}[A] & [B]\\ [B]^T & [D]\end{bmatrix}\begin{Bmatrix}\{\varepsilon_0\}\\ \{\kappa\}\end{Bmatrix}, \tag{1.152}$$

where

$$A_{ij} = \sum_{k=1}^{N}\left(\bar{Q}_{ij}\right)_{(k)}(z_{k+1} - z_k), \quad B_{ij} = \frac{1}{2}\sum_{k=1}^{N}\left(\bar{Q}_{ij}\right)_{(k)}({z_{k+1}}^2 - {z_k}^2), \tag{1.153}$$

$$D_{ij} = \frac{1}{3}\sum_{k=1}^{N}\left(\bar{Q}_{ij}\right)_{(k)}\left({z_{k+1}}^3 - {z_k}^3\right). \tag{1.154}$$

In addition, we have the following laminate constitutive equations:

$$\begin{Bmatrix} Q_y \\ Q_x \end{Bmatrix} = K_s \begin{bmatrix} A_{44} & A_{45} \\ A_{45} & A_{55} \end{bmatrix} \begin{Bmatrix} \gamma_y \\ \gamma_x \end{Bmatrix}, \tag{1.155}$$

where

$$A_{ij} = \sum_{k=1}^{N}\left(\bar{Q}_{ij}\right)_{(k)}\left(z_{k+1} - z_k\right). \tag{1.156}$$

After this step we can consider the plate as an equivalent single layer homogeneous plate, so the next steps are completely identical to those in point 1.2. Thus the value of $5/6$ for the shear correction factor holds for this theory also.

Bending equations

The equilibrium equations of a plate are

$$\frac{\partial Q_x}{\partial x} + \frac{\partial Q_y}{\partial y} + q = 0, \tag{1.157}$$

$$\frac{\partial M_{xy}}{\partial x} - \frac{\partial M_x}{\partial y} + Q_y = 0, \tag{1.158}$$

$$\frac{\partial M_{yx}}{\partial y} + \frac{\partial M_y}{\partial x} - Q_x = 0. \tag{1.159}$$

By substituting the expressions for the curvatures and the out of plane shear stresses from (1.46)-(1.51) into the constitutive equations (1.52) and (1.55), and put this all into the equilibrium equations (1.57)-(1.59) we obtain five separate second order differential equations. The complete form of these equations can be found in the book of Reddy (2004). Because of the limits of the present thesis **we have to restrict our investigation to simpler cases, when some of the rigidities of expression** (1.52) **and** (1.55) **are zero**[10].

1.7. Other theories for laminated composite plates

In the spirit of the single-layer theory, higher order displacement fields can be assumed to predict more accurately the behavior of multi layered moderately thick composite plates. It is to note, that in case of a third-order shear theory the displacement field accommodates cubic variations of transverse shear strain, so there is no more need to use a shear correction factor. Needless to say, these theories cannot be discussed in the framework of this thesis.

Unlike the single-layer theories, the layerwise theories assume a unique displacement field for each layer. Again, the order of the displacement fields can vary.

[10] We will continue this thought in chapter 2, where a solution will be presented for specific lamination schemes.

The most accurate results for stresses and strains can be obtained by three dimensional elastic models

1.8. Theory of sandwich plates

A sandwich type construction refers to laminates that consist of **two high strength layers on the outer faces** and a **relatively weak core between** them. This composition produces usually high bending stiffness with low weight.

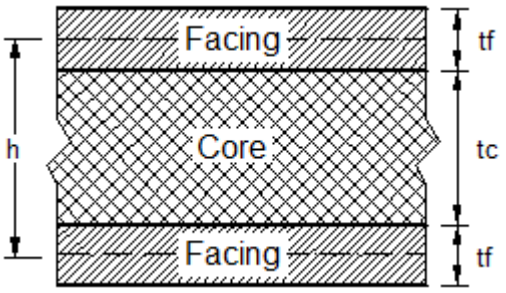


Figure 20 – Geometry of a sandwich plate.

Several theories exists for the bending of flat sandwich plates. These theories differ in how the shear deformations of the core are accounted for. Also, certain stiffnesses of energy contributions may be neglected or taken into account and other idealizations can be made. Therefore the derived formulas can vary greatly in complexity. However it can be shown, that the results from different theories are essentially the same in magnitude. In this point I will introduce a theory developed by Libove and Batdorf in the book of Plantema (1996). This theory assumes, that *the transverse normal stiffness of the plate is infinite and the faces can be considered as membranes*. Furthermore the normal stiffness of the core in x-direction, parallel to the faces, is usually small compared with the nornal stiffness of the faces. For this reason it is assumed, that *the core carries no longitudinal normal stresses*. For practical purposes these assumptions are justified, except in extreme cases, when the wrinkling phenomena occurs. These extreme situations are not the object of the thesis.

Similarly to what was seen in point 1.6, the theory is based on the Mindlin-Reissner plate theory and again we use a statically equivalent layer to substitute the sandwich layers. Therefore the equivalent plate rigidities are needed to use the stress-strain relations of expressions

$$\begin{Bmatrix}\{N\}\\ \{M\}\end{Bmatrix} = \begin{bmatrix}[A] & [B]\\ [B]^T & [D]\end{bmatrix}\begin{Bmatrix}\{\varepsilon_0\}\\ \{\kappa\}\end{Bmatrix}, \tag{1.160}$$

$$\begin{Bmatrix}Q_x\\ Q_y\end{Bmatrix} = K_s\begin{bmatrix}Gt & 0\\ 0 & Gt\end{bmatrix}\begin{Bmatrix}\gamma_x\\ \gamma_y\end{Bmatrix}. \tag{1.161}$$

Note that from the above equations, the first one is not needed to formulate the governing equations. The rigidities for the remaining expressions are formulated in (Plantema, 1996) for sandwich plates with *faces of equal thickness and material properties*, consequently without flexural-extensional coupling. Thus

$$D_{ij} = \int_{-t/2}^{t/2} Q_{ij} z^2 dz \qquad (i,j = 1,2,6), \tag{1.162}$$

$$S_{ij} = \int_{-t/2}^{t/2} Q_{ij} dz \qquad (i,j = 4,5). \tag{1.163}$$

Furthermore, every rigidity has a contribution from the core and the faces also. As a consequence of the introduced idealizations, it is usual to *ignore the contribution of the*

core to the flexural rigidities and the contribution of the faces to the shear rigidity. In the following formulas the ignored terms are written in red. Thus

$$D_{11} = \int_{-t_c/2}^{t_c/2} \frac{E_c z^2}{2(1-\nu_c{}^2)} dz + \int_{t_c/2}^{\frac{t_c}{2}+t_f} \frac{E_f z^2}{2(1-\nu_f{}^2)} dz + \int_{-\frac{t_c}{2}-t_f}^{-t_c/2} \frac{E_f z^2}{2(1-\nu_f{}^2)} dz, \quad (1.166)$$

$$D_{11} = \frac{E_f t_f (t_c{}^2 + 2t_c t_f + 4t_f{}^2/3)}{4(1-\nu_f{}^2)} + \frac{E_c t_c{}^3}{24(1-\nu_f{}^2)} = D_{11}{}^f + D_{11}{}^c, \quad (1.167)$$

$$S_{44} = S_{55} = \int_{-t_c/2}^{t_c/2} G_c \, dz + \int_{t_c/2}^{\frac{t_c}{2}+t_f} G_f \, dz + \int_{-\frac{t_c}{2}-t_f}^{-t_c/2} G_f \, dz, \quad (1.168)$$

$$S_{44} = S_{55} = G_c t_c + 2G_f t_f = S_{44}{}^c + S_{44}{}^f, \quad (1.169)$$

$$D_{12} = \nu_c D_{11}{}^c + \nu_f D_{11}{}^f, \quad (1.170)$$

$$D_{66} = (1-\nu_c) D_{11}{}^c/2 + (1-\nu_f) D_{11}{}^f/2. \quad (1.171)$$

In the expressions t_c and t_f are thicknesses of the core and the faces (Figure 20), E_c and E_f are the elastic moduli and ν_c and ν_f are the Poisson's ratios of the core and the flange respectively.

From now on we can consider the plate as any homogeneous isotropic layer, so the steps of determining the governing equations are identical to those discussed in point 1.1.

1.9. Theory of voided plates

A similar single layer consideration can be applied to voided plates as seen at laminated and sandwich constructions in the previous chapters. Again the point is in the plate constitutive relations, so equivalent plate rigidities has to be determined to be able to treat the plate as a homogeneous isotropic one.

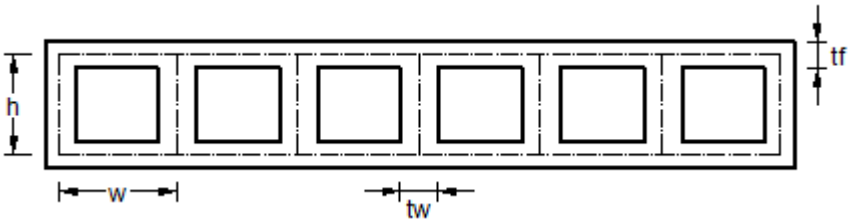


Figure 21 - Section of a voided plate.

The expressions for the evaluation of plate rigidities suggested by Basu and Dawson in (Basu et al, 1970) are now summarized.

To calculate the transverse flexural rigidity D_{11}, only the contribution of the flanges are considered, thus

$$D_{11} = \frac{E t_f h^2}{2(1-\nu^2)}. \quad (1.172)$$

In the longitudinal flexural rigidity D_{12}, the core and the flanges are considered either, so

$$D_{22} = D_{11}\left(1 + \left(\frac{t_w h}{t_f w}\right)\right). \tag{1.173}$$

The coupling rigidity D_{12} and the torsional rigidity D_{66}

$$D_{12} = D_{21} = \nu D_{11}, \qquad D_{66} = \frac{G t_f h^2}{2}. \tag{1.174}$$

The transverse shear rigidity S_{44} is provided by the flexure of the flange and the webs in the transverse cross section as if it were a Vierendeel-type girder, and is obtained by the assumption that there is an inflexion point between the webs:

$$S_{44} = \frac{2E{t_f}^3}{w^2(1 + 2(h/w)(t_f/t_w)^3)}. \tag{1.175}$$

The longitudinal shearing rigidity is obtained on the assumption, that the vertical shear force is resisted only by the webs and the distribution of the shear stresses in the webs is uniform. Therefore

$$S_{55} = G t_f h(1 + t_f/h)/(t_f w/t_w). \tag{1.176}$$

The meaning of the geometrical parameters can be seen on Figure 21.

1.10. Plates on elastic foundation

Many engineering problems can be related to the solution of plates resting on elastic foundation. To simplify this complex problem, let us assume that the supporting medium is isotropic, homogeneous and linearly elastic. Such a type of sub base is called a Winkler-type foundation. The foundation's reaction $q^*(x,y)$ can be calculated by the relationship

$$q^*(x,y) = kw, \tag{1.177}$$

where k represents the bedding constant (in kilonewtons per cubic centimeter) of the foundation material. When a plate is supported by such a foundation, the external load $q(x,y)$ is extended to be $q(x,y) - q^*(x,y) = kw$. Thus the equilibrium equation of vertical forces from expression (Owen et al, 1986) changes to

$$\frac{\partial Q_x}{\partial x} + \frac{\partial Q_y}{\partial y} + q - q^* = 0. \tag{1.178}$$

By substituting expression (70) we obtain

$$\frac{\partial Q_x}{\partial x} + \frac{\partial Q_y}{\partial y} - kw = -q. \tag{1.179}$$

This modification means no more than an additional term in the differential equations, actually in the first. As an example, if we consider the differential equations (1.55)-(1.57) of point 1.2, with this modification the final form of the three differential equation of a Mindlin plate becomes

$$K_s G t\left(\frac{\partial^2 w}{\partial x^2} - \frac{\partial \vartheta_x}{\partial x}\right) + K_s G t\left(\frac{\partial^2 w}{\partial y^2} - \frac{\partial \vartheta_y}{\partial y}\right) - kw = -p(x,y), \tag{1.180}$$

$$K_s Gt\left(\frac{\partial w}{\partial x}-\vartheta_x\right)=-\overline{D}\frac{\partial^2\vartheta_x}{\partial x^2}-\nu\overline{D}\frac{\partial^2\vartheta_y}{\partial x\partial y}-\frac{1-\nu}{2}\overline{D}\left(\frac{\partial\vartheta_y}{\partial y\partial x}+\frac{\partial^2\vartheta_x}{\partial y^2}\right), \quad (1.181)$$

$$K_s Gt\left(\frac{\partial w}{\partial y}-\vartheta_y\right)=-\overline{D}\frac{\partial^2\vartheta_y}{\partial x^2}-\nu\overline{D}\frac{\partial^2\vartheta_x}{\partial x\partial y}-\frac{1-\nu}{2}\overline{D}\left(\frac{\partial\vartheta_x}{\partial y\partial x}+\frac{\partial^2\vartheta_y}{\partial y^2}\right). \quad (1.182)$$

The additional term is marked with green.

2. NUMERICAL SOLUTION

In chapter 1 we introduced the governing equations of plate problems. Now we will continue with the **solution of Mindlin plates** with various cross sections such as homogeneous, sandwich, laminated and voided. The latter three is considered as structures with complex behavior, but substituted with a **statically equivalent single layer** by equivalent plate rigidities, discussed in the previous chapters. Thus we only have to find a solution for homogeneous isotropic plates. Ernest Hinton provided a solution for such a case in (Owen et al, 1986). He also formulated a software in FORTRAN77 computer language capable of calculating rectangular Mindlin plates with simply supported edges resting on elastic Winkler foundation.

2.1. Theoretical background

Although the governing equations are ready, small modifications are needed according to the book of Hinton, these will be summarized now.

Assumptions

The assumptions are identical to those introduced at the Mindlin-Reissner theory in point 1.2.

Strains and displacements

Let consider the case when the mid plane is in the x-y plane and the z axis is pointing downwards. If displacements of the mid-plane due to membranal deformations are not taken into account, the displacement field modifies to

$$u(x_0, y_0, z) = z\vartheta_x, \tag{2.1}$$

$$v(x_0, y_0, z) = z\vartheta_y, \tag{2.2}$$

$$w(x_0, y_0, z) \cong \mathrm{w}(x_0, y_0). \tag{2.3}$$

As a consequence, the elements of the small strain tensor for this case are

$$\varepsilon_{xx} = z\frac{\partial \vartheta_x}{\partial x} = z\,\kappa_x, \tag{2.4}$$

$$\varepsilon_{yy} = z\frac{\partial \vartheta_y}{\partial y} = z\,\kappa_y, \tag{2.5}$$

$$\gamma_{xy} = z\frac{\partial \vartheta_x}{\partial y} + z\frac{\partial \vartheta_x}{\partial x} = z\,\kappa_{xy}, \tag{2.6}$$

$$\gamma_{xz} = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} = \frac{\partial w}{\partial x} + \frac{\partial}{\partial z}(z\vartheta_x) = \frac{\partial w}{\partial x} + \vartheta_x = \gamma_x, \tag{2.7}$$

$$\gamma_{yz} = \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} = \frac{\partial w}{\partial y} + \frac{\partial}{\partial z}(z\vartheta_y) = \frac{\partial w}{\partial y} + \vartheta_y = \gamma_y, \tag{2.8}$$

or in matrix form

$$\{\varepsilon\} = z\{\kappa\} \quad and \quad \{\gamma\} = \begin{Bmatrix}\gamma_y \\ \gamma_x\end{Bmatrix}. \tag{2.9}$$

Constitutive equations

The stress-strain relations are

$$\{\sigma\} = \begin{Bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \sigma_{xy}\end{Bmatrix} = \begin{bmatrix}\bar{Q}_{11} & \bar{Q}_{12} & \bar{Q}_{16}\\ \bar{Q}_{12} & \bar{Q}_{22} & \bar{Q}_{26}\\ \bar{Q}_{16} & \bar{Q}_{26} & \bar{Q}_{66}\end{bmatrix}\{\varepsilon\}, \tag{2.10}$$

$$\{\tau\} = \begin{Bmatrix}\tau_{yz}\\ \tau_{xz}\end{Bmatrix} = \begin{bmatrix}\bar{Q}_{44} & \bar{Q}_{45}\\ \bar{Q}_{45} & \bar{Q}_{55}\end{bmatrix}\{\gamma\}, \tag{2.11}$$

and the plate constitutive equations can be written as

$$\{N\} = \begin{Bmatrix}N_x\\ N_y\\ N_{xy}\end{Bmatrix} = \int_{-t/2}^{t/2}\{\sigma\}dz = \int_{-t/2}^{t/2}[Q]z\{\kappa\}dz = \left[[Q]\frac{z^2}{2}\{\kappa\}\right]_{-t/2}^{t/2} = 0, \tag{2.12}$$

$$\{M\} = \begin{Bmatrix}M_x\\ M_y\\ M_{xy}\end{Bmatrix} = \int_{-t/2}^{t/2}\{\sigma\}zdz = \int_{-t/2}^{t/2}[Q]z^2\{\kappa\}dz = \left[[Q]\frac{z^3}{3}\{\kappa\}\right]_{-t/2}^{t/2} = [Q]\frac{t^2}{12}\{\kappa\}, \tag{2.13}$$

$$\begin{Bmatrix}M_x\\ M_y\\ M_{xy}\end{Bmatrix} = [Q]\frac{t^2}{12}\begin{Bmatrix}\kappa_x\\ \kappa_y\\ \kappa_{xy}\end{Bmatrix} = \begin{bmatrix}D_{11} & D_{12} & D_{16}\\ D_{12} & D_{22} & D_{26}\\ D_{16} & D_{26} & D_{66}\end{bmatrix}\begin{Bmatrix}\kappa_x\\ \kappa_y\\ \kappa_{xy}\end{Bmatrix}, \tag{2.14}$$

$$\{Q_s\} = \begin{Bmatrix}Q_y\\ Q_x\end{Bmatrix} = \int_{-t/2}^{t/2}K_s\{\tau\}dz = \int_{-t/2}^{t/2}K_s\begin{bmatrix}\bar{Q}_{44} & \bar{Q}_{45}\\ \bar{Q}_{45} & \bar{Q}_{55}\end{bmatrix}\{\gamma\}dz = [S]\{\gamma\}, \tag{2.15}$$

$$\begin{Bmatrix}Q_y\\ Q_x\end{Bmatrix} = \begin{bmatrix}S_{44} & S_{45}\\ S_{45} & S_{55}\end{bmatrix}\begin{Bmatrix}\gamma_y\\ \gamma_x\end{Bmatrix}. \tag{2.16}$$

Here we can substitute the expressions for the terms of $\begin{Bmatrix}\kappa_x\\ \kappa_y\\ \kappa_{xy}\end{Bmatrix}$ and $\begin{Bmatrix}\gamma_y\\ \gamma_x\end{Bmatrix}$ from the strain-displacement relations. Thus we have

$$\begin{Bmatrix}M_x\\ M_y\\ M_{xy}\end{Bmatrix} = \begin{bmatrix}D_{11} & D_{12} & D_{16}\\ D_{12} & D_{22} & D_{26}\\ D_{16} & D_{26} & D_{66}\end{bmatrix}\begin{Bmatrix}\dfrac{\partial\vartheta_x}{\partial x}\\ \dfrac{\partial\vartheta_y}{\partial y}\\ \dfrac{\partial\vartheta_x}{\partial y}+\dfrac{\partial\vartheta_x}{\partial x}\end{Bmatrix}, \tag{2.17}$$

and

$$\begin{Bmatrix}Q_y\\ Q_x\end{Bmatrix} = \begin{bmatrix}S_{44} & S_{45}\\ S_{45} & S_{55}\end{bmatrix}\begin{Bmatrix}\dfrac{\partial w}{\partial y}+\vartheta_y\\ \dfrac{\partial w}{\partial x}+\vartheta_x\end{Bmatrix}. \tag{2.18}$$

It is important, that in the equations (2.16) and (2.18), the shear correction factor is included in terms S_{ij}.

Equilibrium and governing equations

The equilibrium equations for a Mindlin plate resting on elastic Winkler foundation of modulus k are

$$\frac{\partial Q_x}{\partial x} + \frac{\partial Q_y}{\partial y} + q + kw = 0, \tag{2.19}$$

$$\frac{\partial M_{xy}}{\partial x} - \frac{\partial M_x}{\partial y} + Q_y = 0, \tag{2.20}$$

$$\frac{\partial M_{yx}}{\partial y} + \frac{\partial M_y}{\partial x} - Q_x = 0. \tag{2.21}$$

If we substitute here the previous expressions for $\{M\}$ and $\{Q_s\}$ we obtain the three separate differential equations for a Mindlin plate. Before doing so I must declare, that we **restrict our object** to those especially **orthotropic plates**, where rigidities $D_{16}, D_{61}, D_{26}, D_{62}$ and S_{45}, S_{54} are zero. Considering these, the governing equations takes the following shape

$$S_{55}\frac{\partial \vartheta_x}{\partial x} + S_{55}\frac{\partial^2 w}{\partial x^2} + S_{44}\frac{\partial \vartheta_y}{\partial y} + S_{44}\frac{\partial^2 w}{\partial y^2} + q + kw = 0, \tag{2.22}$$

$$D_{11}\frac{\partial^2 \vartheta_x}{\partial x^2} + D_{66}\frac{\partial^2 \vartheta_x}{\partial y^2} + (D_{12} + D_{66})\frac{\partial^2 \vartheta_y}{\partial x \partial y} - S_{55}\vartheta_x - S_{55}\frac{\partial w}{\partial x} = 0, \tag{2.23}$$

$$(D_{12} + D_{66})\frac{\partial^2 \vartheta_x}{\partial x \partial y} + D_{66}\frac{\partial^2 \vartheta_y}{\partial x^2} + D_{22}\frac{\partial^2 \vartheta_y}{\partial y^2} - S_{44}\vartheta_y - S_{44}\frac{\partial w}{\partial y} = 0. \tag{2.24}$$

Solution

Now a closed form solution will be presented given by Dobyns, who employed the Navier approach to solve equations (2.22)-(2.24) for a composite plate **simply supported** on **all four edges** subjected to any lateral load. The solution is concerned with plates of uniform thickness and dimensions a and b. For such a plate the boundary conditions are

$$w = \frac{\partial \vartheta_x}{\partial x} = 0 \quad at \quad x = 0, a; \tag{2.25}$$

$$w = \frac{\partial \vartheta_y}{\partial y} = 0 \quad at \quad y = 0, b. \tag{2.26}$$

In the closed form solution, w, ϑ_x and ϑ_y are thought to satisfy equations (2.22)-(2.24) and (2.25)-(2.26). Thus

$$\vartheta_x = A_{mn}\cos\left(\frac{m\pi x}{a}\right)\sin\left(\frac{n\pi y}{b}\right), \tag{2.27}$$

$$\vartheta_y = B_{mn}\sin\left(\frac{m\pi x}{a}\right)\cos\left(\frac{n\pi y}{b}\right), \tag{2.28}$$

$$w = C_{mn}\sin\left(\frac{m\pi x}{a}\right)\sin\left(\frac{n\pi y}{b}\right), \tag{2.29}$$

and the loading function is given as

$$q = q_{mn} \sin\left(\frac{m\pi x}{a}\right) \sin\left(\frac{n\pi y}{b}\right), \tag{2.30}$$

in which q_{mn} are the Fourier coefficients of the applied load.

By substituting (2.27)-(2.30) into the governing equations (2.22)-(2.24), we obtain the matrix equation

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{12} & P_{22} & P_{23} \\ P_{13} & P_{23} & P_{33} \end{bmatrix} \begin{Bmatrix} A_{mn} \\ B_{mn} \\ C_{mn} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ q_{mn} \end{Bmatrix}, \tag{2.31}$$

where

$$P_{11} = D_{11}\left(\frac{m\pi}{a}\right)^2 + D_{66}\left(\frac{n\pi}{b}\right)^2 + S_{55}, \tag{2.32}$$

$$P_{12} = (D_{12} + D_{66})\left(\frac{m\pi}{a}\right)\left(\frac{n\pi}{b}\right), \tag{2.33}$$

$$P_{13} = S_{55}\left(\frac{m\pi}{a}\right), \tag{2.34}$$

$$P_{22} = D_{66}\left(\frac{m\pi}{a}\right)^2 + D_{22}\left(\frac{n\pi}{b}\right)^2 + S_{44}, \tag{2.35}$$

$$P_{23} = S_{44}\left(\frac{n\pi}{b}\right), \tag{2.36}$$

$$P_{33} = S_{55}\left(\frac{m\pi}{a}\right)^2 + S_{44}\left(\frac{n\pi}{b}\right)^2 + k, \tag{2.37}$$

After solving this equation for the coefficients A_{mn}, B_{mn} and C_{mn} we obtain the following expressions

$$A_{mn} = \frac{(P_{12}P_{23} - P_{22}P_{13})q_{mn}}{|P|}, \tag{2.38}$$

$$B_{mn} = \frac{(P_{12}P_{13} - P_{11}P_{23})q_{mn}}{|P|}, \tag{2.39}$$

$$C_{mn} = \frac{(P_{11}P_{22} - {P_{12}}^2)q_{mn}}{|P|}, \tag{2.40}$$

where $|P|$ is the determinant of matrix P in expression (2.31).

With these coefficients it is possible to calculate the displacements, curvatures (hence bending moments) and shear forces at any point of the plate by the expressions

$$w = \sum_m \sum_n C_{mn} \sin\left(\frac{m\pi x}{a}\right) \sin\left(\frac{n\pi y}{b}\right), \tag{2.41}$$

$$\kappa_x = -\sum_m \sum_n A_{mn}\left(\frac{\mathrm{m\pi}}{\mathrm{a}}\right) \cos\left(\frac{m\pi x}{a}\right) \sin\left(\frac{n\pi y}{b}\right), \tag{2.42}$$

$$\kappa_y = -\sum_m \sum_n B_{mn}\left(\frac{\mathrm{n\pi}}{\mathrm{b}}\right) \sin\left(\frac{m\pi x}{a}\right) \cos\left(\frac{n\pi y}{b}\right), \tag{2.43}$$

$$\kappa_{xy} = -\sum_m \sum_n \left(\left(A_{mn}\left(\frac{\mathrm{n\pi}}{\mathrm{b}}\right) + B_{mn}\left(\frac{\mathrm{m\pi}}{\mathrm{a}}\right)\right) \cos\left(\frac{m\pi x}{a}\right) \cos\left(\frac{n\pi y}{b}\right), \tag{2.44}$$

$$Q_x = \sum_m \sum_n S_{55} \left(C_{mn} \left(\frac{\mathrm{m\pi}}{\mathrm{a}} \right) + A_{mn} \right) \cos\left(\frac{m\pi x}{a} \right) \sin\left(\frac{n\pi y}{b} \right), \tag{2.45}$$

$$Q_y = \sum_m \sum_n S_{44} \left(C_{mn} \left(\frac{\mathrm{n\pi}}{\mathrm{b}} \right) + B_{mn} \right) \sin\left(\frac{m\pi x}{a} \right) \cos\left(\frac{n\pi y}{b} \right), \tag{2.46}$$

where the summation implies from $m = 1$ to ∞.

Fourier series representation of the loads

Equation (2.31) requires the loading function to be transformed into purely sinusoidal Fourier series. If the lateral load $q(x, y)$ in equations (2.22)-(2.24) is distributed over the entire lateral surface, then the Euler coefficient, q_{mn} is found to be

$$q_{mn} = \left(\frac{4}{ab} \right) \int_0^a \int_0^b q(x, y) \sin\left(\frac{m\pi x}{a} \right) \sin\left(\frac{n\pi y}{b} \right) dxdy. \tag{2.47}$$

The following expressions are checked with MATLAB and the Fourier representations can be seen on the corresponding figures, each represented with different number of Fourier terms. The first three kind were given in (Owen et al, 1986), while the pyramid load is my own development.

- For a uniform load $q(x, y) = q$:

$$q_{mn} = \left(\frac{4q}{mn\pi^2} \right) (1 - cosm\pi)(1 - cosn\pi). \tag{2.48}$$

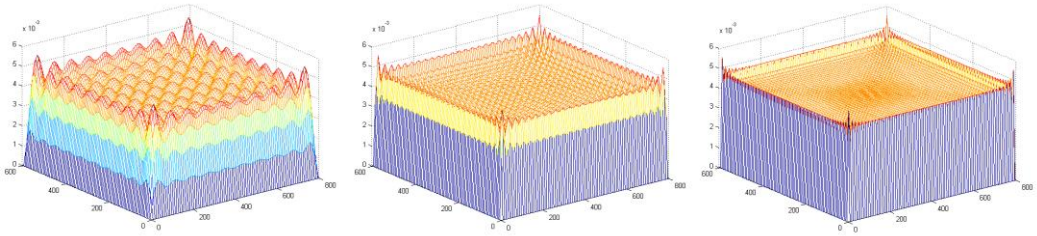


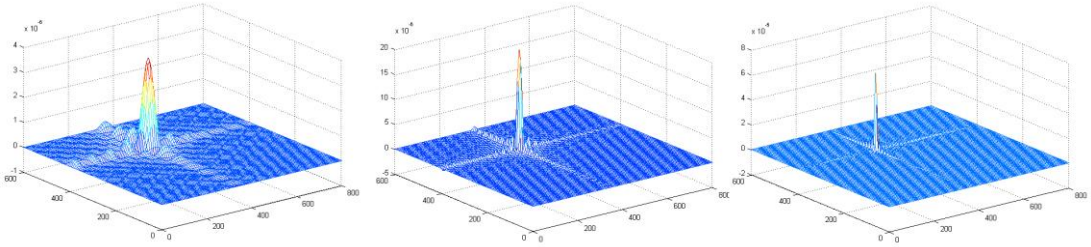
Figure 22 – Fourier representation of a uniformly distributed load with 20, 50 and 100 terms included.

- For a concentrated load P located at $x = \xi$ and $y = \eta$:

$$q_{mn} = \left(\frac{4P}{ab} \right) \sin\left(\frac{m\pi\xi}{a} \right) \sin\left(\frac{\mathrm{n\pi\eta}}{\mathrm{b}} \right). \tag{2.49}$$

- For loads of total intensity P over a rectangular area of side lengths u and v whose center is at $x = \xi$ and $y = \eta$, q_{mn} is given as follows

$$q_{mn} = \left(\frac{16P}{\pi^2 mn} \right) \sin\left(\frac{m\pi\xi}{a} \right) \sin\left(\frac{n\pi\eta}{b} \right) \sin\left(\frac{m\pi u}{2a} \right) \sin\left(\frac{n\pi v}{2b} \right). \tag{2.50}$$



23. Figure – Fourier representation of a concentrated load with 20, 50 and 100 terms included.

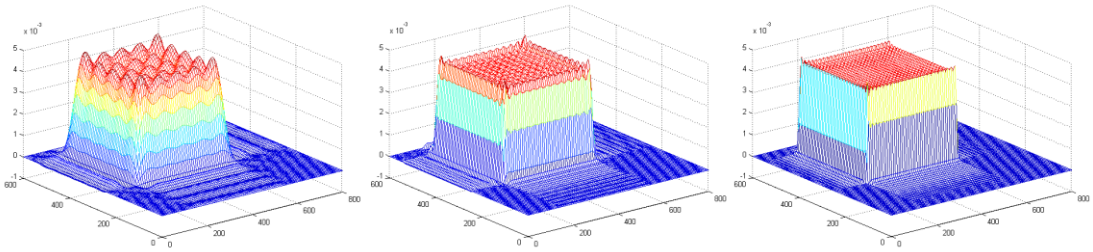
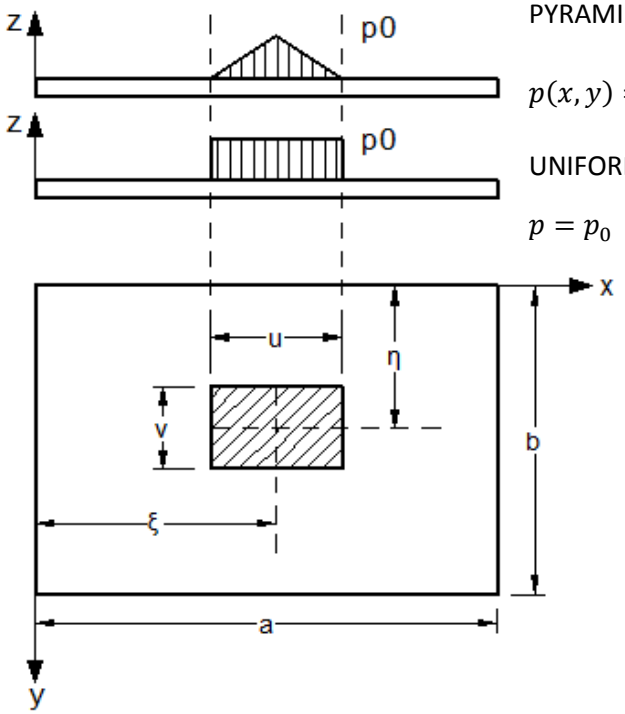


Figure 25 – Fourier representation of a rectangular patch load with 20, 50 and 100 terms included.

➲ In case of a pyramid load of total intensity P acting over a rectangular patch u, v with center located at $x = \xi$ and $y = \eta$:



PYRAMID LOAD

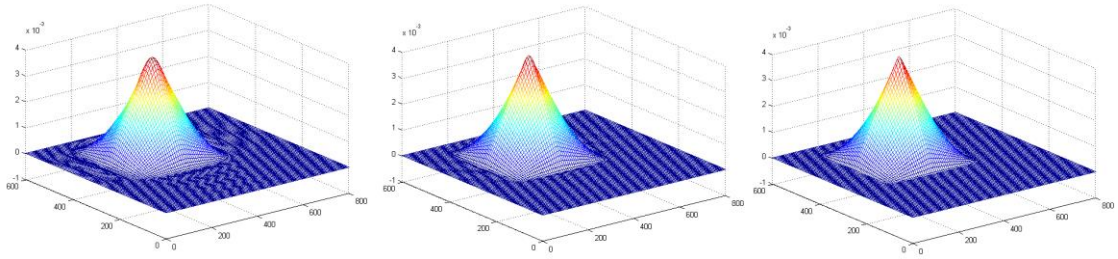
$$p(x,y) = p_0 \cdot \left(1 - \frac{2}{u} \cdot |\xi - x|\right) \cdot \left(1 - \frac{2}{v} \cdot |\eta - y|\right)$$

UNIFORM LOAD

$$p = p_0$$

Figure 24 – Notations of a pyramid and a rectangular patch load.

$$q_{mn} = \frac{128 \cdot P \cdot ab}{\pi^4 \cdot m^2 \cdot n^2 \cdot u \cdot v} \sin\left(\frac{\pi \xi m}{a}\right) \sin\left(\frac{\pi \eta n}{b}\right) \sin\left(\frac{\pi m u}{4a}\right)^2 \left[1 - cos\left(\frac{\pi n v}{2b}\right)\right]. \qquad (2.51)$$



26. Figure – Fourier representation of a pyramid patch load with 20, 50 and 100 terms included.

Of course, any lateral load can be characterized by the use of equation (2.47).

Plate rigidities

The evaluation of plate rigidities was carried out in the previous chapters for homogeneous isotropic, sandwich, laminated and voided plates. In all cases, special orthotropy with respect to the x and y axes is assumed. A short summary is presented now about these stiffness parameters.

a) Homogeneous isotropic plates

$$D_{11} = D_{22} = \frac{Et^3}{12(1-\nu^2)}, \tag{2.52}$$

$$D_{12} = D_{21} = \nu D_{11}, \tag{2.53}$$

$$D_{66} = \frac{(1-\nu)D_{11}}{2}, \tag{2.54}$$

$$S_{44} = S_{55} = \frac{K_s Et}{(2+2\nu)}. \tag{2.55}$$

b) Sandwich Plates

$$D_{11} = D_{22} = \frac{E_f t_f\left({t_c}^2+2t_c t_f+\frac{4{t_f}^2}{3}\right)}{4\left(1-{\nu_f}^2\right)}, \tag{2.56}$$

$$D_{12} = D_{21} = \nu_f D_{11}, \tag{2.57}$$

$$D_{66} = (1-\nu_f)D_{11}/2, \tag{2.58}$$

$$S_{44} = S_{55} = G_c t_c. \tag{2.59}$$

In the expressions t_c and t_f are thicknesses of the core and the faces (Figure 20), E_c and E_f are the elastic moduli and ν_c and ν_f are the Poisson's ratios of the core and the flange respectively. I must emphasize that the not mentioned rigidity components are zero as a consequence of *faces with equal thickness and material properties*. The governing equations and therefore the solution is concerned only for such a case, so we have to restrict our objectives with these special type of sandwich structures.

c) Laminated plates

$$D_{ij} = \frac{1}{3}\sum_{k=1}^{N}\left(\bar{Q}_{ij}\right)_{(k)}\left({z_{k+1}}^3 - {z_k}^3\right), \qquad (i,j = 1,2,6) \tag{2.60}$$

$$S_{ij} = \sum_{k=1}^{N}\left(\bar{Q}_{ij}\right)_{(k)}(z_{k+1} - z_k). \qquad (i,j = 4,5) \tag{2.61}$$

Let's take a look on the plate constitutive equations (1.123) and (1.124). The most important phenomenon is the *coupling phenomenon which exists between stretching and bending* and was discussed in more details at the end of chapter 1.4. By ignoring the membranal deformations in equations (2.1)-(2.2) we rule out the possibility to take into account this phenomenon, therefore we **have to restrict our studies to special lamination schemes**, to guarantee the zero value for all the stiffness coefficients B_{ij}, so to rule out bending-extension coupling. This condition is fulfilled in case of *symmetric laminates*. On the other hand, the governing equations were formulated on the assumption that

$D_{16}, D_{61}, D_{26}, D_{62}$ – which represents bend-twist coupling - and S_{45}, S_{54} are zero. To fulfill this criteria also, the laminates must have *specially orthotropic material*, which means that the **laminate is loaded in its principal directions**.

d) Voided plates

$$D_{11} = \frac{Et_f h^2}{2(1-\nu^2)}, \tag{2.62}$$

$$D_{22} = D_{11}\left(1 + \left(\frac{t_w h}{t_f w}\right)\right), \tag{2.63}$$

$$D_{12} = D_{21} = \nu D_{11}, \tag{2.64}$$

$$D_{66} = G t_f h^2/2, \tag{2.65}$$

$$S_{44} = \frac{2Et_f{}^3}{w^2\left(1+2\left(\frac{h}{w}\right)\left(\frac{t_f}{t_w}\right)^3\right)}, \tag{2.66}$$

$$S_{55} = \frac{t_f h\left(1+\frac{t_f}{h}\right)}{\frac{t_f w}{t_w}}. \tag{2.67}$$

2.2. Computational solution

2.2.1. Introduction

The numerical formulation was carried out with FORTRAN (the name is a blend derived from *The IBM Mathematical **For**mula **Tran**slating System*), which is a general-purpose, imperative computational language, especially *optimized for numerical calculations* and scientific computing. It was developed by IBM in the 1950s in California, USA. FORTRAN has many versions. In this thesis we deal with FORTRAN77 and Fortran90. The former is a very early version, containing only 32 commands. Up to this version the languages were conventionally spelled in all-caps. The capitalization has been dropped in referring to newer versions starting with Fortran90. This article adopts the convention of using the all-caps FORTRAN in referring to all of the versions independently of the version number. It is clear that this language was developed for simple purposes, but it is to mention that now it has object-oriented versions either, the last stable release is from 2008.

E. Hinton already published the program code written in FORTRAN77. Unfortunately this version of the code **contained a lot of theoretical, syntactical and other problems** either, so it was unable to run. These problems, and the capabilities of the newer version of FORTRAN, informally known as Fortran90, pushed me to **rewrite the code** in this new environment to produce a **more efficient and maintainable program structure**. The updated code is presented in Appendix A as one and it is explained in the next pages in smaller sections. According to the mentioned programming errors in the code it would be pointless to attach to the present thesis the original code, but for those, who are interested it can be seen in (Owen et al, 1986).

The main program

Later in this section I will name the main goals and capabilities which I find to be important to have to a numerical solver, but it must be mentioned, that the very first step was to

learn FORTRAN itself. If there is any level below the rookie, then I was there before the thesis, so **to learn programming was a very challenging** and **substantial part of the work**.

From now on each FORTRAN command will be distinguished from the current text by applying `Courier New` font and capital letters. It is to note, that FORTRAN does not distinguish between small and capital letters or spaces between the characters of the commands.

Originally the program consisted of **two parts**, one calculating the **rigidities** and another calculating the **mechaical results** in the desired points of the structure. In each sub-program the user had to type in the necessary parameters one by one. Then the program calculates the results in those points, whose *x* and *y* coordinates were defined by typing into the command line one after another. In addition, the maximum number of these points was hard coded into the program as a consequence of the lack of dynamic memory handling. Furthermore, if the user mistyped a data, the opportunity to correct the mistake was to start from the bottom. After reading through, we can point out that these are not desirable properties of a modern program. Considering all of the available and the desired capabilities of a numerical solver I decided to:

- combine the two programs into one,
- redefine the I/O capabilities,
- raise the dimension of the arrays in the program,
- and supplement the code with dynamic memory handling by the way of memory allocation.

Redefining the arrays is not inevitable, but in my opinion it is more logical to work with matrices, since we assign a value (a result) to a pair of data, let say a point, and this way of discussion could have benefits later.

Each FORTRAN program begins with a `PROGRAM` statement, which consists of the `PROGRAM` command followed by the program name. As a pair the same program should end with an `END` statement, which begins with the `END` command and may be, but not necessarily, followed by the program name. In our case the program should look like:

```
PROGRAM BENSYS
…
CALL SUB1(ARG1, ARG2, …)
CALL SUB2(ARG1, ARG2, …)
…
CONTAINS
SUB1(ARG1, ARG2, …)
SUB2(ARG1, ARG2, …)
…
END PROGRAM BENSYS
```

Between the two mentioned statements there are other commands that should be clarified now.

At a very small program there is no need to further articulate the code, let consider the next example:

```
PROGRAM ADD
INTEGER::x,y
READ*,x
READ*,y
PRINT*,"x+y=",x+y
END PROGRAM ADD
```

In this case the operation of the program is see through, it easy to follow what is the path of the operations. First the two variables *x* and *y* are declared, then values are read from the user through command line input and finally the result is printed out to the screen. Although it is a small one, this could be subdivided to different parts as a *declaration section*, and a section containing the *operations*. In the situation of the BENSYS the case is much more complicated. Simply piling up the commands on each other would produce a very hard to see through bunch of commands. To follow the operations of such a structure is hard and in case of a probable mistake, the identification and the correction also takes much more efforts. For these reasons it is good programming practice to group related sets of code with the help of the so called procedures.

Procedures have to kinds, namely *subroutines* and *functions*. The difference is that a function always returns a value, while a subroutine is only a set of commands to execute on calling. Both kind of procedures are classified as internal procedures or module procedures. A subroutine or a function have to be defined similarly to a program with the difference that in the place of the keyword '`PROGRAM`' we should place the '`SUBROUTINE`' or the '`FUNCTION`' words. The main advantage of procedures is that with their help the same operations can be done to different variables. This can be maintained by declaring arguments to the procedure, in this case these should be enclosed in parenthesis after the defining statements. The procedures can be invoked by the `CALL` command in the executable part, but they are defined only at the end of the program code, between the `CONTAINS` command and the `END PROGRAM` statement.

The lines between the program statement and the executable part of the program is called the *declaration section*. Here we use *type statements* to specify a FORTRAN intrinsic type followed by two colons and a list of variable names separated by commas. FORTRAN supports *integer*, *real*, *complex*, *logical* and *character* variable types. Each of them may be declared to have a particular kind parameter by putting 'kind=' followed by the kind parameter value in parentheses. These kind parameters can be used to switch between single and double precision in case of a variable of type real.

2.2.2. Program details

In the next lines I present the main program code, but before doing so let me have some comments. To help the eye I marked the `PROGRAM` statement, the `CONTAINS` command and the `END PROGRAM` statement with **bold** font. Just to remember, between the first two there are the *declaration section* and the *executable part*, while between the last two the definition of the *different subroutines* take place. The '&' symbol at the end of the lines mean continuation of the command in the next line, while the '!' symbol is used to add comments to the code. The essential part of the subroutines are substituted with '…' and will be introduced later in detail.

```
PROGRAM BENSYS

!*****************
!DECLARATION
!*****************

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION :: A(10),D(6,6),H1(11)

REAL,DIMENSION(:),ALLOCATABLE :: XCORD,YCORD
REAL,DIMENSION(:,:),ALLOCATABLE :: BMX,BMY,BMXY,CX,CY,CXY&
                                   ,QX,QY,W,WUX,WUY

!*****************
!EXECUTABLE PART
!*****************

CALL DATA (A,C,E,E11,E22,EF,GC,G12,G23,HD,H1,IQQ,M,T,TF,TW,&
                  VNU,V12,VF,WD,AA,BB,ETA,ETB,G1,GFS,IQ,IT,NPON,&
            NUM,PI,PZ,U,V)

!ALLOCATION
ALLOCATE(XCORD(NPON),YCORD(NPON),BMX(NPON,NPON),BMY(NPON,NPON)&
            ,BMXY(NPON,NPON),CX(NPON,NPON),CY(NPON,NPON),&
         CXY(NPON,NPON),QX(NPON,NPON),QY(NPON,NPON),W(NPON,NPON)&
         ,WUX(NPON,NPON),WUY(NPON,NPON))

!CREATE COORDINATES OF OUTPUT POINTS
DO IPON=1,NPON
   XCORD(IPON)=(AA/(NPON-1))*(IPON-1)
   YCORD(IPON)=(BB/(NPON-1))*(IPON-1)
END DO

IF (IQQ==1) THEN
            CALL HOMG (D11,D22,D12,D21,D66,E,S44,S55,T,VNU)
     ELSE IF (IQQ==2) THEN
            CALL SAND (C,D11,D22,D12,D21,D66,EF,GC,S44,S55,T,VF)
     ELSE IF (IQQ==3) THEN
            CALL VOID (D11,D22,D12,D66,E,HD,S44,S55,TF,TW,VNU,WD)
     ELSE IF (IQQ==4) THEN
            CALL LAMI (A,D,E11,E22,G12,G23,H1,M,V12)
END IF

CALL INIT (BMX,BMY,BMXY,CX,CY,CXY,NPON,QX,QY,W,WUX,WUY)

DO M=1,NUM,IT
     DO N=1,NUM,IT
     CALL COEF (AA,BB,D11,D22,D12,D66,DET,G1,GFS,M,N,&
                         P11,P12,P13,P22,P23,PI,S44,S55)
     CALL CONS (AA,AMN,BB,BMN,CMN,DET,ETA,ETB,IQ,M,N,PI,PZ,P11,P12,&
                         P13,P22,P23,U,V)
     CALL SUMS (AA,AMN,BB,BMN,BMX,BMY,BMXY,CMN,CX,CY,CXY,D11,D22,&
                         D12,D66,G1,M,N,NPON,QX,QY,PI,S44,S55,W,WUX,&
                   WUY,XCORD,YCORD)
     END DO
END DO

CALL OUTP (NPON,XCORD,YCORD,W,BMX,BMY,BMXY,QX,QY)

STOP

CONTAINS

!************************
!DATA /!READS ALL DATA/
!************************
```

```
SUBROUTINE DATA (A,C,E,E11,E22,EF,GC,G12,G23,HD,H1,IQQ,M,T,TF&
                         ,TW,VNU,V12,VF,WD,AA,BB,ETA,ETB,G1,GFS,IQ,IT&
                 ,NPON,NUM,PI,PZ,U,V)

     ...

END SUBROUTINE DATA

!**********************************************************************
!HOMOGENEOUS RIGIDITIES /!CALCULATES RIGIDITIES FOR HOMOGENEOUS PLATES/
!**********************************************************************

SUBROUTINE HOMG (D11,D22,D12,D21,D66,E,S44,S55,T,VNU)

    ...

END SUBROUTINE HOMG

!*****************************************************************
!SANDWICH RIGIDITIES /!CALCULATES RIGIDITIES FOR SANDWICH PLATES/
!*****************************************************************

SUBROUTINE SAND (C,D11,D22,D12,D21,D66,EF,GC,S44,S55,T,VF)

     ...

END SUBROUTINE SAND

!************************************************************
!VOIDED RIGIDITIES /!CALCULATES RIGIDITIES FOR VOIDED PLATES/
!************************************************************

SUBROUTINE VOID (D11,D22,D12,D66,E,HD,S44,S55,TF,TW,VNU,WD)

     ...

END SUBROUTINE VOID

!*****************************************************************
!LAMINATED RIGIDITIES /!CALCULATES RIGIDITIES FOR LAMINATED PLATES/
!*****************************************************************

SUBROUTINE LAMI (A,D,E11,E22,G12,G23,H1,M,V12)

     ...

!*******************************************
!SUBROUTINE INIT  /INITIALISES VARIOUS ARRAYS/
!*******************************************

SUBROUTINE INIT (BMX,BMY,BMXY,CX,CY,CXY,&
                         NPON,QX,QY,W,WUX,WUY)

     ...

END SUBROUTINE INIT

!*******************************************
!SUBROUTINE CONS  /ALCULATES THE VALUES
!AMN,BMN AND CMN FOR A GIVEN LOADING FUNCTION/
!*******************************************

SUBROUTINE CONS(AA,AMN,BB,BMN,CMN,DET,ETA,ETB,IQ,M,N,PI,PZ,P11,&
                         P12,P13,P22,P23,U,V)
```

```
      ...

END SUBROUTINE CONS

!*************************************************
!COEF /!THIS SUBROUTINE CALCULATES THE COEFFICIENTS
!     (P11,P12,P13,P22,P23,P33) AND THE DETERMINANT/
!*************************************************

SUBROUTINE COEF (AA,BB,D11,D22,D12,D66,DET,G1,GFS,M,&
                         N,P11,P12,P13,P22,P23,PI,S44,S55)

      ...

END SUBROUTINE COEF

!***********************************************
!SUBROUTINE SUMS /SUMS THE VARIOUS FURIER SERIES/
!***********************************************

SUBROUTINE SUMS (AA,AMN,BB,BMN,BMX,BMY,BMXY,CMN,CX,CY,CXY,D11,&
                          D22,D12,D66,G1,M,N,NPON,QX,QY,PI,S44,S55,W,&
                 WUX,WUY,XCORD,YCORD)

      ...

END SUBROUTINE SUMS

!*****************************
!OUTP /PRINTS OUT THE RESULTS/
!*****************************

SUBROUTINE OUTP (NPON,XCORD,YCORD,W,BMX,BMY,BMXY,QX,QY)

       ...

END SUBROUTINE OUTP

END PROGRAM BENSYS
```

After having a clear view of the program structure, let discuss now the individual parts of the program in details

Variable declaration

```
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION :: A(10),D(6,6),H1(11)

REAL,DIMENSION(:),ALLOCATABLE :: XCORD,YCORD
REAL,DIMENSION(:,:),ALLOCATABLE :: BMX,BMY,BMXY,CX,CY,CXY&
                                        ,QX,QY,W,WUX,WUY
```

In the earliest days, FORTRAN did not have type declarations. Instead, variables were assigned a type by implicit typing based on the first letter of their names, and only real or integer variable types were allowed. Although nowadays the programming practice is to declare all variables, those that accidentally or intentionally remain undeclared are still assigned the default types based on their first letter. Implicit typing involves the danger that the misspelled or mistyped variable names will not be detected as errors, they are accepted as new variables. For this reason every program should contain the `IMPLICIT NONE` statement which blocks this default behavior. However breaking this rule is acceptable in very short programs. Accordingly the first line states that the variables with first letters being in the range A-H or O-Z will be defined as real numbers, while the others

will be integers. The '*8' means that the real variables will have double precision, in other words they are defined as *floating point* numbers.

In the second line three arrays are defined. Array 'A' and 'H1' are one dimensional arrays (vectors) correspond to laminated plates, while matrix 'D' is the two dimensional array (matrix) of plate rigidities from expression (2.17). Since we already know the dimensions of these arrays, we can predefine them with the `DIMENSION` command.

In the remaining lines the definition of the other arrays –which are arrays of the results - are maintained. Later I will discuss in details how the results are calculated and threated, let just now accept that size of this arrays is up to the user and will be only defined at run time. However these arrays should be dimensioned somehow and there are two possible approach of this problem. The first is to set the dimensions of these arrays to be a very large number *at compile time*, so the real dimension could only be smaller than this value and is formally known as *static storage allocation*. A way more creative one is to apply *dynamic storage allocation*, which means that storage may be allocated or deallocated *during the execution* of a program. With this, the program can wait until it knows exactly what size array is needed and then allocate only that much space

Executable part

The first operation is to read the necessary data to the calculation. This is done by calling a subroutine with the next command:

```
CALL DATA (A,C,E,E11,E22,EF,GC,G12,G23,HD,H1,IQQ,M,T,TF,TW,&
           VNU,V12,VF,WD,AA,BB,ETA,ETB,G1,GFS,IQ,IT,NPON,&
           NUM,PI,PZ,U,V)
```

The arguments of this subroutine are enclosed in the parenthesis and are geometrical, material, load and other parameters, corresponding to the section and load cases of the previous chapters. The exact meanings of the variables are explained later in this chapter at the place of the first occurrence at the complete definition of the subroutines, and they can be found in the Terminology at the beginning if the thesis either.

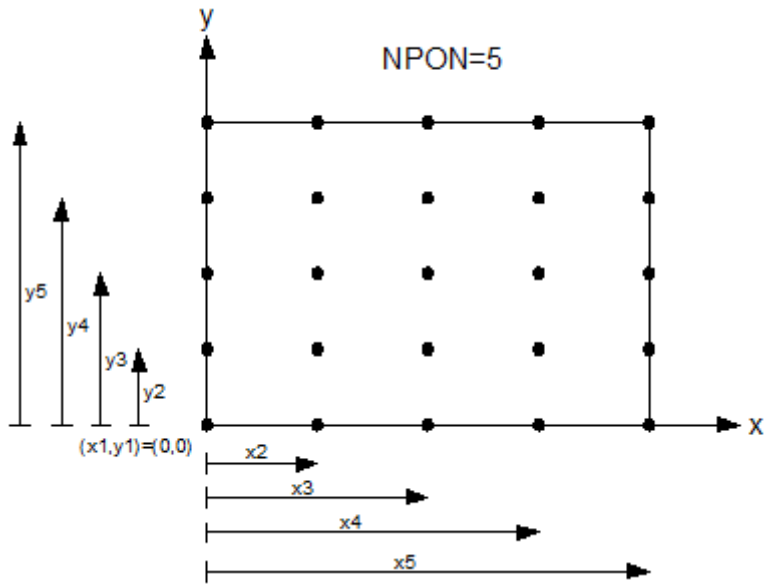


Figure 27 – Meaning of the NPON parameter and the coordinates of the output points.

Thereafter we are in possession of all the data, so we should be able to set the correct dimensions of the previously allocated arrays. This can be done with the `ALLOCATE` command:

```
ALLOCATE(XCORD(NPON),YCORD(NPON),BMX(NPON,NPON),BMY(NPON,NPON)&
             ,BMXY(NPON,NPON),CX(NPON,NPON),CY(NPON,NPON),&
         CXY(NPON,NPON),QX(NPON,NPON),QY(NPON,NPON),W(NPON,NPON)&
         ,WUX(NPON,NPON),WUY(NPON,NPON))
```

Here '*NPON*' means the number of result points in both directions defined by the user, so the matrices of the results can be dimensioned with this parameter.

On the same basis we are now able to calculate the *x* and *y* coordinates of the output points. Accordingly to the previous statements we have $NPON^2$ number of points where we have to calculate the results, therefore:

```
!CREATE COORDINATES OF OUTPUT POINTS
DO IPON=1,NPON
   XCORD(IPON)=(AA/(NPON-1))*(IPON-1)
   YCORD(IPON)=(BB/(NPON-1))*(IPON-1)
END DO
```

The meaning of the *NPON* parameter and the *x* and *y* coordinates is visualized on the next figure (Figure 27). The variables *AA* and *BB* are the two in plane dimensions of the plate in *x* and *y* directions respectively.

The next step is to calculate the plate rigidities depending on the choice of the section type:

```
IF (IQQ==1) THEN
             CALL HOMG (D11,D22,D12,D21,D66,E,S44,S55,T,VNU)
     ELSE IF (IQQ==2) THEN
             CALL SAND (C,D11,D22,D12,D21,D66,EF,GC,S44,S55,T,VF)
     ELSE IF (IQQ==3) THEN
             CALL VOID (D11,D22,D12,D66,E,HD,S44,S55,TF,TW,VNU,WD)
     ELSE IF (IQQ==4) THEN
             CALL LAMI (A,D,E11,E22,G12,G23,H1,M,V12)
END IF
```

Here *IQQ* refers to the type of the cross section and can take values from 1 to 4.

For the calculation of the results it is necessary to set some initial value to the matrices which will store the results. This operation is done by invoking the next subroutine:

```
CALL INIT (BMX,BMY,BMXY,CX,CY,CXY,NPON,QX,QY,W,WUX,WUY)
```

Once done, we can continue with calculating and summing the Fourier terms of expressions (2.41)-(2.46).

```
DO M=1,NUM,IT
     DO N=1,NUM,IT
     CALL COEF (AA,BB,D11,D22,D12,D66,DET,G1,GFS,M,N,&
                      P11,P12,P13,P22,P23,PI,S44,S55)
     CALL CONS (AA,AMN,BB,BMN,CMN,DET,ETA,ETB,IQ,M,N,PI,PZ,P11,P12,&
                      P13,P22,P23,U,V)
     CALL SUMS (AA,AMN,BB,BMN,BMX,BMY,BMXY,CMN,CX,CY,CXY,D11,D22,&
                      D12,D66,G1,M,N,NPON,QX,QY,PI,S44,S55,W,WUX,&
                 WUY,XCORD,YCORD)
     END DO
END DO
```

What we see is a well-known ‘do cycle’ with ‘*M’* as the running parameter. The summation goes from 1 to *NUM* with a step defined by the *IT* parameter. *NUM* means the number of Fourier terms, while *IT* equals 2 if the loading is symmetric and 1 if it is not.

Finally there is nothing left than to write a results to a file. This process is executed by calling the last subroutine:

```
CALL OUTP (NPON,XCORD,YCORD,W,BMX,BMY,BMXY,QX,QY)
```

The next one is the CONTAINS command, which separates the executable part of the program and the definition of the subroutines. So after understanding the sequence and basic purpose of the different subroutines, let have a closer look on each one.

Subroutines

└ SUBROUTINE DATA

This procedure reads in all of the data that is necessary for the forthcoming operations.

```
SUBROUTINE DATA (A,C,E,E11,E22,EF,GC,G12,G23,HD,&
H1,IQQ,M,T,TF,TW,VNU,V12,VF,WD,AA,BB,ETA,ETB,G1,&
GFS,IQ,IT,NPON,NUM,PI,PZ,U,V)

     DIMENSION :: A(10),H1(11)

     !identification of parameters
     OPEN (UNIT=10,FILE="ID.TXT",FORM="FORMATTED",&
     STATUS="OLD",ACTION="READ")
    READ (10,"(I5)") IQQ,IQ,IT,NUM,NPON
    CLOSE (UNIT=10)

     !read geometrical parametrs
    OPEN (UNIT=20,FILE="GEOM.TXT",FORM="FORMATTED",&
 STATUS="OLD",ACTION="READ")
    IF (IQQ==1) THEN
            READ (20,"(F10.5)") AA,BB,T
        ELSE IF (IQQ==2) THEN
                  READ (20,"(F10.5)") AA,BB,C,T
           ELSE IF (IQQ==3) THEN
                  READ (20,"(F10.5)") AA,BB,TF,TW,HD,WD
        ELSE IF (IQQ==4) THEN
          READ (20,"(F10.5)") AA,BB
             READ (20,"(i2)") M
             J=M+1
             DO I=1,J
            READ (20,"(F10.5)") H1(I)
          END DO
          DO K=1,M
            READ (20,"(F10.5)") A(K)
          END DO
    END IF
    CLOSE (20)

    !read material parametrs
    OPEN (UNIT=30,FILE="MAT.TXT",FORM="FORMATTED",&
  STATUS="OLD",ACTION="READ")
    IF (IQQ==1) THEN
            READ (30,"(F10.5)") VNU,E
        ELSE IF (IQQ==2) THEN
                  READ (30,"(F10.5)") VF,GC,EF
          ELSE IF (IQQ==3) THEN
```

```
                    READ (30,"(F10.5)") VNU,E
         ELSE IF (IQQ==4) THEN
                 READ (30,"(F10.5)") E11,E22,V12,G12,G23
    END IF
    CLOSE (30)

    !read load parametrs
    OPEN (UNIT=40,FILE="LOAD.TXT",FORM="FORMATTED",&
  STATUS="OLD",ACTION="READ")
    IF (IQ==1) THEN
           READ (40,"(F10.5)") PZ,GFS,G1
        ELSE IF (IQ==2) THEN
                    READ (40,"(F10.5)") PZ,GFS,G1,ETA,ETB
          ELSE IF (IQ==3) THEN
                    READ (40,"(F10.5)") PZ,GFS,G1,ETA,ETB,U,V
        ELSE IF (IQ==4) THEN
                 READ (40,"(F10.5)") PZ,GFS,G1,ETA,ETB,U,V
    END IF
    CLOSE (40)

       PI=3.141592654
    RETURN

END SUBROUTINE DATA
```

The type of the variables can be easily determined from their first letter and the meanings are:

`A()` – angle of principal laminate direction
`C` – core thickness (see Figure 20)
`E` – Elastic modulus
`E11, E22` – elastic modulus in the first and in the second principal directions (see Figure 12)
`EF` – elastic modulus of the facing (see Figure 20)
`GC` – shear modulus of the core (see Figure 20)
`G12` – in plane shear modulus
`HD` – center to center distance between flanges (see Figure 21)
`WD` – center to center distance between webs (see Figure 21)
`H1()` – z coordinate of boundary of laminate (see Figure 15)
`IQQ` - = 1 for homogeneous plates
= 2 for sandwich plates
= 3 for voided plates
= 4 for laminated plates
`M` – number of laminates in a laminated plate
`T` – plate thickness
`TF` – thickness of the flanges (see Figure 21)
`TW` – thickness of the web (see Figure 21)
`VNU` – Poisson's ratio
`V12` – in plane Poisson's ratio
`VF` – Poisson's ratio of the facing material (see Figure 20)
`AA`, `BB` - in plane dimensions of the plate in *x* and *y* directions respectively
`ETA`, `ETB` – *x* and *y* coordinate of the center of the load
`G1` – shear modification factor
`GFS` – foundation modulus

IQ - = 1 for uniform load
= 2 for concentrated load
= 3 for rectangular patch load
= 4 for pyramid patch load
IT - = 1 for non-symmetric loads
= 2 for symmetric loads
NPON – number of output points
NUM – number of fourier terms
PZ – load intensity
U, V – side length of pathch loads parallel to axis *x* and *y* respectively
PI - π

One thing should be clarified in connection with the variables in FORTRAN. Every variable has a scope, which is the set of lines in the program where the variable's name can be used and refer to the same variable. In general the scope of a variable declared in the main program extends throughout the entire program from the PROGRAM statement to the END PROGRAM statement. A variable declared in an internal procedure has scope extending only in the procedure, not the main program, nor any other internal procedure. A similar rule applies to the IMPLICIT statements. Once declared in the main program it takes effect in the internal procedures as well, so in contradiction with the original code of E. Hinton there is no point in repeating the same IMPLICIT statement in every single internal subroutine. Furthermore the name of an internal procedure its type of arguments and their names are considered as declared in the containing program or procedure, hence their scope extends to the entire program. Let consider the next simple example:

```
SUBROUTINE ADD(A, B)
REAL::A, B, C
C=A+B
END SUBROUTINE ADD
```

In this example *A* and *B* are called *dummy arguments*, while *C* is a *local argument* of the subroutine. This example were shown to illustrate that dummy arguments must be declared in the subroutine even if they have the same name as a variable declared in the containing program. Now it is clear why are arrays *A* and *H1* dimensioned again in SUBROUTINE DATA(), if they were already dimensioned in the main program, because they are dummy arguments now.

After this statement the identification, geometrical, material and load parameters are read in from input files. The creation of these input files will be discussed later, let us assume that we are in possession of some input files with *.txt extension. These files have one record in each line. To read data from txt files, first we have to open them and make a reference by assigning a unit number by the OPEN command. The arguments of the OPEN command specify how the file should be opened. These arguments are:

- UNIT – This number identifies the file and must be unique. We have to avoid number 0, 5 and 6 beacause they are picked to be used by FORTRAN.

- `FILE` – A string that holds the name of the file that we would like to open. If the file to read is in the same directory as the executable, only the filename has to be given.
- `FORM` – A string that holds the type of the file that we would like to open. Text files belong to the group of `FORMATTED` files.
- `STATUS` - A string that holds the status of the file that we would like to open. Note that the default behavior is compiler dependent, so the best is if we always specify one of the types '`NEW`', '`OLD`' and '`REPLACE`'. Now we assume the files to be existing, so the corresponding option is '`OLD`'.
- `ACTION` – Specifies the input/output actions that are permitted to do with the file. One can choose from '`READ`', '`WRITE`' or '`READWRITE`'. Now we want only read from the files, so the option is '`READ`'.

After assigning a unit number to the actual input file, we can read from it with the `READ` command. Let consider the first occurrence:

```
!identification of parameters
      OPEN (UNIT=10,FILE="ID.TXT",FORM="FORMATTED",&
      STATUS="OLD",ACTION="READ")
    READ (10,"(I5)") IQQ,IQ,IT,NUM,NPON
    CLOSE (UNIT=10)
```

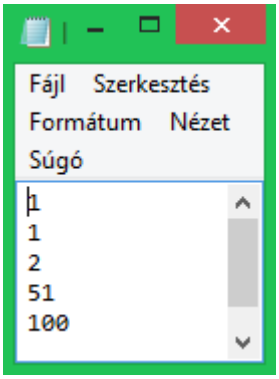


Figure 28 – ID.txt

Here the read command has two arguments. The first refers to the unit number attached to the input file to read from, and the second is a so called *edit descriptor*. In the present case it means that the record in the input file should be of type integer and occupy a total of 5 positions. An example for *ID.txt* can be seen on Figure 28. It should be understood now how the `IQQ`, `IQ`, `IT`, `NUM` and `NPON` variables get their values as 1, 1, 2, 51 and 100 from the *ID.txt* file. When a program is finished with inputting or outputting from or to a file it is prudent to close the file and free the resources of the I/O operation by the close command. The input of the geometrical, material and load parameters happens in the same manner, only the unit number, the filename and the edit descriptors change.

The subroutine ends with the `RETURN` statement. It causes execution of the subroutine to terminate with control given back to the calling program. With the help of modern control constructs, a procedure should terminate by arriving to the end of a procedure. However it is good programming practice to rather using a `RETURN` statement instead of a complicated set of nested `IF` constructs.

└ SUBROUTINE HOMG

```
      SUBROUTINE HOMG (D11,D22,D12,D21,D66,E,S44,S55,T,VNU)

       D11=(E*(T**3))/(12.0*(1.0-VNU*VNU))
     D22=D11
     D12=VNU*D11
     D21=D12
     D66=((1.0-VNU)*D11)/2.0
     S44=(E*T)/(2.0*(1.0+VNU)*(1.2))
     S55=S44
```

```
      RETURN

       END SUBROUTINE HOMG
```

These lines are the same of equations *(2.52)-(2.55)*.

└ SUBROUTINE SAND

```
     SUBROUTINE SAND (C,D11,D22,D12,D21,D66,EF,GC,S44,S55,T,VF)

     D11=(EF*T*((C+T)**2.0))/(2.0*(1.0-VF*VF))
     D22=D11
      D12=VF*D11
      D21=D12
      D66=((1.0-VF)*D11)/2.0
      S44=C*GC/1.2
      S55=S44
      RETURN

      END SUBROUTINE SAND
```

These lines are the same of equations (2.56)-(2.59).

└ SUBROUTINE VOID

```
     SUBROUTINE VOID (D11,D22,D12,D66,E,HD,S44,S55,TF,TW,VNU,WD)

     G=E/(2.0*(1.0+VNU))
     D11=(E*TF*HD*HD)/(2.0*(1.0-VNU*VNU))
     D22=D11*(1.0+(TW*HD)/(6.0*TF*WD))
     D12=VNU*D11
     D21=D12
     D66=(G*TF*HD*HD)/2.0
     S44=(2.0*E*(TF**3))/(WD*WD*(1.0+(2.0*(HD/WD))&
     *((TF/TW)**3))*(1.0-VNU*VNU))
     S55=(G*TF*HD*(1.0+(TF/HD)))/((TF/TW)*WD)
     RETURN

     END SUBROUTINE VOID
```

These lines are the same of equations (2.62)-(2.67).

└ SUBROUTINE LAMI

```
     SUBROUTINE LAMI (A,D,E11,E22,G12,G23,H1,M,V12)

     DIMENSION A(10),AA(6,6),A2(6,6),C1(6,6),&
                   D(6,6),H1(11),Q(6,6)

     V21=V12*E22/E11
     G31=G23
     Q(1,1)=E11/(1.0-V12*V21)
     Q(1,2)=E11*V12/(1.0-V12*V21)
     Q(1,3)=0.0
     Q(1,4)=0.0
     Q(1,5)=0.0
     Q(1,6)=0.0
     Q(2,2)=E22/(1.0-V12*V21)
     Q(2,3)=0.0
     Q(2,4)=0.0
     Q(2,5)=0.0
     Q(2,6)=0.0
     Q(3,3)=0.0
     Q(3,4)=0.0
     Q(3,5)=0.0
```

```
      Q(3,6)=0.0
      Q(4,4)=G23
      Q(4,5)=0.0
      Q(4,6)=0.0
      Q(5,5)=G31
      Q(5,6)=0.0
      Q(6,6)=G12

      Q(2,1)=Q(1,2)
      Q(3,1)=Q(1,3)
      Q(3,2)=Q(2,3)
      Q(4,1)=Q(1,4)
      Q(4,2)=Q(2,4)
      Q(4,3)=Q(3,4)
      Q(5,1)=Q(1,5)
      Q(5,2)=Q(2,5)
      Q(5,3)=Q(3,5)
      Q(5,4)=Q(4,5)
      Q(6,1)=Q(1,6)
      Q(6,2)=Q(2,6)
      Q(6,3)=Q(3,6)
      Q(6,4)=Q(4,6)
      Q(6,5)=Q(5,6)

      DO I=1,6
       DO J=1,6
              D(I,J)=0.0
              DO K=1,M
                     ANGLE=3.1415927*A(K)/180.0
                     CC=COS(ANGLE)
                     SS=SIN(ANGLE)
                     C1(1,1)=0.0
                     C1(1,2)=0.0
                     C1(1,3)=0.0
                     C1(1,4)=0.0
                     C1(1,5)=0.0
                     C1(1,6)=0.0
                     C1(2,2)=0.0
                     C1(2,3)=0.0
                     C1(2,4)=0.0
                     C1(2,5)=0.0
                     C1(2,6)=0.0
                     C1(3,3)=0.0
                     C1(3,4)=0.0
                     C1(3,5)=0.0
                     C1(3,6)=0.0
                     C1(4,4)=0.0
                     C1(4,5)=0.0
                     C1(4,6)=0.0
                     C1(5,5)=0.0
                     C1(5,6)=0.0
                     C1(6,6)=0.0
                    C1(2,1)=C1(1,2)
                    C1(3,1)=C1(1,3)
                    C1(3,2)=C1(2,3)
                    C1(4,1)=C1(1,4)
                    C1(4,2)=C1(2,4)
                    C1(4,3)=C1(3,4)
                    C1(5,1)=C1(1,5)
                    C1(5,2)=C1(2,5)
                    C1(5,3)=C1(3,5)
                    C1(5,4)=C1(4,5)
                    C1(6,1)=C1(1,6)
                    C1(6,2)=C1(2,6)
                    C1(6,3)=C1(3,6)
                    C1(6,4)=C1(4,6)
```

```
                C1(6,5)=C1(5,6)

                C1(1,1)=(Q(1,1)*(CC**4))+(2.0*(Q(1,2)+2*Q(6,6))&
                       *CC*CC*SS*SS)+(Q(2,2)*(SS**4))
                C1(1,2)=((Q(1,1)+Q(2,2)-4.0*Q(6,6))*SS*SS*CC*CC)&
                       +(Q(1,2)*(CC**4+SS**4))
                C1(1,6)=((Q(1,1)-Q(1,2)-2*Q(6,6))*SS*CC**3)&
                       +((Q(1,2)-Q(2,2)+2*Q(6,6))*(SS**3)*CC)
                C1(2,2)=(Q(1,1)*SS**4)+(2.0*(Q(1,2)+2*Q(6,6))&
                       *CC*CC*SS*SS)+(Q(2,2)*CC**4)
                C1(2,6)=((Q(1,1)-Q(1,2)-2*Q(6,6))*(SS**3)*CC)&
                       +((Q(1,2)-Q(2,2)+2*Q(6,6))*SS*(CC**3))
                C1(6,6)=((Q(1,1)+Q(2,2)-2.0*Q(1,2)-2*Q(6,6))&
                       *CC*CC*SS*SS)+(Q(6,6)*(CC**4+SS**4))
                C1(2,1)=C1(1,2)
                C1(6,1)=C1(1,6)
                C1(6,2)=C1(2,6)

                D(I,J)=D(I,J)+(C1(I,J)*(((H1(K+1))**3)-&
                       ((H1(K))**3)))/3.0
            END DO
        END DO
    END DO

    DO I=4,5
     DO J=4,5
            AA(I,J)=0.0
            DO K=1,M
                   ANGLE=3.1415927*A(K)/180.0
                   CC=COS(ANGLE)
                   SS=SIN(ANGLE)
                   A2(4,4)=0.0
                   A2(5,5)=0.0
                   A2(4,5)=0.0
                   A2(5,4)=0.0
                   A2(4,4)=(Q(4,4)*CC*CC)+(Q(5,5)*SS*SS)
                   A2(5,5)=(Q(4,4)*SS*SS)+(Q(5,5)*CC*CC)
                   AA(I,J)=AA(I,J)+(A2(I,J)*&
                          ((H1(K+1))-(H1(K))))
            END DO
        END DO
    END DO
    D(4,4)=AA(4,4)
    D(5,5)=AA(5,5)

    D11=D(1,1)
    D22=D(2,2)
    D12=D(1,2)
    D21=D(2,1)
    D66=D(6,6)
    S44=D(4,4)
    S55=D(5,5)

    RETURN

    END SUBROUTINE LAMI
```

In these lines first we calculate the plane-reduced stiffnesses of equations *(1.73)-(1.78)*. These rigidities refer to the material coordinate system of the individual laminates, so we must apply the transformation introduced in equations (1.101). After carrying out the operations of equations (1.104)-(1.112) we get the elements $\bar{Q}_{ij}$ of expression (1.102) and (1.103). On the go, the summation through the laminates in expressions (1.126) and (1.127) are carried out by applying '`DO`' cycles.

Finally we assign the simple scalar (zero dimensional) variables the corresponding matrix entries for the forthcoming calculations.

└ SUBROUTINE INIT

```
SUBROUTINE INIT (BMX,BMY,BMXY,CX,CY,CXY,&
                      NPON,QX,QY,W,WUX,WUY)

REAL,DIMENSION (:,:) :: BMX,BMY,BMXY,CX,CY,&
                      CXY,QX,QY,WUX,WUY,W

DO IPON=1,NPON
  DO IIPON=1,NPON
  W(IPON,IIPON)=0.0
  WUX(IPON,IIPON)=0.0
  WUY(IPON,IIPON)=0.0
  CX(IPON,IIPON)=0.0
  CY(IPON,IIPON)=0.0
  CXY(IPON,IIPON)=0.0
  BMX(IPON,IIPON)=0.0
  BMY(IPON,IIPON)=0.0
  BMXY(IPON,IIPON)=0.0
  QX(IPON,IIPON)=0.0
  QY(IPON,IIPON)=0.0
  END DO
END DO
RETURN

END SUBROUTINE INIT
```

The arguments of this subroutine are of type real with double precision and they have the next meanings:

`W()` – deflection

`WUX(),WUY()` – rotations of the cross sections in case of the kinematic assumptions of the Mindlin – Reissner theory, see ϑ_x and ϑ_y in equations (2.1) and (2.2)

`CX(),CY(),CXY()` – curvatures with respect to axis *x* and *y* and the mixed curvature parameter, called warping (equation (2.17))

`BMX(),BMY(),BMXY()` – bending moments M_x, M_y and twisting moment M_{xy} in equation (2.17)

`QX(),QY()` – shear forces Q_x and Q_y in equation (2.15)

What happens here is no more than setting to all of the values of the arrays in the argument a zero value. It is necessary, because otherwise in the next subroutines we would try to refer to an entry of these matrices before it has been assigned a value.

└ SUBROUTINE CONS

```
SUBROUTINE CONS(AA,AMN,BB,BMN,CMN,DET,ETA,ETB,IQ,M,N,PI,PZ,P11,&
                      P12,P13,P22,P23,U,V)

IF (IQ==1) THEN
    QMN=(16.0*PZ)/(PI*PI*FLOAT(M)*FLOAT(N))
ELSEIF (IQ==2) THEN
    QMN=((4.0*PZ)/(AA*BB))*SIN((FLOAT(M)*PI*ETA)/AA)&
    *SIN((FLOAT(N)*PI*ETB)/BB)
ELSEIF (IQ==3) THEN
```

```
        QMN=((16.0*PZ*U*V)/(PI*PI*FLOAT(M)*FLOAT(N)*U*V))&
        *SIN((FLOAT(M)*PI*ETA)/AA)*&
            SIN((FLOAT(N)*PI*ETB)/BB)*SIN((FLOAT(M)*PI*U)/(2.0*AA))*&
        SIN((FLOAT(N)*PI*V)/(2.0*BB))
    ELSEIF (IQ==4) THEN
         QMN=(128.0*PZ*aa*bb*sin((pi*eta*FLOAT(M))/aa)&
        *sin((pi*etb*FLOAT(N))/bb)&
        *sin((pi*FLOAT(M)*u)/(4.0*aa))**2.0 - &
        128.0*PZ*aa*bb*sin((pi*eta*FLOAT(M))/aa)&
        *sin((pi*etb*FLOAT(N))/bb)*cos((pi*FLOAT(N)*v)/&
        (2.0*bb))*sin((pi*FLOAT(M)*u)/(4.0*aa))**2.0)&
        /(pi**4.0*FLOAT(M)**2.0*FLOAT(N)**2.0*u*v)
    END IF
    AMN=(P12*P23-P22*P13)*QMN/DET
    BMN=(P12*P13-P11*P23)*QMN/DET
    CMN=(P11*P22-P12*P12)*QMN/DET
    RETURN

 END SUBROUTINE CONS
```

The first part of these operations are equivalent of expressions (2.47)-(2.51), where we calculate the Fourier coefficients of the applied load. After this we determine the Fourier coefficients A_{mn}, B_{mn} and C_{mn} of expressions (2.27)-(2.29) by the means of expressions (2.38)-(2.40).

Because of the implicit typing, the variables `M` and `N`, which are the running parameters in the main program, happen to be of type integer, but in the expressions we need a real value to preserve the accuracy. The change between variable types is called *casting* and in this case it is done by the intrinsic FORTRAN function `FLOAT()`, which converts a variable to the default real type.

└ SUBROUTINE COEF

```
SUBROUTINE COEF (AA,BB,D11,D22,D12,D66,DET,G1,GFS,M,&
                         N,P11,P12,P13,P22,P23,PI,S44,S55)

     P11=D11*(FLOAT(M)*PI/AA)*(FLOAT(M)*PI/AA)&
     +D66*(FLOAT(N)*PI/BB)*(FLOAT(N)*PI/BB)+G1*S55
     P12=(D12+D66)*(FLOAT(M)*PI/AA)*(FLOAT(N)*PI/BB)
     P13=G1*S55*(FLOAT(M)*PI/AA)
     P22=D66*(FLOAT(M)*PI/AA)*(FLOAT(M)*PI/AA)&
     +D22*(FLOAT(N)*PI/BB)*(FLOAT(N)*PI/BB)+G1*S44
     P23=G1*S44*(FLOAT(N)*PI/BB)
     P33=(G1*S55)*(FLOAT(M)*PI/AA)*(FLOAT(M)*PI/AA)&
     +(G1*S44)*(FLOAT(N)*PI/BB)*(FLOAT(N)*PI/BB)+GFS
     DET=P11*(P22*P33-P23*P23)-P12*(P12*P33-P23*P13)&
     +P13*(P12*P23-P22*P13)
     RETURN

END SUBROUTINE COEF
```

These lines are equivalent of equations (2.32)-(2.37).

└ SUBROUTINE SUMS

```
    SUBROUTINE SUMS (AA,AMN,BB,BMN,BMX,BMY,BMXY,CMN,CX,CY,CXY,D11,&
                         D22,D12,D66,G1,M,N,NPON,QX,QY,PI,S44,S55,W,&
                WUX,WUY,XCORD,YCORD)

    REAL,DIMENSION(:) :: XCORD,YCORD
    REAL,DIMENSION(:,:) :: BMX,BMY,BMXY,CX,CY,&
                                      CXY,QX,QY,WUX,WUY,W
```

```
      DO IPON=1,NPON
      DO IIPON=1,NPON
            X=XCORD(IPON)
            Y=YCORD(IIPON)
                  !DEFLECTION W(IPON)
            W(IPON,IIPON)=W(IPON,IIPON)+CMN*SIN((FLOAT(M)*PI*X)/AA)*&
                  SIN((FLOAT(N)*PI*Y)/BB)
            !SLOPE WUX(IPON)
            WUX(IPON,IIPON)=WUX(IPON,IIPON)+AMN*COS((FLOAT(M)*PI*X)/AA)*&
                        SIN((FLOAT(N)*PI*Y)/BB)
            !SLOPE WUY(IPON)
            WUY(IPON,IIPON)=WUY(IPON,IIPON)+BMN*SIN((FLOAT(M)*PI*X)/AA)*&
                        COS((FLOAT(N)*PI*Y)/BB)
            !CURVATURE CX(IPON)
            CX(IPON,IIPON)=CX(IPON,IIPON)+AMN*(FLOAT(M)*PI/AA)*&
                  SIN((FLOAT(M)*PI*X)/AA)*SIN((FLOAT(N)*PI*Y)/BB)
            !CURVATURE CY(IPON)
            CY(IPON,IIPON)=CY(IPON,IIPON)+BMN*(FLOAT(N)*PI/BB)*&
SIN((FLOAT(M)*PI*X)/AA)*SIN((FLOAT(N)*PI*Y)/BB)
            !CURVATURE CXY(IPON)
            CXY(IPON,IIPON)=CXY(IPON,IIPON)+((AMN*(FLOAT(N)*PI/BB))+&
                        (BMN*(FLOAT(M)*PI/AA)))*&
                     COS((FLOAT(M)*PI*X)/AA)*COS((FLOAT(N)*PI*Y)/BB)
            !BENDING MOMENT BMX(IPON)
            BMX(IPON,IIPON)=-D11*CX(IPON,IIPON)-D12*CY(IPON,IIPON)
            !BENDING MOMENT BMY(IPON)
            BMY(IPON,IIPON)=-D22*CY(IPON,IIPON)-D12*CX(IPON,IIPON)
            !TWISTING MOMENT BMXY(IPON)
            BMXY(IPON,IIPON)=D66*CXY(IPON,IIPON)
            !SHEAR FORCE QX(IPON)
            QX(IPON,IIPON)=QX(IPON,IIPON)+G1*S55*&
((CMN*FLOAT(M)*PI/AA)+AMN)*&
                  COS((FLOAT(M)*PI*X)/AA)*SIN((FLOAT(N)*PI*Y)/BB)
            !SHEAR FORCE QY(IPON)
            QY(IPON,IIPON)=QY(IPON,IIPON)+G1*S44*&
((CMN*FLOAT(N)*PI/BB)+BMN)*&
                  SIN((FLOAT(M)*PI*X)/AA)*COS((FLOAT(N)*PI*Y)/BB)
      END DO
      END DO
      RETURN

      END SUBROUTINE SUMS
```

This subroutine is left with the original comments in it, so the meaning of the different matrices should be clear. NPON refers to the number of the output points, hence the summation goes twice from 1 to NPON. Now it is clear that even in the first cycle we already refer to the matrix under discussion, which explains the need for initializing these matrices a few lines before. The equations are equal to those of expressions (2.41)-(2.45).

└ SUBROUTINE OUTP

```
SUBROUTINE OUTP (NPON,XCORD,YCORD,W,BMX,BMY,BMXY,QX,QY)

REAL,DIMENSION (:) :: XCORD,YCORD
REAL,DIMENSION (:,:) :: BMX,BMY,BMXY,QX,QY,W

OPEN (UNIT=60,FILE="RESULT.TXT",FORM="FORMATTED",&
STATUS="REPLACE",ACTION="WRITE")
DO IPON=1,NPON
  DO IIPON=1,NPON
    WRITE (60,"(8F18.5)") XCORD(IPON),YCORD(IIPON),W(IPON,IIPON)&
,BMX(IPON,IIPON),BMY(IPON,IIPON),&
BMXY(IPON,IIPON),QX(IPON,IIPON),&
```

```
QY(IPON,IIPON)
  END DO
END DO
CLOSE (60)
 RETURN

END SUBROUTINE OUTP
```

Finally I wrote a completely new subroutine to handle the output. What I needed here is to create a text file filled up with the results. The `OPEN` statement was mentioned at `SUBROUTINE DATA`, the only difference is that now the `ACTION` specifier is set to `WRITE`. The `WRITE` command is very similar to the `READ` command, as it operates on a device associated with the unit number, which now is the opened text file.

3. GRAPHICAL SOLUTION

The **graphical user interface** (GUI) was programmed in the latest release of the Visual Basic .NET programming environment which is a fully **object oriented** language based on the .NET 4.5 framework of Microsoft. Although terms 'Visual Basic' and 'Visual Basic .NET' are not the same, from now on I will use the term Visual Basic or shortly VB to refer to the language of the graphical user interface.

Software engineers talk about five generations of languages. The first-generation is the machine language: 0s and 1s. Between these levels FORTRAN is sad to be a third-generation language as it provides much more sophisticated language elements such as subroutines, loops and data structures. The Visual Basic development environment belongs to the family of the fifth-generation languages. To be able to place it somewhere among the other common languages, it can be said that aside from a few stylistic differences VB is comparable with C#. In fact, VB is not a whole lot easier to use than C#, and C# is not significantly more powerful, the difference between them is rather one of style. As a matter of fact, all the notions I intended to impellent in the program could be done, I never had to drop an idea because of the deficiency of the VB language skills and this proves the selection to be good.

Again I have to emphasize that in the lack of any previous experience, I had to start **learning VB from scratch**. It was mentioned at FORTRAN either, but it is a *completely different case in the meaning* that now I did not have a good basis code to start from, the programming work started with a really blank sheet.

To the compilation I used the Visual Studio 2012 integrated development environment (IDE) from a Visual Basic developer's point of view. The VB commands in this section will be highlighted by applying `Courier View` font style. The modern IDE's, such as Visual Studio 2012 has word correction options, the typed commands are edited to look like a normal text, therefore the old style full-caps command input is dropped.

3.1 Introduction to Visual Basic .NET

I must adumbrate, that the present thesis does not attempt to give even a basic knowledge of the Visual Basic .NET language, because it could be hardly done under the limit of a reasonable page number. Even so, some very fundamental things must be mentioned to be able to prepare the reader at least to read and

Object Oriented Programming (OOP)

OOP is a programming paradigm that means concepts as "objects", each with its own set of properties, methods and events. Strictly speaking object is a programming structure that encapsulates data and functionality as a single unit.

For example a Button is a control object of VB. It has properties like the text on the button or the site of it and events like pressing it down or releasing it. It can be said what should happen on pressing the button, therefore in may be considered as an independent "machine" with a distinct role.

An object oriented language may be viewed as a collection of interacting objects, as opposed to the imperative languages, in which a program is seen as a list of tasks to execute.

At first sight it must be ambiguous what an object is, but later it will me illustrated with own examples.

understand the general purpose and mechanism of the solutions discussed in the next sections. For those who are interested in the topic I suggest to start with (Foxall, 2010).

An application is a set of instructions directing the computer to perform tasks. The structure of an application is the way how all these instructions are organized and executed.

First of all a glossary is provided in Table 3, after which this section will use many in context.

Keyword	Description
`Namespace`	A collection of classes that provide related capabilities. For example, the `System.Drawing` namespace contains classes associated with graphics.
`Class`	A definition of an object. Includes properties (variables) and methods, each can be `Subs` or `Functions`.
`Sub`	A method that contains a set of commands, allow data to be transferred as parameters, and provides scope around local variables and commands, but does not return a value.
`Function`	A method that contains a set of commands, returns a value, allow data to be transferred as parameters, and provides scope around local variables and commands.
`Return`	Ends the currently executing `Sub` or `Function`. Combined with a return value for functions.
`Dim`	Declares and defines a new variable.
`New`	Creates an instance of an object.
`Nothing`	Used to indicate that a variable has no value. Equivalent to null in other languages and databases.
`Me`	A reference to the instance of the object within which a method is executing.
`Module`	A code block that isn't a class but which can contain `Sub` and `Function` methods. Used when only a single copy of code or data is need in memory.

Table 3 - Most commonly used VB keywords, (Sheldon et al, 2012)

From these, ones have bigger importance from the point of view of the present thesis, so discuss this in details now.

Let's start with `namespace`. VB has a serious number of classes. When .NET was being created, the developers realized that attempting to organize all of these classes requires a system. Therefore a namespace is a system what is used to organize and group classes containing common functionality.

Very shortly a `class` is a definition of a class the same way as the source code is the definition of the application. It designates a common set of data and behavior within the application.

To declare a variable we use the `dim` statement, which is the short form of 'Dimension' and comes from the ancient past of VB. It can be replaced with the commands `Private` or `Public` with which the developer can limit the accessibility of the declared variables in the project. VB supports the general scalar variable types like *integer*, single precision real (*single*), double precision real (*double*), *string*, character (*char*) and *boolean*. Most of them should be familiar from the previous chapter, but the Boolean type variable is new. It is a logical variable which can take the values 'true' or 'false'. Of course arrays and other structures can be declared either, these will be discussed later.

The `nothing` keyword is a way of telling the system that the actual variable does not use any more memory on the heap.

A `sub` or subroutine can be declared with the next statement:

```
Sub Main()
     . . .
End Sub
```

If the subroutine has arguments, they can be specified between the parentheses, while the actions to carry out on invoking the sub can be placed instead of the dots. When all the commands of the subroutine were executed, the control is given back to the code that invoked the sub.

The declaration of a `function` is very similar. The differences are that the function must return a value and we also have to declare the type of that value in advance. The declaration looks like:

```
Function Main() as single
     . . .
     return var1
End Sub
```

This sample function carries out some actions than returns the variable `var1` which is a real number with single precision.

Functions and subroutines can also be combined with the `private` and `public` modifiers to limit the accessibility of the procedures.

A VB program is built up from standard building blocks. A solution can include one or more project. In our case we speak about one project, so there is no need to talk about the higher levels of program structure. A project can contain one or more assemblies,

which are composed of one or more source files. A source file provides the definition and implementation of classes, modules, etc. which altogether contain all the code.

A simple program basically consists of the following parts:

- Namespace declaration
- A class or module
- One or more procedures
- Variables
- The Main procedure
- Statements & Expressions
- Comments

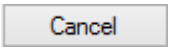
The always first to introduce “Hello World” program has the following structure in VB:

```
' A "Hello, World!" program in Visual Basic.
Module Hello
  Sub Main()
      MsgBox("Hello, World!") ' Display message on computer screen.
  End Sub
End Module
```

Here using namespace is not necessary, but examples will be shown later to this type of statement. In turn we can see the `Main` subroutine, placed on the module `Hello`. This sub uses a message box to print to thee screen the string ‘Hello World!’. Declaring variables also could have been dropped now, but for example the string ‘Hello World’ could be set as a string variable and then this could be used in the argument of the message box. As it has been certainly appeared, the single quotation mark is used to leave comments in the code.

Visual Basic is said to be *event driven*. This means that the procedures can be attached to the event of an arbitrary control, as the clicking of a button or releasing a specific key on the keyboard, etc.

Since I am planning to make a graphical user interface, it is time to speak about the *visual part*. Next to the events, properties and methods, objects do have an interface to allow the user to interact with them at runtime. For example we now that a button has a click event, a text written on it, etc., we can only interact with a button through its interface, which is well known for anyone ever used any program.



Usually the pressing down of this button causes a window to close. It happens because the `close` method of the window is attached to the `click` event of the button. Although a button has its own design, it is not on the fly, it is on a *form*. Forms are essentially windows, and the two terms are often interchangeably. More accurately, window refers to what the user sees and interacts with, whereas form refers to what you see when you design. Think of a form as a canvas on which you can build your program’s interface.

3.2 BENSYS Program structure

The relationship between the hours spent on the two main parts of the program is visualized on Figure 29. In numbers, the length of the code of the graphical user interface is 152 (and growing) in comparison with the 9 pages of FORTRAN. Consequently the all-round introduction of this part cannot be the objective of this thesis. Instead, the most important and fundamental solutions will be introduced.

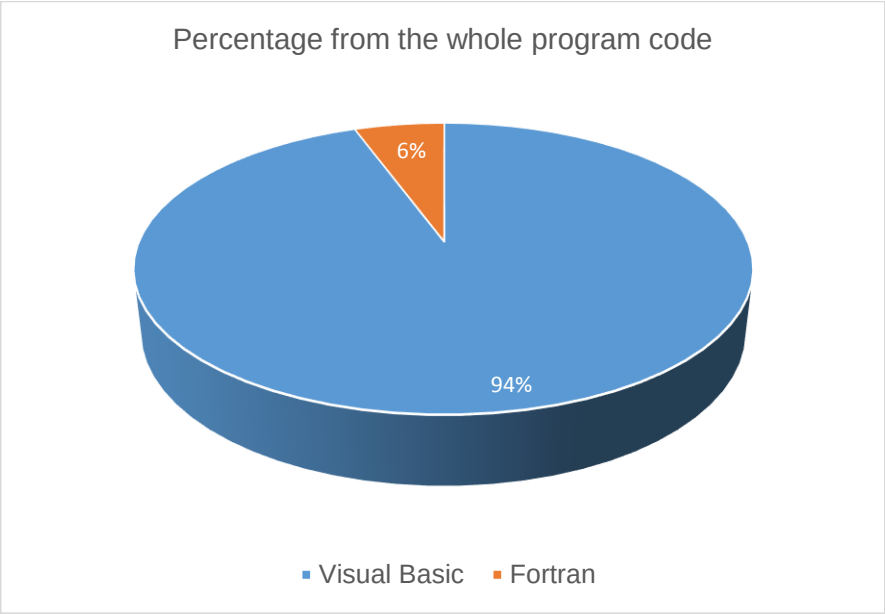


Figure 29 – Percentage from the whole program code

Worth mentioning, that writing a program such BENSYS cannot be done simply from books. Sometimes to find a simple solution needs a very long search and a lot of try on programmer's forums and other sources. In the next subsections I will guide through the user on the path of a general working session and explain the most important tricks and solutions on the go. I assume that the reader is in possess of the program and so he is able to follow the train of thought.

3.1.1. The splash form

Every self-respecting programmer creates a so called splash. It is what appears first when opening an application and can be used to give a hint about the main objectives and the functionality of the program. The splash of BENSYS can be seen on Figure 30. Although it does not contain any control to interact with, actions still can be attached to this form using the `load` event of it. Every form has a `load` event and the commands, functions and subroutines in the event are executed when the form is loaded, in other words it is the first entry of the form. The `load` event of the splash form contains the **registry** operations and the necessary commands to shading the form and giving control to the next form. Thanks to the moderate extent, next to the load event I enclose now the whole class as an example, which is the design view about the form.

```
Public Class Splash

    Public Shared langOpt As String
    Public Shared langOptOld As String
    Public Shared textDir As String

    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles Me.Load

        'registry operations
        langOpt = My.Computer.Registry.GetValue("HKEY_CURRENT_USER\" & _
                        "Software\BENSYS\language\", "LangOpt", "English")
        langOptOld = langOpt
        textDir = My.Computer.Registry.GetValue("HKEY_CURRENT_USER\" & _
                            "Software\BENSYS\directory\", "textDir", "")

        'shading
        Dim sngOpacity As Single
        Me.Show()
        System.Threading.Thread.Sleep(6000)
        For sngOpacity = 1 To 0 Step -0.01
            Me.Opacity = sngOpacity
            'Let the form repaint itself
            Me.Show()
            'Create a delay
            System.Threading.Thread.Sleep(20)
        Next

        ProjectForm.Show()
        Me.Close()


    End Sub
End Class
```

After the class statement there is the declaration section, where 3 string variables are defined and their publicity is set to `public`, because later I will need to access these variables from outside of this form. If I want to use these variables not just outside, but on other forms, then I have to add the specifier `'shared'` to the declaration.

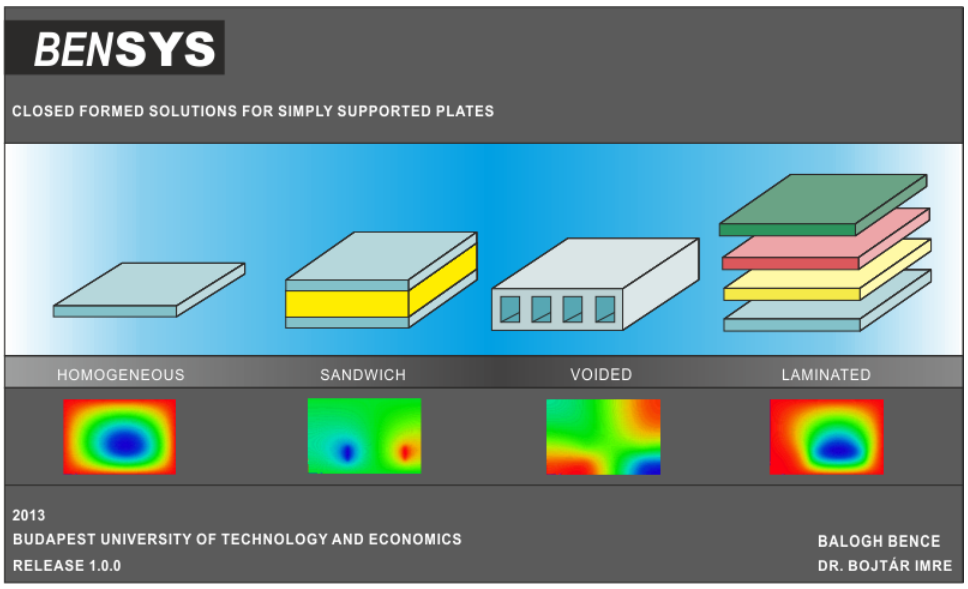


Figure 30 – BENSYS splash screen

After the declarations, the load event takes place. Yes, it is a subroutine, which is attached to the load event of the form, which is named as `form2`, by the words '`Handles`

`Me.Load`' at the end of the subroutine declaration statement. Therefore the commands in the sub are fired when the load is loaded.

The registry is a repository used to store application, user and machine specific information, database connection strings, file locations, and more. It is organized in a hierarchical structure. The top nodes, called the *hives*, are pre-defined and you can't take actions on them. Under each hive, a number of keys are listed. Keys can have subkeys and can contain one or more values. Keys have different data types, but I do not want to spend time with these, since windows always chooses the suitable type when writing to the registry. There are multiple ways to access the registry, the most professional is to use the `Registry` object. This object is an object property of `My.Computer`. The `My` object is basically a shortcut to other useful objects that aren't so easy to be accessed on their own. The `My` object has the object property `Computer` which has the object property called `Registry`. We can use this object to perform all registry operations. To read keys from the registry, the following code is added:

```
langOpt = My.Computer.Registry.GetValue("HKEY_CURRENT_USER\" & _
                        "Software\BENSYS\language\", "LangOpt", "English")
```

On the left side of the equal sign there is a variable that we would like to assign a value. On the right side we can see an example to the use of the `GetValue` method of the `Registry` object. In its argument we have to define the full path of the registry key, the name of the key and the default value. The default value stands for the case when the key does not exist at the moment of the query, for example at the first run of the program. Note, that the hive of the key is `HKEY_CURRENT_USER`, which means that the *language setting can be different for each user of the computer*. The other registry operation is the same, except that is relates to the *working directory* setting. These operations result that the user will start the program with those language and working directory options that he left on closing the program.

After the registry operations, the load event continues with the control of the shading of the form. This can be done with the combination of the use of the `Opacity` property of the form and the `Sleep` method of the `Thread` object under the `System.Threading` namespace. The command

```
System.Threading.Thread.Sleep(6000)
```

means that the system waits 6000 milliseconds until the next command is executed in the code. It means that in the `for…next` cycle the delay between two opacity state is 20 milliseconds, so the speed of the change can be set by the argument of the `Sleep` method. After setting a new opacity value to the form, it must be repainted. It can be achieved on many ways, but the only one that worked well is to force the form to show itself repeatedly with the use of the `Show` method. After the opacity is reduced to zero the cycle finishes and the remaining commands first show the `ProjectForm` form and then close the present form. By these commands not just showing and closing is done, but the control is given to the new form either, as the showing of the form invokes the load event of that form.

3.1.2. The Project Form

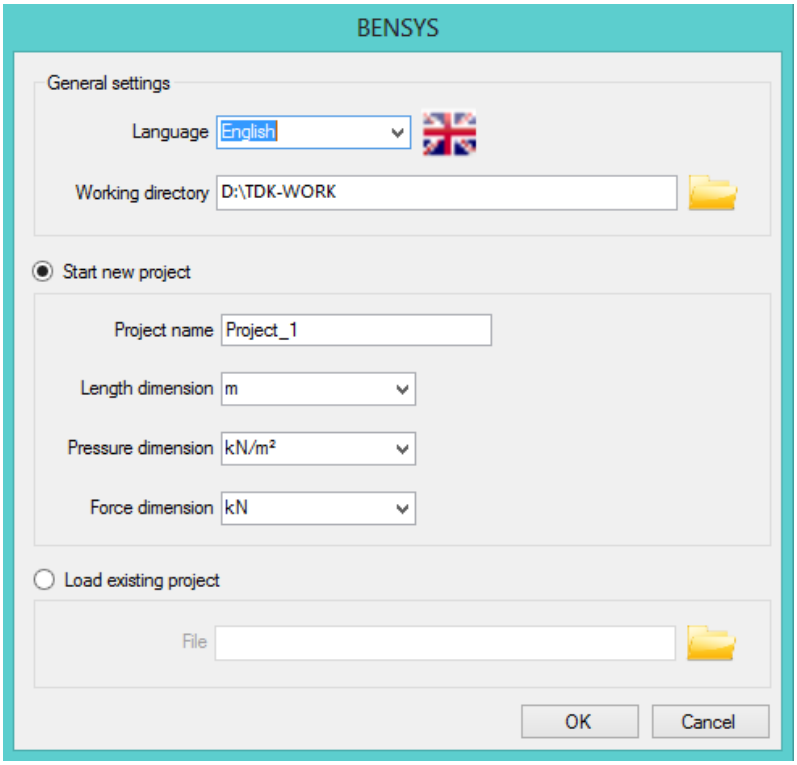


Figure 31 – Project form

Similarly to other software, I wanted a form where the user can initialize the settings for the working session, this can be seen on Figure 31. The user can first set the language, the working directory, but of course the actual values on the form are those that were chosen on the last run. Then the user has two options, these are separated with *radio buttons.* If one decides to start a new working session, then it is able to set the project name and the dimensions for data input. If the choice is to resume an existing work, then the user has to select a BENSYS save file to read from. In this case nothing has to be set because all settings of the program is stored in the save file.

The save and open solutions will be discussed later in details, now let assume that on pressing the folder button next to the textbox a window appears and we can browse for a file and select it. As a result the path of the selected should be written in the textbox, as shown on Figure 32.

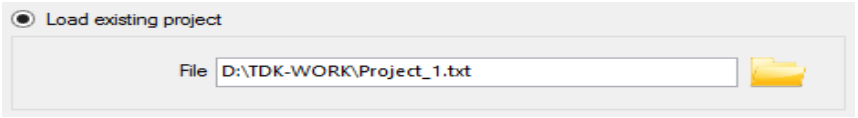


Figure 32 – Loading existing project

By pressing the OK button the control is given to the `Main` form, which is named like this on purpose, this is the core of the application.

3.1.3. The Main Form

The Main form is presented on Figure 33. Actually it is not a form, but a window. The difference was highlighted before, namely the form is the 'design time' equivalent of the window that is visible when running the application.

The selections are controlled by tabs and radio buttons, let's first discuss the geometry.

Geometry

First let's think over what is needed than we can have a look on the solutions. We can choose from 4 section types as 'Homogeneous', 'Sandwich', 'Voided' and 'Laminated'. Each has its own radio button, whose `clicked` property can take values true or false depending on the choice of selection. One of the key features of the radio button is that if

we place more than of it on a panel or a form, than only one of them can be clicked. This means that if we click on one, it will automatically change its clicked property to true and change the other's click property to false. So if we want to have different controls on the form depending on the cross section, we should attach some actions to the `CheckedChanged` property of each radio button. Obviously these commands are going to fire when the radio button's `clicked` property changes, independently from the direction of the change. For instance the `CheckedChanged` event of the first radio button is declared on the `Main` form, and takes the following form:

```
Private Sub rbtHomg_CheckedChanged(sender As Object, e As EventArgs) Handles
rbtHomg.CheckedChanged
        '-------------------------------------------------------------
        'DEFINES WHAT HAPPENS IF THE HOMOGENEOUS BUTTON IS CLICKED
        '-------------------------------------------------------------
        HomogeneousGeom()
End Sub
```

Now the application of the modules gets its meaning. Although the subroutine `HomogeneousGeom()` is invoked on the Main form, its code is placed on the `GeometryModule` module. In this way the path of operations is see through and easy to

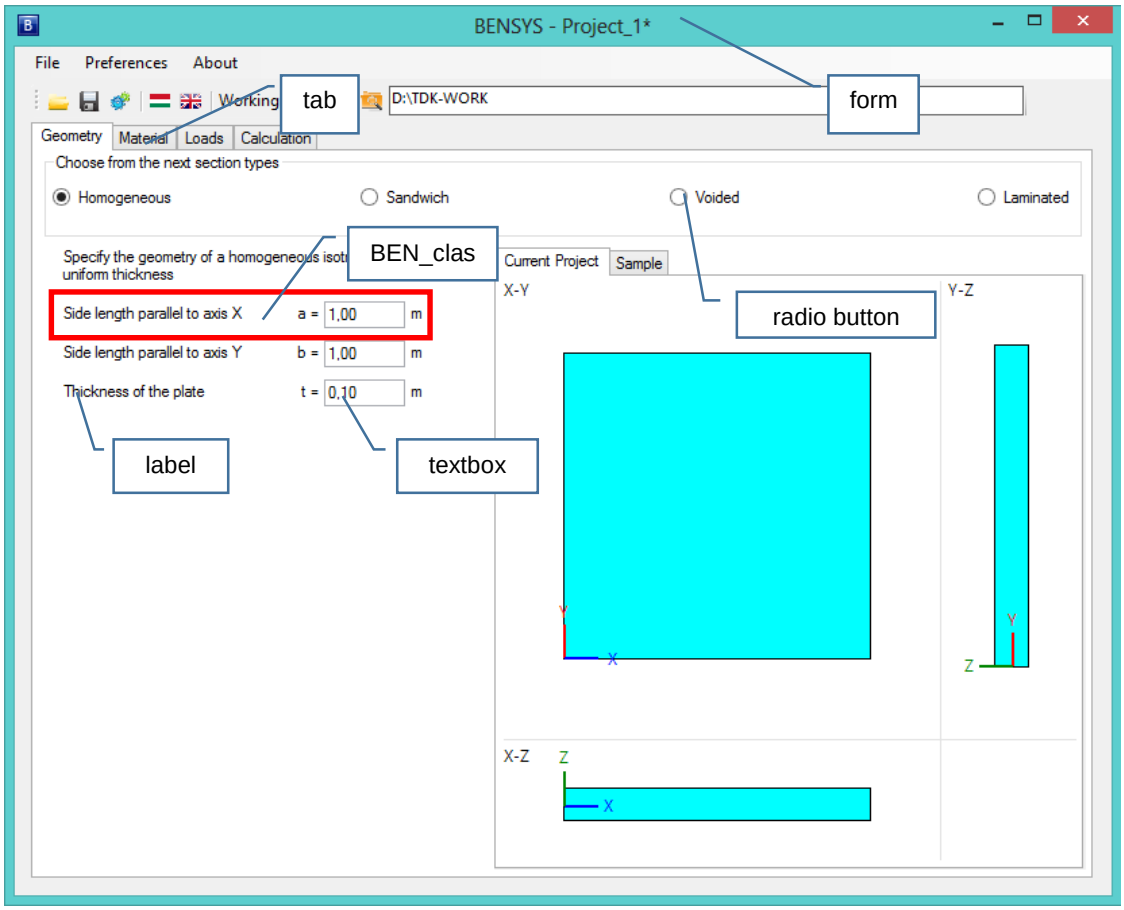


Figure 33 – The Main form after loading with some of the object types marked

follow. Consequently, all of the variables, functions and subroutines that have actions when the focus is on the 'Geometry' tab is located on the `GeometryModule` module. Unfortunately this thought cannot be continued with the introduction of the `HomogeneousGeom()` subroutine, but later we will finish this discussion.

What is important now is to introduce an object, called `BEN_class`. It is one of my own created objects and by getting to know it, the definition of an object will be surely clearer. To understand the need to create such an object, let's think through what happens exactly when changing the section type from 'Homogeneous' to 'Sandwich'. When being in 'Homogeneous' state, there are controls on the form like textboxes and labels, see Figure 33. The geometry of a sandwich plate is not the same, therefore these controls need to be removed and others should be added. For example to add the controls of the 'Homogeneous' choice right after clicking on the OK button of the Project form needs the following steps:

- Declare objects local to the subroutine with the `Dim` statement (13 db).
- Set the location and size of each one.
- Set other properties such as the text (depending on the actual language setting), background color, etc.
- Add handlers to the objects to handle their events. These will be detailed later, but keep in mind that it means a circa hundred lines of code *for each* textbox.
- Add the objects.

Then changing to 'Sandwich' requires:

- Removing the handlers of the 'Homogeneous' controls. It is an **extremely important** step because otherwise the next time when the 'Homogeneous' type would have been chosen, the handlers would be added again and then invoking some event of these controls would execute the corresponding actions twice and third in the next round, etc.
- Removing the 'Homogeneous' controls.
- Repeating the steps of the previous list with the 'Sandwich controls'

Maybe it is not obvious now, but it is a lot to do and it must be done for every four section types, and of course as the geometry gets more complicated, the number of the controls to work with gets higher. So it would be very productive to encapsulate these actions into one object and then reuse the same object everywhere where the same functionality is needed. The elements of the `BEN_class` object can be seen on Figure 33 in the red rectangle. At the end of this point it will be clear that explaining all of the details is not an option to choose. Even so, it is necessary to introduce once the creation of an own object, because actually it is the real deal of programming and provided the body of the thesis.

In the next pages the full code of the `BEN_class` object is presented, the line numbering is meant to help us follow the lead of operations. Before all, I have to apologize for the formatting errors, but the different compilers do not include too improved text formatting tools and inside the program there is no point of margins neither. As it can be seen, the lines begin with increased indentation. Where it doesn't, that means that the command should be regarded as the continuation of the command in the previous line. As the '`&`' in FORTRAN, the symbol of continuation is now the underline, as on the next example.

```
very_very_long_command_can_be_continued_in_the _
next_line
```

Of course the underline can be placed in the command itself, but if it is prevented by a space than in means continuation.

```
Imports BEN___SYS.Splash
Imports BEN___SYS.ProjectForm
Public Class BEN_class

    Private exist = False
    Private WithEvents box As New BEN_box
    Private box_label As New Label
    Private box_label_long As New BEN_label
    Private box_dim As New Label
    Private boxtext As Single = 1.0
    Private box_font As New Font(box_label.Font.Name, box_label.Font.Size)
    Private box_long_font As New Font(box_label_long.Font.Name,
box_label_long.Font.Size)
    Private box_pan As String
    Private box_type As dimtypes

    Private err1() As String = {"A value must be specified", "A mező
kitöltése kötelező"}
    Private err2() As String = {"The value must be greater than 0", "Az érték
0-nál nagyobb kell, hogy legyen"}
    Private err3() As String = {"Only numeric input is allowed!", "Ebbe a
mezőbe csak számot írhat!"}
    Private err4() As String = {"Not numeric!", "Hibás formátum!"}
'-------------------------------------------------------------------------
    Public ReadOnly Property benbox() As BEN_box
        Get
            Return box
        End Get
    End Property
'-------------------------------------------------------------------------
    Public Property text() As String
        Get
            Return box_label.Text
        End Get
        Set(ByVal value As String)
            box_label.Text = value
            box_label.Size = TextRenderer.MeasureText(box_label.Text,
box_font)
        End Set
    End Property
'-------------------------------------------------------------------------
    Public Property value() As Single
        Get
            Return boxtext
        End Get
        Set(value As Single)
            box.Text = String.Format("{0:F2}", value)
            boxtext = value
        End Set
    End Property
'-------------------------------------------------------------------------
    Public Property left() As Integer
        Get
            Return box.Left
        End Get
        Set(value As Integer)
            box.Left = value
            box_label.Left = box.Left - box_label.Width
            box_dim.Left = box.Left + box.Width + 2
        End Set
    End Property
'-------------------------------------------------------------------------
    Public Property height() As Integer
        Get
            Return box.Height
        End Get
        Set(value As Integer)
            box.Height = value
```

```
        End Set
    End Property
'-------------------------------------------------------------------------
    Public Property label() As String
        Get
            Return box_label_long.Text
        End Get
        Set(value As String)
            box_label_long.Text = value
        End Set
    End Property
'-------------------------------------------------------------------------
    Public ReadOnly Property dimlabel() As Label
        Get
            Return box_dim
        End Get
    End Property

    Public ReadOnly Property shortlabel() As Label
        Get
            Return box_label
        End Get
    End Property
'-------------------------------------------------------------------------
    Public Property top() As Integer
        Get
            Return box.Top
        End Get
        Set(value As Integer)
            box.Top = value
            box_label.Top = value + 2
            box_dim.Top = value + 2
        End Set
    End Property
'-------------------------------------------------------------------------
    Public Property dimension() As String
        Get
            Return box_dim.Text
        End Get
        Set(value As String)
            box_dim.Text = value
        End Set
    End Property
'-------------------------------------------------------------------------
    Public Property type() As dimtypes
        Get
            Return box_type
        End Get
        Set(value As dimtypes)
            box_type = value
        End Set
    End Property
'-------------------------------------------------------------------------
    Public Property IsError() As Boolean
        Get
            Return box.IsError
        End Get
        Set(value As Boolean)
            box.IsError = value
        End Set
    End Property
'-------------------------------------------------------------------------
    Public Property existance() As Boolean
        Get
            Return exist
        End Get
```

```
        Set(value As Boolean)
            exist = value
        End Set
    End Property
'-------------------------------------------------------------------------------
    Public Sub Add(pan_ As String, array_ As Array, text_ As String, left_ As
Integer, _
                   top_ As Integer, start_ As Single, Optional ByVal type_ As
dimtypes = dimtypes.dimension)

        Define(pan_, array_, text_, left_, top_, start_, type_)
        JustAdd()

    End Sub
'-------------------------------------------------------------------------------
    Public Sub JustAdd()

        If box_pan = "G" Then
            formMainForm.panGeometry.Controls.Add(box)
            If box_label_long.Text <> "" Then
formMainForm.panGeometry.Controls.Add(box_label_long)
            formMainForm.panGeometry.Controls.Add(box_label)
            If type <> dimtypes.none Then
formMainForm.panGeometry.Controls.Add(box_dim)
        ElseIf box_pan = "L" Then
            formMainForm.panLoad.Controls.Add(box)
            If box_label_long.Text <> "" Then
formMainForm.panLoad.Controls.Add(box_label_long)
            formMainForm.panLoad.Controls.Add(box_label)
            If type <> dimtypes.none Then
formMainForm.panLoad.Controls.Add(box_dim)
        End If

        If box.Text = "" Then box.Focus()
        existance = True

    End Sub
'-------------------------------------------------------------------------------
    Public Sub Define(pan_ As String, array_ As Array, text_ As String, left_
As Integer, _
                      top_ As Integer, start_ As Single, Optional ByVal type_
As dimtypes = dimtypes.dimension)

        type = type_

        box.Width = 60

        If boxtext <> Nothing Or type = dimtypes.none Or type =
dimtypes.foundation Then
            box.Text = String.Format("{0:F2}", boxtext)
        Else
            box.Text = ""
            box.IsError = True
            formMainForm.ErrorProvider1.SetError(box_dim,
err1(formMainForm.LangIndex(langOpt)))
        End If
        box_label.Text = text_

        If array_ Is Nothing Then
            box_label_long.Text = ""
        Else
            box_label_long.array = array_
            box_label_long.lang = langOpt
        End If
        box_label.Size = TextRenderer.MeasureText(box_label.Text, box_font)
```

```
        box_label_long.Size = TextRenderer.MeasureText(box_label_long.Text,
box_long_font)
        box.Left = left_
        box_label.Left = left_ - box_label.Width
        box_label_long.Left = 14
        box_dim.Left = left_ + box.Width + 2
        box.Top = top_
        box_label.Top = top_ + 2
        box_label_long.Top = top_ + 2
        box_dim.Top = top_ + 2
        box_pan = pan_

        If existance = False Then
            value = start_
            'existance = True
        End If

        If type_ = dimtypes.dimension Then
            If existance = True Then
                If box_dim.Text <> dimOpt Then
                    Convert(box_dim.Text, dimOpt)
                    box_dim.Text = dimOpt
                End If
            Else
                box_dim.Text = dimOpt
            End If
        ElseIf type_ = dimtypes.pressure Then
            If existance = True Then
                If box_dim.Text <> presOpt Then
                    Convert(box_dim.Text, presOpt)
                    box_dim.Text = presOpt
                End If
            Else
                box_dim.Text = presOpt
            End If
        ElseIf type_ = dimtypes.force Then
            If existance = True Then
                If box_dim.Text <> forceOpt Then
                    Convert(box_dim.Text, forceOpt)
                    box_dim.Text = forceOpt
                End If
            Else
                box_dim.Text = forceOpt
            End If
        ElseIf type_ = dimtypes.none Then
            box_dim.Text = ""
        ElseIf type_ = dimtypes.foundation Then
            box_dim.Text = "kN/m3"
        End If

        box_dim.Size = TextRenderer.MeasureText(box_dim.Text, box_long_font)

    End Sub
'-------------------------------------------------------------------------
    Public Sub Remove()

        If box.Text <> "" And box.IsError = False Then
            boxtext = CSng(box.Text)
        Else
            boxtext = Nothing
        End If

        If box_pan = "G" Then
            If IsNothing(box_label) = False Then
formMainForm.panGeometry.Controls.Remove(box_label)
            formMainForm.panGeometry.Controls.Remove(box)
```

```
            formMainForm.panGeometry.Controls.Remove(box_label_long)
            If type <> dimtypes.none Then
formMainForm.panGeometry.Controls.Remove(box_dim)
        ElseIf box_pan = "L" Then
            If IsNothing(box_label) = False Then
formMainForm.panLoad.Controls.Remove(box_label)
            formMainForm.panLoad.Controls.Remove(box)
            formMainForm.panLoad.Controls.Remove(box_label_long)
            If type <> dimtypes.none Then
formMainForm.panLoad.Controls.Remove(box_dim)
        End If

    End Sub
'-------------------------------------------------------------------------
    Public Sub Convert(from_ As String, to_ As String)
        box_dim.Text = to_
        box_dim.Size = TextRenderer.MeasureText(box_dim.Text, box_long_font)

        If box.Text <> 0 Then
            If type = dimtypes.dimension Then
                value = CSng(box.Text) * UnitChange(to_) / UnitChange(from_)
            ElseIf type = dimtypes.pressure Then
                value = CSng(box.Text) * UnitChange(to_, type) /
UnitChange(from_, type)
            ElseIf type = dimtypes.force Then
                value = CSng(box.Text) * UnitChange(to_, type) /
UnitChange(from_, type)
            End If
        End If

    End Sub
'-------------------------------------------------------------------------
    Private Function BENbox_onlyNumbers(ByVal KeyChar As Char) As Boolean
        '-----------------------------------------------------------------
        'RETURNS TRUE OR FALSE ON SOME CHARACTERS
        '-----------------------------------------------------------------
        Dim allowedChars As String
        If type = dimtypes.force Or type = dimtypes.pressure Then
            allowedChars = "-0123456789,"
        Else
            allowedChars = "0123456789,"
        End If

        If allowedChars.IndexOf(KeyChar) = -1 And (Asc(KeyChar)) <> 8 Then
            Return True
        Else
            Return False
        End If

    End Function
'-------------------------------------------------------------------------
    Private Function BENbox_onlyonecomma(ByVal KeyChar As Char) As Boolean
        '-----------------------------------------------------------------
        'RETURNS TRUE OR FALSE ON SOME CHARACTERS
        '-----------------------------------------------------------------
        Dim actualtext As String
        actualtext = box.Text

        If actualtext.IndexOf(KeyChar) = -1 And (Asc(KeyChar)) <> 8 Then
            Return False
        Else
            Return True
        End If

    End Function
'-------------------------------------------------------------------------
```

```
    Private Sub BENbox_KeyPress(ByVal sender As Object, ByVal e As
KeyPressEventArgs) Handles box.KeyPress
        '---------------------------------------------------------------------
        'HANDLES A KEYPRESS EVENT ON DECIDING IF IT IS NUMERIC OR NOT
        'HANDLES A KEYPRESS EVENT ON ROUNDING
        '---------------------------------------------------------------------

        If BENbox_onlyNumbers(e.KeyChar) = True Then
            If e.KeyChar = "." Then
                If BENbox_onlyonecomma(",") = True Or box.Text = "" Or
box.Text = "-" Then
                    e.Handled = True
                Else
                    e.KeyChar = ","
                End If
            ElseIf e.KeyChar = Microsoft.VisualBasic.ChrW(Keys.Return) Then
                If box.Text <> "" And box.Text <> "-" Then
                    SendKeys.Send("{TAB}")
                    value = Math.Round(CSng(box.Text), 2)
                Else
                    e.Handled = True
                End If
            Else
                MessageBox.Show(err3(formMainForm.LangIndex(langOpt)), _
                                err4(formMainForm.LangIndex(langOpt)), _
                                MessageBoxButtons.OK,
MessageBoxIcon.Information)
                e.Handled = True
            End If
        ElseIf BENbox_onlyNumbers(e.KeyChar) = False Then
            If e.KeyChar = "," Then
                If BENbox_onlyonecomma(",") = True Or box.Text = "" Or
box.Text = "-" Then
                    e.Handled = True
                Else
                    e.KeyChar = ","
                End If
            End If
        End If

    End Sub
'-------------------------------------------------------------------------------
    Private Sub BENbox_KeyUp(ByVal sender As Object, ByVal e As KeyEventArgs)
Handles box.KeyUp
        '---------------------------------------------------------------------
        'HANDLES A KEYUP EVENT
        '---------------------------------------------------------------------

        If box.Text = "" Or box.Text = "-" Then
            box.IsError = True
            If type <> dimtypes.none Then
                formMainForm.ErrorProvider1.SetError(box_dim,
err1(formMainForm.LangIndex(langOpt)))
            Else
                formMainForm.ErrorProvider1.SetError(box,
err1(formMainForm.LangIndex(langOpt)))
            End If
        ElseIf CSng(box.Text) = 0 Then
            If type = dimtypes.none Or type = dimtypes.foundation Then
                box.IsError = False
                If type <> dimtypes.none Then
                    formMainForm.ErrorProvider1.SetError(box_dim, Nothing)
                Else
                    formMainForm.ErrorProvider1.SetError(box, Nothing)
                End If
            Else
```

```
                box.IsError = True
                formMainForm.ErrorProvider1.SetError(box_dim,
err2(formMainForm.LangIndex(langOpt)))
            End If
        Else
            box.IsError = False
            If type <> dimtypes.none Then
                formMainForm.ErrorProvider1.SetError(box_dim, Nothing)
            Else
                formMainForm.ErrorProvider1.SetError(box, Nothing)
            End If
        End If

    End Sub
'-------------------------------------------------------------------------
    Private Sub BENbox_LostFocus(ByVal sender As Object, ByVal e As
EventArgs) Handles box.LostFocus
        '-----------------------------------------------------------------
        'HANDLES A LOSTFOCUS EVENT
        '-----------------------------------------------------------------
        If box.IsError = True Then
            box.Focus()
        Else
            value = Math.Round(CSng(box.Text), 2)
        End If

    End Sub
'-------------------------------------------------------------------------
    Public Enum dimtypes
        dimension
        none
        force
        pressure
        foundation
    End Enum

End Class
```

The individual procedures and the declaration section are separated by dashed lines.

The object is composed of three `label` object and one `BEN_box` object which is also a self-made one but now it can be regarded as a simple `textbox` object. From lines 24-139 the definition of the different properties take place. Property can be anything what is found to be important and is assumed to be queried or changed from code later. For example if we will have a subroutine in which we want to modify the text in the `BEN_box` of a `BEN_class` object, than we have to make a text property to provide a way to access it. Of course in this case the `BEN_box` have to be the property of the `BEN_class` either, as it is according to the first to properties. With these, if we assume that we have a `BEN_class` object named `benclass1`, than to modify the text to the string 'new text' requires the following code:

```
benclass1.benbox.text='new text'
```

If we set the property of an object to some value, we expect it to be remembered, therefore we have to store the values out own properties either. This is done with the variables in the declaration section (lines 5-23). These are declared with the `Private` keyword, which means that they are local to the class, therefore they can not be accessed from outside of the class, only by changing the value of the corresponding property. When reading a property, the value of the corresponding local variable is returned and writing a property is no more than assigning it a value. Property procedures enables the user to execute code when the property is changed. The basic structure of a property procedure looks like:

```
Public Property propertyname() As datatype
  Get
      ' Code to return the property's value goes here.
  End Get
  Set(ByVal Value As datatype)
      ' Code that accepts a new value goes here.
  End Set
End Property
```

Between the property declaration statement and the `End Property` statement are two constructs: `Get` and `Set`. The `Get` construct is used to place code that returns a value for the property when read by a client. The `Set` construct is where to place code that accepts a new property value from client code. The value between the parenthesis declares the type of the data what is passed to the argument. If we look at the `text()` property at line 42 we can see, that it returns the module level variable `boxtext`, and sets the `text` property of the module level variable `box_label` and sets also its size to match the length of its content. The behavior of the other procedures is the same. The meanings are more or less self explanatory, but one of them is very important, the `IsError()` property. It can take the boolean values true or false and it returns true if the string of the `BEN_box` object does not have input errors as it will be explained later.

The subroutines `Add()`, `JustAdd()` and `Define()` contains the commands which are required to add, position and initialize the mentioned four components of the `BEN_class` object. It is critical to well organize the structure of these subroutines, however these contain nothing special what couldn't have been learned from any book of the topic.

What is more important is how the input of the `BEN_box` object is controlled. Note, that the input assigns a value to a dimension of a plate, therefore we have some expectations towards this input. We want it to be a positive number, but not zero. More exactly zero can

be an input value, it just cannot be the only one, etc. Moreover if the input contains an error, the program should lead the user to the right direction by some notice. These behaviors are encapsulated into the `KeyPress` and the `KeyUp` events of the `BEN_box` object. Before introducing these, two fundamental function should be discussed. The first is called `BENbox_onlynumbers()` and begins at line 298. It attempts a character as an input and returns true if it is not included in the group of the allowed characters which are '0123456789,' in case of dimensions and '-0123456789,' in case of forces. The function `BENbox_onlyonecomma()` also accepts a character as input and returns true if the text of the `BEN_BOX` object includes the input character. Generally the two functions do the same thing, they determine from a string if a character is contained in them or not. Without going into the very details, the command assigned to the `KeyPress` event of the `BEN_BOX` object do the next operations:

- Determines if the character input is beteen the allowed characters with the `BENbox_onlynumbers()` function.
    - If it returns true, but it is a dot, than it determines if the box already contains a comma or not with the `BENbox_onlyonecomma()` function and puts a comma if the result is false, otherwise disables character input and provides an error message.
    - If it returns false and the character is a comma, than it determines if the box already contains a comma or not with the `BENbox_onlyonecomma()` function and neglects the action if the result is true, otherwise enables it and the input character will be placed in the box.

The following operations are attached to releasing up a button on the keyboard with the `KeyUp` event:

- If after pressing a key the box is still empty or contains only a minus sign (because the action was disabled by the `KeyPress` event), than provides an error note.
- Else if the value being shown in the box is zero than two case is possible. If the value under question is a dimension, than an error notice must be provided. If it is a force or something that can take zero value, than remove all the error notes because the content is valid.
- It is very important, that parallel with setting and deleting error notes, the mentioned `IsError()` property is always set to true if the content is valid and false if it is not. Therefore instead of carrying out all the difficult operations above we can always determine from a `BEN_class` object if its content is valid or not.

The last subroutine still worth mentioning. It fires when the box loses control. If the `IsError()` property is false, than it rounds the value to 2 fractional digits.

After this small evasive let's turn back to the geometry tab of the main form. As it was mentioned, clicking on the 'Homogeneous' radio button invokes the `HomogeneousGeom()` sub on the `GeometryModule` module.

```
Public Sub HomogeneousGeom()

        If formMainForm.rbtHomg.Checked = True Then
```

```
            'Add tabpage and picturebox.
            tabpageHomgSample1 = New BEN_Pic

            'Add instruction string
            labinstHomg = New BEN_label
            labinstHomg.array = instHomg
            labinstHomg.lang = langOpt
            labinstHomg.Width = 300
            labinstHomg.Height = 26
            labinstHomg.MaximumSize = labinstHomg.Size
            labinstHomg.AutoSize = True
            labinstHomg.Left = 14
            labinstHomg.Top = 0
            formMainForm.panGeometry.Controls.Add(labinstHomg)

            'Add BENboxes
            numboxHomg1.Add("G", textHomg1, "a =", 210, 40, 1.0)
            numboxHomg2.Add("G", textHomg2, "b =", 210, 69, 1.0)
            numboxHomg3.Add("G", textHomg3, "t =", 210, 98, 0.1)

            'Add gridview items.
            dataMat.Rows.Add(Gsub1(formMainForm.LangIndex(langOpt)), "", "",
"", "", "", "", "", "", "", "")

        ElseIf formMainForm.rbtHomg.Checked = False Then

            numboxHomg1.Remove()
            numboxHomg2.Remove()
            numboxHomg3.Remove()
            tabpageHomgSample1.Remove()
            dataMat.Clear()
            formMainForm.panGeometry.Controls.Remove(labinstHomg)

        End If

    End Sub
```

The red rectangles show the parts of the subroutine which are in connection with the `BEN_class`. With this object the adding, positioning and all the maintenance operations of the controls on the geometry tab simplifies to these statements. Under the '`Add instruction string`' comment we can see the commands necessary to define one single label with the classical methods, namely declaration, parameter settings and adding to the form. The difference speaks for itself…

The commands corresponding to other choices of section type are very similar to what was seen at the uniform one, the main point is the use of the `BEN_class` object.

Leave the rest of this subroutine and the geometry tab and follow with the load tab.

Loads

Being on this tab we can select from the load types uniform, concentrated, rectangular patch and cosine patch. The uniform case provides nothing new, we work again with `BEN_class` objects. Clicking on the '`Concentrated`' radio button makes the form look like on Figure 34.

Here we can see another object of mine, called `BEN_scroll`. It containes 7 labels, 3

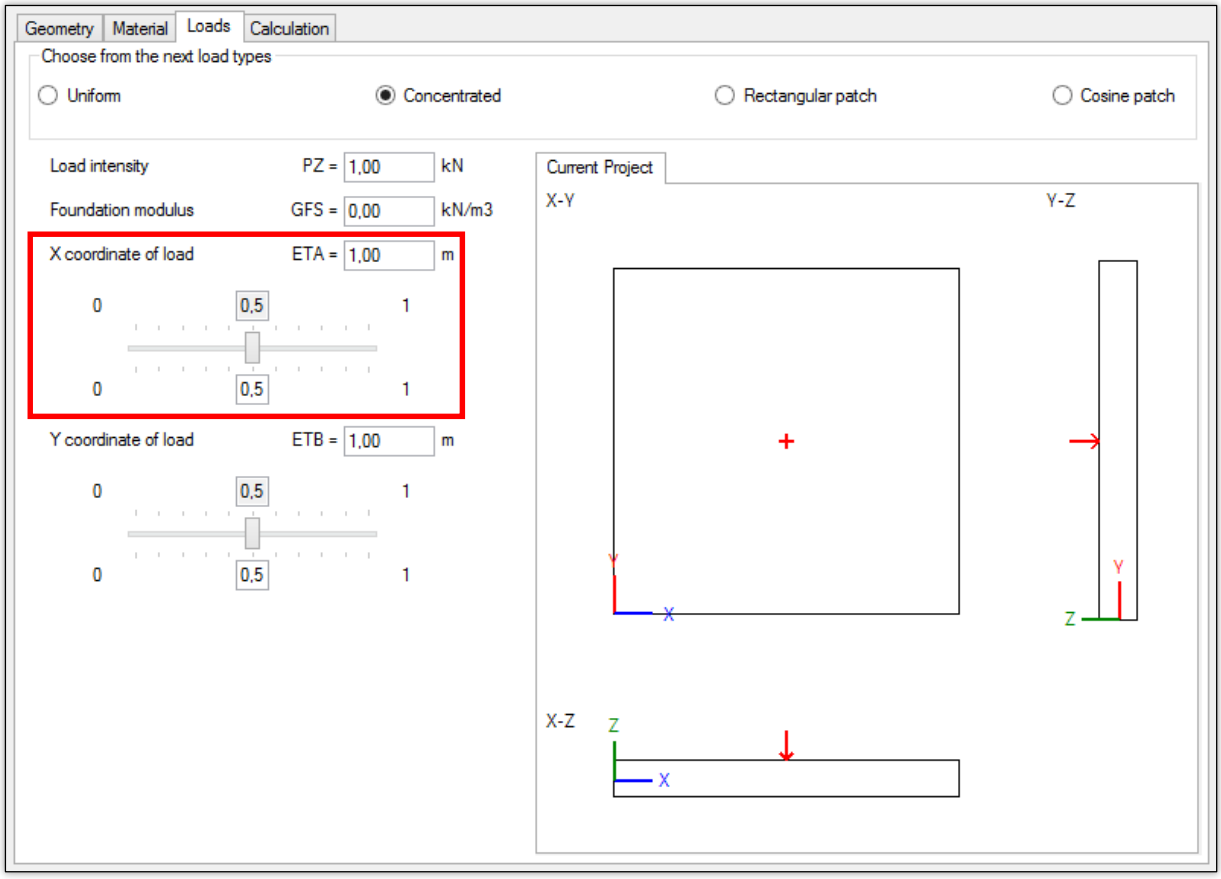


Figure 34 – Clicking on the 'Concentrated' radio button.

textboxes (one of them actually is a `BEN_box`) and a `trackbar`. The first `BEN_scroll` object is bounded by a red rectangle. However the difficulty is not in the number of the components, but their events. Due to its complexity this object cannot be introduced, however the solution of one problem took so long that it worth mentioning.

Figure 36 – Normal and focused appearance of a button.

Sometimes on windows applications an ugly, dotted, gray line appears around a control, for example when it gets focus by pressing the tab button. This behavior can be seen on the middle picture of Figure 35. Indicating to the user that some actions were taken on a

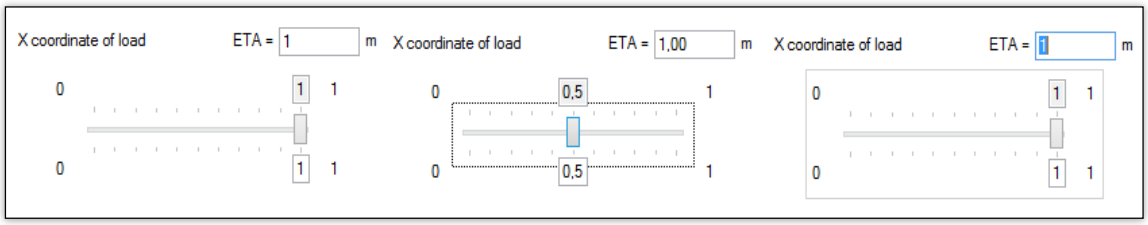


Figure 35 – BEN_scroll object with standard, focused and the corrected appearance.

control, or it has the focus is a good idea, but the dotted rectangle is not. I have tried a lot of ideas to make it disappear, the final solution was to simply add panels with white background and bring them to the front so they simply cover the dotted lines. This is not

the most elegant method, but in fact it works perfectly. In addition I attached to the `GetFocus` events of the controls to draw a large blue rectangle over the `trackbar`.

Material

After clicking on the material tab the program looks like on Figure 37. Most of the tab is

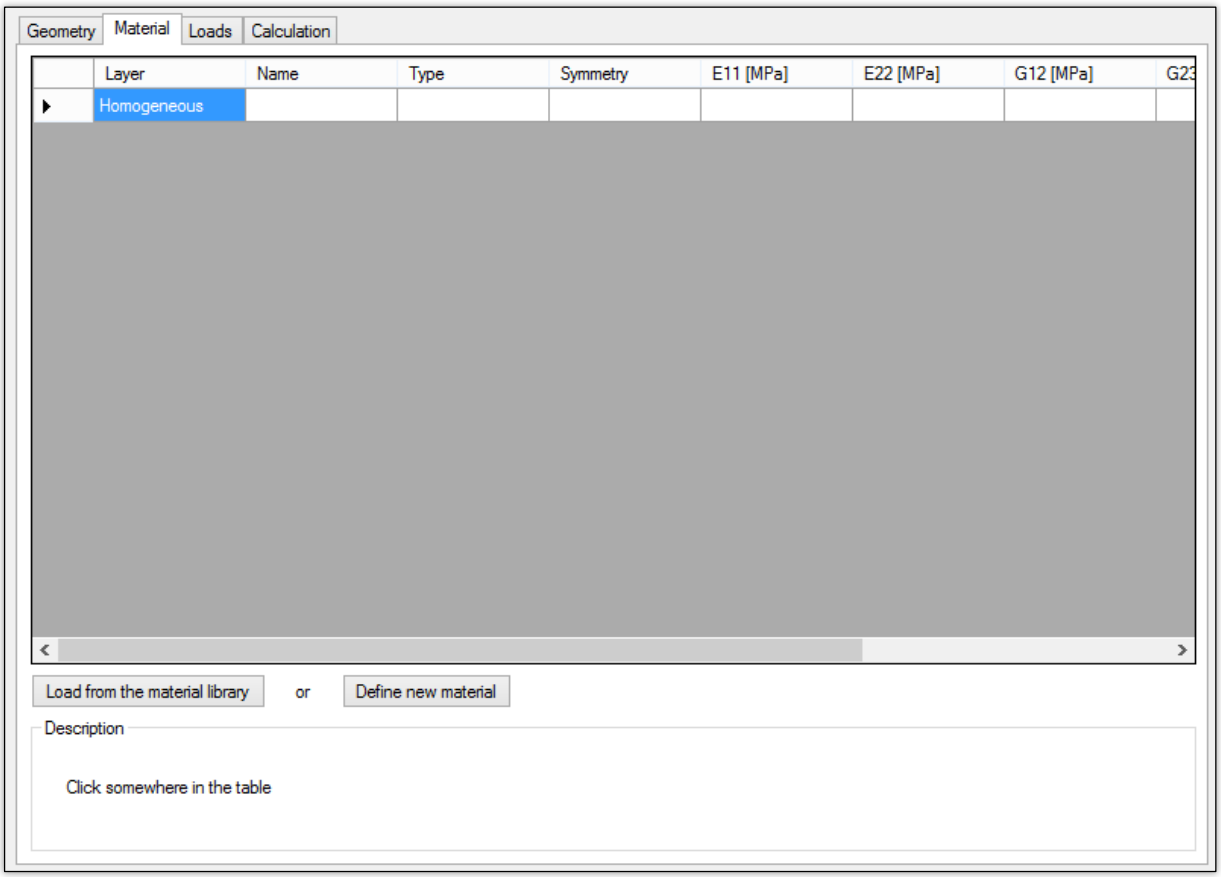


Figure 37 – The 'Material' tab.

possessed by a `DataGridView` object, named `dgvMat`. This object is similar to an excel sheet, but it has no entries, it has instead a source which can be for example a `DataTable` object. Depending on the choice of the section type, the object is filled with one or two empty rows. Upon clicking on some filed in the table, a description helps us to know what is the meaning of the content of the actual cell. Such a description can be managed by creating a string array with as many entries as the number of cells. After it the current state of selection can be determined with the `ColumnIndex` property of the `SelectedCells` property of the `dgvMat` object.

To define a material for the selected layer, the user has two options:

- chose a pre-defined material from the material library or
- define a new material.

Depending on which button was clicked, one of the forms on Figure 38 will pop up.

When defining a new material, the type of material symmetry is already determined by the selected section type, only in the case of a laminated plate is allowed to switch between isotropic or orthotropic material. What worth mentioning is the three radio button

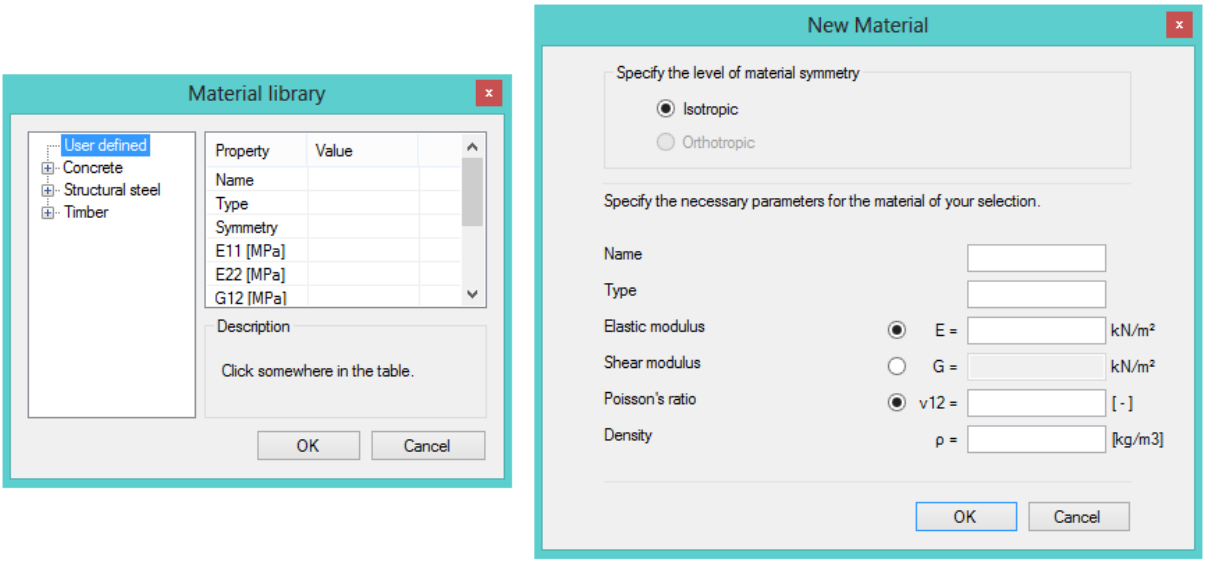


Figure 38 – Material forms.

on the 'New Material' form next to the material parameters E, G and ν_{12}. The perfect choise for this functionality would be the check box object, but due to its misleading visual appearance, the radio button is used instead. As it was mentioned at the introduction of the `ProjectForm` form, when placed on the same container, only one of the applied radio buttons can be clicked by definition. By 'definition' I mean that it is impossible to change that behavior with any property settings. Therefore I applied one panel under each radio button, in which it is placed, so they are contained in different containers and they can be independently modified. However this result is still not desirable, because only 2 of these material parameters are independent, therefore I wrote a simple function which controls the behavior of this three.

In the material library we can see a hive structure on the left. The firs hive is named 'User defined', which is empty by definition. After completing the session in the 'New Material' form, the defined material will take place under this hive. It is important, that changing the section type clears the `dgvMat` object and creates an empty field for the new selection. For this case if the material was a user defined one, it still can be found in the material library for the lifetime of the working session, so until the program is not closed.

Options

Figure 39 – The ToolStrip menu.

The options can be accessed by clicking on the 'gear' button on the `ToolStrip` menu, as on Figure 39. On the appearing form we can set the followings:

- Number of Fourier terms taken into account. The default value is 51.
- Number of output points in one direction. This controls the quality of the isosurfaces that will be introduced in the next chapter, for example on Figure 47. The default value is 50.
- Language.
- Dimension of the geometrical sizes on the geometry tab.

- Dimension of the pressures. This has effect on the distributed loads and the material parameters.
- Dimension of the concentrated forces.
- Dimension of the power type result components, which will be returned from the numerical solver.
- Dimension of the deflection what will be returned from the numerical solver.

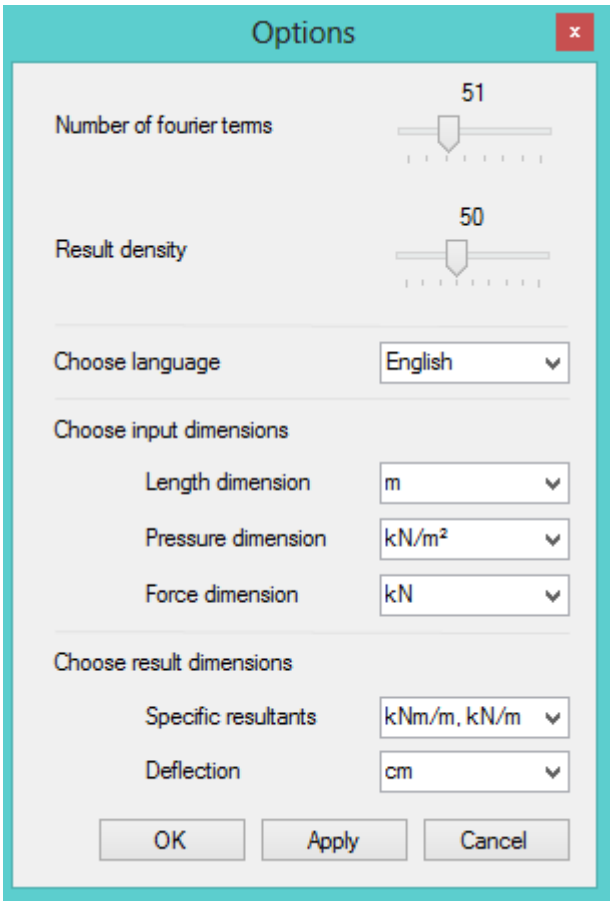


Figure 41 – Options form.

Calculation

Clicking on the 'Calculation' tab, three object welcome us. A `ListView`, a `TabControl` and a `Button` object. The first two is used to facilitate a quick overlook on the settings so far. If these are desirable, we can start the calculation by clicking on the button. The actions what are invoked by the `Click` event of this button deserves a detailed explanation. What is visible is that a new form appears and indicates that some calculations are being carried out in the back. While finishing these, a so called 'knight rider scanner' pleases the eye. This s mall trick is done by setting a gif animation as the `image` property (and not the background) of a `PictureBox`.

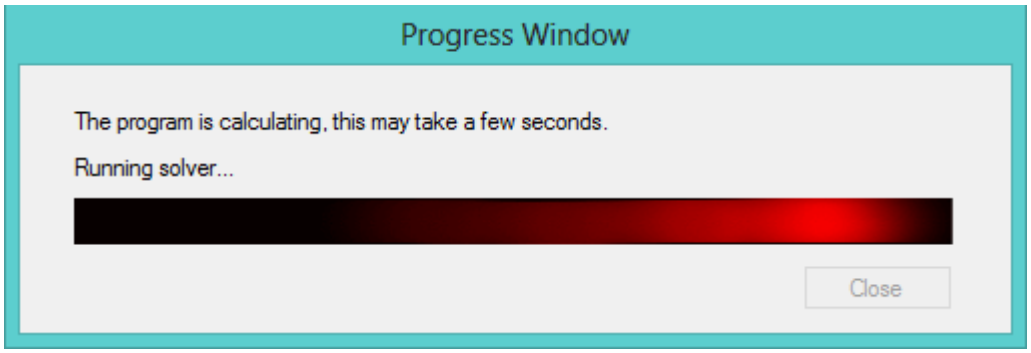


Figure 40 – The Progress Window.

This is an important step, because the numerical solver and the graphical user interface are connected at this point. At first a variable is declared in the declaration section of the form with the following statement:

```
Private p As New Process()
```

When the 'Calcualte' button is clicked, it first writes those txt files, which were mentioned at the introduction of the `DATA()` subroutine of the numerical solver (FORTRAN). The files

are written to that directory which is specified as the working directory (see Figure 39). After, the FORTRAN executable needs to be started at the same place. For this we have to be sure, that this exe file is placed in the working directory, no matter what it is exactly. The code to do is stored in the `Progress()` subroutine on the `ProgressForm` form and is invoked by clicking the 'Calculate' button either. It has the following structure:

```
Public Sub Progress()

        Me.Show()

        lblProg2.Text = Psub1(formMainForm.LangIndex(langOpt))

        textSource = textDir & "\" & executable

        If System.IO.File.Exists(textSource) = False Then
            System.IO.File.WriteAllBytes(textSource, _
                    My.Resources.BENSYS_1_21)
        End If

        p.StartInfo.FileName = textDir & "\" & executable
        p.StartInfo.WorkingDirectory = textDir & "\"
        p.StartInfo.WindowStyle = ProcessWindowStyle.Hidden
        p.Start()

        tim.Enabled = True

    End Sub
```

The interesting part is in the red rectangle. In Visual Studio 2012 every project has resources and these can be managed in the project settings window. The first step is to add the FORTRAN executable to the resources where it will be stored in bytes. In the rectangle, first a string value is assigned to the string variable `textSource`, which should be the full path of the executable with the filename for example 'C:\directory\bensys.exe'. After checking whether the file already exists, the exe is rewritten with the `WriteAllbytes` subroutine under the `System.IO.File` namespace. The next four lines are no mystery. The last command enables a variable that has not mentioned before, a `timer` called `tim`. A timer can be considered as an internal machine which carries out actions when it ticks. The tick frequency of these object determines how often should the object execute the commands that are attached to its tick event. The need for this timer is that we need to know if the process is ended and the FORTRAN executable finished to continue with reading in the results. This could be done with the `WaitForExit` method of the `p` process, but I in that case no any parallel action is enabled, therefore the gif animation (and any other action) will be terminated until the process is working, the system really means to wait for the process to exit. This is outflanked by applying the timer. The `p` progress has a property called `HasExited` which is set to true if the progress is finished. Therefore the timer investigates the value of this property and if it gets 'true', the 'SOLUTION DONE!' message lets know the user that that he can pass to the results.

Results

Since the view that hosts us is quite easy to understand, I won't explain everything but the specialties.

General remarks on creating graphics will be introduced later, but I want to present the method of creating the isosurface picture. First a linear color gradient is produced, composed of 11 color and a linear transition between. When the results are read, the

domain of the plate is divided for small rectangles, the number of them depends on the ‘result quality’ setting, mentioned before. For the first and last color in the ‘gradientbox’ a value is attached, which are always the minimum and the maximum. Similarly a value is attached to every small rectangle, which is the value of the queried result component in that particular point. Then based on the above mentioned bounds of the ‘gradientbox’, the program finds the color to the corresponding result value and fills the rectangle with that color. In limit, when the number of these small rectangles is large enough, the assembly of them gives a smooth 2D isosurface. I find it to be a very good trick, that although only eleven colors were used to create the ‘gradientbox’, the available options for one small rectangle is far more, actually it is bounded by the width of the ‘gradientbox’ only.

The other interesting feature is connected with the save button in the left down corner. When it is pressed, a green message tells the user that the picture is successfully saved and then the text is fading away gradually. I have tried numerous techniques to achieve this and the final solution is by using a timer, actually two. As later it will be introduced, to create graphics I use timers, so there is a timer, let say a global timer, always running when the result tab has the focus, and it ticks in every 10 milliseconds. I declared another color string, but now it contains only shades of green, representing steps of green with decreasing opacity. So the idea is that when the button is clicked, the color of the text is set to green with full opacity, and another timer, let say a local timer is enabled (it was already declared but disabled), which decreases the transparency of the text with one unit in each thick. At the same time the global timer checks the color of the text and determines its opacity. If the global timer perceives that the color of the text is white again, it disables the local timer. For this it is necessary to synchronize the tick frequency of the timers. In

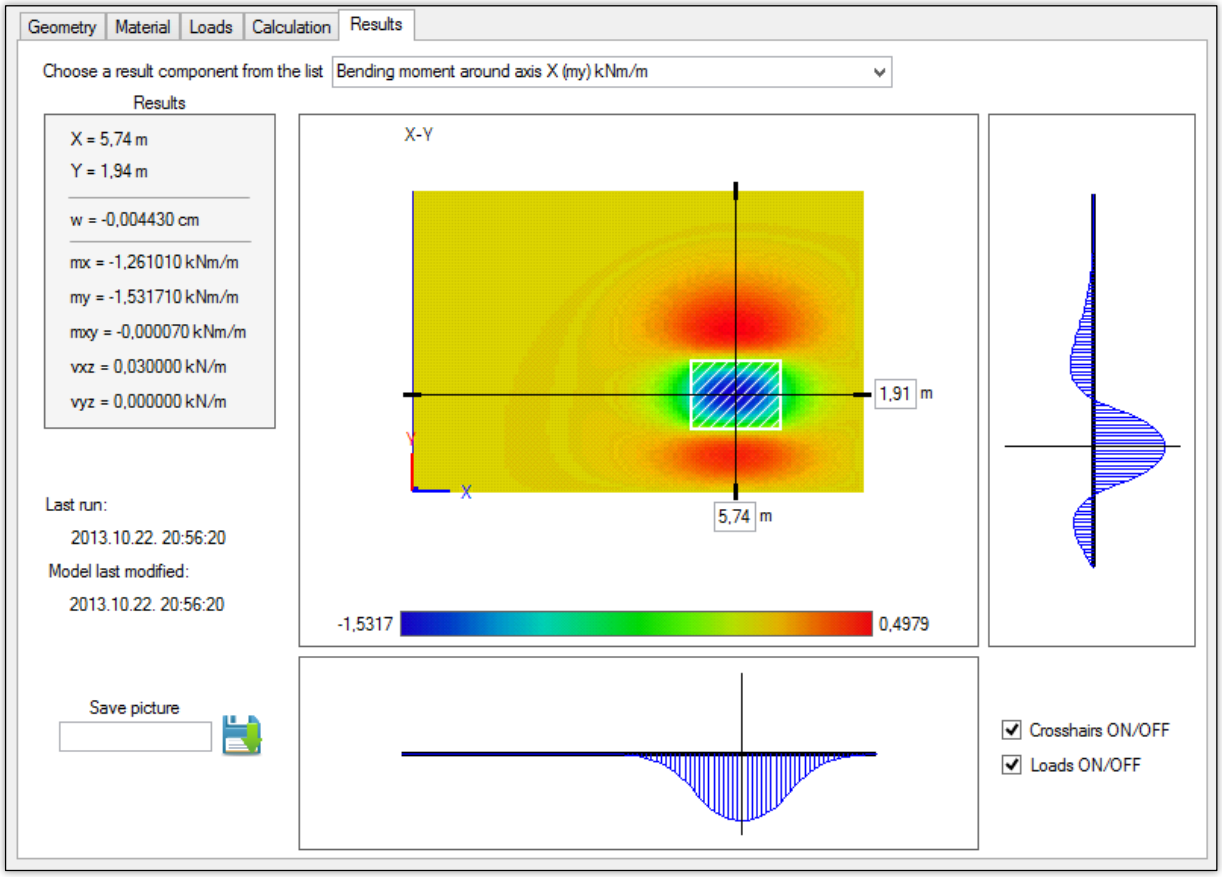


Figure 42 - Results

our case both fires in 10 milliseconds, therefore it cannot be happen that that the global timer does not tick between two ticks of the local timer.

Dimension casting

If we decide to change the unit in the options, it is not a desirable result to leave the already typed values as they were, they should follow the change and be converted. This is solved by defining to arrays. The first is a string array, containing the possible options for a dimension, for example:

```
Public Shared arrayDim() As String = {"m", "dm", "cm", "mm"}
```

The other array should contain integers, as:

```
Public Shared arrayDimVal() As Single = {1, 10, 100, 1000}
```

In this case the base unit is the 'm'. Now every unit has an attached scaling factor and, for instance if a value is in meters, let say 6 m and we want to convert it to centimeters, we have to do the following:

- find the index of 'm' in the first array and search the integer at the specified index in the second array, let it be `integer_m`,
- find the index of 'cm' in the first array and search the integer at the specified index in the second array, let it be `integer_cm`,
- multiple 6 by `integer_m` and divide by `integer_cm`.

The result is of course 0.06. The frequent use of these operations calls for a function. From this point the task is not so complicated, only one thing should be kept in mind. The values in Visual Basic are usually stored as strings, because they are a text of a textbox, etc. Of course when it comes to calculation, these strings are converted to reals for a while, than converted back to string. The importance of this comment will get its meaning later.

I recall the `BEN_class` object now. If one scrolls back, than he will see that the object has its own convert subroutine, which is a method of the object. Hence, if a conversation is needed, it is only necessary to call the `Convert()` method of the `BEN_class` object and the rest is on the code, which had to be formulated only once.

Language casting

The program supports English and Hungarian languages. My solution is the following:

- I defined every text variable as an array, where the same meaning is written in English into the first entry of the array and in Hungarian into the second.
- I defined a function called `LangIndex`, which returns zero or one[11] as the indices corresponding to the English and the Hungarian entries of a string array. The

[11] The arrays in Visual Basic are zero based, so the index of the first entry is zero.

argument of this function should be always one of the strings ‘English’ and ‘Hungarian’. The actual language setting is stored in the variable `langOpt`.

Therefore to assign a value to a string variable has the following form:

```
rbtHomg.Text = sub11(LangIndex(langOpt)),
```

where `sub11()` is a string array and `rbtHomg` is an object with a `text` property.

Graphics

There is no room now to introduce how to create graphics on a form, but they are explained very clearly in the different books of the topic. However there is one thing that worth mentioning.

When being on the load tab let choose the concentrated load. On the right side of the form a picture can be seen, representing the actual position of the defined force. As I mentioned before, to persist graphics on a form I used graphics, however creating a simple picture doesn’t need one. The need for the timer is that I wanted to let the users interact with the picture and define the position of the load with mouse operations only. To introduce the operations of the timers is not feasible due to the simple fact that these are the **longest and most complicated subroutines of the program** by far. The only thing to memorize is that on the action of the tick event of this timer, graphics are created with sizes depending on the textboxes on the left of the pictures. What I did to accomplish the

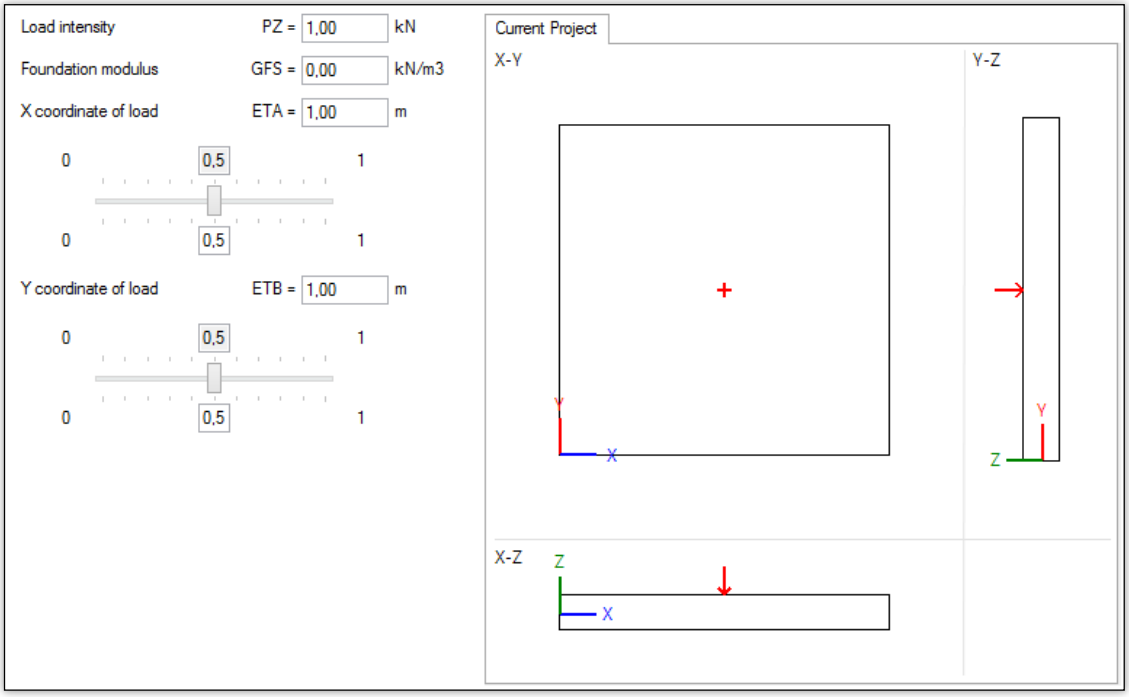


Figure 43 – Concentrated load.

desired functionality is the following:

- I defined a subroutine what determines the necessary values of the ETA and ETB parameters (Figure 43) from the position of the mouse cursor. Here the skill is in the declaration of the subroutine, that is why the commands are missing. Let the arguments be an `object` and a so called `MouseEventArgs` event.

```
Public Sub MousePrint_Load(sender As Object, e As MouseEventArgs)
…
End Sub
```

- I attached the execution of this subroutine to the `MouseClick` and `MouseMove` events of the picturebox that contains the picture.

```
Private Sub pboxLoadXY_MouseClick(sender As Object, e As
MouseEventArgs) Handles pboxLoadXY.MouseClick

        MousePrint_Load(sender, e)

    End Sub

    Private Sub pboxLoadXY_MouseMove(sender As Object, e As
MouseEventArgs) Handles pboxLoadXY.MouseMove

        MousePrint_Load(sender, e)

    End Sub
```

We can see, that the `sender` and `MouseEventArgs` argumentums of the events are passed as arguments of the `MousePrint_Load()` subroutine. In case of an event, the `sender` parameter always holds a reference to the control causing the event, which is now the mouse. The other argument, the `e` parameter always contains some information about the event. For example in the subroutine `MousePrint_Load()` the 'x' position of the mouse cursor can be queried by typing `e.X`, as this position is the 'X' property of the mouse event.

Save and Open

It is hard to imagine a self-respecting program without the ability to save and restore a working session. In this case it is solved by writing and reading txt files with the `SteamWriter` and `SteamReader` classes of the `System.IO` namespace. BENSYS is also capable of determining whether the txt file to open were saved from BENSYS or not. It is done by writing a particular string into the first line of the saved txt file, so when opening, the existence of this string can be checked.

4. VERIFICATION

It is not expectable from any user to use an application whose results are not reliable. For this reason this chapter is dedicated to confirm the results with the help of finite element programs which have already proven. Actually I used for this purpose the AXIS VM for the simpler and ANSYS for the more complicated problems. Next to the numerical accuracy, the quality of the results, namely the skills of graphical representation will be contrasted as well.

4.1. Verification of the load cases

In this section we can kill two birds with one stone. The calculation method is based on the single layer technique, which means that equivalent plate rigidities are calculated for each type of cross section. It entails, that the correctness of the calculations after this step are independent from the correctness of the different cross sections. Simply speaking, if the results of a loading type are proved with one of the cross section types, this remains valid with the other section types too. Therefore it is logical to choose that section type from which we account to get less errors, the homogeneous one.

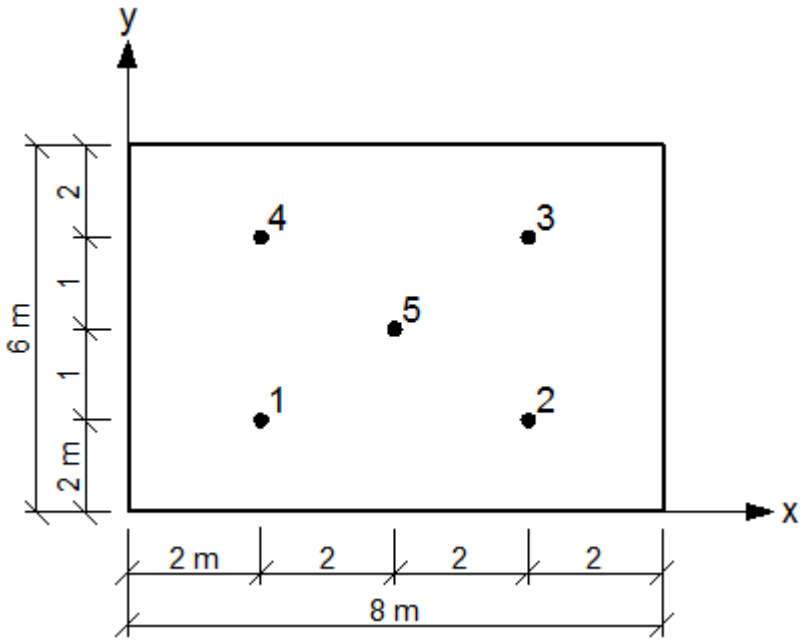


Figure 44 – Points of measurement.

According to its simplicity, the AXIS VM was chosen for this task. For the basis of the comparisons I used an **8m x 6m plate** with an overall thickness of 20 cm and C16/20 concrete material. The results are measured in 5 points on the plate, by the means of Figure 44.

Uniform load

The loading is represented on Figure 45 and the results are summarized in Table 4. The results were obtained with taking into account 51 Fourier terms and the differences can be seen in percentage in Table 5.

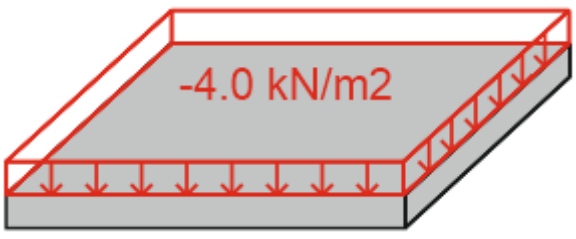


Figure 45 – Uniform load.

It is very important, that the *location of the measurement points in BENSYS and AXIS are not the same*, because of the

Uniform load, PZ = -4 kN/m^2										
	1		2		3		4		5	
	BENSYS	AXIS	BENSYS	AXIS	BENSYS	AXIS	BENSYS	AXIS	BENSYS	AXIS
ez [mm]	-1.1692	-1.1224	-1.1690	-1.1630	-1.1690	-1.2250	-1.1692	-1.1630	-1.8075	-1.8570
mx [kNm/m]	-5.0000	-5.2200	-5.0000	-5.0300	-5.0000	-5.2200	-5.0000	-5.0300	-6.3546	-6.4700
my [kNm/m]	-6.7670	-7.0200	-6.7670	-6.7500	-6.7670	-7.0300	-6.7670	-6.7500	-9.8372	-10.0500
mxy [kNm/m]	1.8383	1.7700	-1.8383	-2.0100	1.8383	1.7700	-1.8383	-2.0100	-0.0012	0.0000
vxz [kN/m]	-2.5160	-2.5300	2.5160	2.5500	2.5160	2.5200	-2.5160	-2.5500	0.0480	-0.0600
vyz [kN/m]	-2.1700	-2.0500	-2.1700	-2.2700	2.1700	2.0600	2.1700	2.2700	-0.0830	-0.0700

Table 4 – Results from BENYS and AXIS for a uniform load.

mentioned 'result quality setting'. Since BENSYS does not interpolate between the results, **a value is valid for a range, instead of a point**. This is not true for AXIS, where according to the interpolation, the results can be obtained at any point of the plate even if the result

Uniform load, PZ = -4 kN/m2					
	1	2	3	4	5
ez [mm]	4.00%	0.51%	4.79%	0.53%	2.74%
mx [kNm/m]	4.40%	0.60%	4.40%	0.60%	1.82%
my [kNm/m]	3.74%	0.25%	3.89%	0.25%	2.16%
mxy [kNm/m]	3.72%	9.34%	3.72%	9.34%	100.00%
vxz [kN/m]	0.56%	1.35%	0.16%	1.35%	225.00%
vyz [kN/m]	5.53%	4.61%	5.07%	4.61%	15.66%

Table 5 – Differences in the results in case of a uniformly distributed load.

will be 'exact' only in the nodes either. For example if we compare the results of m_{xy} in point 5, we see that AXIS produces the correct value which is zero. Since BENSYS does not calculate in the center of the middle plane, the result cannot be zero, which automatically results a 100% difference in the results, but don't let it disturb us. Obviously this phenomenon is true for every measurement point, but due to the double symmetry of the in plane geometry of the plate the **differences are naturally higher in the middle point**.

Deflection of the middle point		
Fourier terms	$e_{z.BENSYS}$ [mm]	$e_{z.AXIS}$ [mm]
30	-2.079	-2.489
40	-2.415	
50	-2.417	
60	-2.418	
70	-2.420	
80	-2.421	
90	-2.422	

Table 6 – Parametric study on the number of Fourier-terms.

To make sure, I made a parametric study on the number of the Fourier-terms taken into account. For this I have chosen the deflection of the middle point as the basis of the comparison. The results are ordered in Table 6. I observed that above 40 terms the accuracy does not improve, and has a degeneracy under, so I made a lower limit of these value in the program to be 50.

Finally I present the results in the form of a 2D isosurface of both AXIS and BENSYS.

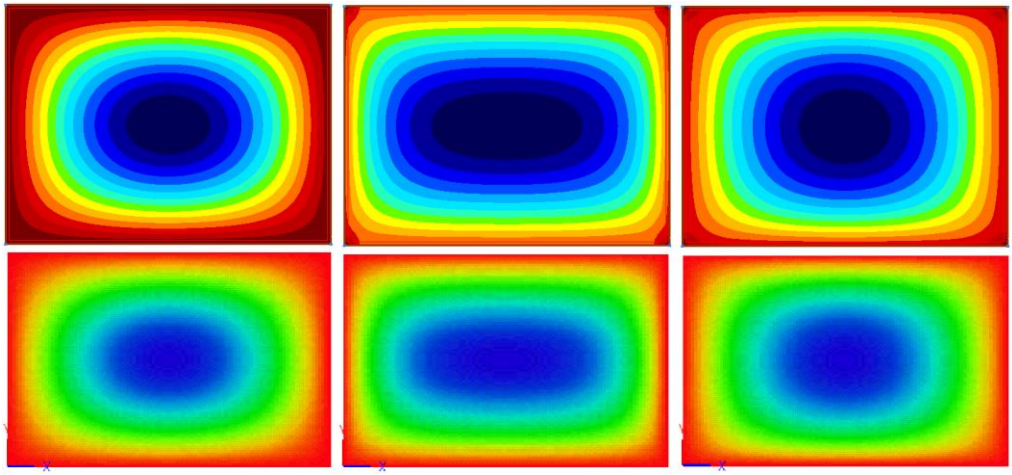


Figure 47 – Isosurfaces about the deflection e_z, bending moment mx and my from AXIS in the top line and from the BENSYS in the bottom line.

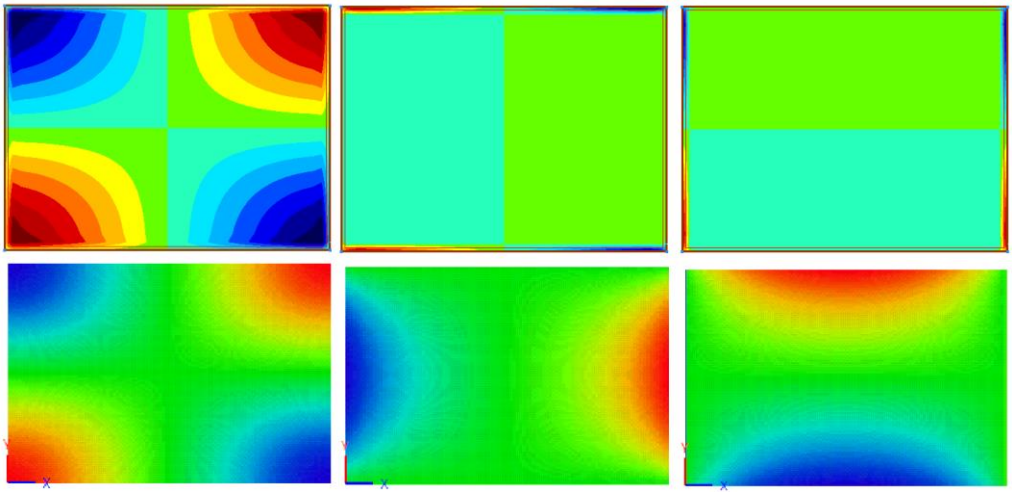


Figure 46– Isosurfaces about the twisting moment mxy and shear forces vxy and vyz from AXIS in the top line and from the BENSYS in the bottom line.

Concentrated load

The loading is represented on Figure 48. The results were obtained with taking into account 51 Fourier terms and the differences can be seen in percentage in Table 8.

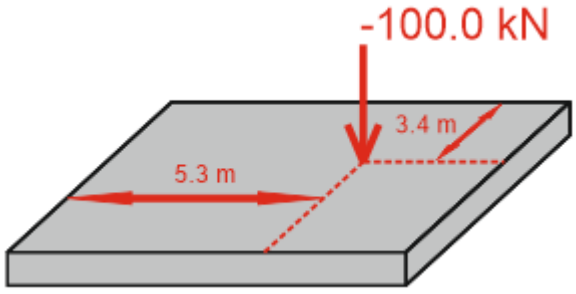


Figure 48 – Concentrated load

In this case the match between the results is excellent, so there were no reason to make a parametric study, the number of 50 for the minimum of the Fourier-terms is acceptable for this case too. However, at point 5 a relatively large difference can be seen in the results, but this is no surprise, since it is the nature of Fourier transformation. Sudden changes and jumps are hard to follow by this technique, and in fact, point 5 is the closest to the position of the concentrated force. I repeated the calculation with 100 Fourier-terms taken into account and the result haven't changed, so we just have to accept that the method – just like any other - has weaknesses and it is one of them.

The results are again presented in 2D isosurfaces on Figure 49 and Figure 50.

Concentrated load, PZ = -100 kN										
	1		2		3		4		5	
	BENSYS	AXIS	BENSYS	AXIS	BENSYS	AXIS	BENSYS	AXIS	BENSYS	AXIS
ez [mm]	-0.8828	-0.8830	-1.7838	-1.8310	-1.5320	-1.5770	-0.8452	-0.8470	-2.0797	-2.4890
mx [kNm/m]	-0.2570	-0.2700	-10.8548	-11.1400	-8.6170	-8.8600	-0.5428	-0.5600	-6.1126	-6.1500
my [kNm/m]	-4.7698	-4.7600	-12.7618	-13.0300	-6.1685	-6.3800	-4.0453	-4.0700	-13.4760	-13.4700
mxy [kNm/m]	2.0546	2.0800	-4.6789	-4.7100	3.7920	3.8000	-2.3255	-2.3400	-1.4296	-1.5400
vxz [kN/m]	-2.8430	-2.8200	13.8030	13.7300	5.1030	5.0400	-2.4500	-2.4400	-10.5150	-10.1500
vyz [kN/m]	-1.1760	-1.1900	-12.6540	-12.3300	8.3630	8.6500	1.3980	1.3700	2.9660	3.0300

Table 8 – Results from BENYS and AXIS for a concentrated load.

Concentrated load, PZ = -100 kN					
	1	2	3	4	5
ez [mm]	0.02%	2.65%	2.94%	0.21%	19.68%
mx [kNm/m]	5.06%	2.63%	2.82%	3.17%	0.61%
my [kNm/m]	0.21%	2.10%	3.43%	0.61%	0.04%
mxy [kNm/m]	1.24%	0.66%	0.21%	0.62%	7.72%
vxz [kN/m]	0.81%	0.53%	1.23%	0.41%	3.47%
vyz [kN/m]	1.19%	2.56%	3.43%	2.00%	2.16%

Table 7 – Differences in the results in case of a concentrated load.

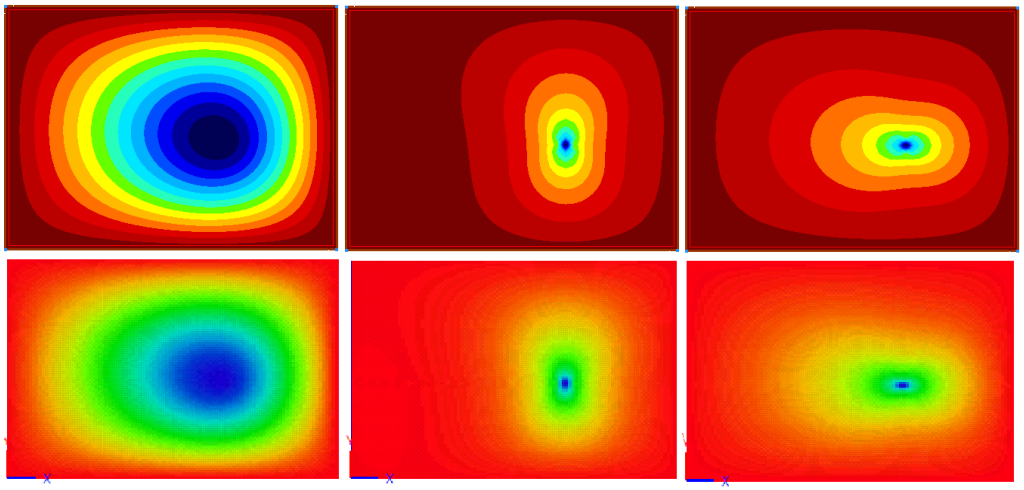


Figure 49 – Isosurfaces about the deflection e_z, bending moment mx and my from AXIS in the top line and from the BENSYS in the bottom line.

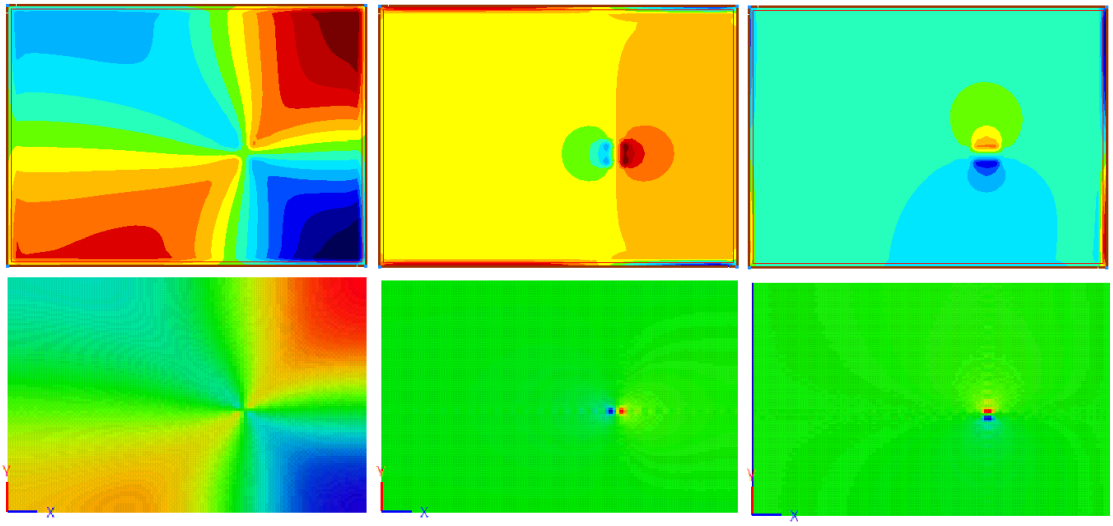


Figure 50 – Isosurfaces about the twisting moment mxy and shear forces vxy and vyz from AXIS in the top line and from the BENSYS in the bottom line.

Rectangular patch

The loading is represented on Figure 51. The results were obtained with taking into account 51 Fourier terms and the differences can be seen in percentage in Table 9.

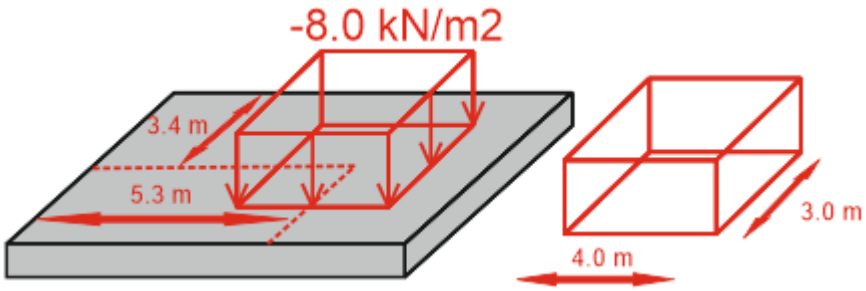


Figure 51 – Rectangular patch.

Rectangular path										
	1		2		3		4		5	
	BENSYS	AXIS	BENSYS	AXIS	BENSYS	AXIS	BENSYS	AXIS	BENSYS	AXIS
ez [mm]	-0.7722	-0.7730	-1.2490	-1.2860	-1.1470	-1.1820	-0.7404	0.7420	-1.6140	-1.6490
mx [kNm/m]	-0.9158	-0.9100	-7.0510	-7.2400	-6.0890	-6.2600	-1.0306	-1.0300	-6.9475	-7.0000
my [kNm/m]	-4.2926	-4.2900	-8.4720	-8.6800	-6.3170	-6.4900	-3.6809	-3.7000	-9.8730	-10.0000
mxy [kNm/m]	1.6806	1.7100	-1.6567	-1.6700	1.8820	1.8900	-1.9147	-1.9400	-0.2013	-0.2400
vxz [kN/m]	-2.9880	-2.9200	2.8950	2.8400	2.4710	2.4400	-2.5410	-2.5000	-3.1340	-3.2200
vyz [kN/m]	-1.1990	-1.1900	-3.2490	-3.2500	6.1010	6.1100	1.4590	1.4300	0.9920	1.1300

Table 9 – Results from BENYS and AXIS for a rectangular patch load.

The differences at this load case does not justify to raise the number of the terms, so again, the value of 50 seems to be appropriate. Again, the statements made at the previous load case are valid now either. As the geometry of the load contains edges, it is logical to expect that from the result functions too and this sharp parts are harder to follow

with Fourier transform. These and the difference in the exact location of the measurement points can cause a few salient difference value in Table 10.

Rectangular path, PZ = -4 kN/m2					
	1	2	3	4	5
ez [mm]	0.10%	2.96%	3.05%	0.22%	2.17%
mx [kNm/m]	0.63%	2.68%	2.81%	0.06%	0.76%
my [kNm/m]	0.06%	2.46%	2.74%	0.52%	1.29%
mxy [kNm/m]	1.75%	0.80%	0.43%	1.32%	19.23%
vxz [kN/m]	2.28%	1.90%	1.25%	1.61%	2.74%
vyz [kN/m]	0.75%	0.03%	0.15%	1.99%	13.91%

Table 10 – Differences in the results in case of a concentrated load.

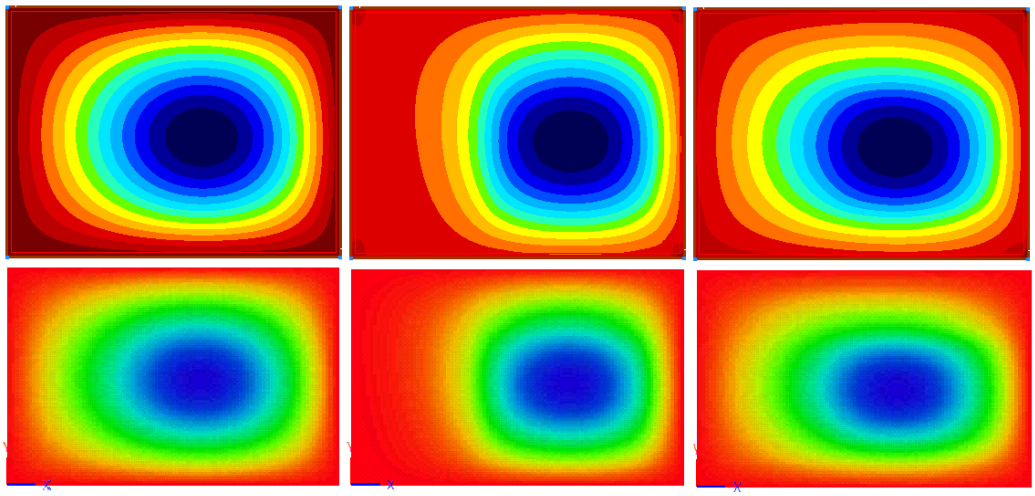


Figure 52– Isosurfaces about the deflection e_z, bending moment mx and my from AXIS in the top line and from the BENSYS in the bottom line.

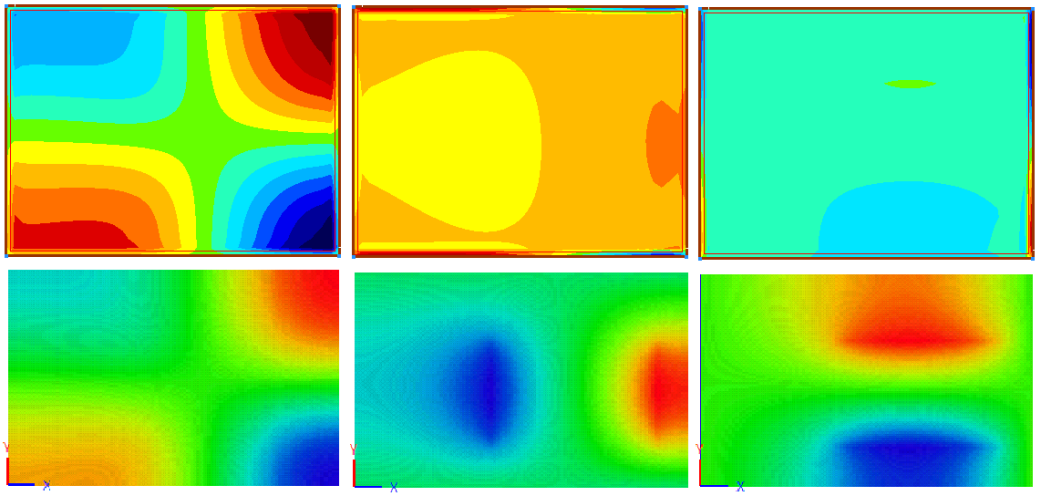


Figure 53 – Isosurfaces about the twisting moment mxy and shear forces vxy and vyz from AXIS in the top line and from the BENSYS in the bottom line.

Pyramid patch

The loading is illustrated on Figure 55. According to the relative complicated load function a different kind of discussion is needed here. The modelling of this load could hardly be done with AXIS, so I wrote a script file in ANSYS and gave the nodal loads of each node

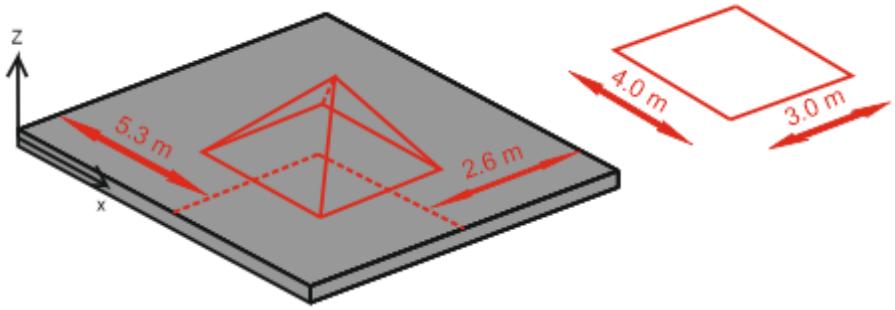


Figure 55 – Pyramid patch load.

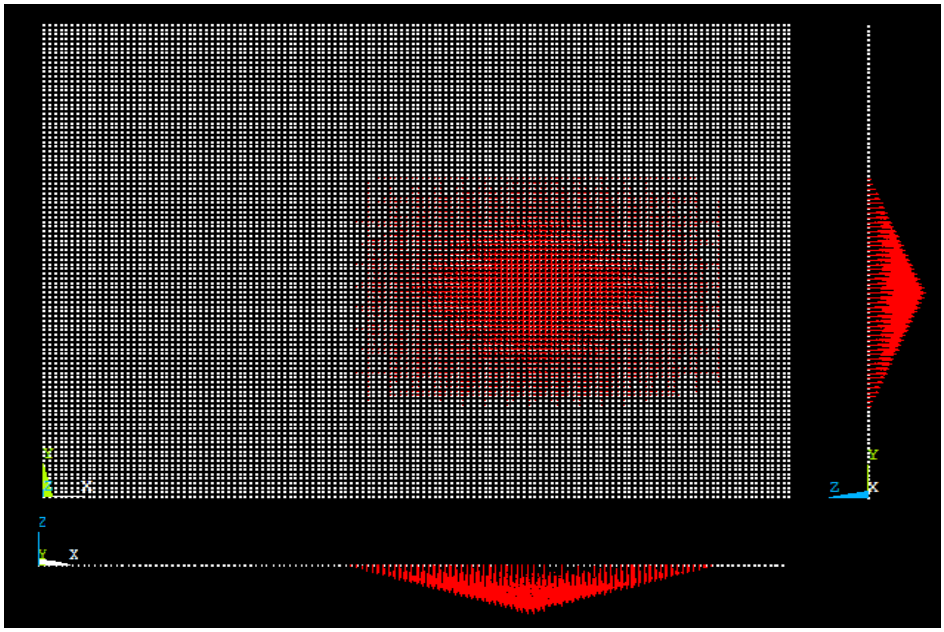


Figure 54 – Nodal forces in ANSYS.

separately, according to the first expression on Figure 24. These loads can be seen from three directions on Figure 54. The results were obtained with taking into account 51 Fourier terms and the differences can be seen in percentage.

Pyramid										
	1		2		3		4		5	
	BENSYS	ANSYS	BENSYS	ANSYS	BENSYS	ANSYS	BENSYS	ANSYS	BENSYS	ANSYS
ez [cm]	-0.1014	-0.1021	-0.1835	-0.1874	-0.1631	-0.1681	-0.0970	-0.0984	-0.2257	-0.2291
error	0.69%		2.13%		3.07%		1.42%		1.51%	

Table 11 – Comparison of the deflections in the measurement points o fthe ANSYS and the BENSYS.

The results show perfect match.

Self-weight

It was already mentioned, that the program is able to calculate the self-weight of a plate based on the geometry and the density of the applied material and then calculate a uniformly distributed load from it. The results can be seen in Table 12.

a=8 m, b=6 m, t=0.2 m			
section type	maximum deflection [cm]		error
	ANSYS	BENSYS	
homogeneous	-0.2727	-0.2668	2.21%
sandwich	-1.4360	-1.4238	0.86%
voided	-0.0976	-0.1002	2.59%
laminated	-25.8440	-25.7440	0.39%

Table 12 – Differences between ANSYS and BENSYS in the maximum deflection for a self-weight load.

Winkler foundation

For the comparison I used now the AXIS program because of the surface support capability what makes it easy to take into account the elastic foundation. The foundation modulus were taken to be 0.25 N/mm^3. I investigated the effect with each load type, except the pyramid patch which is not feasible with axis. The differences in the maximum deflection are represented in Table 13 and also some impressive graphics are compared on Figure 56.

a=8 m, b=6 m, t=0.2 m			
load type	maximum deflection [cm]		error
	AXIS	BENSYS	
uniform	-0.0018	-0.0018	0.00%
concentrated	-0.0021	-0.0020	5.00%
rectangular	-0.0044	-0.0044	0.00%

Table 13 - Differences between AXIS and BENSYS in the maximum deflection for various load types supported by Winkler foundation.

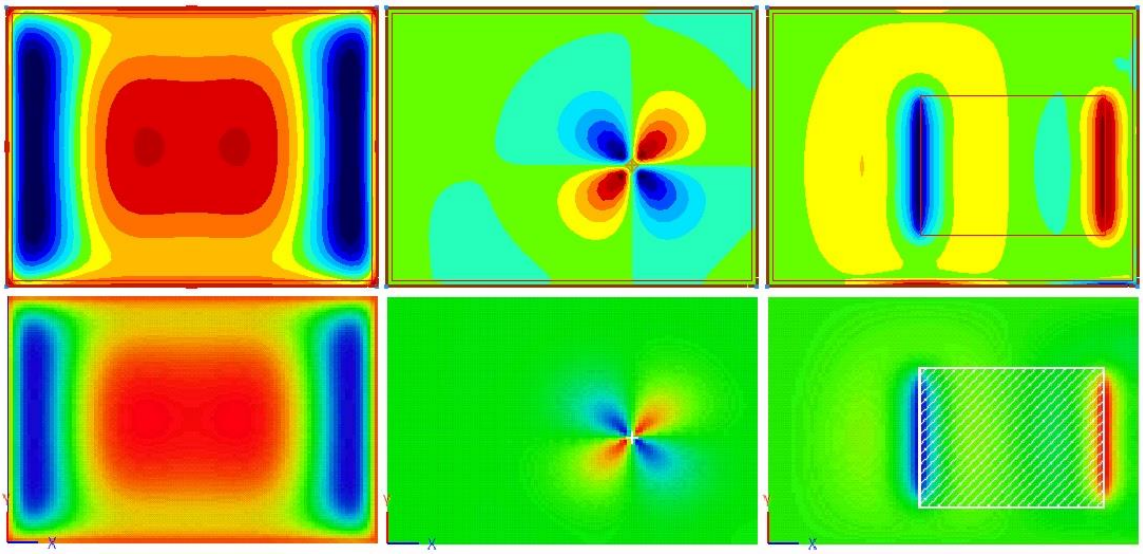


Figure 56 – Isosurfaces of bending moment mx, twisting moment mxy and shear force vxz due to uniform, concentrated and rectangular patch load cases, respectively.

4.2. Verification of the single layer models

In this point the verification of the different equivalent single layer rigidities will be performed. The in plane dimensions and the location of the measurement point will be the same as before, but now only the deflection of the middle point will be analyzed as a result of a uniformly distributed force of intensity -4.0 kN/m^2. Because the calculation method of the result components and the other loading cases has been already proved, it is sufficient to only compare the deflections of the BENSYS and the ANSYS in case of the different section types.

For the ANSYS models I wrote script files in ANSYS APDL and these are attached to the thesis in Appendix B. For the homogeneous, sandwich and laminated types I used a Shell181 element, with multilayer options in the latter two cases. The voided plate was modelled using 3D Solid186 brick element.

Homogeneous plate

The measurements of the previous point were carried out on a homogeneous plate, so there is no need to further prove the correctness of this model.

Sandwich plate

The task is to evidence the correctness of the equivalent single layer rigidities of equations (2.56)-(2.59). Because the theory applied to sandwich plates is an approximation, it follows that it has a range where it provides satisfactory results and a range where it doesn't. Through this investigation fixed the in plane dimensions and regarded the core/face thickness

	material	E [GPa]	v [-]	G [GPa]
face	aluminium	73.400	0.320	27.800
core	pvc foam	0.286	0.300	0.110

Table 15 – Material properties of the analyzed sandwich plate.

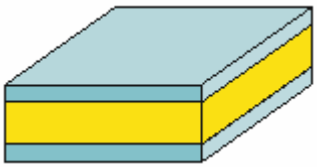


Figure 57 – Sandwich plate.

	defelction [cm]		face material / thickness [cm]	core material / thickness [cm]	core/face thickness	error
	ANSYS	BENSYS				
1	-3.612	-3.803	alu/0.1	pvc foam/15	150	5.29%
2	-2.013	-2.168	alu/0.1	pvc foam/20	200	7.70%
3	-8.170	-8.410	alu/0.1	pvc foam/10	100.00	2.94%
4	-4.251	-4.215	alu/0.2		50.00	0.85%
5	-2.870	-2.817	alu/0.3		33.33	1.85%
6	-2.173	-2.119	alu/0.4		25.00	2.49%
7	-1.746	-1.701	alu/0.5		20.00	2.58%
8	-1.457	-1.424	alu/0.6		16.67	2.26%
9	-1.249	-1.225	alu/0.7		14.29	1.92%
10	-0.870	-0.871	alu/1.0		10.00	0.11%
11	-0.721	-0.735	alu/1.2		8.33	1.94%
12	-0.570	-0.600	alu/1.5		6.67	5.26%
13	-0.498	-0.538	alu/1.7		5.88	8.03%
14	-0.417	-0.469	alu/2.0		5.00	12.47%

Table 14 – Configurations of sandwich plates.

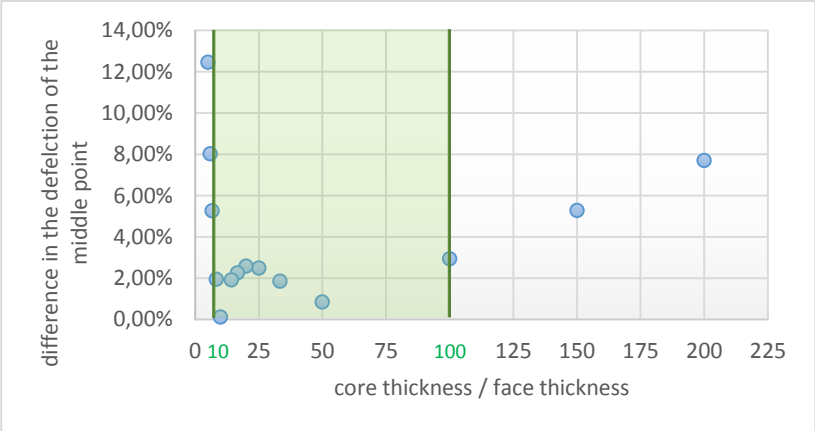


Diagram 1 – Differences in the deflection of the middle point with respect to the core/face thickness ratio.

ratio as a free parameter. The material properties of the core and the face are summarized in Table 15. The results are summarized in Table 14 and are visualized on Diagram 1. It can be diagnosed that for sandwich plates with **core/face thickness ratios in the range of 10 to 100** we will get reliable results.

Voided plate

The case is just the same as above, I had to find the range of exactness of expressions (2.62)-(2.67). The in plane dimensions and the overall thickness were fixed as 8 m, 6 m and 0.3 m, but now the number of free parameters is much more, which state requires to investigate different voided plates in high variety. This variety shows in the different

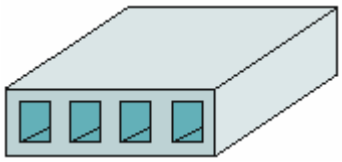


Figure 59 – Voided plate.

a=8 m, b=6 m, t=0,3 m							
	ansys	bensys	width	height	width/height	cells	err
conf1	-0.0595	-0.0669	50	5	10.00	11	✗ 11.00%
conf2	-0.0567	-0.0620	10	5	2.00	40	✗ 8.52%
conf3	-0.0476	-0.0926	50	30	1.67	11	✗ 48.58%
conf4	-0.0513	-0.0936	54	30	1.80	11	✗ 45.25%
conf5	-0.0607	-0.0602	10	15	0.67	30	✓ 0.83%
conf6	-0.0580	-0.0570	5	15	0.33	30	✓ 1.77%
conf7	-0.0593	-0.0575	5	15	0.33	40	✓ 3.13%
conf8	-0.0571	-0.0580	5	15	0.33	50	✓ 1.56%
conf9	-0.0607	-0.0633	15	15	1.00	30	✓ 4.11%
conf10	-0.0609	-0.0607	11	15	0.73	30	✓ 0.33%
conf11	-0.0617	-0.0613	12	15	0.80	30	✓ 0.65%
conf12	-0.0603	-0.0597	9	15	0.60	30	✓ 1.01%
conf13	-0.0597	-0.0593	8	15	0.53	30	✓ 0.67%
conf14	-0.0572	-0.0576	4	15	0.27	30	✓ 0.67%
conf15	-0.0575	-0.0572	3	15	0.20	30	✓ 0.52%
conf16	-0.0545	-0.0591	5	15	0.33	25	✗ 7.78%
conf17	-0.0563	-0.0588	5	15	0.33	26	✓ 4.25%
conf18	-0.0573	-0.0586	5	15	0.33	27	✓ 2.22%
conf19	-0.0591	-0.0576	5	15	0.33	35	✓ 2.60%
conf20	-0.0587	-0.0577	5	15	0.33	33	✓ 1.73%
conf21	-0.0593	-0.0574	5	15	0.33	40	✓ 3.31%
conf22	-0.0575	-0.0613	5	15	0.33	20	✗ 6.20%

Table 16 – Different configurations of voided plates.

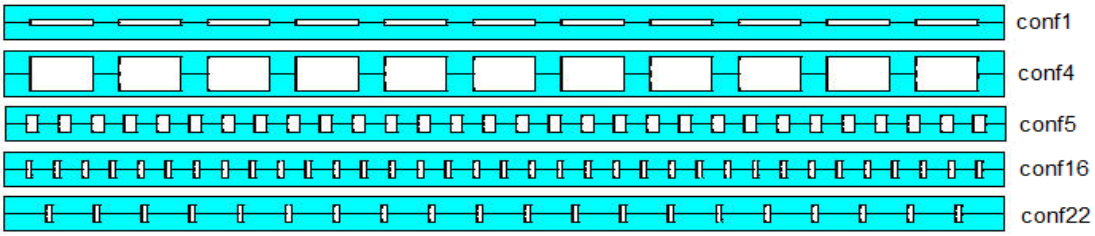


Figure 58 – Cross section of some configurations.

geometry and spacing of the voids. The different configurations are summarized in Table 16 and some of them is visualized on Fgure 58. The table is divided into 3 parts. In the first part the width/height ratio was over 1 and wrong results were obtained. In the second part I fixed the height and number of the voids on a constant value and investigated the results on changing the width of the voids. In the last part only the number of the voids was a free parameter, the other three was taken like at the best results at the second part.

It can be stated, that to get correct results, the width/height ratio of the voids must be under 1.0. Moreover, if the plate has reasonable cross sectional dimensions, the accuracy of the calculations is not in danger. To determine whether a cross section is reasonable or not is not a strict thing, however at now it seems that the web thickness should be in magnitude with the flange thickness.

Laminated plate

Although the code of the corresponding calculations is the longest, laminated plates have a clear theory with less simplifications. The in plane dimensions are again 8 m and 6 m, while in each configuration the overall thickness of the layers is 20 cm. The configurations and the results can be seen in Table 17. The measurements were carried out on a plate with orthotropic material, subjected to a uniformly distributed load. The obtained results fulfill the expectations and shows excellent match with the finite element calculations of ANSYS.

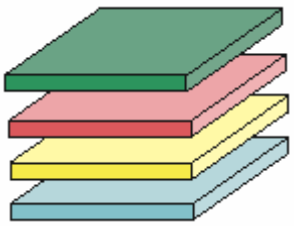


Figure 60 – Laminated plate.

a=8 m, b=6 m, t=0.2 m						
	maximum deflection [cm]		layers	orientation	thickness	error
	ansys	bensys				
conf1	-20.4524	-20.4486	2	00	equal	0.0%
conf2	-20.4524	-20.4486	4	0000	equal	0.0%
conf3	-20.4524	-20.4486	10	0000000000	equal	0.0%
conf4	-8.9434	-9.1001	10	1111111111	equal	-1.8%
conf5	-11.6040	-11.6376	10	1010110101	equal	-0.3%
conf6	-17.3387	-17.2474	10	0010110100	equal	-0.5%
conf7	-18.5613	-18.4896	10	0010110101	varying	-0.4%

Table 17 – Different configurations of laminated plates.

5. OUTCOME

5.1. Comparison with other software

Let me compare the different software that were used through the chapter of the thesis from the point of view of overall completion time of an arbitrary task. As the problem to solve I have chosen a moderately difficult one, the users had to solve a homogeneous plate subjected to a rectangular path load. It were considered as solved when the user could manage to determine the deflection in the middle point of the plate. At each test I measured the necessary time to complete the task. The results can be seen on Diagram 2, where the time values are represented in minutes. I tested the programs with 4 students and 3 advanced users of each program under discussion to have more or less

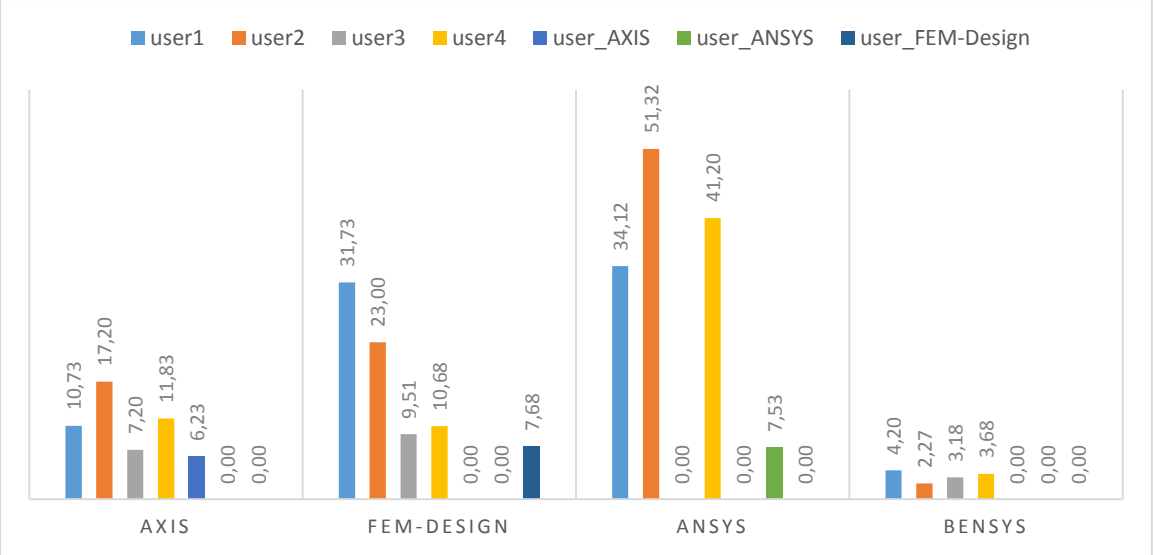


Diagram 2 – Comparison of solution time.

representative results. The experts solved the task only with their favorite, that's why zeros occur in the diagram. However, I want to emphasize, that these are **very subjective results**, concerning only one particular type of problem which is actually the objective of BENSYS and this comparison cannot provide a basis to set up an order between the programs. Though, as it was mentioned in the preliminary of the thesis, the major objective was precisely to target some configurations and provide an easy way of calculation to them. From this point of view the above table shows that this goal is reached as the solution time remained under the 5 min limit in each case. Generally it can be sad that the results are satisfactory for the aimed purposes and the graphics became clearly a strength of the program.

5.2. Future plans

Programming such a software is a never ending story in the meaning that it only can be discontinued, but not finished. Accordingly, a lot of ideas are waiting to be implemented in the program. First of all, it is considered to rewrite the numerical solver in a modern language to avoid compatibility problems with the different systems. Next to technical development, the program could be complemented with the following abilities:

- Take into account different boundary conditions.

- Use superposition to allow the definition of load groups.
- Derive the expressions for other load cases and section types.
- Create a module to calculate the necessary amount of reinforcement of a concrete plate.
- Level-up the graphics to 3D.
- Improve data storage by using one or more database.
- Improve the visual appearance.

If any of these plans come true in the future, the up to date program will be available for download at the web site of the Department of Structural Mechanics (http://www.me.bme.hu).

BIBLIOGRAPHY

Airoldi, A. (n.d.). Plates 1 - Classical Plate Theory. POLITECNICO DI MILANO, Italy.

Airoldi, A. (n.d.). PLates 2 - Reissner - Mindlin Plate Theory. POLITECNICO DI MILANO, Italy.

Ashton, J. E. (n.d.). *Theory of Laminated Plates.* USA: Technomic.

Basu-Dawson. (1970). Orthotropic Sandwich Plates. *Proc. Instn. Civ. Engrs.*, 87-115.

BigResource. (n.d.). Retrieved from http://www.bigresource.com

Brainerd-Goldberg-Adams. (1996). *Programmer's Gude to Fortran 90* (Third Edition ed.). USA: Springer.

Bytes. (n.d.). Retrieved from http://www.bytes.com

Foxall, J. (2010). *Sams Teach Yourself Visual Basic in 24 Hours.* USA: Pearson Education.

Johnsson, B. (2013). *Professional Visual Studio 2012.* USA: John Wiley & Sons.

Jones, R. M. (1999). *Mechanics of Composite Materials.* USA: Taylor & Francis, Inc.

Lengyel András, K. F. (2012). *Lecture Notes on Structural Analysis Theory.* Source: http://www.me.bme.hu/sites/default/files/courses/notes_stanth.pdf

Lőcs-Vigassy. (1985). *A FORTRAN programozási nyelv* (6. bővített kiadás. kiad.). Budapest: Műszaki Könyvkiadó.

Mauer, L. (2002). *Sams Teach Yourself More Visual Basic .NET in 21 Days.* USA: Sams Publishing.

Microsoft Developer Network. (n.d.). Retrieved from http://www.msdn.microsoft.com

Owen-Hinton. (1986). *Finite Element Software for Plates and Shells.* Swansea: Pineridge Press.

Planet Source Code. (n.d.). Retrieved from http://www.planet-source-code.com

Plantema, F. (1996). *Sandwich Construction.* USA: John Wiley & Sons.

Reddy, J. (2004). *Mechanics of Laminated Composite Plates and Shells.* CRC Press.

Sheldon, H. W. (2013). *Professional Visual Basic 2012 and .NET 4.5 Programming.* USA: John Wiley & Sons.

Stack Overflow. (n.d.). Retrieved from http://www.stackoverflow.com

Stephen Timoshenko, S. W.-K. (1989). *Theory of plates and shells.* USA: McGraw - Hill Book Company.

Stephens, R. (2012). *Visual Basic 2012 Programmer's Reference.* USA: John Wiley & Sons.

Szilard, R. (2004). *Theories and Applications of Plate Analysis.* Hoboken, New Jersey: John Wiley & Sons.

The VB Programmer. (n.d.). Retrieved from http://www.thevbprogrammer.com

VB Helper. (n.d.). Retrieved from http://www.vb-helper.com

Vinson, J. (2005). *Plate and Panel Structures of Isotropic, Composite and Piezoelectric Materials, Including Sandwich Construction.* USA: Springer.

APPENDIX A

```
PROGRAM BENSYS

!*****************
!DECLARATION
!*****************

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION :: A(10),D(6,6),H1(11)

REAL,DIMENSION(:),ALLOCATABLE :: XCORD,YCORD
REAL,DIMENSION(:,:),ALLOCATABLE :: BMX,BMY,BMXY,CX,CY,CXY&
                                                    ,QX,QY,W,WUX,WUY

!*****************
!EXECUTABLE PART
!*****************

CALL DATA (A,C,E,E11,E22,EF,GC,G12,G23,HD,H1,IQQ,M,T,TF,TW,VNU,V12,VF,WD,&

     AA,BB,ETA,ETB,G1,GFS,IQ,IT,NPON,NUM,PI,PZ,U,V)

!ALLOCATION
ALLOCATE(XCORD(NPON),YCORD(NPON),BMX(NPON,NPON),BMY(NPON,NPON),BMXY(NPON,NPO
N),CX(NPON,NPON),&

     CY(NPON,NPON),CXY(NPON,NPON),QX(NPON,NPON),QY(NPON,NPON),W(NPON,NPON),
WUX(NPON,NPON),WUY(NPON,NPON))

!CREATE COORDINATES OF OUTPUT POINTS
DO IPON=1,NPON
   XCORD(IPON)=(AA/(NPON-1))*(IPON-1)
   YCORD(IPON)=(BB/(NPON-1))*(IPON-1)
END DO

IF (IQQ==1) THEN
            CALL HOMG (D11,D22,D12,D21,D66,E,S44,S55,T,VNU)
     ELSE IF (IQQ==2) THEN
            CALL SAND (C,D11,D22,D12,D21,D66,EF,GC,S44,S55,T,VF)
     ELSE IF (IQQ==3) THEN
            CALL VOID (D11,D22,D12,D66,E,HD,S44,S55,TF,TW,VNU,WD)
     ELSE IF (IQQ==4) THEN
            CALL LAMI (A,D,E11,E22,G12,G23,H1,M,V12)
END IF

CALL INIT (BMX,BMY,BMXY,CX,CY,CXY,NPON,QX,QY,W,WUX,WUY)

DO M=1,NUM,IT
     DO N=1,NUM,IT
     CALL COEF (AA,BB,D11,D22,D12,D66,DET,G1,GFS,M,N,&
                          P11,P12,P13,P22,P23,PI,S44,S55)
     CALL CONS (AA,AMN,BB,BMN,CMN,DET,ETA,ETB,IQ,M,N,PI,PZ,P11,P12,&
                          P13,P22,P23,U,V)
     CALL SUMS (AA,AMN,BB,BMN,BMX,BMY,BMXY,CMN,CX,CY,CXY,D11,D22,&
                          D12,D66,G1,M,N,NPON,QX,QY,PI,S44,S55,W,WUX,&
                WUY,XCORD,YCORD)
           END DO
END DO

CALL OUTP (NPON,XCORD,YCORD,W,BMX,BMY,BMXY,QX,QY)

STOP

CONTAINS
```

```
!***********************
!DATA /!READS ALL DATA/
!***********************

SUBROUTINE DATA
(A,C,E,E11,E22,EF,GC,G12,G23,HD,H1,IQQ,M,T,TF,TW,VNU,V12,VF,WD,&

     AA,BB,ETA,ETB,G1,GFS,IQ,IT,NPON,NUM,PI,PZ,U,V)

     DIMENSION :: A(10),H1(11)

     !identification of parameters /plate type, loading type,
    !                              symmetry condition, fourier terms, result
density /
     OPEN
(UNIT=10,FILE="ID.TXT",FORM="FORMATTED",STATUS="OLD",ACTION="READ")
    READ (10,"(I5)") IQQ,IQ,IT,NUM,NPON
    CLOSE (UNIT=10)

     !read geometrical parametrs
    OPEN
(UNIT=20,FILE="GEOM.TXT",FORM="FORMATTED",STATUS="OLD",ACTION="READ")
    IF (IQQ==1) THEN
            READ (20,*) AA,BB,T
        ELSE IF (IQQ==2) THEN
                  READ (20,*) AA,BB,C,T
            ELSE IF (IQQ==3) THEN
                  READ (20,*) AA,BB,TF,TW,HD,WD
        ELSE IF (IQQ==4) THEN
          READ (20,*) AA,BB
              READ (20,"(i2)") M
              J=M+1
              DO I=1,J
          READ (20,*) H1(I)
          END DO
          DO K=1,M
            READ (20,*) A(K)
          END DO
    END IF
    CLOSE (20)

    !read material parametrs
    OPEN
(UNIT=30,FILE="MAT.TXT",FORM="FORMATTED",STATUS="OLD",ACTION="READ")
    IF (IQQ==1) THEN
            READ (30,*) VNU,E
        ELSE IF (IQQ==2) THEN
                  READ (30,*) VF,GC,EF
            ELSE IF (IQQ==3) THEN
                  READ (30,*) VNU,E
        ELSE IF (IQQ==4) THEN
                READ (30,*) E11,E22,V12,G12,G23
    END IF
    CLOSE (30)

    !read load parametrs
    OPEN
(UNIT=40,FILE="LOAD.TXT",FORM="FORMATTED",STATUS="OLD",ACTION="READ")
    IF (IQ==1) THEN
            READ (40,*) PZ,GFS,G1
        ELSE IF (IQ==2) THEN
                  READ (40,*) PZ,GFS,G1,ETA,ETB
            ELSE IF (IQ==3) THEN
                  READ (40,*) PZ,GFS,G1,ETA,ETB,U,V
        ELSE IF (IQ==4) THEN
                READ (40,*) PZ,GFS,G1,ETA,ETB,U,V
```

```
    END IF
    CLOSE (40)

     PI=3.141592654
    RETURN

END SUBROUTINE DATA

!**********************************************************************
!HOMOGENEOUS RIGIDITIES /!CALCULATES RIGIDITIES FOR HOMOGENEOUS PLATES/
!**********************************************************************

SUBROUTINE HOMG (D11,D22,D12,D21,D66,E,S44,S55,T,VNU)

     D11=(E*(T**3))/(12.0*(1.0-VNU*VNU))
     D22=D11
    D12=VNU*D11
     D21=D12
     D66=((1.0-VNU)*D11)/2.0
     S44=(E*T)/(2.0*(1.0+VNU)*(1.2))
     S55=S44
     RETURN

END SUBROUTINE HOMG

!**************************************************************
!SANDWICH RIGIDITIES /!CALCULATES RIGIDITIES FOR SANDWICH PLATES/
!**************************************************************

SUBROUTINE SAND (C,D11,D22,D12,D21,D66,EF,GC,S44,S55,T,VF)

     D11=(EF*T*((C+T)**2.0))/(2.0*(1.0-VF*VF))
    !D11=EF*T*(C**2.0+2.0*C*T+4.0*(T**2.0)/3.0)/(4.0*(1-VF**2))
     D22=D11
     D12=VF*D11
     D21=D12
     D66=((1.0-VF)*D11)/2.0
     S44=C*GC/1.2
     S55=S44
     RETURN

END SUBROUTINE SAND

!************************************************************
!VOIDED RIGIDITIES /!CALCULATES RIGIDITIES FOR VOIDED PLATES/
!************************************************************

SUBROUTINE VOID (D11,D22,D12,D66,E,HD,S44,S55,TF,TW,VNU,WD)

    G=E/(2.0*(1.0+VNU))
    D11=(E*TF*HD*HD)/(2.0*(1.0-VNU*VNU))
    D22=D11*(1.0+(TW*HD)/(6.0*TF*WD))
    D12=VNU*D11
    D21=D12
    D66=(G*TF*HD*HD)/2.0
    S44=(2.0*E*(TF**3))/(WD*WD*(1.0+(2.0*(HD/WD))*((TF/TW)**3))*(1.0-
VNU*VNU))
    S55=(G*TF*HD*(1.0+(TF/HD)))/((TF/TW)*WD)
    RETURN

END SUBROUTINE VOID

!**************************************************************
!LAMINATED RIGIDITIES /!CALCULATES RIGIDITIES FOR LAMINATED PLATES/
!**************************************************************
```

```
SUBROUTINE LAMI (A,D,E11,E22,G12,G23,H1,M,V12)

    DIMENSION A(10),AA(6,6),A2(6,6),C1(6,6),D(6,6),H1(11),Q(6,6)

    V21=V12*E22/E11
    G31=G23
    Q(1,1)=E11/(1.0-V12*V21)
    Q(1,2)=E11*V21/(1.0-V12*V21)
    Q(1,3)=0.0
    Q(1,4)=0.0
    Q(1,5)=0.0
    Q(1,6)=0.0
    Q(2,2)=E22/(1.0-V12*V21)
    Q(2,3)=0.0
    Q(2,4)=0.0
    Q(2,5)=0.0
    Q(2,6)=0.0
    Q(3,3)=0.0
    Q(3,4)=0.0
    Q(3,5)=0.0
    Q(3,6)=0.0
    Q(4,4)=G23
    Q(4,5)=0.0
    Q(4,6)=0.0
    Q(5,5)=G31
    Q(5,6)=0.0
    Q(6,6)=G12

     Q(2,1)=Q(1,2)
     Q(3,1)=Q(1,3)
    Q(3,2)=Q(2,3)
     Q(4,1)=Q(1,4)
    Q(4,2)=Q(2,4)
    Q(4,3)=Q(3,4)
     Q(5,1)=Q(1,5)
    Q(5,2)=Q(2,5)
    Q(5,3)=Q(3,5)
    Q(5,4)=Q(4,5)
    Q(6,1)=Q(1,6)
    Q(6,2)=Q(2,6)
    Q(6,3)=Q(3,6)
    Q(6,4)=Q(4,6)
    Q(6,5)=Q(5,6)

    DO I=1,6
     DO J=1,6
            D(I,J)=0.0
            DO K=1,M
                  ANGLE=3.1415927*A(K)/180.0
                  CC=COS(ANGLE)
                  SS=SIN(ANGLE)
                        C1(1,1)=0.0
                  C1(1,2)=0.0
                  C1(1,3)=0.0
                  C1(1,4)=0.0
                  C1(1,5)=0.0
                  C1(1,6)=0.0
                  C1(2,2)=0.0
                  C1(2,3)=0.0
                  C1(2,4)=0.0
                  C1(2,5)=0.0
                  C1(2,6)=0.0
                  C1(3,3)=0.0
                  C1(3,4)=0.0
                  C1(3,5)=0.0
                  C1(3,6)=0.0
```

```
                    C1(4,4)=0.0
                    C1(4,5)=0.0
                    C1(4,6)=0.0
                    C1(5,5)=0.0
                    C1(5,6)=0.0
                    C1(6,6)=0.0
                  C1(2,1)=C1(1,2)
                  C1(3,1)=C1(1,3)
                  C1(3,2)=C1(2,3)
                  C1(4,1)=C1(1,4)
                  C1(4,2)=C1(2,4)
                  C1(4,3)=C1(3,4)
                  C1(5,1)=C1(1,5)
                  C1(5,2)=C1(2,5)
                  C1(5,3)=C1(3,5)
                  C1(5,4)=C1(4,5)
                  C1(6,1)=C1(1,6)
                  C1(6,2)=C1(2,6)
                  C1(6,3)=C1(3,6)
                  C1(6,4)=C1(4,6)
                  C1(6,5)=C1(5,6)


      C1(1,1)=(Q(1,1)*(CC**4))+(2.0*(Q(1,2)+2*Q(6,6))*CC*CC*SS*SS)+(Q(2,2)*(
SS**4))
                    C1(1,2)=((Q(1,1)+Q(2,2)-
4.0*Q(6,6))*SS*SS*CC*CC)+(Q(1,2)*(CC**4+SS**4))
                    C1(1,6)=((Q(1,1)-Q(1,2)-2*Q(6,6))*SS*CC**3)+((Q(1,2)-
Q(2,2)+2*Q(6,6))*(SS**3)*CC)

      C1(2,2)=(Q(1,1)*SS**4)+(2.0*(Q(1,2)+2*Q(6,6))*CC*CC*SS*SS)+(Q(2,2)*CC*
*4)
                    C1(2,6)=((Q(1,1)-Q(1,2)-2*Q(6,6))*(SS**3)*CC)+((Q(1,2)-
Q(2,2)+2*Q(6,6))*SS*(CC**3))
                    C1(6,6)=((Q(1,1)+Q(2,2)-2.0*Q(1,2)-
2*Q(6,6))*CC*CC*SS*SS)+(Q(6,6)*(CC**4+SS**4))
                    C1(2,1)=C1(1,2)
                    C1(6,1)=C1(1,6)
                    C1(6,2)=C1(2,6)

                    D(I,J)=D(I,J)+(C1(I,J)*(((H1(K+1))**3)-((H1(K))**3)))/3.0
             END DO
          END DO
      END DO


      DO I=4,5
       DO J=4,5
             AA(I,J)=0.0
             DO K=1,M
                    ANGLE=3.1415927*A(K)/180.0
                    CC=COS(ANGLE)
                    SS=SIN(ANGLE)
                    A2(4,4)=0.0
                    A2(5,5)=0.0
                    A2(4,5)=0.0
                    A2(5,4)=0.0
                    A2(4,4)=(Q(4,4)*CC*CC)+(Q(5,5)*SS*SS)
                    A2(5,5)=(Q(4,4)*SS*SS)+(Q(5,5)*CC*CC)
                    AA(I,J)=AA(I,J)+(A2(I,J)*((H1(K+1))-(H1(K))))
             END DO
          END DO
      END DO
      D(4,4)=AA(4,4)
      D(5,5)=AA(5,5)
```

```
      D11=D(1,1)
      D22=D(2,2)
     D12=D(1,2)
      D21=D(2,1)
      D66=D(6,6)
      S44=D(4,4)
      S55=D(5,5)
     RETURN

END SUBROUTINE LAMI

!*******************************************
!SUBROUTINE INIT  /INITIALISES VARIOUS ARRAYS/
!*******************************************

SUBROUTINE INIT (BMX,BMY,BMXY,CX,CY,CXY,&
                          NPON,QX,QY,W,WUX,WUY)

    REAL,DIMENSION (:,:) :: BMX,BMY,BMXY,CX,CY,&
                          CXY,QX,QY,WUX,WUY,W

    DO IPON=1,NPON
      DO IIPON=1,NPON
      W(IPON,IIPON)=0.0
      WUX(IPON,IIPON)=0.0
             WUY(IPON,IIPON)=0.0
      CX(IPON,IIPON)=0.0
      CY(IPON,IIPON)=0.0
             CXY(IPON,IIPON)=0.0
      BMX(IPON,IIPON)=0.0
      BMY(IPON,IIPON)=0.0
      BMXY(IPON,IIPON)=0.0
      QX(IPON,IIPON)=0.0
      QY(IPON,IIPON)=0.0
       END DO
    END DO
     RETURN

END SUBROUTINE INIT

!*******************************************
!SUBROUTINE CONS  /ALCULATES THE VALUES
!AMN,BMN AND CMN FOR A GIVEN LOADING FUNCTION/
!*******************************************

SUBROUTINE CONS(AA,AMN,BB,BMN,CMN,DET,ETA,ETB,IQ,M,N,PI,PZ,P11,&
                          P12,P13,P22,P23,U,V)

    IF (IQ==1) THEN
        QMN=(16.0*PZ)/(PI*PI*FLOAT(M)*FLOAT(N))
     ELSEIF (IQ==2) THEN
     QMN=((4.0*PZ)/(AA*BB))*SIN((FLOAT(M)*PI*ETA)/AA)&
        *SIN((FLOAT(N)*PI*ETB)/BB)
     ELSEIF (IQ==3) THEN
     QMN=((16.0*PZ)/(PI*PI*FLOAT(M)*FLOAT(N)))&
        *SIN((FLOAT(M)*PI*ETA)/AA)*&
            SIN((FLOAT(N)*PI*ETB)/BB)*SIN((FLOAT(M)*PI*U)/(2.0*AA))*&
        SIN((FLOAT(N)*PI*V)/(2.0*BB))
    ELSEIF (IQ==4) THEN
            QMN=(256*PZ*aa*bb*sin((pi*eta*FLOAT(M))&
        /aa)*sin((pi*etb*FLOAT(N))/bb)*sin((pi*FLOAT(M)*u)/(4*aa))**2&
        *sin((pi*FLOAT(N)*v)/(4*bb))**2)/(pi**4*FLOAT(M)**2*FLOAT(N)**2*u*v)
    END IF
    AMN=(P12*P23-P22*P13)*QMN/DET
    BMN=(P12*P13-P11*P23)*QMN/DET
    CMN=(P11*P22-P12*P12)*QMN/DET
```

```
    RETURN

END SUBROUTINE CONS

!*************************************************
!COEF /!THIS SUBROUTINE CALCULATES THE COEFFICIENTS
!     (P11,P12,P13,P22,P23,P33) AND THE DETERMINANT/
!*************************************************

SUBROUTINE COEF (AA,BB,D11,D22,D12,D66,DET,G1,GFS,M,&
                         N,P11,P12,P13,P22,P23,PI,S44,S55)

     P11=D11*(FLOAT(M)*PI/AA)*(FLOAT(M)*PI/AA)&
     +D66*(FLOAT(N)*PI/BB)*(FLOAT(N)*PI/BB)+G1*S55
    P12=(D12+D66)*(FLOAT(M)*PI/AA)*(FLOAT(N)*PI/BB)
    P13=G1*S55*(FLOAT(M)*PI/AA)
    P22=D66*(FLOAT(M)*PI/AA)*(FLOAT(M)*PI/AA)&
     +D22*(FLOAT(N)*PI/BB)*(FLOAT(N)*PI/BB)+G1*S44
    P23=G1*S44*(FLOAT(N)*PI/BB)
    P33=(G1*S55)*(FLOAT(M)*PI/AA)*(FLOAT(M)*PI/AA)&
     +(G1*S44)*(FLOAT(N)*PI/BB)*(FLOAT(N)*PI/BB)+GFS
    DET=P11*(P22*P33-P23*P23)-P12*(P12*P33-P23*P13)&
     +P13*(P12*P23-P22*P13)
    RETURN

END SUBROUTINE COEF

!*************************************************
!SUBROUTINE SUMS /SUMS THE VARIOUS FURIER SERIES/
!*************************************************

SUBROUTINE SUMS (AA,AMN,BB,BMN,BMX,BMY,BMXY,CMN,CX,CY,CXY,D11,&
                         D22,D12,D66,G1,M,N,NPON,QX,QY,PI,S44,S55,W,&
                 WUX,WUY,XCORD,YCORD)

    REAL,DIMENSION(:) :: XCORD,YCORD
    REAL,DIMENSION(:,:) :: BMX,BMY,BMXY,CX,CY,&
                                       CXY,QX,QY,WUX,WUY,W

     DO IPON=1,NPON
     DO IIPON=1,NPON
            X=XCORD(IPON)
            Y=YCORD(IIPON)
                   !DEFLECTION W(IPON)
            W(IPON,IIPON)=W(IPON,IIPON)+CMN*SIN((FLOAT(M)*PI*X)/AA)*&
                   SIN((FLOAT(N)*PI*Y)/BB)
            !SLOPE WUX(IPON)
            WUX(IPON,IIPON)=WUX(IPON,IIPON)+AMN*COS((FLOAT(M)*PI*X)/AA)*&
                         SIN((FLOAT(N)*PI*Y)/BB)
            !SLOPE WUY(IPON)
            WUY(IPON,IIPON)=WUY(IPON,IIPON)+BMN*SIN((FLOAT(M)*PI*X)/AA)*&
                         COS((FLOAT(N)*PI*Y)/BB)
            !CURVATURE CX(IPON)

     CX(IPON,IIPON)=CX(IPON,IIPON)+AMN*(FLOAT(M)*PI/AA)*SIN((FLOAT(M)*PI*X)
/AA)*&
                         SIN((FLOAT(N)*PI*Y)/BB)
            !CURVATURE CY(IPON)

     CY(IPON,IIPON)=CY(IPON,IIPON)+BMN*(FLOAT(N)*PI/BB)*SIN((FLOAT(M)*PI*X)
/AA)*&
                         SIN((FLOAT(N)*PI*Y)/BB)
            !CURVATURE CXY(IPON)
            CXY(IPON,IIPON)=CXY(IPON,IIPON)+((AMN*(FLOAT(N)*PI/BB))+&
                          (BMN*(FLOAT(M)*PI/AA)))*&
                      COS((FLOAT(M)*PI*X)/AA)*COS((FLOAT(N)*PI*Y)/BB)
```

```
            !BENDING MOMENT BMX(IPON)
            BMX(IPON,IIPON)=-D11*CX(IPON,IIPON)-D12*CY(IPON,IIPON)
            !BENDING MOMENT BMY(IPON)
            BMY(IPON,IIPON)=-D22*CY(IPON,IIPON)-D12*CX(IPON,IIPON)
            !TWISTING MOMENT BMXY(IPON)
            BMXY(IPON,IIPON)=D66*CXY(IPON,IIPON)
            !SHEAR FORCE QX(IPON)

     QX(IPON,IIPON)=QX(IPON,IIPON)+G1*S55*((CMN*FLOAT(M)*PI/AA)+AMN)*&
                  COS((FLOAT(M)*PI*X)/AA)*SIN((FLOAT(N)*PI*Y)/BB)
            !SHEAR FORCE QY(IPON)

     QY(IPON,IIPON)=QY(IPON,IIPON)+G1*S44*((CMN*FLOAT(N)*PI/BB)+BMN)*&
                  SIN((FLOAT(M)*PI*X)/AA)*COS((FLOAT(N)*PI*Y)/BB)
      END DO
    END DO
      RETURN

END SUBROUTINE SUMS

!*****************************
!OUTP /PRINTS OUT THE RESULTS/
!*****************************

SUBROUTINE OUTP (NPON,XCORD,YCORD,W,BMX,BMY,BMXY,QX,QY)

     REAL,DIMENSION (:) :: XCORD,YCORD
     REAL,DIMENSION (:,:) :: BMX,BMY,BMXY,&
                                      QX,QY,W

OPEN
(UNIT=60,FILE="RESULT.TXT",FORM="FORMATTED",STATUS="REPLACE",ACTION="WRITE")
DO IPON=1,NPON
  DO IIPON=1,NPON
    WRITE (60,"(8F18.5)")
XCORD(IPON),YCORD(IIPON),W(IPON,IIPON),BMX(IPON,IIPON), &

     BMY(IPON,IIPON),BMXY(IPON,IIPON),QX(IPON,IIPON),QY(IPON,IIPON)
  END DO
END DO
CLOSE (60)
      RETURN

END SUBROUTINE OUTP

END PROGRAM BENSYS
```

APPENDIX B

```
!HOMOGENEOUS PLATE / UNIFORM
LOAD

FINISH
/CLEAR,START

!precrocessing
/PREP7

AA=800
BB=600
TT=20
EXY=2750
FZ=1.0/10000
NU=0.2
DENS=2500e-6

!geometry
RECTNG,0,AA,0,BB

!material
MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,EXY
MPDATA,PRXY,1,,NU
MPDATA,DENS,1,,DENS

!element
ET,1,SHELL181
KEYOPT,1,8,2
SECTYPE,1,SHELL
SECDATA,TT,1,,9

!mesh
MSHAPE,0,2D
MSHKEY,2
LESIZE,ALL,MIN(AA/50,BB/50)
AATT,1,,1,0,1
AMESH,ALL

!load
SFA,ALL,1,PRES,-FZ
acel,,,9.81e-3

!bc
LSEL,S,LOC,X,0
LSEL,A,LOC,X,AA
NSLL,S,1
D,ALL,UX
D,ALL,UY
D,ALL,UZ
ALLSEL,ALL
LSEL,S,LOC,Y,0
LSEL,A,LOC,Y,BB
NSLL,S,1
D,ALL,UX
D,ALL,UY
D,ALL,UZ
ALLSEL,ALL

FINISH

!solution
/SOLU
```

```
ANTYPE,0
PSTRES,ON
SOLVE

FINISH

/POST1

!postprocessing
PLNSOL, U,Z,0,1.0

*GET,w1,NODE,NODE(200,200,0),U
,Z
*GET,w2,NODE,NODE(600,200,0),U
,Z
*GET,w3,NODE,NODE(600,400,0),U
,Z
*GET,w4,NODE,NODE(200,400,0),U
,Z
*GET,w5,NODE,NODE(400,300,0),U
,Z
```

```
!HOMOGENEOUS PLATE /
RECTANGULAR LOAD

FINISH
/CLEAR,START

/PREP7

AA=800
BB=800
TT=20
ETA=530
ETB=260
UU=200
VV=414
EXY=2600
FZ=40/10000
NU=0.3
DENS=7.85e-9

!geometry
RECTNG,0,AA,0,BB
*GET,maxkey,kp,,num,max
k,maxkey+1,0,ETB-VV/2,0
k,maxkey+2,AA,ETB-VV/2,0
L,maxkey+1,maxkey+2
*GET,maxkey,kp,,num,max
k,maxkey+1,0,ETB+VV/2,0
k,maxkey+2,AA,ETB+VV/2,0
L,maxkey+1,maxkey+2
lsel,s,line,,5,6,1
*GET,maxkey,kp,,num,max
k,maxkey+1,ETA-UU/2,0,0
k,maxkey+2,ETA-UU/2,BB,0
L,maxkey+1,maxkey+2
*GET,maxkey,kp,,num,max
k,maxkey+1,ETA+UU/2,0,0
k,maxkey+2,ETA+UU/2,BB,0
L,maxkey+1,maxkey+2
```

```
lsel,s,line,,5,8,1
asbl,all,all,,delete,delete
nummrg,all
allsel,all

!material
MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,EXY
MPDATA,PRXY,1,,NU
MPDATA,DENS,1,,DENS

!element
ET,1,SHELL181
KEYOPT,1,8,2
SECTYPE,1,SHELL
SECDATA,TT,1,,9

!mesh
MSHAPE,0,2D
MSHKEY,2
LESIZE,ALL,MIN(AA/50,BB/50)
AATT,1,,1,0,1
AMESH,ALL

!load
ASEL,s,loc,x,ETA-UU/2,ETA+UU/2
ASEL,r,loc,y,ETB-VV/2,ETB+VV/2
SFA,ALL,1,PRES,-FZ
ALLSEL,ALL

!bc
LSEL,S,LOC,X,0
LSEL,A,LOC,X,AA
NSLL,S,1
D,ALL,UX
D,ALL,UY
D,ALL,UZ
ALLSEL,ALL
LSEL,S,LOC,Y,0
LSEL,A,LOC,Y,BB
NSLL,S,1
D,ALL,UX
D,ALL,UY
D,ALL,UZ
ALLSEL,ALL

FINISH

!solution
/SOLU

ANTYPE,0
PSTRES,ON
SOLVE

FINISH

/POST1

!postprocess
PLNSOL, U,Z,0,1.0
```

```
!HOMOGENEOUS PLATE /
CONCENTRATED LOAD

FINISH
/CLEAR,START

/PREP7

AA=800
BB=800
TT=20
ETA=530
ETB=260
EXY=2750
FZ=100
NU=0.3
DENS=7.85e-9

!geometry
RECTNG,0,ETA,0,ETB
RECTNG,ETA,AA,0,ETB
RECTNG,0,ETA,ETB,BB
RECTNG,ETA,AA,ETB,BB
NUMMRG,KP

!material
MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,EXY
MPDATA,PRXY,1,,NU
MPDATA,DENS,1,,DENS

!element
ET,1,SHELL181
KEYOPT,1,8,2
SECTYPE,1,SHELL
SECDATA,TT,1,,9

!mesh
MSHAPE,0,2D
MSHKEY,2
LESIZE,ALL,MIN(AA/50,BB/50)
AATT,1,,1,0,1
AMESH,ALL

!load
nsel,s,node,,node(ETA,ETB,0)
F,ALL,FZ,-FZ
ALLSEL,ALL

!bc
LSEL,S,LOC,X,0
LSEL,A,LOC,X,AA
NSLL,S,1
D,ALL,UX
D,ALL,UY
D,ALL,UZ
ALLSEL,ALL
LSEL,S,LOC,Y,0
LSEL,A,LOC,Y,BB
NSLL,S,1
D,ALL,UX
D,ALL,UY
D,ALL,UZ
ALLSEL,ALL
```

```
FINISH

!solution
/SOLU

ANTYPE,0
PSTRES,ON
SOLVE

FINISH
```

```
/POST1

!postprocess
PLNSOL, U,Z,0,1.0
```

```
!HOMOGENEOUS PLATE / PYRAMID LOAD

FINISH
/CLEAR,START

/PREP7

pi=3.14159265358979323846264338327950288419716939937510

AA=800
BB=600
TT=20
ETA=530
ETB=260
UU=400
VV=300
EXY=2750
FZ=40/10000
NU=0.2
DENS=7.85e-9

!geometry
RECTNG,0,AA,0,BB
*GET,maxkey,kp,,num,max
k,maxkey+1,0,ETB-VV/2,0
k,maxkey+2,AA,ETB-VV/2,0
L,maxkey+1,maxkey+2
*GET,maxkey,kp,,num,max
k,maxkey+1,0,ETB+VV/2,0
k,maxkey+2,AA,ETB+VV/2,0
L,maxkey+1,maxkey+2
lsel,s,line,,5,6,1
*GET,maxkey,kp,,num,max
k,maxkey+1,ETA-UU/2,0,0
k,maxkey+2,ETA-UU/2,BB,0
L,maxkey+1,maxkey+2
*GET,maxkey,kp,,num,max
k,maxkey+1,ETA+UU/2,0,0
k,maxkey+2,ETA+UU/2,BB,0
L,maxkey+1,maxkey+2
lsel,s,line,,5,8,1
asbl,all,all,,delete,delete
nummrg,all
allsel,all

!material
MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,EXY
MPDATA,PRXY,1,,NU
MPDATA,DENS,1,,DENS

!element
ET,1,SHELL181
KEYOPT,1,8,2
SECTYPE,1,SHELL
SECDATA,TT,1,,9

!mesh
MSHAPE,0,2D
MSHKEY,2
LESIZE,ALL,MIN(AA/50,BB/50)
AATT,1,,1,0,1
AMESH,ALL

!load
NSEL,s,loc,x,ETA-UU/2,ETA+UU/2
NSEL,r,loc,y,ETB-VV/2,ETB+VV/2
```

```
CM,reszlet,NODE
CMSEL,S,reszlet,NODE

*GET,xmin,NODE,0,MNLOC,X
*GET,xmax,NODE,0,MXLOC,X
*GET,ymin,NODE,0,MNLOC,Y
*GET,ymax,NODE,0,MXLOC,Y

NSEL,R,LOC,Y,ymin
*GET,xnbrnode,NODE,0,COUNT
*DIM,xcoord,ARRAY,xnbrnode,1
*DO,i,1,xnbrnode
   *GET,xcoord(i),NODE,0,MNLOC,X
   NSEL,U,LOC,X,xcoord(i)-0.1,xcoord(i)+0.1,0.01
*ENDDO

CMSEL,S,reszlet,NODE
NSEL,R,LOC,X,xmin
*GET,ynbrnode,NODE,0,COUNT
*DIM,ycoord,ARRAY,ynbrnode,1
*DO,j,1,ynbrnode
   *GET,ycoord(j),NODE,0,MNLOC,Y
   NSEL,U,LOC,Y,ycoord(j)-0.1,ycoord(j)+0.1,0.01
*ENDDO

CMSEL,S,reszlet,NODE

*DO,i,1,xnbrnode
     *DO,j,1,ynbrnode

     *IF,j,EQ,1,THEN
            *IF,i,EQ,1,THEN
                  CMSEL,S,reszlet,NODE
                  NSEL,R,LOC,X,xcoord(i)-0.1,xcoord(i)+0.1,0.01
                  NSEL,R,LOC,Y,ycoord(j)-0.1,ycoord(j)+0.1,0.01
                  forc=-FZ*(1-2/UU*abs(eta-xcoord(i)))*(1-2/VV*abs(etb-
ycoord(j)))
                  F,ALL,FZ,forc*((ycoord(j+1)-ycoord(j))/2)*((xcoord(i+1)-
xcoord(i))/2)
            *ELSEIF,i,NE,1,AND,i,NE,xnbrnode,THEN
                  CMSEL,S,reszlet,NODE
                  NSEL,R,LOC,X,xcoord(i)-0.1,xcoord(i)+0.1,0.01
                  NSEL,R,LOC,Y,ycoord(j)-0.1,ycoord(j)+0.1,0.01
                  forc=-FZ*(1-2/UU*abs(eta-xcoord(i)))*(1-2/VV*abs(etb-
ycoord(j)))
                  F,ALL,FZ,forc*((ycoord(j+1)-ycoord(j))/2)*((xcoord(i)-
xcoord(i-1))/2+(xcoord(i+1)-xcoord(i))/2)
            *ELSEIF,i,EQ,xnbrnode,THEN
                  CMSEL,S,reszlet,NODE
                  NSEL,R,LOC,X,xcoord(i)-0.1,xcoord(i)+0.1,0.01
                  NSEL,R,LOC,Y,ycoord(j)-0.1,ycoord(j)+0.1,0.01
                  forc=-FZ*(1-2/UU*abs(eta-xcoord(i)))*(1-2/VV*abs(etb-
ycoord(j)))
                  F,ALL,FZ,forc*((ycoord(j+1)-ycoord(j))/2)*((xcoord(i)-
xcoord(i-1))/2)
            *ENDIF
     *ELSEIF,j,NE,1,AND,j,NE,ynbrnode,THEN
             *IF,i,EQ,1,THEN
                  CMSEL,S,reszlet,NODE
                  NSEL,R,LOC,X,xcoord(i)-0.1,xcoord(i)+0.1,0.01
                  NSEL,R,LOC,Y,ycoord(j)-0.1,ycoord(j)+0.1,0.01
                  forc=-FZ*(1-2/UU*abs(eta-xcoord(i)))*(1-2/VV*abs(etb-
ycoord(j)))
                  F,ALL,FZ,forc*((ycoord(j)-ycoord(j-1))/2+(ycoord(j+1)-
ycoord(j))/2)*((xcoord(i+1)-xcoord(i))/2)
                *ELSEIF,i,NE,1,AND,i,NE,xnbrnode,THEN
```

```
                  CMSEL,S,reszlet,NODE
                  NSEL,R,LOC,X,xcoord(i)-0.1,xcoord(i)+0.1,0.01
                  NSEL,R,LOC,Y,ycoord(j)-0.1,ycoord(j)+0.1,0.01
                  forc=-FZ*(1-2/UU*abs(eta-xcoord(i)))*(1-2/VV*abs(etb-
ycoord(j)))
                  F,ALL,FZ,forc*((ycoord(j)-ycoord(j-1))/2+(ycoord(j+1)-
ycoord(j))/2)*((xcoord(i)-xcoord(i-1))/2+(xcoord(i+1)-xcoord(i))/2)
               *ELSEIF,i,EQ,xnbrnode,THEN
                  CMSEL,S,reszlet,NODE
                  NSEL,R,LOC,X,xcoord(i)-0.1,xcoord(i)+0.1,0.01
                  NSEL,R,LOC,Y,ycoord(j)-0.1,ycoord(j)+0.1,0.01
                  forc=-FZ*(1-2/UU*abs(eta-xcoord(i)))*(1-2/VV*abs(etb-
ycoord(j)))
                  F,ALL,FZ,forc*((ycoord(j)-ycoord(j-1))/2+(ycoord(j+1)-
ycoord(j))/2)*((xcoord(i)-xcoord(i-1))/2)
               *ENDIF
     *ELSEIF,j,EQ,ynbrnode,THEN
               *IF,i,EQ,1,THEN
                  CMSEL,S,reszlet,NODE
                  NSEL,R,LOC,X,xcoord(i)-0.1,xcoord(i)+0.1,0.01
                  NSEL,R,LOC,Y,ycoord(j)-0.1,ycoord(j)+0.1,0.01
                  forc=-FZ*(1-2/UU*abs(eta-xcoord(i)))*(1-2/VV*abs(etb-
ycoord(j)))
                  F,ALL,FZ,p*((ycoord(j)-ycoord(j-1))/2)*((xcoord(i+1)-
xcoord(i))/2)
               *ELSEIF,i,NE,1,AND,i,NE,xnbrnode,THEN
                  CMSEL,S,reszlet,NODE
                  NSEL,R,LOC,X,xcoord(i)-0.1,xcoord(i)+0.1,0.01
                  NSEL,R,LOC,Y,ycoord(j)-0.1,ycoord(j)+0.1,0.01
                  forc=-FZ*(1-2/UU*abs(eta-xcoord(i)))*(1-2/VV*abs(etb-
ycoord(j)))
                  F,ALL,FZ,p*((ycoord(j)-ycoord(j-1))/2)*((xcoord(i)-
xcoord(i-1))/2+(xcoord(i+1)-xcoord(i))/2)
               *ELSEIF,i,EQ,xnbrnode,THEN
                  CMSEL,S,reszlet,NODE
                  NSEL,R,LOC,X,xcoord(i)-0.1,xcoord(i)+0.1,0.01
                  NSEL,R,LOC,Y,ycoord(j)-0.1,ycoord(j)+0.1,0.01
                  forc=-FZ*(1-2/UU*abs(eta-xcoord(i)))*(1-2/VV*abs(etb-
ycoord(j)))
                  F,ALL,FZ,p*((ycoord(j)-ycoord(j-1))/2)*((xcoord(i)-
xcoord(i-1))/2)
               *ENDIF
        *ENDIF

     *ENDDO
*ENDDO
ALLSEL,ALL

!bc
LSEL,S,LOC,X,0
LSEL,A,LOC,X,AA
NSLL,S,1
D,ALL,UX
D,ALL,UY
D,ALL,UZ
ALLSEL,ALL
LSEL,S,LOC,Y,0
LSEL,A,LOC,Y,BB
NSLL,S,1
D,ALL,UX
D,ALL,UY
D,ALL,UZ
ALLSEL,ALL

FINISH
```

```
!solution
/SOLU

ANTYPE,0
PSTRES,ON
SOLVE

FINISH

/POST1

!postprocess
PLNSOL, U,Z,0,1.0

*GET,EZ1,NODE,NODE(200,200,0),U,Z
*GET,EZ2,NODE,NODE(600,200,0),U,Z
*GET,EZ3,NODE,NODE(600,400,0),U,Z
*GET,EZ4,NODE,NODE(200,400,0),U,Z
*GET,EZ5,NODE,NODE(400,300,0),U,Z
*GET,EZR,NODE,NODE(379.79,121.21,0),U,Z
```