THEORETICAL AND EXPERIMENTAL INVESTIGATION OF THE FREE VIBRATION OF PARALLELOGRAM PLATES WITH SIMPLY SUPPORTED AND CLAMPED BOUNDARY CONDITIONS

Thesis presented to the School of Graduate Studies
as partial fulfillment of the requirements
for the Degree of Master of Applied Science
in Mechanical Engineering

© Stefanie D. Stangier

University of Ottawa
Ottawa, Ontario, Canada
August, 1997

0-612-28462-X

Abstract

A systematic approach to the free vibration analysis of thin, flat, non-rectangular quadrilateral plates with combinations of simple and clamped supports is presented, using the parallelogram plate as an example. Modifications that Saliba made for the right-triangular plate to the building block superposition method developed by Gorman are implemented. The superposition method is an analytical solution. No simplifications are made to take advantage of the point symmetry of the parallelogram plate, keeping the solution general. The whole plate is divided into two right-triangular and one rectangular segment. Rectangular building blocks are superimposed for each of the segments to meet the required net boundary conditions and the conditions of continuity along the segment interfaces. The Lévy-type solution to the eight building blocks used are given, along with the necessary Fourier expansions and a comprehensive guide to assembling the eigenvalue matrix. Numerical results for a wide variety of plates are presented, including numerous mesh and contour plots. Comparison is made to previously published data. The results for the fully clamped parallelogram plate are supported by experimental results. Six aluminum plates were tested for the first six resonance frequencies and the first three mode shapes. Details of the simple, yet highly accurate experimental method are included.

Acknowledgments

First of all, I have to thank Dr. H.T. Saliba of Lakehead University for giving me the opportunity to do my research at Lakehead, for his guidance, and for having the confidence that I would eventually get this thesis finished. I would also like to thank Dr. D.J. Gorman of the University of Ottawa for his counsel.

Thanks also go to the School of Engineering at Lakehead University for allowing me the use of the university facilities and for providing assistance in many other ways. I would like to thank Kailash Bhatia for his expert guidance in the machine shop. Mark Blezzard did a fine job of assisting me in the shop as well. Jean Eric Mesmain assisted with the execution of experiments, as did Andrew Saikaley. Andrew also gave me some very helpful programming ideas, and much needed moral support.

The Natural Science and Engineering Research Council supported this work financially..

Table of Contents

List of Figures

List of Tables

List of Symbols

a	plate dimension in the x-direction.
b	plate dimension in the y-direction.
D	plate flexural rigidity, $Eh^3/12(1-\nu^2)$.
E	modulus of elasticity of the plate material.
f	circular frequency of plate vibration in Hz.
h	plate thickness.
q(x,y)	lateral loading of the plate surface.
r	subscript referring to the rectangular segment of a parallelogram plate segment
t	subscript referring to the triangular segment of a parallelogram plate.
W(x,y)	amplitude of the plate lateral deflection.
W(ξ,η)	dimensionless plate lateral deflection, $W(x,y)/a$.
x, y	spatial coordinates.
α	angle of the skew side of a parallelogram plate to the y- or η-axis.
ξ	dimensionless coordinate in the x-direction, x/a.
η	dimensionless coordinate in the y-direction, y/b.
φ	aspect ratio, b/a.
ν	Poisson's ratio of the plate material; ν = 0.3 is assumed in this work.
ν*	= 2-ν.
ρ	mass density per unit area of the plate material.
ω	circular frequency of plate vibration in rad/s.
λ^2	eigenvalue, non-dimensional natural frequency, $\omega a^2\sqrt{\rho/D}$.

1 - Introduction

1. Problem Definition

Thin plates of all shapes and sizes are undoubtedly one of, if not the most common structural component in engineering applications. They are found in everything, from seemingly simple casings to circuit boards to solar panels. Knowing the static response of these structural elements is not sufficient to ensure proper, even safe, performance. The dynamic behaviour of plates must be included in any design project that incorporates them. The ability to predict the vibration behaviour without testing saves time and money spent on trial and error approaches. The resonance frequencies have to be known so that they can be dealt with in one of three ways. They can be avoided in the operation of the structure using the plate. The plate can be re-designed to withstand the vibration, or the supports or physical properties of the plate can be changed to change its vibration behaviour.

In the last few years, the finite element method has made the very problem dependent solution methods to plate vibration obsolete in the eyes of the uninitiated. Most other approximate solutions, in particular the earliest ones, appear to be limited to special problems. However, these solutions have a place as a check for the finite element approach. It should not be forgotten that they are still only approximate in nature. Particularly in the case of skew plates, they all do not satisfy either the boundary conditions or the governing differential equation completely.

So far, the building block superposition method is the only analytical solution to the problem of thin, flat, quadrilateral plates. Solutions by this method always satisfy the governing differential equation, and all boundary conditions to the desired accuracy. Each rectangular building block meets this criterion for being an analytical solution to the rectangular plate. The building block solutions are superimposed to satisfy the net boundary conditions, and they are positioned adjacent to one another to accommodate irregularly shaped plates. Continuity of displacement, slope, bending moment and vertical edge reaction are accounted for by building blocks as well.

The special case of the thin, flat parallelogram plates looked at in this work serves as an example for the general quadrilateral plate with combinations of simple and clamped supports. With this in mind, no assumptions are made about the symmetry or antisymmetry of the mode shapes, something that could be used to reduce the computational effort. A parallelogram is point symmetric about the centre point (the intersection of the diagonals). For support combinations that are symmetric about the centre point as well, such as the all simply supported plate, the mode shapes will have symmetry or antisymmetry about this centre point.

In addition to demonstrating the application of the building block superposition method to the isotropic, thin, flat parallelogram plate, experimental results are presented for the fully clamped case to support the theoretical results. The brief literature review that follows shows not only the lack of an exact solution to the problem at hand, but also to an absence of experimental results.

2. Literature Review

A very detailed review of published work dealing with the vibration of plates, including thin, flat parallelogram plates, has been done by Leissa [1,2,3,4] up to the year 1985. The reader is referred there for a more in-depth guide to the available literature up to that time than can be presented here. Some of this earlier work is highlighted here, before more recent research is described. Theoretical and experimental works are dealt with separately.

2.1 Theory

Two types of solution exist: approximate and analytical. Analytical solutions are the most desirable, but not always easily attainable, as in the case of parallelogram plates.

Early solutions especially are highly dependent on the actual shape of the plate. The difficulty lies in the requirement to satisfy both, the governing differential equation and all of the boundary conditions exactly.

Approximate Solutions

The approximate methods can be grouped into continuum types like the Rayleigh-Ritz method, and discrete types like finite element and finite difference methods. Even though finite element solutions have become the first choice for many researchers with the ever decreasing cost of computing, Rayleigh-Ritz type approaches are still popular. The aim there is to lower the upper bound result of the solution. Many authors are also satisfied with the fundamental frequency only, and rhombic plates appear to be the test subject of choice.

It is well known that the Rayleigh-Ritz method yields upper-bounds only. Kaul and Cadambe [5] and Pnuelli [6] attempted to find lower bounds. In [5], the upper bounds for three plate configurations are found by the Rayleigh-Ritz method with a single product function approximation. Kato's principle is used to find the lower bounds. Durvasula [7] commented that their frequency expressions for skew plates are faulty. Pnuelli [6] used the membrane analogy for simply supported and clamped plates, among a variety of other shapes. The resulting algebraic equations depend on the plate area only.

Durvasula is one of the more prolific researchers on the vibration of skew plates. In Reference [8], he applied the Galerkin method, a method of weighted residuals, using a double series of beam characteristic functions to describe the deflection of the plate. The functions are in oblique coordinates. They satisfy the boundary conditions of the clamped plate at the edges. Results for various parallelogram plate dimensions are given. Nair and Durvasula [7] used a double series of beam characteristic functions with the Ritz method for a number of clamped and simple support combinations with various aspect ratios and skew angles. This references includes a very comprehensive set of nodal patterns, something lacking in other studies. The effect of the skew angle on the nodal patterns and the resonance frequencies is well illustrated. Durvasula and Nair [9] used the partition or subdomain method. It is a modified form of areal collocation. They describe the deflection of the plate with a polynomial that satisfies the boundary conditions. The governing differential equation is then satisfied as best as possible over each of the subdomains. They find the convergence in the cases where they solved the resulting set of linear homogeneous equations to be satisfactory.

Mizusawa et al. [10] used B-spline shape functions with the Rayleigh-Ritz method. The B-spline functions do not satisfy the boundary conditions on their own, and so the method of artificial springs is applied to deal with arbitrary boundary conditions. For each edge, one of the three springs corresponds to the displacement and each of the other two to the slope in the x- and y-directions. The coefficients of the springs are set to zero for free edges, and to infinity for clamped edges. The energy contribution is added to the equation for the maximum potential energy of the skew plate. Final results of the analysis are presented for various plate dimensions and skew angles.

Kuttler and Sigillito [11] turned their attention to finding lower bounds to the resonance frequencies. Linear combinations of trial functions that are not required to satisfy the boundary conditions are combined and optimized in a Ritz-like manner. Results are given for the fully symmetric modes of the clamped rhombic plate. The span between the bounds is quite small, but increases with increasing skew angle.

Nagaya's [12] solution is of interest, because the exact differential equation is used, and the boundary conditions are satisfied by using Fourier expansions to get around the problem created by the polar coordinates used. However, due to algebraic complications, integrations in the solution have to be performed numerically. Results given for simply supported and clamped rhombic and trapezoidal plates are given.

Sakata [13] derived an approximation formula for the natural frequencies of simply supported orthotropic parallelogram plates from the natural frequencies of membranes of the same shape. The solution to the governing differential equation of the membrane is easier than to that of the plate. The author applies his reduction method to obtain approximations for the simply supported isotropic skew plate from the clamped skew membrane. Results for orthotropic plates are included as well.

Subrahmanyam and Wah [14] looked at general quadrilateral plates, and gave the fundamental frequencies of parallelogram plates with various support combinations. The plate is divided along one diagonal into two triangular regions. The differential equation and the boundary conditions along all four edges are satisfied exactly. The frequency is determined by enforcing the continuity condition along the diagonal.

The newer methods, particularly the discrete-element ones, are increasingly general in nature, applying to more general shapes, or handling various support conditions or even free and forced vibration. Geannakakes [15] uses the finite strip method. Arbitrarily shaped plates are mapped into a natural coordinate plane using Serendipity functions. Results are presented for trapezoidal and rhombic plates, among others. Katsikadelis [16,17] applies a static boundary element method to the general problem of free and forced vibration of plates. No results for skew plates are offered, but the method applies to any shape. Matsucla et al.[18] find the fundamental frequencies only for some combination of clamped and simple support, including a combination of these two support types along the same edge.

Huang, McGee, Leissa and Kim [19] focus on highly skewed, simply supported rhombic plates. They reduce the increased error found in most solutions for this type of plate by taking into account the stress singularities in the obtuse angles. Corner functions are used in addition to polynomial shape functions in the Ritz method. The corner functions are found to improve the convergence of the solution. The results are still upper bounds, as is typical of the Ritz method. They are however consistently lower than the results of other upper bound solutions. The authors point to the importance of considering the stress singularities in both, continuum and discrete-element based methods.

Liew and Lam [20] used the Rayleigh-Ritz method. The set of two-dimensional orthogonal shape functions is generated with the Gram-Schmidt orthogonalization process. Results are given for various rhombic plates with combinations of simply supported and clamped edges, as well as cantilevered rhombic plates. They are upper bounds, and agree well with results obtained by Durvasula and Nair [9] and others. Lee and Pon [21] also used a Gram-Schmidt orthogonalization process, but to obtain a set of orthogonal beam functions. Only the geometric boundary conditions are satisfied. Singh and Chakraverty [22] mapped rectangular and skew plates into a unit square. The solution from then on is the same for all plates with the same boundary conditions, and therefore has to be done only once. Orthogonal polynomials that are used in the Rayleigh-Ritz method satisfy the boundary conditions. Results are included for a wide variety of plates with combinations of simply supported, clamped and free edges. The paper contains a comprehensive list of references in the field of thin plate vibration.

A different approach to plate vibration has been taken by Wang, Striz and Bert [23]. Their application of the differential quadrature method to the vibration and buckling of skew plates is similar to a mixed collocation method in that the boundary conditions are only satisfied point-wise. For a given grid, the weighting coefficients have to be calculated only once. The results for the simply supported rhombic plate are in good agreement with those of Gorman [24].

Analytical Solutions

Gorman applied the building block superposition method to fully simply supported and fully clamped rhombic plates [24] and to the simply supported parallelogram plate [25]. He pioneered the method with rectangular plates [26] and applied it to right triangular plates as well [27,28]. With the rhombic plate [24], Gorman makes full use of the symmetry of the plate about both diagonals: only one quarter of the rhombus, that is, one right triangle, needs to be looked at, but for all four possible combinations of symmetry and antisymmetry about the axes. In Reference [25], Gorman uses the point-symmetry of the plate to simplify the analysis. However, this simplification reduces the applicability of the solution to more general cases, for example those featuring different boundary conditions on the various edges. Only half of the plate is actually analyzed. This half is segmented into a rectangular and a right triangular part.

Saliba applied the superposition method to simply supported and fully clamped symmetrical trapezoidal plates [29,30,31], as an example for the general quadrilateral plate. Again, only half of the plate was looked at for the particular cases presented, and that half was divided into rectangular and right triangular segments. The solution to the triangular segment is taken from Gorman [27,28], with two building blocks being rotated to position the forcing functions along the skew edges, as illustrated in Figure 1-1a.

Saliba simplified the superposition solution for the right triangular plate by enforcing the boundary conditions along oblique lines within rectangular building blocks that are not

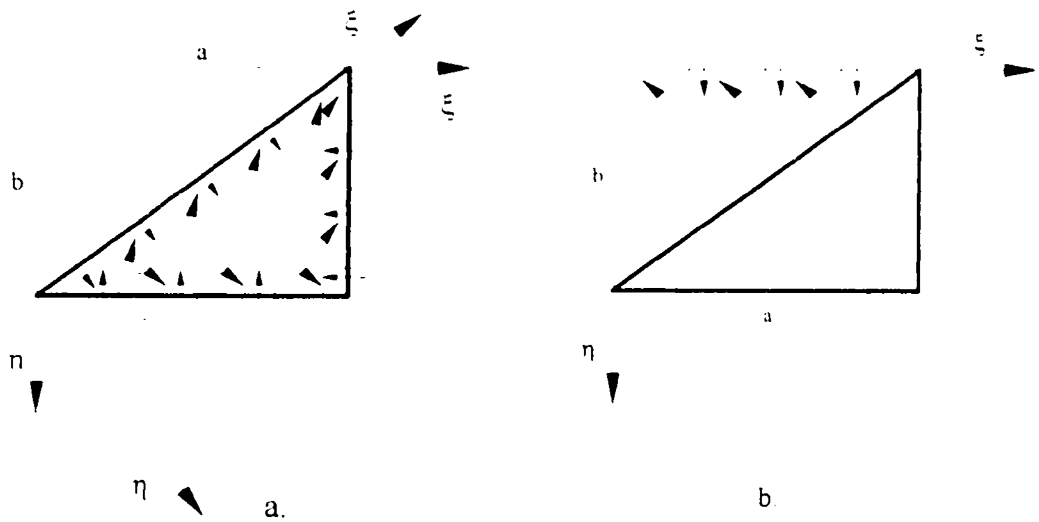


Figure 1-1 Building block combinations for the right triangular plate as used by a. Gorman [27] and b. Saliba [32]. The lower triangle represents the area of interest.

rotated [32,33]. The simplification is from both a derivation and a computation aspect. The number of building blocks is reduced, and the rotated building blocks are eliminated. In fact, only two building blocks along the longer edge, outside of the area of interest, are required to enforce the boundary conditions of simple or clamped support along the hypotenuse, as shown in Figure 1-1b. In [34], clamped supports are added. To clamp either cathesus of the right triangle, a building block with a forced moment along that edge is added. However, Saliba found that with clamping, two additional building blocks with forcing functions along the remaining edge outside of the area of interest was needed for good results with aspect ratios close to unity. He also noted in [34] that doing so improves the mode shapes for simply supported triangular plates. In this work, this simplified triangular solution is used for the triangular sections of the parallelogram plate.

2.2 Experiment

No experimental data for parallelogram plates with clamped supports on all edges exists in the literature. The cantilevered skew plate does receive some attention, possibly because of its similarity to the swept-back wings of airplanes. The present work represents a first by presenting experimental results for the fully clamped parallelogram plate.

3. Basic Assumptions

The plates studied here are considered to be thin, that is, the plate thickness is small compared to the lateral dimensions, as well as to the distance between the nodal lines. Furthermore, the plate displacement is small relative to the thickness of the plate. The effects of rotary inertia are neglected. In-plane forces are considered absent in the theoretical analysis.

4. Plate Geometry

The plates under consideration in this work are parallelograms. The method is readily applied to the general thin quadrilateral plate where only two opposite edges are parallel. For the purpose of generality, no simplifications are made to take advantage of the point symmetry found with some support combinations. The plates are divided into three segments, two triangular and one rectangular, as shown in Figure 1-2. Many plates can be divided in this manner in two ways. The advantage of one over the other is discussed in a later section. The plate size is labeled using either the skew angle α or the aspect ratio of the triangular segments, ϕ_t, and the rectangular aspect ratio ϕ_r. Conversions between the dimensioning conventions of other authors and this work are available in Appendix 4.

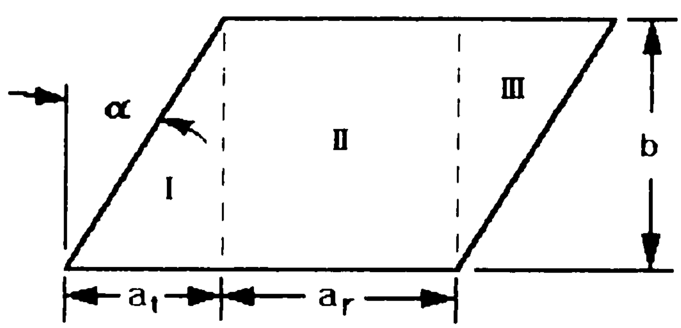


Figure 1-2 Plate layout.

5. Labeling Conventions

Support conditions for plates are labeled either 'S' for simple support or 'C' for clamped support. The edges are labeled starting with the lower edge, moving around the plate in a counter-clockwise manner. For example, a plate with the base clamped and the remaining edges simply supported would be designated as 'CSSS'. The SSCS plate would be identical to the CSSS plate, with the only difference being the rotation by 180°. Similarly, the SSSC plate is the SCSS plate after a 180° rotation. In fact, the SSSC plate can also be rotated by 90° and analyzed as a CSSS plate. The advantages of one over the other are discussed in later sections. Results for nine plate configurations are provided in this work:

- with simply supported base: SSSS, SSSC, SCSC;
- with clamped base and simply supported upper edge: CSSS, CSSC, CCSC;
- with clamped base and top edge: CSCS, CSCC, CCCC.

Boundary conditions are abbreviated in figures as:

- D - displacement,
- M - moment,
- S - slope,
- V - vertical edge reaction.

In the software, edges are numbered, starting from the $\eta = 1$ edge:

- 1: $\eta = 1$
- 3: $\eta = 0$,
- 2: $\xi = 1$
- 4: $\xi = 0$,
- 12: $\eta = 1 - \xi$.

In figures of the building blocks, all edges are simply supported, unless indicated otherwise. The coordinate system used is shown in Figure 1-3 below.

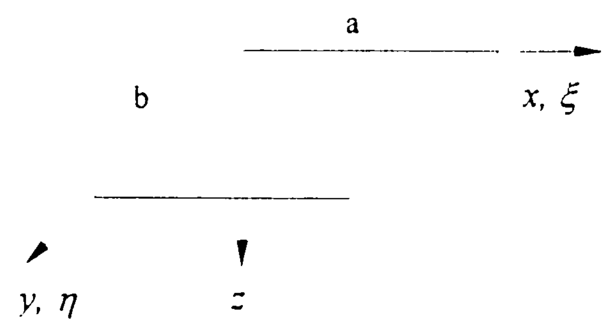


Figure 1-3 Coordinate system.

2 - Theory

1. The Governing Differential Equation

The governing equation for the free vibration of thin, flat plates is a partial differential equation of the fourth order:

$$\frac{\partial^4 W(x,y,t)}{\partial x^4}+2\frac{\partial^4 W(x,y,t)}{\partial x^2 y^2}+\frac{\partial^4 W(x,y,t)}{\partial y^4}+\frac{\rho}{D}\frac{\partial^2 W(x,y,t)}{\partial t^2}=0 \qquad (2\text{-}1)$$

The last term on the left-hand side of the equation accounts for the inertia. It alone is a function of time. By separation of variables using $W(x,y,t)=W(x,y)\,T(t)$, two equations are obtained:

$$\frac{\partial^4 W(x,y)}{\partial x^4}+2\frac{\partial^4 W(x,y)}{\partial x^2 y^2}+\frac{\partial^4 W(x,y)}{\partial y^4}-\omega^2\frac{\rho}{D}W(x,y)=0 \qquad (2\text{-}2)$$

and

$$\frac{\partial^2 T(\tau)}{\partial \tau^2}+\omega^2 T(\tau)=0. \qquad (2\text{-}3)$$

From equation (2-2) it is obvious that the mode shape is independent of time. Equation (2-3) is of no interest to the present analysis. For convenience, Equation (2-2) is non-dimensionalized with respect to the x-dimension of the segment in question. The non-dimensional displacement amplitude becomes

$$W(\xi,\eta)=\frac{W(x,y)}{a}, \qquad (2\text{-}4)$$

and the aspect ratio $\phi=\frac{b}{a}$ is introduced.

Finally, the differential equation follows:

$$\frac{\partial^4 W(\xi,\eta)}{\partial \xi^4}+\frac{2}{\phi^2}\frac{\partial^4 W(\xi,\eta)}{\partial \xi^2 \eta^2}+\frac{1}{\phi^4}\frac{\partial^4 W(\xi,\eta)}{\partial \eta^4}-\lambda^4 W(\xi,\eta)=0 \qquad (2\text{-}5)$$

where $\lambda^2=\omega a^2\sqrt{\rho/D}$ is the eigenvalue, or non-dimensional resonance frequency;

$\omega=2\pi f$ is the circular frequency in rad/s;

ρ is the mass per unit area;

and D is the flexural rigidity of the plate.

2. Boundary Conditions

The following are the boundary conditions for the classical edge conditions in dimensional and non-dimensional forms that are encountered in this work.

2.1 Simple Support

For this support condition, there is no moment or displacement along the edge,

Simple Support along the x- or ξ-axis

$$W(a,y)=\left.\frac{\partial^2 W(x,y)}{\partial x^2}\right|_{x=a}=0 \qquad W(1,\eta)=\left.\frac{\partial^2 W(\xi,\eta)}{\partial \xi^2}\right|_{\xi=1}=0$$

Simple Support along the y- or η-axis

$$W(x,b)=\left.\frac{\partial^2 W(x,y)}{\partial y^2}\right|_{y=b}=0 \qquad W(\xi,1)=\left.\frac{\partial^2 W(\xi,\eta)}{\partial \eta^2}\right|_{\eta=1}=0$$

2.2 Clamped Edges

There is no displacement or slope along the clamped edge.

Along the x- or ξ-axis

$$W(a,y)=\left.\frac{\partial W(x,y)}{\partial x}\right|_{x=a}=0 \qquad W(1,\eta)=\left.\frac{\partial W(\xi,\eta)}{\partial \xi}\right|_{\xi=1}=0$$

along the y- or η-axis

$$W(x,b)=\left.\frac{\partial W(x,y)}{\partial y}\right|_{y=b}=0 \qquad W(\xi,1)=\left.\frac{\partial W(\xi,\eta)}{\partial \eta}\right|_{\eta=1}=0$$

2.3 Slope

Along an oblique line:

$$\frac{\partial W(x,y)}{\partial n}=\frac{\partial W(x,y)}{\partial x}\cos\alpha+\frac{\partial W(x,y)}{\partial y}\sin\alpha$$

$$\frac{\partial W(\xi,\eta)}{\partial n}=\frac{\partial W(\xi,\eta)}{\partial \xi}\cos\alpha+\frac{1}{\phi}\frac{\partial W(\xi,\eta)}{\partial \eta}\sin\alpha$$

The direction of the normal vector n and the location of α are the same as used for the bending moment, shown in Figure 2-2.

2.4 Bending Moment

The positive directions are shown in Figure 2-1.

Along the x- or ξ-axis:

$$M_x = -D\left[\frac{\partial^2 W(x,y)}{\partial x^2} + \nu\frac{\partial^2 W(x,y)}{\partial y^2}\right] \qquad \frac{M_\xi a}{D} = -\left[\frac{\partial^2 W(\xi,\eta)}{\partial \xi^2} + \frac{\nu}{\phi^2}\frac{\partial^2 W(\xi,\eta)}{\partial \eta^2}\right]$$

Along the y- or η-axis:

$$M_y = -D\left[\frac{\partial^2 W(x,y)}{\partial y^2} + \nu\frac{\partial^2 W(x,y)}{\partial x^2}\right] \qquad \frac{M_\eta b^2}{aD} = -\left[\frac{\partial^2 W(\xi,\eta)}{\partial \eta^2} + \nu\phi^2\frac{\partial^2 W(\xi,\eta)}{\partial \xi^2}\right]$$

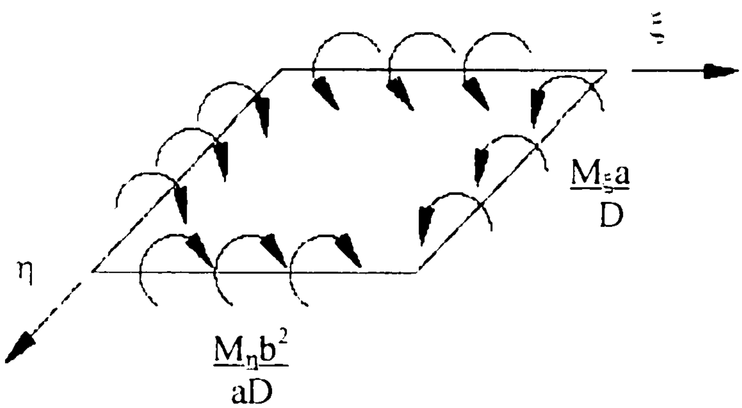


Figure 2-1 Bending moments on the outside edges of a rectangular plate.

In any direction:

Figure 2-2 shows the proper directions.

$$M_n = M_x\cos^2\alpha + M_y\sin^2\alpha - 2M_{xy}\sin\alpha\cos\alpha$$

$$\frac{M_n b^2}{aD} = -\left[\theta_{11}\phi^2\frac{\partial^2 W(\xi,\eta)}{\partial \xi^2} + \theta_{12}\frac{\partial^2 W(\xi,\eta)}{\partial \eta^2} + \theta_{13}\phi\frac{\partial^2 W(\xi,\eta)}{\partial \xi\,\partial \eta}\right]$$

where $\theta_{11} = \cos^2\alpha + \nu\sin^2\alpha$

$\theta_{12} = \sin^2\alpha + \nu\cos^2\alpha$

$\theta_{13} = (1-\nu)\sin 2\alpha$

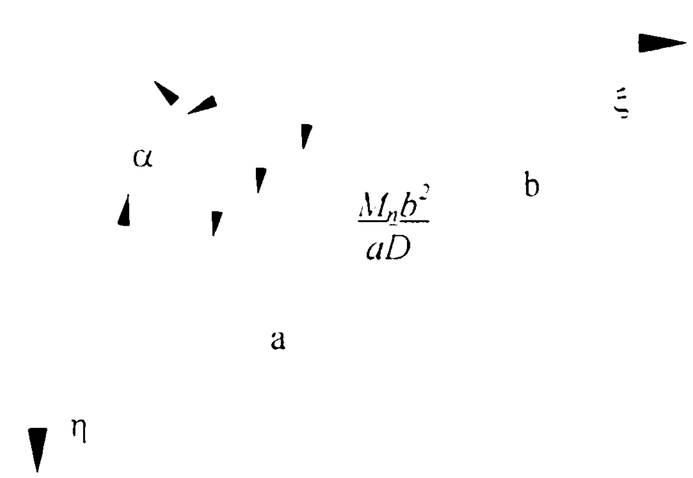


Figure 2-2 Bending moment along an oblique line.

2.5 Vertical Edge Reaction

Figure 2-3 shows the positive orientation of the vertical edge reactions.

Along the x- or ξ-axis:

$$V_x = -D\left[\frac{\partial^3 W(x,y)}{\partial x^3} + \nu^* \frac{\partial^3 W(x,y)}{\partial x\,\partial y^2}\right] \qquad \frac{V_\xi a^2}{D} = -\left[\frac{\partial^3 W(\xi,\eta)}{\partial \xi^3} + \frac{\nu^*}{\phi^2}\frac{\partial^3 W(\xi,\eta)}{\partial \xi\,\partial \eta^2}\right]$$

Along the y- or η-axis:

$$V_y = -D\left[\frac{\partial^3 W(x,y)}{\partial y^3} + \nu^* \frac{\partial^3 W(x,y)}{\partial x^2\,\partial y}\right] \qquad \frac{V_\eta \phi b^2}{D} = -\left[\frac{\partial^3 W(\xi,\eta)}{\partial \eta^3} + \nu^* \phi^2 \frac{\partial^3 W(\xi,\eta)}{\partial \xi^2\,\partial \eta}\right]$$

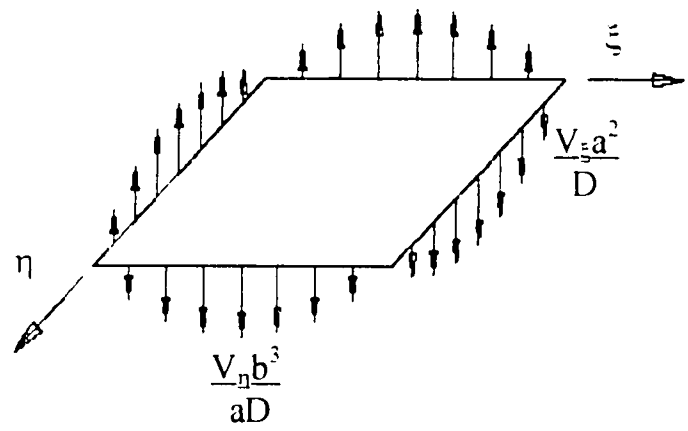


Figure 2-3 Vertical edge reactions on a rectangular plate.

along oblique lines:

$$V_n = -D\left[v_1 \frac{\partial^3 W(x,y)}{\partial x^3} + v_2 \frac{\partial^3 W(x,y)}{\partial y^3} + v_3 \frac{\partial^3 W(x,y)}{\partial x^2 \partial y} + v_4 \frac{\partial^3 W(x,y)}{\partial x \partial y^2}\right]$$

where $v_1 = \cos\alpha + (1-\nu)\dfrac{\sin\alpha \sin 2\alpha}{2}$

$$v_2 = \sin\alpha + (1-\nu)\frac{\cos\alpha \sin 2\alpha}{2}$$

$$v_3 = \sin\alpha - (1-\nu)\left(\sin\alpha\cos 2\alpha + \frac{\cos\alpha\sin 2\alpha}{2}\right)$$

$$v_4 = \cos\alpha + (1-\nu)\left(\cos\alpha\sin 2\alpha + \frac{\sin\alpha\sin 2\alpha}{2}\right)$$

$$\frac{V_n a^2}{D} = -\left[V_1 \frac{\partial^3 W(\xi,\eta)}{\partial \xi^3} + V_2 \frac{\partial^3 W(\xi,\eta)}{\partial \eta^3} + V_3 \frac{\partial^3 W(\xi,\eta)}{\partial \xi \partial \eta^2} + V_4 \frac{\partial^3 W(\xi,\eta)}{\partial \xi^2 \partial \eta}\right]$$

where $V_1 = \cos\alpha + (1-\nu)\dfrac{\sin\alpha\sin 2\alpha}{2}$

$$V_2 = \frac{1}{\phi^3}\left[\sin\alpha + (1-\nu)\frac{\cos\alpha\sin 2\alpha}{2}\right]$$

$$V_3 = \frac{1}{\phi}\left[\sin\alpha - (1-\nu)\left(\sin\alpha\cos 2\alpha + \frac{\cos\alpha\sin 2\alpha}{2}\right)\right]$$

$$V_4 = \frac{1}{\phi^2}\left[\cos\alpha + (1-\nu)\left(\cos\alpha\sin 2\alpha + \frac{\sin\alpha\sin 2\alpha}{2}\right)\right]$$

3. The Lévy-Type Solution

The Lévy-type solution for the free vibration of a rectangular plate requires that the boundary conditions on two opposite edges can be satisfied by a trigonometric function. When two opposite edges are simply supported, this condition is automatically satisfied by the single Fourier series solution of the form

$$W(\xi,\eta) = \sum_{m=1}^{\infty} Y_m(\eta)\sin(m\pi\xi) \qquad (2\text{-}6)$$

This is the original Lévy-type solution for simple support along the $\xi = 0$ and $\xi = 1$ edges. Simple support and slip shear (zero slope and zero vertical edge reaction) conditions can be modeled with cosine, half-sine or half-cosine series. For the present work, only the sine series is used.

Upon substitution of this solution into the governing differential equation (2-5), two solutions for $Y_m(\eta)$ are found, dependent on whether $\lambda^2 - (m\pi)^2$ is positive or negative.

For the case of $\lambda^2 - (m\pi)^2 \quad 0$,

$$Y_m(\eta) = A_m \cosh(\beta_m \eta) + B_m \sinh(\beta_m \eta) + C_m \cos(\gamma_m \eta) + D_m \sin(\gamma_m \eta). \qquad (2\text{-}7)$$

If $\lambda^2 - (m\pi)^2 \quad 0$,

$$Y_m(\eta) = A_m \cosh(\beta_m \eta) + B_m \sinh(\beta_m \eta) + C_m \cosh(\gamma_m \eta) + D_m \sinh(\gamma_m \eta). \qquad (2\text{-}9)$$

For both solutions, $\beta_m = \phi\sqrt{\lambda^2 + (m\pi)^2}$

and $\gamma_m = \phi\sqrt{\lambda^2 - (m\pi)^2}$

or $\gamma_m = \phi\sqrt{(m\pi)^2 - \lambda^2}$, whichever is real.

The constants A_m, B_m, C_m and D_m are determined by satisfying the boundary conditions along the remaining two edges. Possible boundary conditions include simple and clamped support conditions, slip shear, and forced conditions such as forced displacement or slope. Solutions that have simple support conditions on one edge and either a forced harmonic displacement or a forced harmonic moment are used in this work. The forced moment was chosen over the forced rotation for reasons that are discussed in more detail in the next section.

4. The Superposition Method

The Lévy-type solution alone cannot be used to model a non-rectangular plate or every possible support combination on a rectangular plate. However, the Lévy-type solutions to a plate can be a building block. Several building blocks can be superimposed so that the net boundary conditions are met exactly.

The solution can also be applied to simple shapes like right-triangular plates, as seen in Figure 1-1 . The forcing functions on the outside edges of rectangular building blocks are adjusted to enforce the boundary conditions along oblique lines within the building block boundary. This oblique line is usually conveniently chosen as one of the diagonals. Beyond rectangular and triangular plates, plate solutions can be joined by requiring continuity of displacement, slope, and moment as well as a zero net vertical edge reaction along the interface of segments. Superposition solutions to triangular and rectangular plates can be linked to analyze general quadrilateral plates.

In this work, the parallelogram plates is used as an example. As can be seen in later sections, only eight building blocks are used. Actually, the solution to two building blocks is rotated by 90°, 180° and 270° to obtain the other six. Each building block has three sides simply supported, and a forced displacement or a forced bending moment along the remaining edge. These forcing functions are both sine functions. Bending moment is a second derivative of displacement. The second derivative of the sine function is a sine function also. That simplifies the solution process greatly.

5. Building Block Solutions

5.1 Building Block 1

Building Block 1 features a forced displacement along the η=1 edge of the form

$$W_1(\xi,\eta)\big|_{\eta=1} = \sum_{m=1}^{\infty} E_{1m}\sin(m\pi\xi). \qquad (2\text{-}9)$$

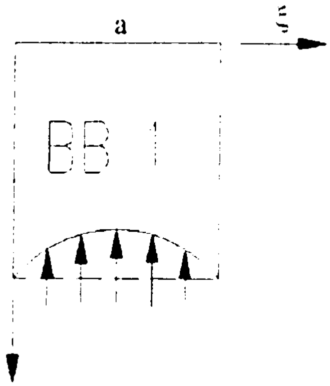


Figure 2-4 Building Block 1.

The bending moment along this edge is zero. The remaining edges are simply supported, as shown in Figure 2-4. The solution then becomes

$$\begin{aligned} W_1(\xi,\eta) = & \sum_{m=1}^{k^*} E_{1m}\theta_{11m}\left[\sinh(\beta_m\eta) + C_{11m}\sin(\gamma_m\eta)\right]\sin(m\pi\xi) \\ & + \sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}\left[\sinh(\beta_m\eta) + C_{12m}\sinh(\gamma_m\eta)\right]\sin(m\pi\xi) \end{aligned} \qquad (2\text{-}10)$$

where $\beta_m = \phi\sqrt{\lambda^2 + (m\pi)^2}$,

$\gamma_m = \phi\sqrt{\lambda^2 - (m\pi)^2}$ or $\gamma_m = \phi\sqrt{(m\pi)^2 - \lambda^2}$, whichever is real.

and

$$C_{11m} = \frac{\beta_m^2 - \nu\phi^2(m\pi)^2}{\gamma_m^2 + \nu\phi^2(m\pi)^2}\,\frac{\sinh\beta_m}{\sin\gamma_m}$$

$$C_{12m} = \frac{\nu\phi^2(m\pi)^2 - \beta_m^2}{\gamma_m^2 - \nu\phi^2(m\pi)^2}\,\frac{\sinh\beta_m}{\sinh\gamma_m}$$

$$\theta_{11m} = \frac{1}{\sinh\beta_m + C_{11m}\sin\gamma_m}$$

$$\theta_{12m} = \frac{1}{\sinh\beta_m + C_{12m}\sinh\gamma_m}$$

5.2 Building Block 2

Building Block 2 is shown in Figure 2-5. It has no displacement along the edge $\eta=1$. Instead, the function

$$\left.\frac{Mb^2}{aD}\right|_{\eta=1} = \sum_{m=1}^{\tau} E_{2m}\sin(m\pi\xi) \qquad (2\text{-}11)$$

describes the forced moment. The remaining edges are simply supported. The solution for this building block then becomes

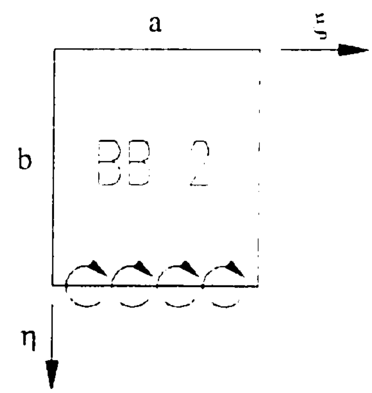


Figure 2-5 Building Block 2.

$$\begin{aligned} W_2(\xi,\eta) &= \sum_{m=1}^{k^*} E_{2m}\theta_{21m}\left[\sinh(\beta_m\eta) + C_{21m}\sin(\gamma_m\eta)\right]\sin(m\pi\xi) \\ &+ \sum_{m=k^*+1}^{\tau} E_{2m}\theta_{22m}\left[\sinh(\beta_m\eta) + C_{22m}\sinh(\gamma_m\eta)\right]\sin(m\pi\xi) \end{aligned} \qquad (2\text{-}12)$$

with $\beta_m = \phi\sqrt{\lambda^2 + (m\pi)^2}$,

$\gamma_m = \phi\sqrt{\lambda^2 - (m\pi)^2}$ or $\gamma_m = \phi\sqrt{(m\pi)^2 - \lambda^2}$, whichever is real,

and

$$C_{21m} = -\frac{\sinh\beta_m}{\sin\gamma_m}$$

$$C_{22m} = -\frac{\sinh\beta_m}{\sinh\gamma_m}$$

$$\theta_{21m} = \frac{-1}{\beta_m^2\sinh\beta_m - C_{21m}\gamma_m^2\sin\gamma_m}$$

$$\theta_{22m} = \frac{-1}{\beta_m^2\sinh\beta_m + C_{21m}\gamma_m^2\sinh\gamma_m}$$

5.3 Building Block 3

Building Block 3, shown in Figure 2-6, is found by rotating Building Block 1 by 90°. The displacement along the $\xi=1$ edge is forced using the function

$$W_3(\xi,\eta)\big|_{\xi=1} = \sum_{m=1}^{\infty} E_{3m}\sin(m\pi\eta)\,. \qquad (2\text{-}13)$$

The moment along this edge is zero. The remaining edges are simply supported.

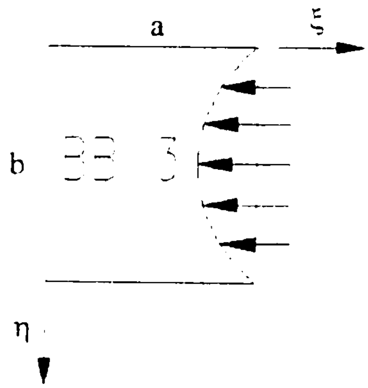


Figure 2-6 Building Block 3.

The solution then becomes

$$\begin{aligned} W_3(\xi,\eta) &= \sum_{m=1}^{k^*} E_{3m}\theta_{31m}\left[\sinh(\beta_{3m}\xi) + C_{31m}\sin(\gamma_{3m}\xi)\right]\sin(m\pi\eta) \\ &\quad + \sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}\left[\sinh(\beta_{3m}\xi) + C_{32m}\sinh(\gamma_{3m}\xi)\right]\sin(m\pi\eta) \end{aligned} \qquad (2\text{-}14)$$

where $\beta_{3m} = \dfrac{1}{\phi}\sqrt{(\phi\lambda)^2 + (m\pi)^2}$,

$\gamma_{3m} = \dfrac{1}{\phi}\sqrt{(\phi\lambda)^2 - (m\pi)^2}$ or $\gamma_{3m} = \dfrac{1}{\phi}\sqrt{(m\pi)^2 - (\phi\lambda)^2}$, whichever is real,

and

$$C_{31m} = \frac{\phi^2\beta_{3m}^2 - \nu(m\pi)^2}{\phi^2\gamma_{3m}^2 + \nu(m\pi)^2}\,\frac{\sinh\beta_{3m}}{\sin\gamma_{3m}}$$

$$C_{32m} = -\frac{\phi^2\beta_{3m}^2 - \nu(m\pi)^2}{\phi^2\gamma_{3m}^2 - \nu(m\pi)^2}\,\frac{\sinh\beta_{3m}}{\sinh\gamma_{3m}}$$

$$\theta_{31m} = \frac{1}{\sinh\beta_{3m} + C_{31m}\sin\gamma_{3m}}$$

$$\theta_{32m} = \frac{1}{\sinh\beta_{3m} + C_{32m}\sinh\gamma_{3m}}$$

5.4 Building Block 4

Building Block 4 has no displacement along the edge ξ=1, as indicated in Figure 2-7. The function

$$\left.\frac{Mb^2}{aD}\right|_{\xi=1} = \sum_{m=1}^{\infty} E_{4m}\sin(m\pi\xi) \qquad (2\text{-}15)$$

describes the forced moment along this edge. The remaining edges are simply supported. The solution for this building block then is that of Building Block 2, rotated by 90°:

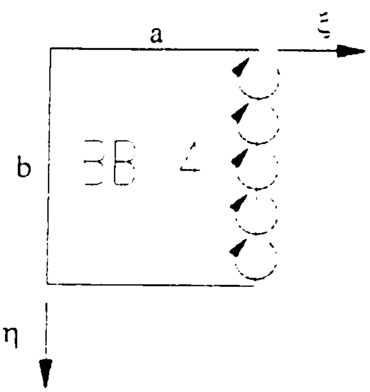


Figure 2-7 Building Block 4.

$$\begin{aligned} W_4(\xi,\eta) &= \sum_{m=1}^{k^*} E_{4m}\theta_{41m}\left[\sinh(\beta_{3m}\xi) + C_{41m}\sin(\gamma_{3m}\xi)\right]\sin(m\pi\eta) \\ &+ \sum_{m=k^*+1}^{\infty} E_{4m}\theta_{42m}\left[\sinh(\beta_{3m}\xi) + C_{42m}\sinh(\gamma_{3m}\xi)\right]\sin(m\pi\eta) \end{aligned} \qquad (2\text{-}16)$$

with $\beta_{3m} = \frac{1}{\phi}\sqrt{(\phi\lambda)^2 + (m\pi)^2}$ and $\gamma_{3m} = \frac{1}{\phi}\sqrt{(\phi\lambda)^2 - (m\pi)^2}$ or

$\gamma_{3m} = \frac{1}{\phi}\sqrt{(m\pi)^2 - (\phi\lambda)^2}$, whichever is real, as defined for Building Block 3,

and

$$C_{41m} = -\frac{\sinh\beta_{3m}}{\sin\gamma_{3m}}$$

$$C_{42m} = -\frac{\sinh\beta_{3m}}{\sinh\gamma_{3m}}$$

$$\theta_{41m} = \frac{-1}{\beta_{3m}^2\sinh\beta_{3m} - C_{41m}\gamma_{3m}^2\sin\gamma_{3m}}$$

$$\theta_{42m} = \frac{-1}{\beta_{3m}^2\sinh\beta_{3m} + C_{42m}\gamma_{3m}^2\sinh\gamma_{3m}}$$

5.5 Building Block 5

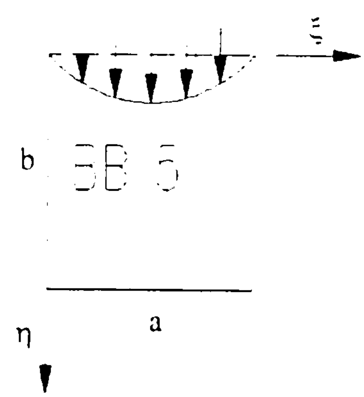


Figure 2-8 Building Block 5.

Building Block 5 is obtained by rotating building blocks 1 by 180°. The forced displacement is applied to the η = 0 edge, as seen in Figure (2-8):

$$W_5(\xi,\eta)\big|_{\eta=0} = \sum_{m=1}^{r} E_{5m}\sin(m\pi(1-\xi)) \qquad (2\text{-}17)$$

The remaining edges are simply supported. The displacement for Building Block 5 becomes

$$\begin{aligned} W_5(\xi,\eta) &= \sum_{m=1}^{k^*} E_{5m}\theta_{51m}[\sinh(\beta_m\eta) + C_{51m}\sin(\gamma_m\eta)]\sin(m\pi(1-\xi)) \\ &+ \sum_{m=k^*+1}^{r} E_{5m}\theta_{52m}[\sinh(\beta_m\eta) + C_{52m}\sinh(\gamma_m\eta)]\sin(m\pi(1-\xi)) \end{aligned} \qquad (2\text{-}18)$$

where $\beta_m = \phi\sqrt{\lambda^2 + (m\pi)^2}$

and $\gamma_m = \phi\sqrt{\lambda^2 - (m\pi)^2}$ or $\gamma_m = \phi\sqrt{(m\pi)^2 - \lambda^2}$, whichever is real.

The coefficients are identical to the corresponding ones for building block 1:

$$C_{51m} = \frac{\beta_m^2 - v\phi^2(m\pi)^2}{\gamma_m^2 + v\phi^2(m\pi)^2}\,\frac{\sinh\beta_m}{\sin\gamma_m}$$

$$C_{52m} = \frac{v\phi^2(m\pi)^2 - \beta_m^2}{\gamma_m^2 - v\phi^2(m\pi)^2}\,\frac{\sinh\beta_m}{\sinh\gamma_m}$$

$$\theta_{51m} = \frac{1}{\sinh\beta_m + C_{51m}\sin\gamma_m}$$

$$\theta_{52m} = \frac{1}{\sinh\beta_m + C_{52m}\sinh\gamma_m}$$

5.6 Building Block 6

To obtain Building Block 6, Building Block 2 is rotated by 180°. As seen in Figure 2-9, the forced bending moment is now applied to the $\eta = 0$ edge:

$$\left.\frac{Mb^2}{aD}\right|_{\eta=0} = \sum_{m=1}^{\infty} E_{6m} \sin(m\pi(1-\xi)) \qquad (2\text{-}19)$$

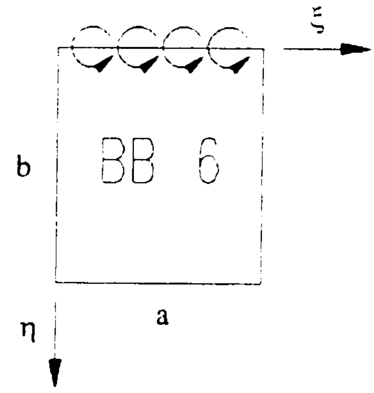


Figure 2-9 Building Block 6.

The remaining edges are simply supported. The solution for the displacement anywhere on the plate follows:

$$\begin{aligned} W_6(\xi,\eta) &= \sum_{m=1}^{k^*} E_{6m}\theta_{61m}[\sinh(\beta_m\eta) + C_{61m}\sin(\gamma_m\eta)]\sin(m\pi(1-\xi)) \\ &+ \sum_{m=k^*+1}^{\infty} E_{6m}\theta_{62m}[\sinh(\beta_m\eta) + C_{62m}\sinh(\gamma_m\eta)]\sin(m\pi(1-\xi)) \end{aligned} \qquad (2\text{-}20)$$

with $\beta_m = \phi\sqrt{\lambda^2 + (m\pi)^2}$

$\gamma_m = \phi\sqrt{\lambda^2 - (m\pi)^2}$ or $\gamma_m = \phi\sqrt{(m\pi)^2 - \lambda^2}$, whichever is real.

and

$$C_{61m} = -\frac{\sinh\beta_m}{\sin\gamma_m}$$

$$C_{61m} = -\frac{\sinh\beta_m}{\sinh\gamma_m}$$

$$\theta_{61m} = \frac{-1}{\beta_m^2\sinh\beta_m - C_{61m}\gamma_m^2\sin\gamma_m}$$

$$\theta_{62m} = \frac{-1}{\beta_m^2\sinh\beta_m + C_{62m}\gamma_m^2\sinh\gamma_m}$$

These coefficients are identical to those for building block 2.

5.7 Building Block 7

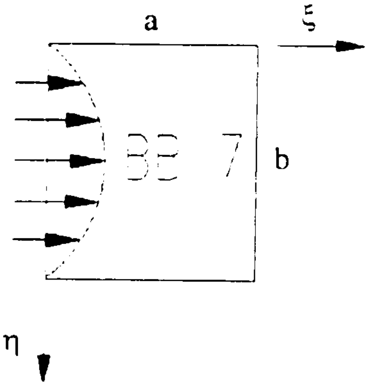


Figure 2-10 Building Block 7.

Building Block 7 is shown in Figure 2-10. It is found by rotating Building Block 3 by 180°. The displacement along the $\xi = 0$ edge is forced using the function

$$W_7(\xi,\eta)\big|_{\xi=0} = \sum_{m=1}^{\infty} E_{7m} \sin(m\pi(1-\eta)). \qquad (2\text{-}21)$$

The moment along this edge is zero. The remaining edges are simply supported. The solution then becomes

$$\begin{aligned} W_7(\xi,\eta) &= \sum_{m=1}^{k^*} E_{7m}\theta_{71m}\left[\sinh(\beta_{3m}(1-\xi)) + C_{71m}\sin(\gamma_{3m}(1-\xi))\right]\sin(m\pi\eta) \\ &+ \sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}\left[\sinh(\beta_{3m}(1-\xi)) + C_{72m}\sinh(\gamma_{3m}(1-\xi))\right]\sin(m\pi\eta) \end{aligned} \qquad (2\text{-}22)$$

where $\beta_{3m} = \dfrac{1}{\phi}\sqrt{(\phi\lambda)^2 + (m\pi)^2}$,

$\gamma_{3m} = \dfrac{1}{\phi}\sqrt{(\phi\lambda)^2 - (m\pi)^2}$ or $\gamma_{3m} = \dfrac{1}{\phi}\sqrt{(m\pi)^2 - (\phi\lambda)^2}$, whichever is real,

and

$$C_{71m} = \frac{\phi^2\beta_{3m}^2 - \nu(m\pi)^2}{\phi^2\gamma_{3m}^2 + \nu(m\pi)^2}\,\frac{\sinh\beta_{3m}}{\sin\gamma_{3m}}$$

$$C_{72m} = -\frac{\phi^2\beta_{3m}^2 - \nu(m\pi)^2}{\phi^2\gamma_{3m}^2 - \nu(m\pi)^2}\,\frac{\sinh\beta_{3m}}{\sinh\gamma_{3m}}$$

$$\theta_{71m} = \frac{1}{\sinh\beta_{3m} + C_{71m}\sin\gamma_{3m}}$$

$$\theta_{72m} = \frac{1}{\sinh\beta_{3m} + C_{72m}\sinh\gamma_{3m}}$$

5.8 Building Block 8

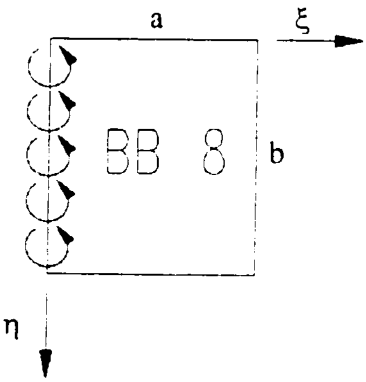


Figure 2-11 Building Block 8.

Building Block 8 has no displacement along the edge $\xi = 0$. The function

$$\left.\frac{Mb^2}{aD}\right|_{\xi=0} = \sum_{m=1}^{\infty} E_{8m} \sin(m\pi(1-\xi)) \qquad (2\text{-}23)$$

describes the forced moment along this edge, while the remaining edges are simply supported. This is shown in Figure (2-11). The solution for this building block then is that of Building Block 4, rotated by 180°:

$$W_8(\xi,\eta) = \sum_{m=1}^{k^*} E_{8m}\theta_{81m}\left[\sinh(\beta_{3m}(1-\xi)) + C_{81m}\sin(\gamma_{3m}(1-\xi))\right]\sin(m\pi\eta)$$
$$+ \sum_{m=k^*+1}^{\infty} E_{8m}\theta_{82m}\left[\sinh(\beta_{3m}(1-\xi)) + C_{82m}\sinh(\gamma_{3m}(1-\xi))\right]\sin(m\pi\eta) \qquad (2\text{-}24)$$

with $\beta_{3m} = \frac{1}{\phi}\sqrt{(\phi\lambda)^2 + (m\pi)^2}$

and $\gamma_{3m} = \frac{1}{\phi}\sqrt{(\phi\lambda)^2 - (m\pi)^2}$ or $\gamma_{3m} = \frac{1}{\phi}\sqrt{(m\pi)^2 - (\phi\lambda)^2}$, whichever is real,

as defined for Building Block 3, and

$$C_{81m} = -\frac{\sinh\beta_{3m}}{\sin\gamma_{3m}}$$

$$C_{82m} = -\frac{\sinh\beta_{3m}}{\sinh\gamma_{3m}}$$

$$\theta_{81m} = \frac{-1}{\beta_{3m}^2\sinh\beta_{3m} - C_{81m}\gamma_{3m}^2\sin\gamma_{3m}}$$

$$\theta_{82m} = \frac{-1}{\beta_{3m}^2\sinh\beta_{3m} + C_{82m}\gamma_{3m}^2\sinh\gamma_{3m}}$$

6. Building Block Selection

The building blocks have to be chosen with the net boundary conditions on the edges of the plate to be modeled and the continuity between segments in mind. Figure 2-12 shows the combinations chosen for this work the parallelogram plate. There is not necessarily one possible combination to choose from. In fact, the solution can be checked by using a different set of building blocks to solve the same problem. The number of distinct building blocks that are required and computational considerations come to play.

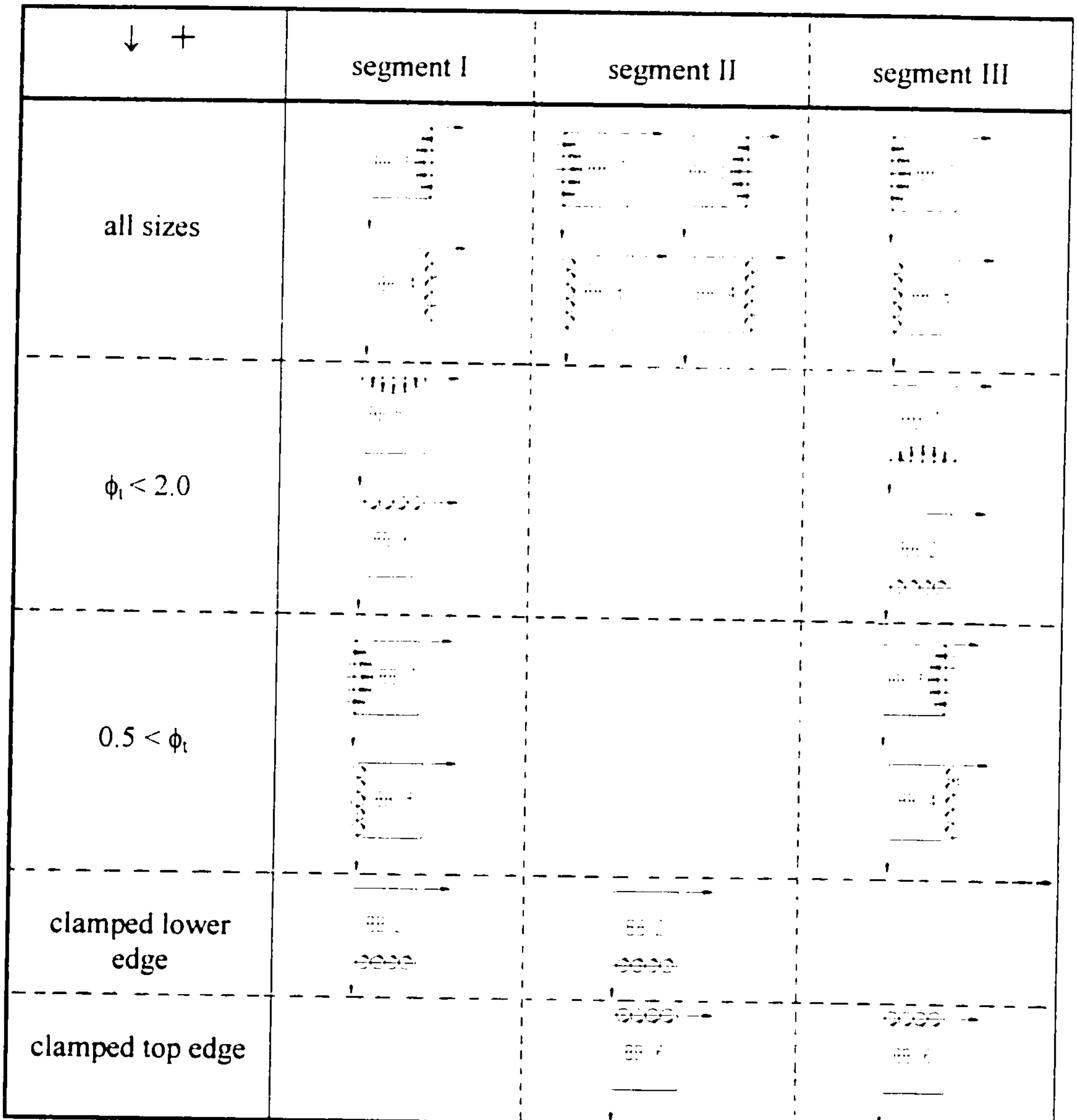


Figure 2-12 Building block combination overview.

There is a basic set that all plate sizes require to meet the requirements for continuity between the segments. The conditions for simple support along the upper and lower edges are satisfied automatically. Additional building blocks are needed to enforce the boundary conditions along the skew edges.

When the skew angle is very large or very small, it suffices to have the forcing functions needed to obtain the boundary conditions along the skew edge along the longer side only. In the case of a large value for ϕ_t (small skew angle) for example, Building Blocks 7 and 8 alone are sufficient to control the skew edge of segment I.

For clamping the skew edges, only the boundary conditions have to be changed. The building block combination remains the same. If however the lower or top edges are to be clamped, the building blocks with the forced bending moments must be added to force the net slope to zero.

7. Continuity Between Segments

Four conditions have to be met along the interfaces of the segments of the quadrilateral plate:

- continuity of displacement,
- continuity of slope,
- continuity of bending moment,
- and force equilibrium.

Continuity of displacement can be taken care of by the coefficients E_m of each of the building block solutions.

At the edge common to segments I and II: $W_I - W_{II} = 0,$ (2-25a)

or at the interface between segments II and III, $W_{II} - W_{III} = 0.$ (2-25b)

To maintain continuity of slope, the contribution of the rectangular segment, or in the case of a general quadrilateral plate, also the third segment, has to be adjusted by the ratio of the aspect ratios of the adjacent segments, ϕ_r ϕ_t:

$$W'_I \cdot (\phi_r/\phi_t) W'_{II} \quad 0 \tag{2-26a}$$

$$\text{and} \quad (\phi_r/\phi_t) W'_{II} \cdot W'_{III} \quad 0. \tag{2-26b}$$

For the bending moment, the adjustment factor becomes $(\phi_r \ \phi_t)^2$:

$$W''_I \cdot (\phi_r/\phi_t)^2 W''_{II} \quad 0 \tag{2-27a}$$

$$\text{and} \quad (\phi_r/\phi_t)^2 W''_{II} \cdot W''_{III} \quad 0. \tag{2-27b}$$

Vertical edge reactions of segment II are multiplied by a factor of $(\phi_r \ \phi_t)^3$:

$$W'''_I \cdot (\phi_r/\phi_t)^3 W'''_{II} \quad 0 \tag{2-28a}$$

$$\text{and} \quad (\phi_r/\phi_t)^3 W'''_{II} \cdot W'''_{III} - 0. \tag{2-28b}$$

The contributions of the building blocks are computed with the appropriate values of ϕ and λ^2 for that segment. Since in this work the reference dimension for the eigenvalue is chosen as the base length of the triangular segment a_t, λ^2 has to be changed for computations involving the rectangular segment:

$$\lambda_r^2 = \lambda_t^2 \frac{a_r^2}{a_t^2} = \lambda_t^2 \frac{\phi_t^2}{\phi_r^2} \tag{2-29}$$

In the analysis of a general quadrilateral plate, λ^2 would have to be adjusted for any other segments as well. The reference dimension for segments I and III for the parallelogram plate is the same, and so this adjustment is not needed.

8. Integral Expansions

When the building blocks are superimposed, their contributions to the various boundary conditions have to be of a compatible form. This is achieved by expanding them in an appropriate Fourier series, where required. The expansion is carried out for each particular edge and condition. A list of integrals that are frequently used in the process is given in Appendix 1. Once the expansions are obtained, the eigenvalue matrix can be put together. The selection of building blocks is explained in Section 6. Details of the eigenvalue matrix assembly are presented in Section 9. To understand the organization of the expansions it should be mentioned here that this eigenvalue matrix consists of submatrices $AS(i,j)$, where i indicates the row, and j, the column. Each combination of edge, e.g. $\eta = 0$, and condition, e.g. slope, has a submatrix associated with it. As the expansions are given, their place in the appropriate submatrix is also given.

In the following, the expansions are grouped by pairs of building blocks. The expansions for building blocks with the forcing functions located along the same edge differ only in the coefficients and in the contribution to the displacement or moment along the edge along which the forcing functions are located. For example, the expansions for Building Block 1 are almost identical to those for Building Block 2.

8.1 Building Block 1

8.1.1 Displacement

a. Displacement along the η = 1 edge

For Building Block 1, there is a forced displacement along this edge,

$$W_1(\xi,\eta)\Big|_{\eta=1} = \sum_{m=1}^{\infty} E_{1m}\sin(m\pi\xi) \qquad (2\text{-}9)$$

which is already a sine in the desired terms series. Hence no expansion is required.

$$AS(j,j) = 1.0, \qquad (2\text{-}30a)$$

and where $i \neq j$,

$$AS(i,j) = 0. \qquad (2\text{-}30b)$$

b. Displacement along the $\xi = 1$, $\eta = 0$, $\xi = 0$ edges

There is no contribution by Building Block 1 to the displacement along these edges:

$$AS(i,j) = 0. \tag{2-31}$$

c. Displacement along $\eta = 1 - \xi$

This line represents one of the diagonals of the rectangular building blocks. The unexpanded solution is

$$W_1(\xi,\eta) = \sum_{m=1}^{k^*} E_{1m}\theta_{11m}\left[\sinh(\beta_m(1-\xi)) + C_{11m}\sin(\gamma_m(1-\xi))\right]\sin(m\pi\xi)$$
$$+ \sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}\left[\sinh(\beta_m(1-\xi)) + C_{12m}\sinh(\gamma_m(1-\xi))\right]\sin(m\pi\xi)$$

After expansion, this solution becomes

$$\begin{aligned} W_1(\xi,\eta) &= \sum_{m=1}^{k^*} E_{1m}\theta_{11m}\left[ex1 + ex2 + C_{11m}(ex3 + ex4)\right] \\ &+ \sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}\left[ex1 + ex2 + C_{12m}(ex5 + ex6)\right] \end{aligned} \tag{2-32}$$

where

$$ex1 = \frac{\beta_m\left[\cosh\beta_m - \cos(m\pi - n\pi)\right]}{\beta_m^2 + (m\pi - n\pi)^2}$$

$$ex2 = \frac{\beta_m\left[\cos(m\pi + n\pi) - \cosh\beta_m\right]}{\beta_m^2 + (m\pi + n\pi)^2}$$

$$ex3 = \frac{\gamma_m\left[\cos(m\pi - n\pi) - \cos\gamma_m\right]}{\gamma_m^2 - (m\pi - n\pi)^2}$$

$$ex4 = \frac{\gamma_m\left[\cos\gamma_m - \cos(m\pi + n\pi)\right]}{\gamma_m^2 - (m\pi + n\pi)^2}$$

$$ex5 = \frac{\gamma_m\left[\cosh\gamma_m - \cos(m\pi - n\pi)\right]}{\gamma_m^2 + (m\pi - n\pi)^2}$$

$$ex6 = \frac{\gamma_m\left[\cos(m\pi + n\pi) - \cosh\gamma_m\right]}{\gamma_m^2 + (m\pi + n\pi)^2}$$

The elements of the submatrix then become

$$AS(i,j) = \theta_{11m}\left[ex1 + ex2 + C_{11m}(ex3 + ex4)\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0, \tag{2-33a}$$

or

$$AS(i,j) = \theta_{12m}\left[ex1 + ex2 + C_{12m}(ex5 + ex6)\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0. \tag{2-33b}$$

8.1.2 Slope

a. Slope along the η = 1 edge

$$\left.\frac{\partial W_1(\xi,\eta)}{\partial \eta}\right|_{\eta=1} = \sum_{m=1}^{k^*} E_{1m}\theta_{11m}\left[\beta_m \cosh\beta_m + C_{11m}\gamma_m \cos\gamma_m\right]\sin(m\pi\xi) + \sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}\left[\beta_m \cosh\beta_m + C_{12m}\gamma_m \cosh\gamma_m\right]\sin(m\pi\xi)$$

All elements that are not on the diagonal of the submatrix are zero:

$$AS(i,j) \quad 0.0. \tag{2-34a}$$

On the diagonal,

$$AS(i,j) = \theta_{11m}(\beta_m \cosh\beta_m + C_{11m}\gamma_m \cos\gamma_m) \qquad \text{for } \lambda^2 - (m\pi)^2 > 0, \tag{2-34b}$$

and $$AS(i,j) = \theta_{12m}(\beta_m \cosh\beta_m + C_{12m}\gamma_m \cosh\gamma_m) \qquad \text{for } \lambda^2 - (m\pi)^2 < 0. \tag{2-34c}$$

b. Slope along the ξ = 1 edge

$$\left.\frac{\partial W_1(\xi,\eta)}{\partial \xi}\right|_{\xi=1} = \sum_{m=1}^{k^*} E_{1m}\theta_{11m}(m\pi)\left[\sinh\beta_m\eta + C_{11m}\sin\gamma_m\eta\right]\cos(m\pi) + \sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}(m\pi)\left[\sinh\beta_m\eta + C_{12m}\sinh\gamma_m\eta\right]\cos(m\pi)$$

In expanded form, this becomes

$$AS(i,j) = \theta_{11m}(m\pi)\cos(m\pi)\left[ex1 + C_{11m}ex2\right] \qquad \text{for } \lambda^2 - (m\pi)^2 > 0, \tag{2-35a}$$

and $$AS(i,j) = \theta_{12m}(m\pi)\cos(m\pi)\left[ex1 + C_{12m}ex3\right] \qquad \text{for } \lambda^2 - (m\pi)^2 < 0, \tag{2-35b}$$

where

$$ex1 = \frac{-2(n\pi)\cos(n\pi)\sinh\beta_m}{\beta_m^2 + (n\pi)^2}$$

$$ex2 = \frac{2(n\pi)\cos(n\pi)\sin\gamma_m}{\gamma_m^2 - (n\pi)^2}$$

$$ex3 = \frac{-2(n\pi)\cos(n\pi)\sinh\gamma_m}{\gamma_m^2 + (n\pi)^2}$$

c. Slope along the $\eta = 0$ edge

$$\left.\frac{\partial W_1(\xi,\eta)}{\partial \eta}\right|_{\eta=0} = \sum_{m=1}^{K^*} E_{1m}\theta_{11m}\left[\beta_m + C_{11m}\gamma_m\right]\sin(m\pi\xi) + \sum_{m=K^*+1}^{\infty} E_{1m}\theta_{12m}\left[\beta_m + C_{12m}\gamma_m\right]\sin(m\pi\xi)$$

Elements off the diagonal of the submatrix are zero: $AS(i,j) = 0$. (2-36a)

The elements on the diagonal are

$$AS(i,j) = \theta_{11m}(\beta_m + C_{11m}\gamma_m) \qquad \text{for } \lambda^2 - (m\pi)^2 > 0, \qquad (2\text{-}36\text{b})$$

and

$$AS(i,j) = \theta_{12m}(\beta_m + C_{12m}\gamma_m) \qquad \text{for } \lambda^2 - (m\pi)^2 < 0. \qquad (2\text{-}36\text{c})$$

d. Slope along the $\xi = 0$ edge

$$\left.\frac{\partial W_1(\xi,\eta)}{\partial \xi}\right|_{\xi=0} = \sum_{m=1}^{K^*} E_{1m}\theta_{11m}(m\pi)\left[\sinh\beta_m\eta + C_{11m}\sin\gamma_m\eta\right] + \sum_{m=K^*+1}^{\infty} E_{1m}\theta_{12m}(m\pi)\left[\sinh\beta_m\eta + C_{12m}\sinh\gamma_m\eta\right]$$

In expanded for, this then becomes

$$AS(i,j) = \theta_{11m}(m\pi)\left[ex1 + C_{11m}ex2\right] \qquad \text{for } \lambda^2 - (m\pi)^2 > 0, \qquad (2\text{-}37\text{a})$$

and

$$AS(i,j) = \theta_{12m}(m\pi)\left[ex1 + C_{12}ex3\right] \qquad \text{for } \lambda^2 - (m\pi)^2 < 0, \qquad (2\text{-}37\text{b})$$

where *ex1*, *ex2* and *ex3* are identical to *ex1*, *ex2* and *ex3* for $\xi = 1$ found in section 8.1.2.b:

$$ex1 = \frac{-2(n\pi)\cos(n\pi)\sinh\beta_m}{\beta_m^2 + (n\pi)^2}$$

$$ex2 = \frac{2(n\pi)\cos(n\pi)\sin\gamma_m}{\gamma_m^2 - (n\pi)^2}$$

$$ex3 = \frac{-2(n\pi)\cos(n\pi)\sinh\gamma_m}{\gamma_m^2 + (n\pi)^2}$$

e. Slope along the $\eta = 1 - \xi$ diagonal

The slope along any oblique line is

$$\frac{\partial W_1(\xi,\eta)}{\partial n} = \frac{\partial W_1(\xi,\eta)}{\partial \xi}\cos\alpha + \frac{1}{\phi}\frac{\partial W_1(\xi,\eta)}{\partial \xi}\sin\alpha$$

where the derivatives for this building block along the diagonal in question are

$$\left.\frac{\partial W_1(\xi,\eta)}{\partial \xi}\right|_{\eta=1-\xi} = \sum_{m=1}^{k^*} E_{1m}\theta_{11m}(m\pi)\left[\sinh\beta_m(1-\xi) + C_{11m}\sin\gamma_m(1-\xi)\right]\cos(m\pi\xi)$$

$$+\sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}(m\pi)\left[\sinh\beta_m(1-\xi) + C_{12m}\sinh\gamma_m(1-\xi)\right]\cos(m\pi\xi)$$

$$\left.\frac{\partial W_1(\xi,\eta)}{\partial \eta}\right|_{\xi=1-\eta} = \sum_{m=1}^{k^*} E_{1m}\theta_{11m}\left[\beta_m\cosh\beta_m(1-\xi) + C_{11m}\gamma_m\cos\gamma_m(1-\xi)\right]\sin(m\pi\xi)$$

$$+\sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}\left[\beta_m\cosh\beta_m(1-\xi) + C_{12m}\gamma_m\cosh\gamma_m(1-\xi)\right]\sin(m\pi\xi)$$

After the expansion is performed, the elements of the submatrix can be found from

$$AS(i,j) = A1N\cos\alpha + \frac{1}{\phi}A2N\sin\alpha\,, \tag{2-38}$$

where $A1N = \theta_{11m}(m\pi)\left[ex1 + ex2 + C_{11m}(ex3 + ex4)\right]$

$A2N = \theta_{11m}\left[\beta_m(y1 + y2) + C_{11m}\gamma_m(y3 + y4)\right]$ for $\lambda^2 - (m\pi)^2 > 0$,

or $A1N = \theta_{12m}(m\pi)\left[ex1 + ex2 + C_{12m}(ex5 + ex6)\right]$

$A2N = \theta_{12m}\left[\beta_m(y1 + y2) + C_{12m}\gamma_m(y5 + y6)\right]$ for $\lambda^2 - (m\pi)^2 < 0$.

$$ex1 = \frac{(m\pi + n\pi)\sinh\beta_m}{\beta_m^2 + (m\pi + n\pi)^2} \qquad y1 = \frac{\beta_m\sinh\beta_m}{\beta_m^2 + (m\pi - n\pi)^2}$$

$$ex2 = \frac{-(m\pi - n\pi)\sinh\beta_m}{\beta_m^2 + (m\pi - n\pi)^2} \qquad y2 = \frac{-\beta_m\sinh\beta_m}{\beta_m^2 + (m\pi + n\pi)^2}$$

$$ex3 = \frac{(m\pi + n\pi)\sin\gamma_m}{(m\pi + n\pi)^2 - \gamma_m^2} \qquad y3 = \frac{\gamma_m\sin\gamma_m}{\gamma_m^2 - (m\pi - n\pi)^2}$$

$$ex4 = \frac{(m\pi - n\pi)\sin\gamma_m}{\gamma_m^2 - (m\pi - n\pi)^2} \qquad y4 = \frac{\gamma_m\sin\gamma_m}{(m\pi + n\pi)^2 - \gamma_m^2}$$

$$ex5 = \frac{(m\pi + n\pi)\sinh\gamma_m}{\gamma_m^2 + (m\pi + n\pi)^2} \qquad y5 = \frac{\gamma_m\sinh\gamma_m}{\gamma_m^2 + (m\pi - n\pi)^2}$$

$$ex6 = \frac{-(m\pi - n\pi)\sinh\gamma_m}{\gamma_m^2 + (m\pi - n\pi)^2} \qquad y6 = \frac{-\gamma_m\sinh\gamma_m}{\gamma_m^2 + (m\pi + n\pi)^2}$$

8.1.3 Moment

a. Moment along the η = 1, ξ = 1, η = 0 and ξ = 0

There is no contribution by Building Block 1 to the bending moment along any of the outside edges: $AS(i,j) = 0.$ (2-39)

b. Moment along η = 1 - ξ

The moment expression that has to be expanded for Building Block 1 is

$$\frac{M_n \phi h}{D} = -\left[\theta_1 \phi^2 \frac{\partial^2 W_1'(\xi,\eta)}{\partial \xi^2} + \theta_2 \frac{\partial^2 W_1(\xi,\eta)}{\partial \eta^2} + \theta_3 \phi \frac{\partial^2 W_1'(\xi,\eta)}{\partial \xi \partial \eta}\right]$$

$$\left.\frac{\partial^2 W_1'(\xi,\eta)}{\partial \xi^2}\right|_{\eta=1-\xi} = -\sum_{m=1}^{k^*} E_{1m}\theta_{11m}(m\pi)^2\left[\sinh\beta_m(1-\xi) + C_{11m}\sin\gamma_m(1-\xi)\right]\sin(m\pi\xi)$$
$$-\sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}(m\pi)^2\left[\sinh\beta_m(1-\xi) + C_{12m}\sinh\gamma_m(1-\xi)\right]\sin(m\pi\xi)$$

$$\left.\frac{\partial^2 W_1(\xi,\eta)}{\partial \eta^2}\right|_{\eta=1-\xi} = \sum_{m=1}^{k^*} E_{1m}\theta_{11m}\left[\beta_m^2\sinh\beta_m(1-\xi) - C_{11m}\gamma_m^2\sin\gamma_m(1-\xi)\right]\sin(m\pi\xi)$$
$$+\sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}\left[\beta_m^2\sinh\beta_m(1-\xi) + C_{12m}\gamma_m^2\sinh\gamma_m(1-\xi)\right]\sin(m\pi\xi)$$

$$\left.\frac{\partial^2 W_1'(\xi,\eta)}{\partial \xi \partial \eta}\right|_{\eta=1-\xi} = \sum_{m=1}^{k^*} E_{1m}\theta_{11m}(m\pi)\left[\beta_m\cosh\beta_m(1-\xi) + C_{11m}\gamma_m\cos\gamma_m(1-\xi)\right]\cos(m\pi\xi)$$
$$+\sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}(m\pi)\left[\beta_m\cosh\beta_m(1-\xi) + C_{12m}\gamma_m\cosh\gamma_m(1-\xi)\right]\cos(m\pi\xi)$$

Upon expansion, the elements of the submatrix can be calculated from

$$AS(i,j) = -\left[\theta_1\phi^2\, A1N + \theta_2\, A2N + \theta_3\phi\, A3N\right], \tag{2-40}$$

where

$$\theta_1 = \cos^2\alpha + \nu\sin^2\alpha$$
$$\theta_2 = \sin^2\alpha + \nu\cos^2\alpha$$
$$\theta_3 = (1-\nu)\sin 2\alpha$$

and $A1N = -\theta_{11m}(m\pi)^2\left[ex1 + ex2 + C_{11m}(ex3 + ex4)\right]$

$$A2N = \theta_{11m}\left[\beta_m^2(ex1 + ex2) - C_{11m}\gamma_m^2(ex3 + ex4)\right]$$

$$A3N = \theta_{11m}(m\pi)\left[\beta_m(y1 + y2) + C_{11m}\gamma_m(y3 + y4)\right] \qquad \text{for } \lambda^2 - (m\pi)^2 > 0,$$

or
$$A1N = -\theta_{12m}(m\pi)^2\left[ex1 + ex2 + C_{12m}(ex5 + ex6)\right]$$

$$A2N = \theta_{12m}\left[\beta_m^2(ex1 + ex2) + C_{12m}\gamma_m^2(ex5 + ex6)\right]$$

$$A3N = \theta_{12m}(m\pi)\left[\beta_m(y1 + y2) + C_{12m}\gamma_m(y5 + y6)\right] \qquad \text{for } \lambda^2 - (m\pi)^2 < 0.$$

The expressions for $ex1$ to $ex6$ are identical to those for the displacement of Building Block 1 along the diagonal $\eta = 1 - \xi$, found in section 8.1.1.c. The remaining expressions are:

$$y1 = \frac{(m\pi + n\pi)\left[\cosh\beta_m - \cos(m\pi + n\pi)\right]}{\beta_m^2 + (m\pi + n\pi)^2}$$

$$y2 = \frac{(m\pi - n\pi)\left[\cos(m\pi - n\pi) - \cosh\beta_m\right]}{\beta_m^2 + (m\pi - n\pi)^2}$$

$$y3 = \frac{(m\pi + n\pi)\left[\cos\gamma_m - \cos(m\pi + n\pi)\right]}{(m\pi + n\pi)^2 - \gamma_m^2}$$

$$y4 = \frac{(m\pi - n\pi)\left[\cos\gamma_m - \cos(m\pi - n\pi)\right]}{\gamma_m^2 - (m\pi - n\pi)^2}$$

$$y5 = \frac{(m\pi + n\pi)\left[\cosh\gamma_m - \cos(m\pi + n\pi)\right]}{\gamma_m^2 + (m\pi + n\pi)^2}$$

$$y6 = \frac{(m\pi - n\pi)\left[\cos(m\pi - n\pi) - \cosh\gamma_m\right]}{\gamma_m^2 + (m\pi - n\pi)^2}$$

8.1.4 Vertical Edge Reaction

a. Vertical edge reaction along the $\eta = 1$ edge

$$\frac{V_\eta \phi b^2}{D} = -\left[\frac{\partial^3 W_1(\xi,\eta)}{\partial\eta^3} + (2-\nu)\phi^2\frac{\partial^3 W_1(\xi,\eta)}{\partial\xi^2\,\partial\eta}\right]$$

$$\left.\frac{\partial^3 W_1(\xi,\eta)}{\partial\eta^3}\right|_{\eta=1} = \sum_{m=1}^{k^*} E_{1m}\theta_{11m}\left[\beta_m^3\cosh\beta_m - C_{11m}\gamma_m^3\cos\gamma_m\right]\sin(m\pi\xi)$$

$$+ \sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}\left[\beta_m^3\cosh\beta_m + C_{12m}\gamma_m^3\cosh\gamma_m\right]\sin(m\pi\xi)$$

$$\left.\frac{\partial^3 W_1(\xi,\eta)}{\partial \xi^2\, \partial \eta}\right|_{\eta=1} = -\sum_{m=1}^{k^*} E_{1m}\theta_{11m}(m\pi)^2\left[\beta_m \cosh\beta_m + C_{11m}\gamma_m \cos\gamma_m\right]\sin(m\pi\xi)$$

$$-\sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}(m\pi)^2\left[\beta_m \cosh\beta_m + C_{12m}\gamma_m \cosh\gamma_m\right]\sin(m\pi\xi)$$

Only the elements on the diagonal of the submatrix are non-zero:

$$AS(i,j) \quad 0 \text{ for } i \neq j. \tag{2-41a}$$

$$AS(j,j) = -\left[A1N + (2-\nu)\phi^2 A2N\right], \tag{2-41b}$$

where $A1N = \theta_{11m}\left[\beta_m^3 \cosh\beta_m - C_{11m}\gamma_m^3 \cos\gamma_m\right]$

$A2N = -\theta_{11m}(m\pi)^2\left[\beta_m \cosh\beta_m + C_{11m}\gamma_m \cos\gamma_m\right]$ for $\lambda^2 - (m\pi)^2 > 0$,

and $A1N = \theta_{12m}\left[\beta_m^3 \cosh\beta_m + C_{12m}\gamma_m^3 \cosh\gamma_m\right]$

$A2N = -\theta_{12m}(m\pi)^2\left[\beta_m \cosh\beta_m + C_{12m}\gamma_m \cos h\gamma_m\right]$ for $\lambda^2 - (m\pi)^2 < 0$.

b. Vertical Edge Reaction along the ξ = 1 edge

$$\frac{V_\eta a^2}{D} = -\left[\frac{\partial^3 W_1(\xi,\eta)}{\partial \xi^3} + \frac{(2-\nu)}{\phi^2}\frac{\partial^3 W_1(\xi,\eta)}{\partial \eta^2\, \partial \xi}\right]$$

$$\left.\frac{\partial^3 W_1(\xi,\eta)}{\partial \xi^3}\right|_{\xi=1} = -\sum_{m=1}^{k^*} E_{1m}\theta_{11m}(m\pi)^3\left[\sinh\beta_m\eta + C_{11m}\sin\gamma_m\eta\right]\cos(m\pi)$$

$$-\sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}(m\pi)^3\left[\sinh\beta_m\eta + C_{12m}\sinh\gamma_m\eta\right]\cos(m\pi)$$

$$\left.\frac{\partial^3 W_1(\xi,\eta)}{\partial \xi\, \partial \eta^2}\right|_{\xi=1} = \sum_{m=1}^{k^*} E_{1m}\theta_{11m}(m\pi)\left[\beta_m^2 \sinh\beta_m\eta - C_{11m}\gamma_m^2 \sin\gamma_m\eta\right]\cos(m\pi)$$

$$+\sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}(m\pi)\left[\beta_m^2 \sinh\beta_m\eta + C_{12m}\gamma_m^2 \sinh\gamma_m\eta\right]\cos(m\pi)$$

Here the submatrix elements are defined by

$$AS(j,j) = -\left[A1N + \frac{(2-\nu)}{\phi^2}A2N\right], \tag{2-42}$$

where $A1N = -\theta_{11m}(m\pi)^3\cos(m\pi)\left[ex1 + C_{11m}ex2\right]$

$A2N = \theta_{11m}(m\pi)\cos(m\pi)\left[\beta_m^2 ex1 - C_{11m}\gamma_m^2 ex2\right]$ for $\lambda^2 - (m\pi)^2 > 0$,

and $A1N = -\theta_{12m}(m\pi)^3\cos(m\pi)\left[ex1 + C_{12m}ex3\right]$

$$A2N = \theta_{12m}(m\pi)\cos(m\pi)\left[\beta_m^2 ex1 + C_{12m}\gamma_m^2 ex3\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0.$$

The expressions $ex1$, $ex2$ and $ex3$ are identical to those for the slope contribution of Building Block 1 along $\xi = 1$, found in section 8.1.2.b.

c. Vertical edge reaction along the η = 0 edge

$$\frac{V_\eta \phi h^2}{D} = -\left[\frac{\partial^3 W_1(\xi,\eta)}{\partial\eta^3} + (2-\nu)\phi^2 \frac{\partial^3 W_1(\xi,\eta)}{\partial\xi^2\,\partial\eta}\right]$$

$$\left.\frac{\partial^3 W_1(\xi,\eta)}{\partial\eta^3}\right|_{\eta=0} = \sum_{m=1}^{k^*} E_{1m}\theta_{11m}\left[\beta_m^3 - C_{11m}\gamma_m^3\right]\sin(m\pi\xi)$$

$$+\sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}\left[\beta_m^3 + C_{12m}\gamma_m^3\right]\sin(m\pi\xi)$$

$$\left.\frac{\partial^3 W_1(\xi,\eta)}{\partial\xi^2\,\partial\eta}\right|_{\eta=0} = -\sum_{m=1}^{k^*} E_{1m}\theta_{11m}(m\pi)^2\left[\beta_m + C_{11m}\gamma_m\right]\sin(m\pi\xi)$$

$$-\sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}(m\pi)^2\left[\beta_m + C_{12m}\gamma_m\right]\sin(m\pi\xi)$$

Elements not on the diagonal of the submatrix are zero: $AS(i,j) \quad 0.$ (2-43a)

On the diagonal of AS, the elements are determined from

$$AS(j,j) = -\left[A1N + (2-\nu)\phi^2 A2N\right], \tag{2-43b}$$

where $A1N = \theta_{11m}\left[\beta_m^3 - C_{11m}\gamma_m^3\right]$

$$A2N = -\theta_{11m}(m\pi)^2\left[\beta_m + C_{11m}\gamma_m\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0,$$

and $A1N = \theta_{12m}\left[\beta_m^3 + C_{12m}\gamma_m^3\right]$

$$A2N = -\theta_{12m}(m\pi)^2\left[\beta_m + C_{12m}\gamma_m\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0.$$

d. Vertical Edge Reaction along the ξ = 0 edge

$$\frac{V_\eta a^2}{D} = -\left[\frac{\partial^3 W_1(\xi,\eta)}{\partial\xi^3} + \frac{(2-\nu)}{\phi^2}\frac{\partial^3 W_1(\xi,\eta)}{\partial\eta^2\,\partial\xi}\right]$$

$$\left.\frac{\partial^3 W_1(\xi,\eta)}{\partial\xi^3}\right|_{\xi=0} = -\sum_{m=1}^{k^*} E_{1m}\theta_{11m}(m\pi)^3\left[\sinh\beta_m\eta + C_{11m}\sin\gamma_m\eta\right]$$

$$-\sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}(m\pi)^3\left[\sinh\beta_m\eta + C_{12m}\sinh\gamma_m\eta\right]$$

$$\left.\frac{\partial^3 W_1(\xi,\eta)}{\partial\xi\,\partial\eta^2}\right|_{\xi=0} = \sum_{m=1}^{k^*} E_{1m}\theta_{11m}(m\pi)\left[\beta_m^2\sinh\beta_m\eta - C_{11m}\gamma_m^2\sin\gamma_m\eta\right]$$

$$+\sum_{m=k^*+1}^{\infty} E_{1m}\theta_{12m}(m\pi)\left[\beta_m^2\sinh\beta_m\eta + C_{12m}\gamma_m^2\sinh\gamma_m\eta\right]$$

Here the submatrix elements are defined by

$$AS(j,j) = -\left[A1N + \frac{(2-\nu)}{\phi^2}A2N\right], \qquad (2\text{-}44)$$

where $A1N = -\theta_{11m}(m\pi)^3\left[ex1 + C_{11m}ex2\right]$

$A2N = \theta_{11m}(m\pi)\left[\beta_m^2 ex1 - C_{11m}\gamma_m^2 ex2\right]$ for $\lambda^2 - (m\pi)^2 > 0$,

and $A1N = -\theta_{12m}(m\pi)^3\left[ex1 + C_{12m}ex3\right]$

$A2N = \theta_{12m}(m\pi)\left[\beta_m^2 ex1 + C_{12m}\gamma_m^2 ex3\right]$ for $\lambda^2 - (m\pi)^2 < 0$.

The terms *ex1*, *ex2* and *ex3* are identical to those for the displacement contribution of Building Block 1 along this line, found in section 8.1.1.c.

8.2 Building Block 2

The expansions for Building Block 2 are identical to those for Building Block 1, except where noted in the next subsections. Of course the coefficients for Building Block 2, θ_{21m}, C_{21m}, θ_{22m} and C_{22m}, replace the corresponding coefficients for Building Block 1: θ_{11m}, C_{11m}, θ_{12m} and C_{12m}.

8.2.1 Displacement along the $\eta = 1$ edge

There is no displacement along this edge: $AS(i,j) = 0$. (2-45)

8.2.2 Moment along the $\eta = 1$ edge

Building Block 1 has a forced bending moment along this edge:

$$\left.\frac{Mb^2}{aD}\right|_{\eta=1} = \sum_{m=1}^{\infty} E_{2m}\sin(m\pi\xi) \qquad (2\text{-}11)$$

which is already a sine series. No expansion is required:

$$AS(j,j) = 1.0, \qquad (2\text{-}46a)$$

and where $i \neq j$, $AS(i,j) = 0.$ (2-46b)

8.3 Building Block 3

8.3.1 Displacement

a. Displacement along the η = 1, η = 0 and ξ = 0 edges

There is no contribution by Building Block 3 to the displacement along these edges.

$$AS(i,j) = 0. \tag{2-47}$$

b. Displacement along the ξ = 1 edge

Building Block 3 features a forced displacement along this edge,

$$W_3(\xi,\eta)\big|_{\xi=1} = \sum_{m=1}^{\infty} E_{3m} \sin(m\pi\eta) \tag{2-13}$$

which does not require expansion, as it is already the appropriate sine series.

$$AS(j,j) = 1.0, \tag{2-48a}$$

and where $i \neq j$,

$$AS(i,j) = 0. \tag{2-48b}$$

c. Displacement along the η = 1 - ξ diagonal

The unexpanded solution is

$$W_3(\xi,\eta) = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}\left[\sinh(\beta_{3m}\eta) + C_{31m}\sin(\gamma_{3m}\eta)\right]\sin(m\pi(1-\xi))$$
$$+ \sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}\left[\sinh(\beta_{3m}\eta) + C_{32m}\sinh(\gamma_{3m}\eta)\right]\sin(m\pi(1-\xi))$$

The elements of the submatrix then become

$$AS(i,j) = \theta_{31m}\left[ex1 + ex2 + C_{31m}\,(ex3 + ex4)\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0, \tag{2-49a}$$

or

$$AS(i,j) = \theta_{32m}\left[ex1 + ex2 + C_{32m}\,(ex5 + ex6)\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0. \tag{2-49b}$$

where

$$ex1 = \beta_{3m}\frac{\cosh\beta_{3m}\cos(n\pi) - \cos(m\pi)}{\beta_{3m}^2 + (m\pi + n\pi)^2}$$

$$ex2 = \beta_{3m}\frac{\cos(m\pi) - \cosh\beta_{3m}\cos(n\pi)}{\beta_{3m}^2 + (m\pi - n\pi)^2}$$

$$ex3 = \gamma_{3m}\frac{\cos(m\pi) - \cos(\gamma_{3m} + n\pi)}{\gamma_{3m}^2 - (m\pi + n\pi)^2}$$

8.3.2 Slope

a. Slope along the η = 1 edge

$$\left.\frac{\partial W_3(\xi,\eta)}{\partial \eta}\right|_{\eta=1} = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}(m\pi)\left[\sinh\beta_{3m}\xi + C_{31m}\sin\gamma_{3m}\xi\right]\cos(m\pi)$$

$$+\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}(m\pi)\left[\sinh\beta_{3m}\xi + C_{32m}\sinh\gamma_{3m}\xi\right]\cos(m\pi)$$

In expanded form, this becomes

$$AS(i,j) = \theta_{31m}(m\pi)\cos(m\pi)\left[ex1 + C_{31m}ex2\right] \quad \text{for } \lambda^2-(m\pi)^2>0, \qquad (2\text{-}50a)$$

and $$AS(i,j) = \theta_{32m}(m\pi)\cos(m\pi)\left[ex1 + C_{31m}ex3\right] \quad \text{for } \lambda^2-(m\pi)^2<0, \qquad (2\text{-}50b)$$

where

$$ex1 = \frac{-2(n\pi)\cos(n\pi)\sinh\beta_{3m}}{\beta_{3m}^2+(n\pi)^2} \qquad ex4 = \gamma_{3m}\frac{\cos(\gamma_{3m}+n\pi)-\cos(m\pi)}{\gamma_{3m}^2-(m\pi-n\pi)^2}$$

$$ex2 = \frac{2(n\pi)\cos(n\pi)\sin\gamma_{3m}}{\gamma_{3m}^2-(n\pi)^2} \qquad ex5 = \gamma_{3m}\frac{\cosh\gamma_{3m}\cos(n\pi)-\cos(m\pi)}{\gamma_{3m}^2+(m\pi+n\pi)^2}$$

$$ex3 = \frac{-2(n\pi)\cos(n\pi)\sinh\gamma_{3m}}{\gamma_{3m}^2+(n\pi)^2} \qquad ex6 = \gamma_{3m}\frac{\cos(m\pi)-\cosh\gamma_{3m}\cos(n\pi)}{\gamma_{3m}^2+(m\pi-n\pi)^2}$$

b. Slope along the ξ = 1 edge

$$\left.\frac{\partial W_3(\xi,\eta)}{\partial \xi}\right|_{\xi=1} = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}\left[\beta_{3m}\cosh\beta_{3m} + C_{31m}\gamma_{3m}\cos\gamma_{3m}\right]\sin(m\pi\eta)$$

$$+\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}\left[\beta_{3m}\cosh\beta_{3m} + C_{32m}\gamma_{3m}\cosh\gamma_{3m}\right]\sin(m\pi\eta)$$

All elements that are not on the diagonal of the submatrix are zero:

$$AS(i,j) = 0.0 \quad \text{for } i \neq j. \qquad (2\text{-}51a)$$

On the diagonal,

$$AS(i,j) = \theta_{31m}\left[\beta_{3m}\cosh\beta_{3m} + C_{31m}\gamma_{3m}\cos\gamma_{3m}\right] \quad \text{for } \lambda^2-(m\pi)^2>0, \qquad (2\text{-}51b)$$

and $$AS(i,j) = \theta_{32m}\left[\beta_{3m}\cosh\beta_{3m} + C_{32m}\gamma_{3m}\cosh\gamma_{3m}\right] \quad \text{for } \lambda^2-(m\pi)^2<0. \qquad (2\text{-}51c)$$

c. Slope along the η = 0 edge

$$\left.\frac{\partial W_3(\xi,\eta)}{\partial\eta}\right|_{\eta=0} = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}(m\pi)\left[\sinh\beta_{3m}\xi + C_{31m}\sin\gamma_{3m}\xi\right]$$

$$+\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}(m\pi)\left[\sinh\beta_{3m}\xi + C_{32m}\sinh\gamma_{3m}\xi\right]$$

In expanded for, this then becomes

$$AS(i,j) = \theta_{31m}(m\pi)\left[ex1 + C_{31m}ex2\right] \qquad \text{for } \lambda^2 - (m\pi)^2 > 0, \tag{2-52a}$$

and $$AS(i,j) = \theta_{32m}(m\pi)\left[ex1 + C_{32m}ex3\right] \qquad \text{for } \lambda^2 - (m\pi)^2 < 0, \tag{2-52b}$$

where *ex1, ex2* and *ex3* are identical to those of section 8.3.2.a.:

$$ex1 = \frac{-2(n\pi)\cos(n\pi)\sinh\beta_{3m}}{\beta_{3m}^2 + (n\pi)^2}$$

$$ex2 = \frac{2(n\pi)\cos(n\pi)\sin\gamma_m}{\gamma_{3m}^2 - (n\pi)^2}$$

$$ex3 = \frac{-2(n\pi)\cos(n\pi)\sinh\gamma_{3m}}{\gamma_{3m}^2 + (n\pi)^2}$$

d. Slope along the ξ = 0 edge

$$\left.\frac{\partial W_3(\xi,\eta)}{\partial\eta}\right|_{\eta=0} = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}\left[\beta_{3m} + C_{31m}\gamma_{3m}\right]\sin(m\pi\xi)$$

$$+\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}\left[\beta_{3m} + C_{32m}\gamma_{3m}\right]\sin(m\pi\xi)$$

Elements off the diagonal of the submatrix are zero:

$$AS(i,j) = 0 \quad \text{for } i \neq j. \tag{2-53a}$$

The elements on the diagonal are

$$AS(i,j) = \theta_{31m}(\beta_{3m} + C_{31m}\gamma_{3m}) \qquad \text{for } \lambda^2 - (m\pi)^2 > 0, \tag{2-53b}$$

and $$AS(i,j) = \theta_{32m}(\beta_{3m} + C_{32m}\gamma_{3m}) \qquad \text{for } \lambda^2 - (m\pi)^2 < 0. \tag{2-53c}$$

e. Slope along the η = 1 - ξ diagonal

The slope along any oblique line is

$$\frac{\partial W_3(\xi,\eta)}{\partial n} = \frac{\partial W_3(\xi,\eta)}{\partial\xi}\cos\alpha + \frac{1}{\phi}\frac{\partial W_3(\xi,\eta)}{\partial\xi}\sin\alpha$$

where the derivatives for this building block are

$$\left.\frac{\partial W_3(\xi,\eta)}{\partial \xi}\right|_{\eta=1-\xi} = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}\left[\beta_{3m}\cosh\beta_{3m}\xi + C_{31m}\gamma_{3m}\cos\gamma_{3m}\xi\right]\sin(m\pi(1-\xi))$$
$$+\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}\left[\beta_{3m}\cosh\beta_{3m}\xi + C_{32m}\gamma_{3m}\cosh\gamma_{3m}\xi\right]\sin(m\pi(1-\xi))$$

$$\left.\frac{\partial W_3(\xi,\eta)}{\partial \eta}\right|_{\xi=1-\eta} = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}(m\pi)\left[\sinh\beta_{3m}\xi + C_{31m}\sin\gamma_{3m}\xi\right]\cos(m\pi(1-\xi))$$
$$+\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}(m\pi)\left[\sinh\beta_{3m}\xi + C_{32m}\sinh\gamma_{3m}\xi\right]\cos(m\pi(1-\xi))$$

After the expansion is performed, the elements of the submatrix can be found from the following:

$$AS(i,j) = A1N\cos\alpha + \frac{1}{\phi}A2N\sin\alpha\,, \tag{2-54}$$

where $A1N = \theta_{31m}(m\pi)\left[\beta_{3m}(ex1+ex2) + C_{31m}\gamma_{3m}(ex3+ex4)\right]$

$A2N = \theta_{31m}\left[y1+y2+C_{31m}(y3+y4)\right]$ for $\lambda^2-(m\pi)^2>0$,

or $A1N = \theta_{32m}(m\pi)\left[\beta_{3m}(ex1+ex2) + C_{32m}\gamma_{3m}(ex5+ex6)\right]$

$A2N = \theta_{32m}\left[y1+y2+C_{32m}(y5+y6)\right]$ for $\lambda^2-(m\pi)^2<0$.

$$ex1 = \frac{\beta_{3m}\cos(n\pi)\sinh\beta_{3m}}{\beta_{3m}^2+(m\pi+n\pi)^2} \qquad ex4 = \frac{\gamma_{3m}\sin(\gamma_{3m}+n\pi)}{(m\pi-n\pi)^2-\gamma_{3m}^2}$$

$$ex2 = -\frac{\beta_{3m}\cos(n\pi)\sinh\beta_{3m}}{\beta_{3m}^2+(m\pi-n\pi)^2} \qquad ex5 = \frac{\gamma_{3m}\cos(n\pi)\sinh\gamma_{3m}}{\gamma_{3m}^2+(m\pi+n\pi)^2}$$

$$ex3 = \frac{\gamma_{3m}\sin(\gamma_{3m}+n\pi)}{\gamma_{3m}^2-(m\pi+n\pi)^2} \qquad ex6 = -\frac{\gamma_{3m}\cos(n\pi)\sinh\gamma_{3m}}{\gamma_{3m}^2+(m\pi-n\pi)^2}$$

$$y1 = -\frac{(m\pi+n\pi)\cos(n\pi)\sinh\beta_{3m}}{\beta_{3m}^2+(m\pi+n\pi)^2} \qquad y4 = \frac{(m\pi-n\pi)\sin(\gamma_{3m}+n\pi)}{(m\pi-n\pi)^2-\gamma_{3m}^2}$$

$$y2 = \frac{(m\pi-n\pi)\cos(n\pi)\sinh\beta_{3m}}{\beta_{3m}^2+(m\pi-n\pi)^2} \qquad y5 = -\frac{(m\pi+n\pi)\cos(n\pi)\sinh\gamma_{3m}}{\gamma_{3m}^2+(m\pi+n\pi)^2}$$

$$y3 = \frac{(m\pi+n\pi)\sin(\gamma_{3m}+n\pi)}{\gamma_{3m}^2-(m\pi+n\pi)^2} \qquad y6 = \frac{(m\pi-n\pi)\cos(n\pi)\sinh\gamma_{3m}}{\gamma_{3m}^2+(m\pi-n\pi)^2}$$

8.3.3 Moment

a. Moment along the η = 1, ξ = 1, η = 0 and ξ = 0 edges

There is no contribution by Building Block 3 to the bending moment along the outside edges of the building block: $AS(i,j)$ 0. (2-55)

b. Moment along η = 1 - ξ

$$\frac{M_n\phi b}{D} = -\left[\theta_1\phi^2\frac{\partial^2 W_3(\xi,\eta)}{\partial\xi^2} + \theta_2\frac{\partial^2 W_3(\xi,\eta)}{\partial\eta^2} + \theta_3\phi\frac{\partial^2 W_3(\xi,\eta)}{\partial\xi\partial\eta}\right]$$

$$\left.\frac{\partial^2 W_3(\xi,\eta)}{\partial\xi^2}\right|_{\eta=1-\xi} = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}\left[\beta_{3m}^2\sinh\beta_{3m}\xi - C_{31m}\gamma_{3m}^2\sin\gamma_{3m}\xi\right]\sin(m\pi(1-\xi))$$

$$+\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}\left[\beta_{3m}^2\sinh\beta_{3m}\xi + C_{32m}\gamma_{3m}^2\sinh\gamma_{3m}\xi\right]\sin(m\pi(1-\xi))$$

$$\left.\frac{\partial^2 W_3(\xi,\eta)}{\partial\eta^2}\right|_{\eta=1-\xi} = -\sum_{m=1}^{k^*} E_{3m}\theta_{31m}(m\pi)^2\left[\sinh\beta_{3m}\xi + C_{31m}\sin\gamma_{3m}\xi\right]\sin(m\pi(1-\xi))$$

$$-\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}(m\pi)^2\left[\sinh\beta_{3m}\xi + C_{32m}\sinh\gamma_{3m}\xi\right]\sin(m\pi(1-\xi))$$

$$\left.\frac{\partial^2 W_3(\xi,\eta)}{\partial\xi\partial\eta}\right|_{\eta=1-\xi} = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}(m\pi)\left[\beta_{3m}\cosh\beta_{3m}\xi + C_{31m}\gamma_{3m}\cos\gamma_{3m}\xi\right]\cos(m\pi(1-\xi))$$

$$+\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}(m\pi)\left[\beta_{3m}\cosh\beta_{3m}\xi + C_{32m}\gamma_{3m}\cosh\gamma_{3m}\xi\right]\cos(m\pi(1-\xi))$$

Upon expansion, the elements of the submatrix can be calculated from

$$AS(i,j) = -\left[\theta_1\phi^2\,A1N + \theta_2\,A2N + \theta_3\phi\,A3N\right], \tag{2-56}$$

where $\theta_1 = \cos^2\alpha + \nu\sin^2\alpha$

$\theta_2 = \sin^2\alpha + \nu\cos^2\alpha$

$\theta_3 = (1-\nu)\sin 2\alpha$

and $A1N = \theta_{31m}\left[\beta_{3m}{}^2(ex1+ex2) - C_{31m}\gamma_{3m}{}^2(ex3+ex4)\right]$

$A2N = -\theta_{31m}(m\pi)^2\left[ex1 + ex2 + C_{31m}(ex3+ex4)\right]$

$A3N = \theta_{31m}(m\pi)\left[\beta_{3m}(y1+y2) + C_{31m}\gamma_{3m}(y3+y4)\right]$ for $\lambda^2-(m\pi)^2>0$, or $$A1N = \theta_{32m}\left[\beta_{3m}^2(ex1 + ex2) + C_{32m}\gamma_{3m}^2(ex5 + ex6)\right]$$

$$A2N = -\theta_{32m}(m\pi)^2\left[ex1 + ex2 + C_{32m}(ex5 + ex6)\right]$$

$$A3N = \theta_{32m}(m\pi)\left[\beta_{3m}(y1 + y2) + C_{32m}\gamma_{3m}(y5 + y6)\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0 .$$

The expressions for $ex1$ to $ex6$ are identical to those for the displacement of Building Block 3 along $\eta = 1 - \xi$, found in section 8.3.1.c. The remaining expressions are:

$$y1 = \frac{(m\pi - n\pi)\left[\cos(n\pi)\cosh\beta_{3m} - \cos(m\pi)\right]}{\beta_{3m}^2 + (m\pi - n\pi)^2}$$

$$y2 = \frac{(m\pi + n\pi)\left[\cos(m\pi) - \cos(n\pi)\cosh\beta_{3m}\right]}{\beta_{3m}^2 + (m\pi + n\pi)^2}$$

$$y3 = \frac{(m\pi - n\pi)\left[\cos(n\pi - \gamma_{3m}) - \cos(m\pi)\right]}{(m\pi - n\pi)^2 - \gamma_{3m}^2}$$

$$y4 = \frac{(m\pi + n\pi)\left[\cos(m\pi) - \cos(n\pi - \gamma_{3m})\right]}{(m\pi + n\pi)^2 - \gamma_{3m}^2}$$

$$y5 = \frac{(m\pi - n\pi)\left[\cos(n\pi)\cosh\gamma_{3m} - \cos(m\pi)\right]}{\gamma_{3m}^2 + (m\pi - n\pi)^2}$$

$$y6 = \frac{(m\pi + n\pi)\left[\cos(m\pi) - \cos(n\pi)\cosh\gamma_{3m}\right]}{\gamma_{3m}^2 + (m\pi + n\pi)^2}$$

8.3.4 Vertical Edge Reaction

a. Vertical edge reaction along the $\eta = 1$ edge

$$\frac{V_\eta \phi b^2}{D} = -\left[\frac{\partial^3 W_3(\xi,\eta)}{\partial \eta^3} + (2 - \nu)\phi^2 \frac{\partial^3 W_3(\xi,\eta)}{\partial \xi^2 \partial \eta}\right]$$

$$\left.\frac{\partial^3 W_3(\xi,\eta)}{\partial \eta^3}\right|_{\eta=1} = -\sum_{m=1}^{k^*} E_{3m}\theta_{31m}(m\pi)^3\left[\sinh\beta_{3m}\xi + C_{31m}\sin\gamma_{3m}\xi\right]\cos(m\pi)$$
$$- \sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}(m\pi)^3\left[\sinh\beta_{3m}\xi + C_{32m}\sinh\gamma_{3m}\xi\right]\cos(m\pi)$$

$$\left.\frac{\partial^3 W_3(\xi,\eta)}{\partial \xi^2 \partial \eta}\right|_{\eta=1} = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}(m\pi)\left[\beta_{3m}^2\sinh\beta_{3m}\xi - C_{31m}\gamma_{3m}^2\sin\gamma_{3m}\xi\right]\cos(m\pi)$$
$$+ \sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}(m\pi)\left[\beta_{3m}^2\sinh\beta_{3m}\xi + C_{32m}\gamma_{3m}^2\sinh\gamma_{3m}\xi\right]\cos(m\pi)$$

The submatrix elements are defined by

$$AS(j,j) = -\left[A1N + (2-\nu)\phi^2 A2N\right], \tag{2-57}$$

where $A1N = -\theta_{31m}(m\pi)^3\cos(m\pi)\left[ex1 + C_{31m}ex2\right]$

$$A2N = \theta_{31m}(m\pi)\cos(m\pi)\left[\beta_{3m}^2 ex1 - C_{31m}\gamma_{3m}^2 ex2\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0,$$

and $A1N = -\theta_{32m}(m\pi)^3\cos(m\pi)\left[ex1 + C_{32m}ex3\right]$

$$A2N = \theta_{32m}(m\pi)\cos(m\pi)\left[\beta_{3m}^2 ex1 + C_{32m}\gamma_{3m}^2 ex3\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0.$$

The expressions *ex1*, *ex2* and *ex3* are identical to those for the slope contribution of Building Block 3 along $\xi = 1$, found in section 8.3.2.b.

b. Vertical edge reaction along the $\xi = 1$ edge

$$\frac{V_\eta a^2}{D} = -\left[\frac{\partial^3 W_3(\xi,\eta)}{\partial\xi^3} + \frac{(2-\nu)}{\phi^2}\frac{\partial^3 W_3(\xi,\eta)}{\partial\eta^2\,\partial\xi}\right]$$

$$\left.\frac{\partial^3 W_3(\xi,\eta)}{\partial\xi^3}\right|_{\xi=1} = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}\left[\beta_{3m}^3\cosh\beta_{3m} - C_{31m}\gamma_{3m}^3\cos\gamma_{3m}\right]\sin(m\pi\eta)$$

$$+\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}\left[\beta_{3m}^3\cosh\beta_{3m} + C_{32m}\gamma_{3m}^3\cosh\gamma_{3m}\right]\sin(m\pi\eta)$$

$$\left.\frac{\partial^3 W_3(\xi,\eta)}{\partial\xi\,\partial\eta^2}\right|_{\xi=1} = -\sum_{m=1}^{k^*} E_{3m}\theta_{31m}(m\pi)^2\left[\beta_{3m}\cosh\beta_{3m} + C_{31m}\gamma_{3m}\cos\gamma_{3m}\right]\sin(m\pi\eta)$$

$$-\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}(m\pi)^2\left[\beta_{3m}\cosh\beta_{3m} + C_{32m}\gamma_{3m}\cosh\gamma_{3m}\right]\sin(m\pi\eta)$$

Only the elements on the diagonal of the submatrix are non-zero:

$$AS(i,j) = 0 \quad \text{for } i \neq j. \tag{2-58a}$$

On the diagonal,

$$AS(j,j) = -\left[A1N + \frac{(2-\nu)}{\phi^2}A2N\right], \tag{2-58b}$$

where $A1N = \theta_{31m}\left[\beta_{3m}^3\cosh\beta_{3m} - C_{31m}\gamma_{3m}^3\cos\gamma_{3m}\right]$

$$A2N = -\theta_{31m}(m\pi)^2\left[\beta_{3m}\cosh\beta_{3m} + C_{31m}\gamma_{3m}\cos\gamma_{3m}\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0,$$

and $A1N = \theta_{32m}\left[\beta_{3m}^3\cosh\beta_{3m} + C_{32m}\gamma_{3m}^3\cosh\gamma_{3m}\right]$

$$A2N = -\theta_{32m}(m\pi)^2\left[\beta_{3m}\cosh\beta_{3m} + C_{32m}\gamma_{3m}\cosh\gamma_{3m}\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0.$$

c. Vertical edge reaction along the η = 0 edge

$$\frac{V_\eta \phi b^2}{D} = -\left[\frac{\partial^3 W_3(\xi,\eta)}{\partial \eta^3} + (2-\nu)\phi^2 \frac{\partial^3 W_3(\xi,\eta)}{\partial \xi^2 \partial \eta}\right]$$

$$\left.\frac{\partial^3 W_3(\xi,\eta)}{\partial \eta^3}\right|_{\eta=0} = -\sum_{m=1}^{k^*} E_{3m}\theta_{31m}(m\pi)^3\left[\sinh\beta_{3m}\xi + C_{31m}\sin\gamma_{3m}\xi\right]$$

$$-\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}(m\pi)^3\left[\sinh\beta_{3m}\xi + C_{32m}\sinh\gamma_{3m}\xi\right]$$

$$\left.\frac{\partial^3 W_3(\xi,\eta)}{\partial \xi^2 \partial \eta}\right|_{\eta=0} = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}(m\pi)\left[\beta_{3m}^2\sinh\beta_{3m}\xi - C_{31m}\gamma_{3m}^2\sin\gamma_{3m}\xi\right]$$

$$+\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}(m\pi)\left[\beta_{3m}^2\sinh\beta_{3m}\xi + C_{32m}\gamma_{3m}^2\sinh\gamma_{3m}\xi\right]$$

Here the submatrix elements are defined by

$$AS(j,j) = -\left[A1N + (2-\nu)\phi^2 A2N\right], \tag{2-59}$$

where $A1N = -\theta_{31m}(m\pi)^3\left[ex1 + C_{31m}ex2\right]$

$A2N = \theta_{31m}(m\pi)\left[\beta_{3m}^2 ex1 - C_{31m}\gamma_{3m}^2 ex2\right]$ for $\lambda^2 - (m\pi)^2 > 0$,

and $A1N = -\theta_{32m}(m\pi)^3\left[ex1 + C_{32m}ex3\right]$

$A2N = \theta_{32m}(m\pi)\left[\beta_{3m}^2 ex1 + C_{32m}\gamma_{3m}^2 ex3\right]$ for $\lambda^2 - (m\pi)^2 < 0$.

d. Vertical edge reaction along the ξ = 0 edge

$$\frac{V_\eta a^2}{D} = -\left[\frac{\partial^3 W_3(\xi,\eta)}{\partial \xi^3} + \frac{(2-\nu)}{\phi^2}\frac{\partial^3 W_3(\xi,\eta)}{\partial \eta^2 \partial \xi}\right]$$

$$\left.\frac{\partial^3 W_3(\xi,\eta)}{\partial \xi^3}\right|_{\xi=0} = \sum_{m=1}^{k^*} E_{3m}\theta_{31m}\left[\beta_{3m}^3 - C_{31m}\gamma_{3m}^3\right]\sin(m\pi\eta)$$

$$+\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}\left[\beta_{3m}^3 + C_{32m}\gamma_{3m}^3\right]\sin(m\pi\eta)$$

$$\left.\frac{\partial^3 W_3(\xi,\eta)}{\partial \xi \partial \eta^2}\right|_{\xi=0} = -\sum_{m=1}^{k^*} E_{3m}\theta_{31m}(m\pi)^2\left[\beta_{3m} + C_{31m}\gamma_{3m}\right]\sin(m\pi\eta)$$

$$-\sum_{m=k^*+1}^{\infty} E_{3m}\theta_{32m}(m\pi)^2\left[\beta_{3m} + C_{12m}\gamma_{3m}\right]\sin(m\pi\eta)$$

Elements not on the diagonal of the submatrix are zero:

$$AS(i,j) = 0 \text{ for } i \neq j. \tag{2-60a}$$

On the diagonal, the elements are determined by

$$AS(j,j) = -\left[A1N + \frac{(2-\nu)}{\phi^2} A2N\right], \tag{2-60b}$$

where

$$A1N = \theta_{31m}\left[\beta_{3m}^3 - C_{31m}\gamma_{3m}^3\right]$$

$$A2N = -\theta_{31m}(m\pi)^2\left[\beta_{3m} + C_{31m}\gamma_{3m}\right] \qquad \text{for } \lambda^2 - (m\pi)^2 > 0,$$

and

$$A1N = \theta_{32m}\left[\beta_{3m}^3 + C_{32m}\gamma_{3m}^3\right]$$

$$A2N = -\theta_{32m}(m\pi)^2\left[\beta_{3m} + C_{32m}\gamma_{3m}\right] \qquad \text{for } \lambda^2 - (m\pi)^2 < 0.$$

8.4. Building Block 4

The coefficients θ_{41m}, C_{41m}, θ_{42m} and C_{42m} are substitued for θ_{31m}, C_{31m}, θ_{32m} and C_{32m}, respectively, in the contribution terms for Building Block 3 for all cases except those specified in the subsequent sections.

8.4.1 Displacement along the $\xi = 1$ edge

There is no contribution to the displacement by Building Block 4 along this edge.

$$AS(i,j) = 0. \tag{2-61}$$

8.4.2 Moment along the $\xi = 1$ edge

The forced harmonic bending moment

$$\left.\frac{Mb^2}{aD}\right|_{\xi=1} = \sum_{m=1}^{\infty} E_{4m}\sin(m\pi\xi) \tag{2-15}$$

is already a sine series in the proper terms. The submatrix elements on the diagonal are

$$AS(j,j) = 1. \tag{2-62a}$$

For the remaining elements where $i \neq j$,

$$AS(i,j) = 0. \tag{2-62b}$$

8.5 Building Block 5

8.5.1 Displacement

a. Displacement along the η = 1, ξ = 1 and ξ = 0 edges

There is no contribution by Building Block 5 to the displacement along these edges:

$$AS(i,j) \quad 0. \tag{2-63}$$

b. Displacement along the η = 0 edge

Building Block 5 has a forced displacement along this edge:

$$W_5(\xi,\eta)\big|_{\eta=0} = \sum_{m=1}^{\infty} E_{5m}\sin(m\pi(1-\xi)) \tag{2-17}$$

and so

$$AS(j,j) \quad 1.0. \tag{2-64a}$$

For all remaining submatrix elements for which $i \neq j$,

$$AS(i,j) \quad 0. \tag{2-64b}$$

c. Displacement along η = 1 - ξ

The unexpanded solution along this diagonal is

$$\begin{aligned} W_5(\xi,\eta) = {} & \sum_{m=1}^{k^*} E_{5m}\theta_{51m}\left[\sinh(\beta_m(1-\xi)) + C_{51m}\sin(\gamma_m(1-\xi))\right]\sin(m\pi(1-\xi)) \\ & + \sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}\left[\sinh(\beta_m(1-\xi)) + C_{52m}\sinh(\gamma_m(1-\xi))\right]\sin(m\pi(1-\xi)) \end{aligned}$$

After expansion, this solution becomes

$$\begin{aligned} W_5(\xi,\eta) = {} & \sum_{m=1}^{k^*} E_{5m}\theta_{51m}\left[ex1 + ex2 + C_{51m}\,(ex3 + ex4)\right] \\ & + \sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}\left[ex1 + ex2 + C_{52m}\,(ex5 + ex6)\right] \end{aligned}$$

where

$$ex1 = \beta_m \frac{\cosh\beta_m \cos(m\pi - n\pi) - 1}{\beta_m^2 + (m\pi - n\pi)^2}$$

$$ex2 = \beta_m \frac{1 - \cosh\beta_m \cos(m\pi + n\pi)}{\beta_m^2 + (m\pi + n\pi)^2}$$

$$ex3 = \gamma_m \frac{1 - \cos\gamma_m \cos(m\pi - n\pi)}{\gamma_m^2 - (m\pi - n\pi)^2}$$

$$ex4 = \gamma_m \frac{\cos\gamma_m \cos(m\pi + n\pi) - 1}{\gamma_m^2 - (m\pi + n\pi)^2}$$

$$ex5 = \gamma_m \frac{\cosh\gamma_m \cos(m\pi - n\pi) - 1}{\gamma_m^2 + (m\pi - n\pi)^2}$$

$$ex6 = \gamma_m \frac{1 - \cosh\gamma_m \cos(m\pi + n\pi)}{\gamma_m^2 + (m\pi + n\pi)^2}$$

The elements of the submatrix then become

$$AS(i,j) = \theta_{51m}\left[ex1 + ex2 + C_{51m}(ex3 + ex4)\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0, \qquad (2\text{-}65a)$$

or
$$AS(i,j) = \theta_{52m}\left[ex1 + ex2 + C_{52m}(ex5 + ex6)\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0. \qquad (2\text{-}65b)$$

8.5.2 Slope

a. Slope along the $\eta = 1$ edge

$$\left.\frac{\partial W_5(\xi,\eta)}{\partial\eta}\right|_{\eta=1} = -\sum_{m=1}^{k^*} E_{5m}\theta_{51m}\left[\beta_m + C_{51m}\gamma_m\right]\sin(m\pi\xi)$$
$$-\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}\left[\beta_m + C_{52m}\gamma_m\right]\sin(m\pi\xi)$$

All elements that are not on the diagonal of the submatrix are zero:

$$AS(i,j) = 0.0 \text{ for } i \neq j. \qquad (2\text{-}66a)$$

On the diagonal,

$$AS(i,j) = -\theta_{51m}\left[\beta_m + C_{51m}\gamma_m\right] \qquad \text{for } \lambda^2 - (m\pi)^2 > 0, \qquad (2\text{-}66b)$$

and
$$AS(i,j) = \theta_{52m}\left[\beta_m + C_{52m}\gamma_m\right] \qquad \text{for } \lambda^2 - (m\pi)^2 < 0. \qquad (2\text{-}66c)$$

b. Slope along $\xi = 1$

$$\left.\frac{\partial W_5(\xi,\eta)}{\partial\xi}\right|_{\xi=1} = \sum_{m=1}^{k^*} E_{5m}\theta_{51m}(m\pi)\left[\sinh\beta_m(1-\eta) + C_{51m}\sin\gamma_m(1-\eta)\right]\cos(m\pi)$$
$$+\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}(m\pi)\left[\sinh\beta_m(1-\eta) + C_{52m}\sinh\gamma_m(1-\eta)\right]\cos(m\pi)$$

In expanded form, this becomes

$$AS(i,j) = \theta_{51m}(m\pi)\cos(m\pi)\left[ex1 + C_{51m}ex2\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0, \qquad (2\text{-}67a)$$

and $$AS(i,j) = \theta_{52m}(m\pi)\cos(m\pi)\left[ex1 + C_{52m}ex3\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0, \qquad (2\text{-}67b)$$

where

$$ex1 = \frac{2(n\pi)\sinh\beta_m}{\beta_m^2 + (n\pi)^2}$$

$$ex2 = \frac{2(n\pi)\sin\gamma_m}{(n\pi)^2 - \gamma_m^2}$$

$$ex3 = \frac{2(n\pi)\sinh\gamma_m}{\gamma_m^2 + (n\pi)^2}$$

c. Slope along the $\eta = 0$ edge

$$\left.\frac{\partial W_5(\xi,\eta)}{\partial\eta}\right|_{\eta=0} = -\sum_{m=1}^{k^*} E_{5m}\theta_{51m}\left[\beta_m\cosh\beta_m + C_{51m}\gamma_m\cos\gamma_m\right]\sin(m\pi\xi)$$
$$-\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}\left[\beta_m\cosh\beta_m + C_{52m}\gamma_m\cosh\gamma_m\right]\sin(m\pi\xi)$$

Elements off the diagonal of the submatrix are zero:

$$AS(i,j) \quad 0 \quad \text{for } i \neq j. \qquad (2\text{-}68a)$$

The elements on the diagonal are

$$AS(i,j) = -\theta_{51m}\left[\beta_m\cosh\beta_m + C_{51m}\gamma_m\cos\gamma_m\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0, \qquad (2\text{-}68b)$$

and $$AS(i,j) = -\theta_{52m}\left[\beta_m\cosh\beta_m + C_{52m}\gamma_m\cosh\gamma_m\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0. \qquad (2\text{-}68c)$$

d. Slope along the $\xi = 0$ edge

$$\left.\frac{\partial W_5(\xi,\eta)}{\partial\xi}\right|_{\xi=0} = \sum_{m=1}^{k^*} E_{5m}\theta_{51m}(m\pi)\left[\sinh\beta_m(1-\eta) + C_{51m}\sin\gamma_m(1-\eta)\right]$$
$$+\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}(m\pi)\left[\sinh\beta_m(1-\eta) + C_{52m}\sinh\gamma_m(1-\eta)\right]$$

In expanded for, this then becomes

$$AS(i,j) = \theta_{51m}(m\pi)\left[ex1 + C_{51m}ex2\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0, \qquad (2\text{-}69a)$$

and $$AS(i,j) = \theta_{52m}(m\pi)\left[ex1 + C_{52m}ex3\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0, \qquad (2\text{-}69b)$$

where $ex1$, $ex2$ and $ex3$ are identical to those of section 8.5.2.b:

$$ex1 = \frac{2(n\pi)\sinh\beta_m}{(n\pi)^2 + \beta_m^2}$$

$$ex2 = \frac{2(n\pi)\sin\gamma_m}{(n\pi)^2 - \gamma_m^2}$$

$$ex3 = \frac{2(n\pi)\sinh\gamma_m}{(n\pi)^2 + \gamma_m^2}$$

e. Slope along η = 1 - ξ

The slope along any oblique line is

$$\frac{\partial W_5(\xi,\eta)}{\partial n} = \frac{\partial W_5(\xi,\eta)}{\partial \xi}\cos\alpha + \frac{1}{\phi}\frac{\partial W_5(\xi,\eta)}{\partial \xi}\sin\alpha$$

where

$$\left.\frac{\partial W_5(\xi,\eta)}{\partial \xi}\right|_{\eta=1-\xi} = \sum_{m=1}^{k^*} E_{5m}\theta_{51m}(m\pi)\left[\sinh\beta_m\xi + C_{51m}\sin\gamma_m\xi\right]\cos(m\pi\xi)$$
$$+ \sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}(m\pi)\left[\sinh\beta_m\xi + C_{52m}\sinh\gamma_m\xi\right]\cos(m\pi\xi)$$

$$\left.\frac{\partial W_5(\xi,\eta)}{\partial \eta}\right|_{\xi=1-\eta} = -\sum_{m=1}^{k^*} E_{5m}\theta_{51m}\left[\beta_m\cosh\beta_m\xi + C_{51m}\gamma_m\cos\gamma_m\xi\right]\sin(m\pi\xi)$$
$$- \sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}\left[\beta_m\cosh\beta_m\xi + C_{52m}\gamma_m\cosh\gamma_m\xi\right]\sin(m\pi\xi)$$

for this building block and this line. After the expansion is performed, the elements of the submatrix can be found from the following:

$$AS(i,j) = A1N\cos\alpha + \frac{1}{\phi}A2N\sin\alpha\,, \tag{2-70}$$

where $A1N = \theta_{51m}(m\pi)\left[ex1 + ex2 + C_{51m}(ex3 + ex4)\right]$

$A2N = \theta_{51m}\left[\beta_m(y1 + y2) + C_{51m}\gamma_m(y3 + y4)\right]$ for $\lambda^2 - (m\pi)^2 > 0$,

or $A1N = \theta_{52m}(m\pi)\left[ex1 + ex2 + C_{52m}(ex5 + ex6)\right]$

$A2N = \theta_{52m}\left[\beta_m(y1 + y2) + C_{52m}\gamma_m(y5 + y6)\right]$ for $\lambda^2 - (m\pi)^2 < 0$.

$$ex1 = \frac{(m\pi + n\pi)\sinh\beta_m}{\beta_m^2 + (m\pi + n\pi)^2} \qquad y1 = \frac{\beta_m \sinh\beta_m}{\beta_m^2 + (m\pi - n\pi)^2}$$

$$ex2 = \frac{-(m\pi - n\pi)\sinh\beta_m}{\beta_m^2 + (m\pi - n\pi)^2} \qquad y2 = \frac{-\beta_m \sinh\beta_m}{\beta_m^2 + (m\pi + n\pi)^2}$$

$$ex3 = \frac{(m\pi + n\pi)\sin\gamma_m}{(m\pi + n\pi)^2 - \gamma_m^2} \qquad y3 = \frac{\gamma_m \sin\gamma_m}{\gamma_m^2 - (m\pi - n\pi)^2}$$

$$ex4 = \frac{(m\pi - n\pi)\sin\gamma_m}{\gamma_m^2 - (m\pi - n\pi)^2} \qquad y4 = \frac{\gamma_m \sin\gamma_m}{(m\pi + n\pi)^2 - \gamma_m^2}$$

$$ex5 = \frac{(m\pi + n\pi)\sinh\gamma_m}{\gamma_m^2 + (m\pi + n\pi)^2} \qquad y5 = \frac{\gamma_m \sinh\gamma_m}{\gamma_m^2 + (m\pi - n\pi)^2}$$

$$ex6 = \frac{-(m\pi - n\pi)\sinh\gamma_m}{\gamma_m^2 + (m\pi - n\pi)^2} \qquad y6 = \frac{-\gamma_m \sinh\gamma_m}{\gamma_m^2 + (m\pi + n\pi)^2}$$

8.5.3 Moment

a. Moment along the η = 1, ξ = 1, η = 0 and ξ = 0 edges

There is no contribution by Building Block 5 to the bending moment along the edges under consideration: *AS(i,j)* 0.

b. Moment along η = 1 - ξ

$$\frac{M_n\phi b}{D} = -\left[\theta_1\,\phi^2\frac{\partial^2 W_5(\xi,\eta)}{\partial\xi^2} + \theta_2\frac{\partial^2 W_5(\xi,\eta)}{\partial\eta^2} + \theta_3\,\phi\frac{\partial^2 W_5(\xi,\eta)}{\partial\xi\partial\eta}\right]$$

$$\left.\frac{\partial^2 W_5(\xi,\eta)}{\partial\xi^2}\right|_{\eta=1-\xi} = -\sum_{m=1}^{k^*} E_{5m}\theta_{51m}(m\pi)^2\left[\sinh\beta_m\xi + C_{51m}\sin\gamma_m\xi\right]\sin(m\pi\xi)$$
$$-\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}(m\pi)^2\left[\sinh\beta_m\xi + C_{52m}\sinh\gamma_m\xi\right]\sin(m\pi\xi)$$

$$\left.\frac{\partial^2 W_5(\xi,\eta)}{\partial\eta^2}\right|_{\eta=1-\xi} = \sum_{m=1}^{k^*} E_{5m}\theta_{51m}\left[\beta_m^2\sinh\beta_m\xi - C_{51m}\gamma_m^2\sin\gamma_m\xi\right]\sin(m\pi\xi)$$
$$+\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}\left[\beta_m^2\sinh\beta_m\xi + C_{52m}\gamma_m^2\sinh\gamma_m\xi\right]\sin(m\pi\xi)$$

$$\left.\frac{\partial^2 W_5(\xi,\eta)}{\partial\xi\partial\eta}\right|_{\eta=1-\xi} = -\sum_{m=1}^{K^*} E_{5m}\theta_{51m}(m\pi)\left[\beta_m \cosh\beta_m\xi + C_{51m}\gamma_m \cos\gamma_m\xi\right]\cos(m\pi\xi)$$

$$-\sum_{m=K^*+1}^{\infty} E_{5m}\theta_{52m}(m\pi)\left[\beta_m \cosh\beta_m\xi + C_{52m}\gamma_m \cosh\gamma_m\xi\right]\cos(m\pi\xi)$$

Upon expansion, the elements of the submatrix can be calculated from

$$AS(i,j) = -\left[\theta_1\phi^2\, A1N + \theta_2\, A2N + \theta_3\phi\, A3N\right], \tag{2-71}$$

where $A1N = -\theta_{51m}(m\pi)^2\left[ex1 + ex2 + C_{51m}(ex3 + ex4)\right]$

$$A2N = \theta_{51m}\left[\beta_m^2(ex1 + ex2) - C_{51m}\gamma_m^2(ex3 + ex4)\right]$$

$$A3N = -\theta_{51m}(m\pi)\left[\beta_m(y1 + y2) + C_{51m}\gamma_m(y3 + y4)\right] \qquad \text{for } \lambda^2 - (m\pi)^2 > 0,$$

or $A1N = -\theta_{52m}(m\pi)^2\left[ex1 + ex2 + C_{52m}(ex5 + ex6)\right]$

$$A2N = \theta_{52m}\left[\beta_m^2(ex1 + ex2) + C_{52m}\gamma_m^2(ex5 + ex6)\right]$$

$$A3N = -\theta_{52m}(m\pi)\left[\beta_m(y1 + y2) + C_{52m}\gamma_m(y5 + y6)\right] \qquad \text{for } \lambda^2 - (m\pi)^2 < 0.$$

The expressions for $ex1$ to $ex6$ are identical to those for the displacement of Building Block 5 along $\eta = 1 - \xi$, found in section 8.5.1.c. The remaining expressions are:

$$y1 = \frac{(m\pi + n\pi)\left[\cosh\beta_m - \cos(m\pi + n\pi)\right]}{\beta_m^2 + (m\pi + n\pi)^2}$$

$$y2 = \frac{(m\pi - n\pi)\left[\cos(m\pi - n\pi) - \cosh\beta_m\right]}{\beta_m^2 + (m\pi - n\pi)^2}$$

$$y3 = \frac{(m\pi + n\pi)\left[\cos\gamma_m - \cos(m\pi + n\pi)\right]}{(m\pi + n\pi)^2 - \gamma_m^2}$$

$$y4 = \frac{(m\pi - n\pi)\left[\cos\gamma_m - \cos(m\pi - n\pi)\right]}{\gamma_m^2 - (m\pi - n\pi)^2}$$

$$y5 = \frac{(m\pi + n\pi)\left[\cosh\gamma_m - \cos(m\pi + n\pi)\right]}{\gamma_m^2 + (m\pi + n\pi)^2}$$

$$y6 = \frac{(m\pi - n\pi)\left[\cos(m\pi - n\pi) - \cosh\gamma_m\right]}{\gamma_m^2 + (m\pi - n\pi)^2}$$

$$\theta_1 = \cos^2\alpha + \nu\sin^2\alpha$$
$$\theta_2 = \sin^2\alpha + \nu\cos^2\alpha$$
$$\theta_3 = (1-\nu)\sin 2\alpha$$

8.5.4 Vertical Edge Reaction

a. Vertical Edge Reaction along the η = 1 edge

$$\frac{V_\eta \phi b^2}{D} = -\left[\frac{\partial^3 W_1(\xi,\eta)}{\partial\eta^3} + (2-\nu)\phi^2\frac{\partial^3 W_1(\xi,\eta)}{\partial\xi^2\partial\eta}\right]$$

$$\left.\frac{\partial^3 W_5(\xi,\eta)}{\partial\eta^3}\right|_{\eta=1} = -\sum_{m=1}^{k^*} E_{5m}\theta_{51m}\left[\beta_m^3 - C_{51m}\gamma_m^3\right]\sin(m\pi\xi)$$
$$-\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}\left[\beta_m^3 + C_{52m}\gamma_m^3\right]\sin(m\pi\xi)$$

$$\left.\frac{\partial^3 W_5(\xi,\eta)}{\partial\xi^2\partial\eta}\right|_{\eta=1} = \sum_{m=1}^{k^*} E_{51m}\theta_{51m}(m\pi)^2\left[\beta_m + C_{51m}\gamma_m\right]\sin(m\pi\xi)$$
$$+\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}(m\pi)^2\left[\beta_m + C_{52m}\gamma_m\right]\sin(m\pi\xi)$$

Only the elements on the diagonal of the submatrix are non-zero:

$$AS(i,j) \quad 0 \text{ for } i \neq j. \qquad (2\text{-}72a)$$

$$AS(j,j) = -\left[A1N + (2-\nu)\phi^2 A2N\right], \qquad (2\text{-}72b)$$

where $A1N = -\,\theta_{51m}\left[\beta_m^3 - C_{51m}\gamma_m^3\right]$

$$A2N = \theta_{51m}(m\pi)^2\left[\beta_m + C_{51m}\gamma_m\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0,$$

and $A1N = -\,\theta_{52m}\left[\beta_m^3 + C_{52m}\gamma_m^3\right]$

$$A2N = \theta_{52m}(m\pi)^2\left[\beta_m + C_{52m}\gamma_m\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0.$$

b. Vertical Edge Reaction along the ξ = 1 edge

$$\frac{V_\eta a^2}{D} = -\left[\frac{\partial^3 W_5(\xi,\eta)}{\partial\xi^3} + \frac{(2-\nu)}{\phi^2}\frac{\partial^3 W_5(\xi,\eta)}{\partial\eta^2\partial\xi}\right]$$

$$\left.\frac{\partial^3 W_5(\xi,\eta)}{\partial\xi^3}\right|_{\xi=1} = -\sum_{m=1}^{k^*} E_{5m}\theta_{51m}(m\pi)^3\left[\sinh\beta_m(1-\eta) + C_{51m}\sin\gamma_m(1-\eta)\right]\cos(m\pi)$$

$$-\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}(m\pi)^3\left[\sinh\beta_m(1-\eta) + C_{52m}\sinh\gamma_m(1-\eta)\right]\cos(m\pi)$$

$$\left.\frac{\partial^3 W_5(\xi,\eta)}{\partial\xi\,\partial\eta^2}\right|_{\xi=1} = \sum_{m=1}^{k^*} E_{5m}\theta_{51m}(m\pi)\left[\beta_m^2\sinh\beta_m(1-\eta) - C_{51m}\gamma_m^2\sin\gamma_m(1-\eta)\right]\cos(m\pi)$$

$$+\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}(m\pi)\left[\beta_m^2\sinh\beta_m(1-\eta) + C_{52m}\gamma_m^2\sinh\gamma_m(1-\eta)\right]\cos(m\pi)$$

The submatrix elements are now defined by

$$AS(J,J) = -\left[A1N + \frac{(2-\nu)}{\phi^2}A2N\right], \qquad (2\text{-}73)$$

where $A1N = -\theta_{51m}(m\pi)^3\cos(m\pi)\left[ex1 + C_{51m}ex2\right]$

$A2N = \theta_{51m}(m\pi)\cos(m\pi)\left[\beta_m^2 ex1 - C_{51m}\gamma_m^2 ex2\right]$ for $\lambda^2 - (m\pi)^2 > 0$,

and $A1N = -\theta_{52m}(m\pi)^3\cos(m\pi)\left[ex1 + C_{52m}ex3\right]$

$A2N = \theta_{52m}(m\pi)\cos(m\pi)\left[\beta_m^2 ex1 + C_{52m}\gamma_m^2 ex3\right]$ for $\lambda^2 - (m\pi)^2 < 0$.

The expressions $ex1$, $ex2$ and $ex3$ are identical to those for the slope contribution of Building Block 5 along the $\xi = 1$ edge, found in section 8.5.2.b.

c. Vertical edge reaction along the $\eta = 0$ edge

$$\frac{V_\eta\phi b^2}{D} = -\left[\frac{\partial^3 W_5(\xi,\eta)}{\partial\eta^3} + (2-\nu)\phi^2\frac{\partial^3 W_5(\xi,\eta)}{\partial\xi^2\,\partial\eta}\right]$$

$$\left.\frac{\partial^3 W_5(\xi,\eta)}{\partial\eta^3}\right|_{\eta=0} = -\sum_{m=1}^{k^*} E_{5m}\theta_{51m}\left[\beta_m^3\cosh\beta_m - C_{51m}\gamma_m^3\cos\gamma_m\right]\sin(m\pi\xi)$$

$$-\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}\left[\beta_m^3\cosh\beta_m + C_{52m}\gamma_m^3\cosh\gamma_m\right]\sin(m\pi\xi)$$

$$\left.\frac{\partial^3 W_5(\xi,\eta)}{\partial\xi^2\,\partial\eta}\right|_{\eta=0} = \sum_{m=1}^{k^*} E_{5m}\theta_{51m}(m\pi)^2\left[\beta_m\cosh\beta_m + C_{51m}\gamma_m\cos\gamma_m\right]\sin(m\pi\xi)$$

$$+\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}(m\pi)^2\left[\beta_m\cosh\beta_m + C_{52m}\gamma_m\cosh\gamma_m\right]\sin(m\pi\xi)$$

Elements not on the diagonal of the submatrix are zero:

$$AS(i,j) = 0 \text{ for } i \neq j. \tag{2-74a}$$

On the diagonal, the elements are determined by

$$AS(j,j) = -\left[A1N + (2-\nu)\phi^2 A2N\right], \tag{2-74b}$$

where $A1N = -\,\theta_{51m}\left[\beta_m^3 \cosh\beta_m - C_{51m}\gamma_m^3 \cos\gamma_m\right]$

$$A2N = \theta_{51m}(m\pi)^2\left[\beta_m \cosh\beta_m + C_{51m}\gamma_m \cos\gamma_m\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0,$$

and $A1N = -\,\theta_{52m}\left[\beta_m^3 \cosh\beta_m + C_{52m}\gamma_m^3 \cosh\gamma_m\right]$

$$A2N = \theta_{52m}(m\pi)^2\left[\beta_m \cosh\beta_m + C_{52m}\gamma_m \cosh\gamma_m\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0.$$

d. Vertical Edge Reaction along the $\xi = 0$ edge

$$\frac{V_\eta a^2}{D} = -\left[\frac{\partial^3 W_5(\xi,\eta)}{\partial\xi^3} + \frac{(2-\nu)}{\phi^2}\frac{\partial^3 W_5(\xi,\eta)}{\partial\eta^2\,\partial\xi}\right]$$

$$\left.\frac{\partial^3 W_5(\xi,\eta)}{\partial\xi^3}\right|_{\xi=0} = -\sum_{m=1}^{k^*} E_{5m}\theta_{51m}(m\pi)^3\left[\sinh\beta_m(1-\eta) + C_{51m}\sin\gamma_m(1-\eta)\right]$$

$$-\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}(m\pi)^3\left[\sinh\beta_m(1-\eta) + C_{52m}\sinh\gamma_m(1-\eta)\right]$$

$$\left.\frac{\partial^3 W_5(\xi,\eta)}{\partial\xi\,\partial\eta^2}\right|_{\xi=0} = \sum_{m=1}^{k^*} E_{5m}\theta_{51m}(m\pi)\left[\beta_m^2 \sinh\beta_m(1-\eta) + C_{51m}\gamma_m^2 \sin\gamma_m(1-\eta)\right]$$

$$+\sum_{m=k^*+1}^{\infty} E_{5m}\theta_{52m}(m\pi)\left[\beta_m^2 \sinh\beta_m(1-\eta) + C_{52m}\gamma_m^2 \sinh\gamma_m(1-\eta)\right]$$

The submatrix elements are defined by

$$AS(j,j) = -\left[A1N + \frac{(2-\nu)}{\phi^2}A2N\right], \tag{2-75}$$

where $A1N = -\theta_{51m}(m\pi)^3\left[ex1 + C_{51m}ex2\right]$

$$A2N = \theta_{51m}(m\pi)\left[\beta_m^2 ex1 - C_{51m}\gamma_m^2 ex2\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0,$$

and $A1N = -\theta_{52m}(m\pi)^3\left[ex1 + C_{52m}ex3\right]$

$$A2N = \theta_{52m}(m\pi)\left[\beta_m^2 ex1 + C_{52m}\gamma_m^2 ex3\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0.$$

The expressions $ex1$, $ex2$ and $ex3$ are identical to those for the slope contribution of Building Block 5 along this edge and also along $\xi = 1$, found in sections 8.5.2.b. and 8.5.2.d.

8.6. Building Block 6

To obtain the contributions of Building Block 6, the constants θ_{61m}, C'_{61m}, θ_{62m} and C'_{62m} replace θ_{51m}, C'_{51m}, θ_{52m} and C'_{52m}, respectively, in the expressions for Building Block 5, unless specified otherwise in the following subsections.

8.6.1 Displacement along the $\eta = 0$ edge

Building Block 6 does not have a displacement along this edge:

$$AS(i,j) = 0 \quad \text{for } i \neq j. \tag{2-76}$$

8.6.2 Moment along the $\eta = 0$ edge

Elements that are not on the diagonal in the submatrix are zero:

$$AS(i,j) = 0 \quad \text{for } i \neq j. \tag{2-77}$$

Building Block 6 has a forced rotation of the form

$$\left.\frac{Mb^2}{aD}\right|_{\eta=1} = \sum_{m=1}^{\infty} E_{6m} \sin(m\pi\xi) \tag{2-19}$$

along this edge. Hence, the elements on the diagonal of the submatrix for Building Block 6 are

$$AS(j,j) = 1. \tag{2-78}$$

8.7 Building Block 7

8.7.1 Displacement

a. Displacement along the η = 1, ξ = 1 and η = 0 edges

There is no contribution by Building Block 7 to the displacement along these edges.

$$AS(i,j) = 0. \qquad (2\text{-}79)$$

b. Displacement along ξ = 0

For Building Block 7, there is a forced displacement along this edge:

$$W_7(\xi,\eta)\big|_{\xi=0} = \sum_{m=1}^{\infty} E_{7m}\sin(m\pi\eta) \qquad (2\text{-}21)$$

No Fourier expansion is required, since the series is already in the desired form.

$$AS(j,j) = 1.0, \qquad (2\text{-}80a)$$

and where $i \neq j$,

$$AS(i,j) = 0. \qquad (2\text{-}80b)$$

c. Displacement along η = 1 - ξ

The unexpanded solution is

$$W_7(\xi,\eta) = \sum_{m=1}^{k^*} E_{7m}\theta_{71m}\left[\sinh(\beta_{3m}(1-\eta)) + C_{71m}\sin(\gamma_{3m}(1-\eta))\right]\sin(m\pi(1-\xi))$$
$$+ \sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}\left[\sinh(\beta_{3m}(1-\eta)) + C_{72m}\sinh(\gamma_{3m}(1-\eta))\right]\sin(m\pi(1-\xi))$$

The elements of the submatrix then become

$$AS(i,j) = \theta_{71m}\left[ex1 + ex2 + C_{71m}(ex3 + ex4)\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0, \qquad (2\text{-}81a)$$

or

$$AS(i,j) = \theta_{72m}\left[ex1 + ex2 + C_{72m}(ex5 + ex6)\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0. \qquad (2\text{-}81b)$$

where

$$ex1 = \beta_{3m}\frac{\cosh\beta_{3m}\cos(m\pi) - \cos(n\pi)}{\beta_{3m}^2 + (m\pi + n\pi)^2} \qquad ex4 = \gamma_{3m}\frac{\cos(m\pi)\cos\gamma_{3m} - \cos(n\pi)}{\gamma_{3m}^2 - (m\pi - n\pi)^2}$$

$$ex2 = \beta_{3m}\frac{\cos(n\pi) - \cosh\beta_{3m}\cos(m\pi)}{\beta_{3m}^2 + (m\pi - n\pi)^2} \qquad ex5 = \gamma_{3m}\frac{\cosh\gamma_{3m}\cos(m\pi) - \cos(n\pi)}{\gamma_{3m}^2 + (m\pi + n\pi)^2}$$

$$ex3 = \gamma_{3m}\frac{\cos(n\pi) - \cos(m\pi)\cos\gamma_{3m}}{\gamma_{3m}^2 - (m\pi + n\pi)^2} \qquad ex6 = \gamma_{3m}\frac{\cos(n\pi) - \cosh\gamma_{3m}\cos(m\pi)}{\gamma_{3m}^2 + (m\pi - n\pi)^2}$$

8.7.2 Slope

a. Slope along the η = 1 edge

$$\left.\frac{\partial W_7(\xi,\eta)}{\partial \eta}\right|_{\eta=1} = \sum_{m=1}^{k^*} E_{7m}\theta_{71m}(m\pi)\left[\sinh\beta_{3m}(1-\xi) + C_{71m}\sin\gamma_{3m}(1-\xi)\right]\cos(m\pi)$$
$$+\sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}(m\pi)\left[\sinh\beta_{3m}(1-\xi) + C_{72m}\sinh\gamma_{3m}(1-\xi)\right]\cos(m\pi)$$

In expanded form, this becomes

$$AS(i,j) = \theta_{71m}(m\pi)\cos(m\pi)\left[ex1 + C_{71m}ex2\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0, \qquad (2\text{-}82a)$$

and $$AS(i,j) = \theta_{72m}(m\pi)\cos(m\pi)\left[ex1 + C_{71m}ex3\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0, \qquad (2\text{-}82b)$$

where $$ex1 = \frac{2(n\pi)\sinh\beta_{3m}}{\beta_{3m}^2 + (n\pi)^2}$$
$$ex2 = \frac{2(n\pi)\sin\gamma_{3m}}{(n\pi)^2 - \gamma_{3m}^2}$$
$$ex3 = \frac{2(n\pi)\sinh\gamma_{3m}}{\gamma_{3m}^2 + (n\pi)^2}$$

b. Slope along the ξ = 1 edge

$$\left.\frac{\partial W_7(\xi,\eta)}{\partial \xi}\right|_{\xi=1} = \sum_{m=1}^{k^*} E_{7m}\theta_{71m}\left[\beta_{3m} + C_{71m}\gamma_{3m}\right]\sin(m\pi\eta)$$
$$+\sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}\left[\beta_{3m} + C_{72m}\gamma_{3m}\right]\sin(m\pi\eta)$$

All elements that are not on the diagonal of the submatrix are zero:

$$AS(i,j) = 0 \quad \text{for } i \neq j. \qquad (2\text{-}83a)$$

On the diagonal, $$AS(i,j) = \theta_{71m}\left[\beta_{3m} + C_{71m}\gamma_{3m}\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0, \qquad (2\text{-}83b)$$

and $$AS(i,j) = \theta_{72m}\left[\beta_{3m} + C_{72m}\gamma_{3m}\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0. \qquad (2\text{-}83c)$$

c. Slope along the η = 0 edge

$$\left.\frac{\partial W_7(\xi,\eta)}{\partial \eta}\right|_{\eta=0} = \sum_{m=1}^{k^*} E_{7m}\theta_{71m}(m\pi)\left[\sinh\beta_{3m}(1-\xi) + C_{71m}\sin\gamma_{3m}(1-\xi)\right]$$
$$+\sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}(m\pi)\left[\sinh\beta_{3m}(1-\xi) + C_{72m}\sinh\gamma_{3m}(1-\xi)\right]$$

In expanded for, this then becomes

$$AS(i,j) = \theta_{71m}(m\pi)\left[ex1 + C_{71m}ex2\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0, \tag{2-84a}$$

and $$AS(i,j) = \theta_{72m}(m\pi)\left[ex1 + C_{72m}ex3\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0, \tag{2-84b}$$

where *ex1*, *ex2* and *ex3* are identical to those of section 8.7.2.a.:

$$ex1 = \frac{2(n\pi)\sinh\beta_{3m}}{\beta_{3m}^2 + (n\pi)^2}$$

$$ex2 = \frac{2(n\pi)\sin\gamma_{3m}}{(n\pi)^2 - \gamma_{3m}^2}$$

$$ex3 = \frac{2(n\pi)\sinh\gamma_{3m}}{\gamma_{3m}^2 + (n\pi)^2}$$

d. Slope along the $\xi = 0$ edge

Elements off of the diagonal of the submatrix are zero:

$$AS(i,j) \quad 0 \text{ for } i \neq j. \tag{2-85a}$$

The elements on the diagonal are

$$AS(i,j) = \theta_{71m}\left[\beta_{3m}\cosh\beta_{3m} + C_{71m}\gamma_{3m}\cos\gamma_{3m}\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0, \tag{2-85b}$$

and $$AS(i,j) = \theta_{72m}\left[\beta_{3m}\cosh\beta_{3m} + C_{72m}\gamma_{3m}\cosh\gamma_{3m}\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0. \tag{2-85c}$$

e. Slope along $\eta = 1 - \xi$

The slope along the diagonal is given by

$$\frac{\partial W_7(\xi,\eta)}{\partial n} = \frac{\partial W_7(\xi,\eta)}{\partial \xi}\cos\alpha + \frac{1}{\phi}\frac{\partial W_7(\xi,\eta)}{\partial \xi}\sin\alpha$$

where

$$\left.\frac{\partial W_7(\xi,\eta)}{\partial \xi}\right|_{\eta=1-\xi} = -\sum_{m=1}^{k^*} E_{7m}\theta_{71m}\left[\beta_{3m}\cosh\beta_{3m}(1-\xi) + C_{71m}\gamma_{3m}\cos\gamma_{3m}(1-\xi)\right]\sin(m\pi(1-\xi))$$
$$- \sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}\left[\beta_{3m}\cosh\beta_{3m}(1-\xi) + C_{72m}\gamma_{3m}\cosh\gamma_{3m}(1-\xi)\right]\sin(m\pi(1-\xi))$$

$$\left.\frac{\partial W_7(\xi,\eta)}{\partial \eta}\right|_{\xi=1-\eta} = \sum_{m=1}^{k^*} E_{7m}\theta_{71m}(m\pi)\left[\sinh\beta_{3m}(1-\xi) + C_{71m}\sin\gamma_{3m}(1-\xi)\right]\cos(m\pi(1-\xi))$$
$$+ \sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}(m\pi)\left[\sinh\beta_{3m}(1-\xi) + C_{72m}\sinh\gamma_{3m}(1-\xi)\right]\cos(m\pi(1-\xi))$$

After the expansion is performed, the elements of the submatrix can be found from:

$$AS(i,j) = A1N\cos\alpha + \frac{1}{\phi}A2N\sin\alpha\,, \qquad (2\text{-}86)$$

where $A1N = \theta_{71m}\left[\beta_{3m}(ex1 + ex2) + C_{71m}\,\gamma_{3m}(ex3 + ex4)\right]$

$A2N = \theta_{71m}(m\pi)\left[y1 + y2 + C_{71m}(y3 + y4)\right]$ for $\lambda^2 - (m\pi)^2 > 0$,

or $A1N = \theta_{72m}\left[\beta_{3m}(ex1 + ex2) + C_{72m}\gamma_{3m}(ex5 + ex6)\right]$

$A2N = \theta_{72m}(m\pi)\left[y1 + y2 + C_{72m}(y5 + y6)\right]$ for $\lambda^2 - (m\pi)^2 < 0$.

$$ex1 = \frac{\beta_{3m}\cos(m\pi)\sinh\beta_{3m}}{\beta_{3m}^2 + (m\pi - n\pi)^2} \qquad y1 = \frac{(m\pi + n\pi)\cos(m\pi)\sinh\beta_{3m}}{\beta_{3m}^2 + (m\pi + n\pi)^2}$$

$$ex2 = -\frac{\beta_{3m}\cos(m\pi)\sinh\beta_{3m}}{\beta_{3m}^2 + (m\pi + n\pi)^2} \qquad y2 = -\frac{(m\pi - n\pi)\cos(m\pi)\sinh\beta_{3m}}{\beta_{3m}^2 + (m\pi - n\pi)^2}$$

$$ex3 = \frac{\gamma_{3m}\cos(m\pi)\sin\gamma_{3m}}{\gamma_{3m}^2 - (m\pi - n\pi)^2} \qquad y3 = \frac{(m\pi + n\pi)\cos(m\pi)\sin\gamma_{3m}}{(m\pi + n\pi)^2 - \gamma_{3m}^2}$$

$$ex4 = \frac{\gamma_{3m}\cos(m\pi)\sin\gamma_{3m}}{(m\pi + n\pi)^2 - \gamma_{3m}^2} \qquad y4 = \frac{(m\pi - n\pi)\cos(m\pi)\sin\gamma_{3m}}{\gamma_{3m}^2 - (m\pi - n\pi)^2}$$

$$ex5 = \frac{\gamma_{3m}\cos(m\pi)\sinh\gamma_{3m}}{\gamma_{3m}^2 + (m\pi - n\pi)^2} \qquad y5 = \frac{(m\pi + n\pi)\cos(m\pi)\sinh\gamma_{3m}}{\gamma_{3m}^2 + (m\pi + n\pi)^2}$$

$$ex6 = -\frac{\gamma_{3m}\cos(m\pi)\sinh\gamma_{3m}}{\gamma_{3m}^2 + (m\pi + n\pi)^2} \qquad y6 = -\frac{(m\pi - n\pi)\cos(m\pi)\sinh\gamma_{3m}}{\gamma_{3m}^2 + (m\pi - n\pi)^2}$$

8.7.3 Moment

a. Moment along the η = 1, ξ = 1, η = 0 and ξ = 0 edges

There is no contribution by Building Block 7 to the bending moment along the outside edges:

$$AS(i,j) = 0. \qquad (2\text{-}87)$$

b. Moment along η = 1 - ξ

$$\frac{M_n\phi b}{D} = -\left[\theta_1\phi^2\frac{\partial^2 W_7(\xi,\eta)}{\partial\xi^2} + \theta_2\frac{\partial^2 W_7(\xi,\eta)}{\partial\eta^2} + \theta_3\phi\frac{\partial^2 W_7(\xi,\eta)}{\partial\xi\,\partial\eta}\right]$$

$$\left.\frac{\partial^2 W_7(\xi,\eta)}{\partial \xi^2}\right|_{\eta=1-\xi} = \sum_{m=1}^{k^*} E_{7m}\theta_{71m}\left[\beta_{3m}^2 \sinh\beta_{3m}(1-\xi) - C_{71m}\gamma_{3m}^2 \sin\gamma_{3m}(1-\xi)\right]\sin(m\pi(1-\xi))$$

$$+ \sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}\left[\beta_{3m}^2 \sinh\beta_{3m}(1-\xi) + C_{72m}\gamma_{3m}^2 \sinh\gamma_{3m}(1-\xi)\right]\sin(m\pi(1-\xi))$$

$$\left.\frac{\partial^2 W_7(\xi,\eta)}{\partial \eta^2}\right|_{\eta=1-\xi} = -\sum_{m=1}^{k^*} E_{7m}\theta_{71m}(m\pi)^2\left[\sinh\beta_{3m}(1-\xi) + C_{71m}\sin\gamma_{3m}(1-\xi)\right]\sin(m\pi(1-\xi))$$

$$- \sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}(m\pi)^2\left[\sinh\beta_{3m}(1-\xi) + C_{72m}\sinh\gamma_{3m}(1-\xi)\right]\sin(m\pi(1-\xi))$$

$$\left.\frac{\partial^2 W_7(\xi,\eta)}{\partial \xi\,\partial \eta}\right|_{\eta=1-\xi} = \sum_{m=1}^{k^*} E_{7m}\theta_{71m}(m\pi)\left[\beta_{3m}\cosh\beta_{3m}(1-\xi) + C_{31m}\gamma_{3m}\cos\gamma_{3m}(1-\xi)\right]\cos(m\pi(1-\xi))$$

$$+ \sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}(m\pi)\left[\beta_{3m}\cosh\beta_{3m}(1-\xi) + C_{32m}\gamma_{3m}\cosh\gamma_{3m}(1-\xi)\right]\cos(m\pi(1-\xi))$$

Upon expansion, the elements of the submatrix can be calculated from

$$AS(i,j) = -\left[\theta_1\,\phi^2\,A1N + \theta_2\,A2N + \theta_3\phi\,A3N\right], \tag{2-88}$$

where

$$\theta_1 = \cos^2\alpha + \nu\sin^2\alpha$$
$$\theta_2 = \sin^2\alpha + \nu\cos^2\alpha$$
$$\theta_3 = (1-\nu)\sin 2\alpha$$

and $A1N = \theta_{71m}\left[\beta_{3m}^{\ 2}(ex1 + ex2) - C_{71m}\gamma_{3m}^{\ 2}(ex3 + ex4)\right]$

$A2N = -\,\theta_{71m}(m\pi)^2\left[ex1 + ex2 + C_{71m}(ex3 + ex4)\right]$

$A3N = -\,\theta_{71m}(m\pi)\left[\beta_{3m}(y1 + y2) + C_{71m}\gamma_{3m}(y3 + y4)\right]$ for $\lambda^2 - (m\pi)^2 > 0$,

or $A1N = \theta_{72m}\left[\beta_{3m}^{\ 2}(ex1 + ex2) + C_{72m}\gamma_{3m}^{\ 2}(ex5 + ex6)\right]$

$A2N = -\theta_{72m}(m\pi)^2\left[ex1 + ex2 + C_{72m}(ex5 + ex6)\right]$

$A3N = -\,\theta_{72m}(m\pi)\left[\beta_{3m}(y1 + y2) + C_{72m}\gamma_{3m}(y5 + y6)\right]$ for $\lambda^2 - (m\pi)^2 < 0$.

The expressions for *ex1* to *ex6* are identical to those for the displacement of Building Block 1 along $\eta = 1 - \xi$, found in section 8.1.3. The remaining expressions are:

$$y1=\frac{(m\pi+n\pi)\left[\cos(m\pi)\cosh\beta_{3m}-\cos(n\pi)\right]}{\beta_{3m}^2+(m\pi+n\pi)^2}$$

$$y2=\frac{(m\pi-n\pi)\left[\cos(n\pi)-\cos(m\pi)\cosh\beta_{3m}\right]}{\beta_{3m}^2+(m\pi-n\pi)^2}$$

$$y3=\frac{(m\pi+n\pi)\left[\cos(m\pi)\cos\gamma_{3m}-\cos(n\pi)\right]}{(m\pi+n\pi)^2-\gamma_{3m}^2}$$

$$y4=\frac{(m\pi-n\pi)\left[\cos(n\pi)-\cos(m\pi)\cos\gamma_{3m}\right]}{(m\pi-n\pi)^2-\gamma_{3m}^2}$$

$$y5=\frac{(m\pi+n\pi)\left[\cos(m\pi)\cosh\gamma_{3m}-\cos(n\pi)\right]}{\gamma_{3m}^2+(m\pi+n\pi)^2}$$

$$y6=\frac{(m\pi-n\pi)\left[\cos(n\pi)-\cos(m\pi)\cosh\gamma_{3m}\right]}{\gamma_{3m}^2+(m\pi-n\pi)^2}$$

8.7.4 Vertical Edge Reaction

a. Vertical edge reaction along the η = 1 edge

$$\frac{V_\eta \phi h^2}{D}=-\left[\frac{\partial^3 W_7(\xi,\eta)}{\partial\eta^3}+(2-\nu)\phi^2\frac{\partial^3 W_7(\xi,\eta)}{\partial\xi^2\,\partial\eta}\right]$$

$$\left.\frac{\partial^3 W_7(\xi,\eta)}{\partial\eta^3}\right|_{\eta=1}=-\sum_{m=1}^{k^*}E_{7m}\theta_{71m}(m\pi)^3\left[\sinh\beta_{3m}(1-\xi)+C_{71m}\sin\gamma_{3m}(1-\xi)\right]\cos(m\pi)$$
$$-\sum_{m=k^*+1}^{\infty}E_{7m}\theta_{72m}(m\pi)^3\left[\sinh\beta_{3m}(1-\xi)+C_{72m}\sinh\gamma_{3m}(1-\xi)\right]\cos(m\pi)$$

$$\left.\frac{\partial^3 W_7(\xi,\eta)}{\partial\xi^2\,\partial\eta}\right|_{\eta=1}=\sum_{m=1}^{k^*}E_{7m}\theta_{71m}(m\pi)\left[\beta_{3m}^2\sinh\beta_{3m}(1-\xi)-C_{71m}\gamma_{3m}^2\sin\gamma_{3m}(1-\xi)\right]\cos(m\pi)$$
$$+\sum_{m=k^*+1}^{\infty}E_{7m}\theta_{72m}(m\pi)\left[\beta_{3m}^2\sinh\beta_{3m}(1-\xi)+C_{72m}\gamma_{3m}^2\sinh\gamma_{3m}(1-\xi)\right]\cos(m\pi)$$

Here the submatrix elements are defined by

$$AS(j,j)=-\left[A1N+(2-\nu)\phi^2 A2N\right], \tag{2-89}$$

where

$$A1N=-\theta_{71m}(m\pi)^3\cos(m\pi)\left[ex1+C_{71m}ex2\right]$$

$$A2N=\theta_{71m}(m\pi)\cos(m\pi)\left[\beta_{3m}^2 ex1-C_{71m}\gamma_{3m}^2 ex2\right]\ \text{for}\ \lambda^2-(m\pi)^2>0,$$

and $\qquad A1N = -\theta_{72m}(m\pi)^3\cos(m\pi)\left[ex1 + C_{72m}ex3\right]$

$$A2N = \theta_{72m}(m\pi)\cos(m\pi)\left[\beta_{3m}^2 ex1 + C_{72m}\gamma_{3m}^2 ex3\right] \text{ for } \lambda^2 - (m\pi)^2 < 0.$$

The expressions ex1, ex2 and ex3 are identical to those for the slope contribution of Building Block 7 along η = 1, found in section 8.7.2.a.

b. Vertical Edge Reaction along the ξ = 1 edge

$$\frac{V_\eta a^2}{D} = -\left[\frac{\partial^3 W_7(\xi,\eta)}{\partial\xi^3} + \frac{(2-\nu)}{\phi^2}\frac{\partial^3 W_7(\xi,\eta)}{\partial\eta^2\,\partial\xi}\right]$$

$$\left.\frac{\partial^3 W_7(\xi,\eta)}{\partial\xi^3}\right|_{\xi=1} = -\sum_{m=1}^{k^*} E_{7m}\theta_{71m}\left[\beta_{3m}^3 - C_{71m}\gamma_{3m}^3\right]\sin(m\pi\eta)$$

$$-\sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}\left[\beta_{3m}^3 + C_{72m}\gamma_{3m}^3\right]\sin(m\pi\eta)$$

$$\left.\frac{\partial^3 W_7(\xi,\eta)}{\partial\xi\,\partial\eta^2}\right|_{\xi=1} = \sum_{m=1}^{k^*} E_{7m}\theta_{71m}(m\pi)^2\left[\beta_{3m} + C_{71m}\gamma_{3m}\right]\sin(m\pi\eta)$$

$$+\sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}(m\pi)^2\left[\beta_{3m} + C_{72m}\gamma_{3m}\right]\sin(m\pi\eta)$$

Only the elements on the diagonal of the submatrix are non-zero:

$$AS(i,j) \quad 0 \text{ for } i \neq j. \tag{2-90a}$$

On the diagonal, $\qquad AS(j,j) = -\left[A1N + \dfrac{(2-\nu)}{\phi^2}A2N\right],$ $\qquad$ (2-90b)

where $\qquad A1N = -\theta_{71m}\left[\beta_{3m}^{\ 3} - C_{71m}\gamma_{3m}^{\ 3}\right]$

$$A2N = \theta_{71m}(m\pi)^2\left[\beta_{3m} + C_{71m}\gamma_{3m}\right] \quad \text{for } \lambda^2 - (m\pi)^2 > 0,$$

and $\qquad A1N = -\theta_{72m}\left[\beta_{3m}^{\ 3} + C_{72m}\gamma_{3m}^{\ 3}\right]$

$$A2N = \theta_{72m}(m\pi)^2\left[\beta_{3m} + C_{72m}\gamma_{3m}\right] \quad \text{for } \lambda^2 - (m\pi)^2 < 0.$$

c. Vertical edge reaction along the η = 0 edge

$$\frac{V_\eta\phi b^2}{D} = -\left[\frac{\partial^3 W_7(\xi,\eta)}{\partial\eta^3} + (2-\nu)\phi^2\frac{\partial^3 W_7(\xi,\eta)}{\partial\xi^2\,\partial\eta}\right]$$

$$\left.\frac{\partial^3 W_7(\xi,\eta)}{\partial \eta^3}\right|_{\eta=0} = -\sum_{m=1}^{k^*} E_{7m}\theta_{71m}(m\pi)^3\left[\sinh\beta_{3m}(1-\xi) + C_{71m}\sin\gamma_{3m}(1-\xi)\right]$$

$$-\sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}(m\pi)^3\left[\sinh\beta_{3m}(1-\xi) + C_{72m}\sinh\gamma_{3m}(1-\xi)\right]$$

$$\left.\frac{\partial^3 W_7(\xi,\eta)}{\partial \xi^2\,\partial \eta}\right|_{\eta=0} = \sum_{m=1}^{k^*} E_{7m}\theta_{71m}(m\pi)\left[\beta_{3m}^2\sinh\beta_{3m}(1-\xi) - C_{71m}\gamma_{3m}^2\sin\gamma_{3m}(1-\xi)\right]$$

$$+\sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}(m\pi)\left[\beta_{3m}^2\sinh\beta_{3m}(1-\xi) + C_{72m}\gamma_{3m}^2\sinh\gamma_{3m}(1-\xi)\right]$$

The submatrix elements are defined by

$$AS(j,j) = -\left[A1N + (2-\nu)\phi^2 A2N\right], \tag{2-91}$$

where $A1N = -\theta 7(m\pi)^3\left[ex1 + C_{71m}ex2\right]$

$$A2N = \theta_{71m}(m\pi)\left[\beta_{3m}^2 ex1 - C_{71m}\gamma_{3m}^2 ex2\right] \quad \text{for } \lambda^2-(m\pi)^2>0,$$

and $A1N = -\theta_{72m}(m\pi)^3\left[ex1 + C_{72m}ex3\right]$

$$A2N = \theta_{72m}(m\pi)\left[\beta_{3m}^2 ex1 + C_{72m}\gamma_{3m}^2 ex3\right] \quad \text{for } \lambda^2-(m\pi)^2<0.$$

d.Vertical edge reaction along the $\xi = 0$ edge

$$\frac{V_\eta a^2}{D} = -\left[\frac{\partial^3 W_7(\xi,\eta)}{\partial\xi^3} + \frac{(2-\nu)}{\phi^2}\frac{\partial^3 W_7(\xi,\eta)}{\partial\eta^2\,\partial\xi}\right]$$

$$\left.\frac{\partial^3 W_7(\xi,\eta)}{\partial \xi^3}\right|_{\xi=0} = -\sum_{m=1}^{k^*} E_{7m}\theta_{71m}\left[\beta_{3m}^3\cosh\beta_{3m} - C_{71m}\gamma_{3m}^3\cos\gamma_{3m}\right]\sin(m\pi\eta)$$

$$-\sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}\left[\beta_{3m}^3\cosh\beta_{3m} + C_{72m}\gamma_{3m}^3\cosh\gamma_{3m}\right]\sin(m\pi\eta)$$

$$\left.\frac{\partial^3 W_7(\xi,\eta)}{\partial \xi\,\partial\eta^2}\right|_{\xi=0} = \sum_{m=1}^{k^*} E_{7m}\theta_{71m}(m\pi)^2\left[\beta_{3m}\cosh\beta_{3m} + C_{31m}\gamma_{3m}\cos\gamma_{3m}\right]\sin(m\pi\eta)$$

$$+\sum_{m=k^*+1}^{\infty} E_{7m}\theta_{72m}(m\pi)^2\left[\beta_{3m}\cosh\beta_{3m} + C_{12m}\gamma_{3m}\cosh\gamma_{3m}\right]\sin(m\pi\eta)$$

Elements not on the diagonal of the submatrix are zero:

$$AS(i,j) = 0 \text{ for } i \neq j. \tag{2-92a}$$

On the diagonal, the elements are determined by

$$AS(j,j) = -\left[a1n + \frac{(2-\nu)}{\phi^2}a2n\right], \qquad (2\text{-}92b)$$

where $A1N = -\theta_{71m}\left[\beta_{3m}^{\;3}\cosh\beta_{3m} - C_{71m}\gamma_{3m}^{\;3}\cos\gamma_{3m}\right]$

$A2N = \theta_{71m}(m\pi)^2\left[\beta_{3m}\cosh\beta_{3m} + C_{71m}\gamma_{3m}\cos\gamma_{3m}\right]$ for $\lambda^2 - (m\pi)^2 > 0$,

and $A1N = -\theta_{72m}\left[\beta_{3m}^{\;3}\cosh\beta_{3m} + C_{72m}\gamma_{3m}^{\;3}\cosh\gamma_{3m}\right]$

$A2N = \theta_{72m}(m\pi)^2\left[\beta_{3m}\cosh\beta_{3m} + C_{72m}\gamma_{3m}\cosh\gamma_{3m}\right]$ for $\lambda^2 - (m\pi)^2 < 0$.

8.8. Building Block 8

For Building Block 8, the coefficients θ_{81m}, C_{81m}, θ_{82m} and C_{82m} replace θ_{71m}, C_{71m}, θ_{72m} and C_{72m}, respectively, in the expressions for Building Block 7, unless specified otherwise in the following sections.

8.8.1 Displacement along the $\xi = 0$ edge

There is no contribution for Building Block 4 along this edge: $AS(i,j) = 0$.

8.8.2 Moment along the $\xi = 0$ edge

On this edge, Building Block 4 has the forced moment

$$\left.\frac{Mb^2}{aD}\right|_{\xi=0} = \sum_{m=1}^{\infty} E_{8m}\sin(m\pi\xi) \qquad (2\text{-}23)$$

which is already a sine series in the proper terms. The submatrix elements on the diagonal are

$$AS(j,j) = 1. \qquad (2\text{-}93a)$$

For the remaining elements where $i \neq j$,

$$AS(i,j) = 0. \qquad (2\text{-}93b)$$

9. Eigenvalue Matrix Assembly

With the building blocks chosen and the building block contributions expanded in the same type of Fourier series, a set of linear algebraic equations can be assembled from the net boundary conditions on the outside edges of the plate and the continuity conditions at the segment interfaces. Each condition/edge combination results in a set of k or $2k$ equations, where k is the number of terms used in the driving force and moment expansions. The $2k$ equations are required for the conditions imposed on the diagonals to obtain as many equations as there are unknowns E_m, the coefficients of the driving functions in the building block solutions. The set of equations is best solved as an eigenvalue problem.

The number of terms used, k, depends on the desired accuracy of the results and the convergence of the eigenvalues for the plate under consideration. In general, 12 terms are sufficient for most cases. The eigenvalues are obtained by requiring the determinant of the eigenvalue matrix to be equal to zero.

Examples of eigenvalue matrices for various parallelogram plates are presented in the subsections that follow. Only non-zero values are shown, and only three terms are used in the examples. FORTRAN 77 computer source code found in Appendix 5 performs the eigenvalue matrix assembly as part of the eigenvalue search as well as the mode shape generation.

9.1 Eigenvalue matrices for SSSS Parallelogram Plates

Figures 2-13, 2-14 and 2-15 show the eigenvalue matrices for the parallelogram plate with simple supports on all four edges.

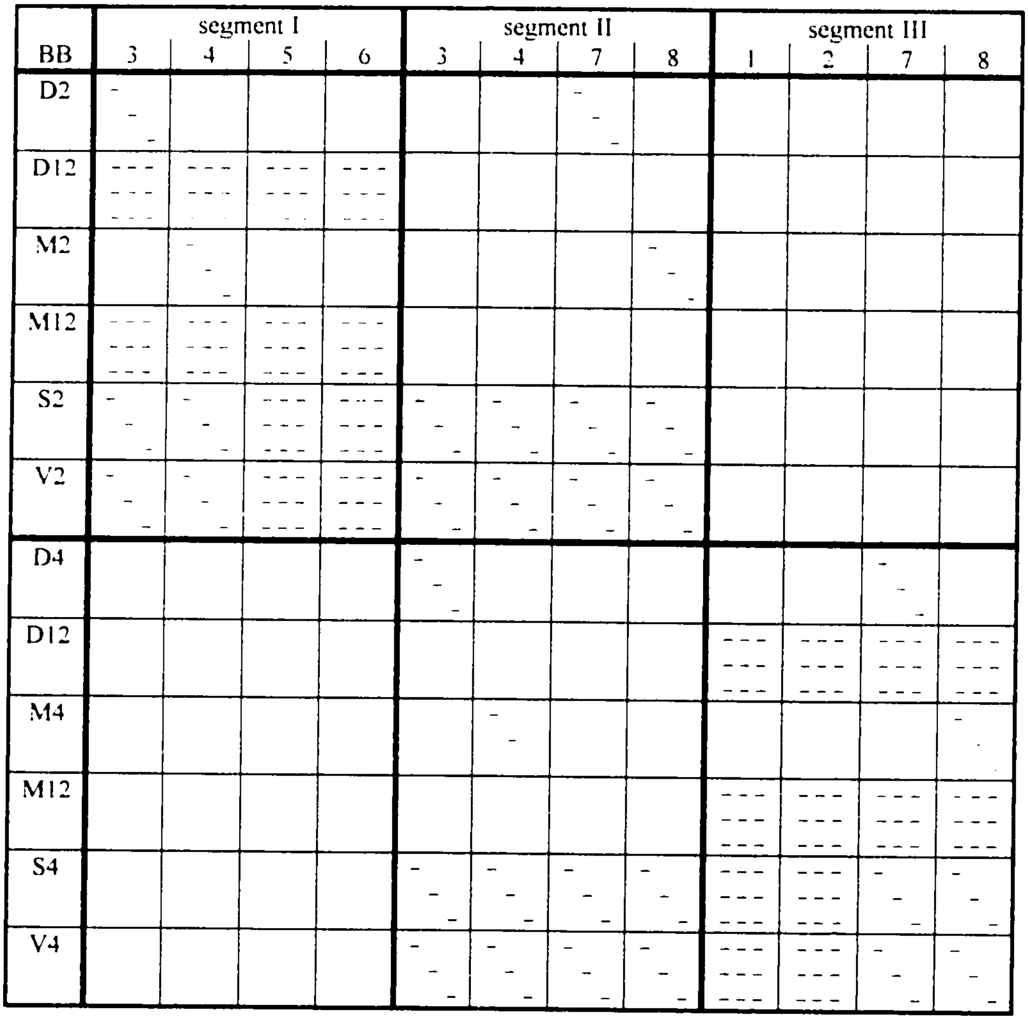


Figure 2-13 Eigenvalue matrix for SSSS parallelogram plate, $\phi_t \leq 0.5$.

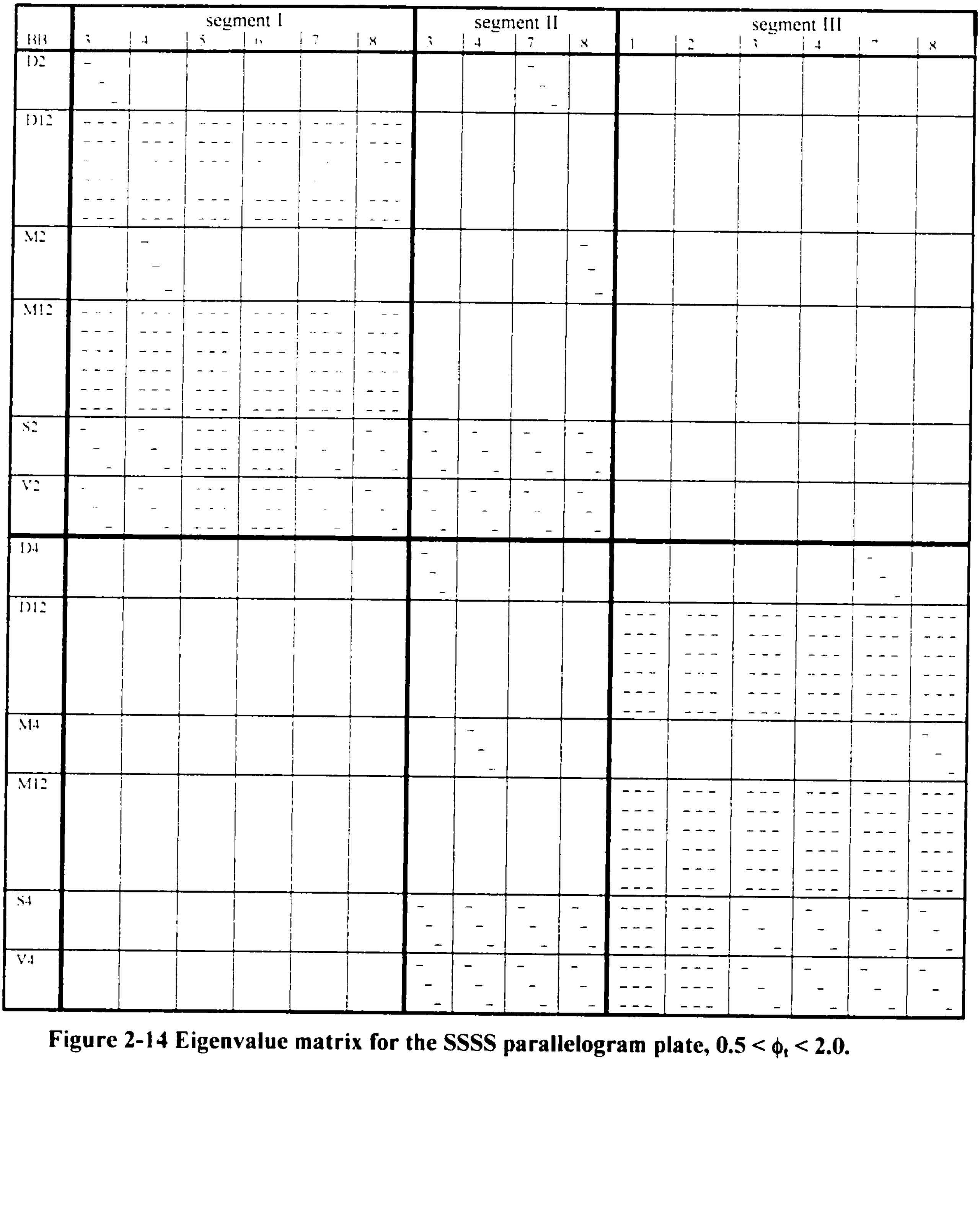


Figure 2-14 Eigenvalue matrix for the SSSS parallelogram plate, 0.5 < ϕ_t < 2.0.

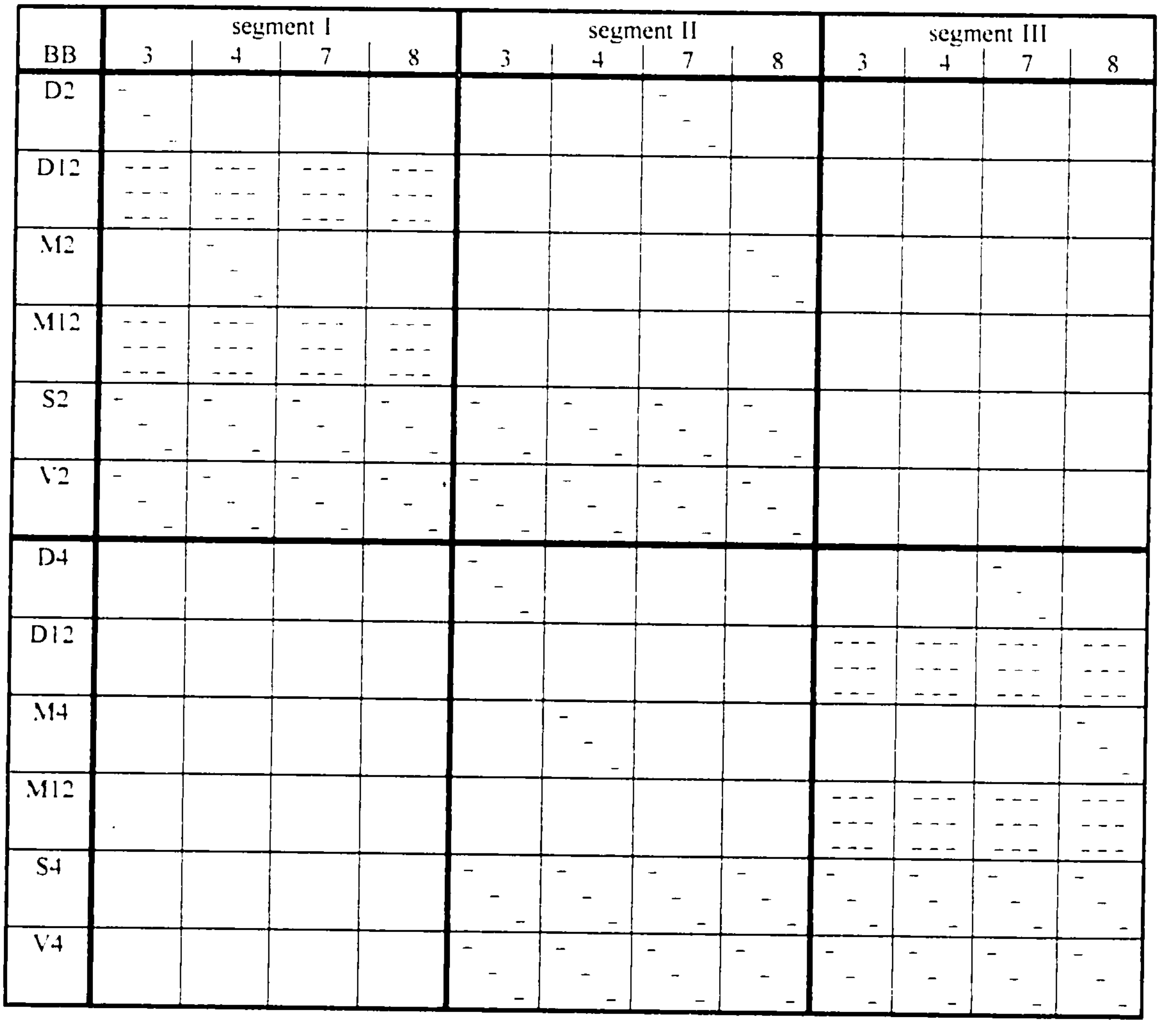


Figure 2-15 Eigenvalue matrix for the SSSS parallelogram plate, $\phi_t \geq 2.0$.

9.2 Eigenvalue Matrices for the SSSC Parallelogram Plate

The eigenvalue matrices for the parallelogram plate with the left skew edge clamped and all other edges simply supported is shown in Figures 2-16, 2-17 and 2-18. Note that the combination of building blocks chosen is identical to that for the SSSS plate. Only the boundary conditions along the one clamped edge differ.

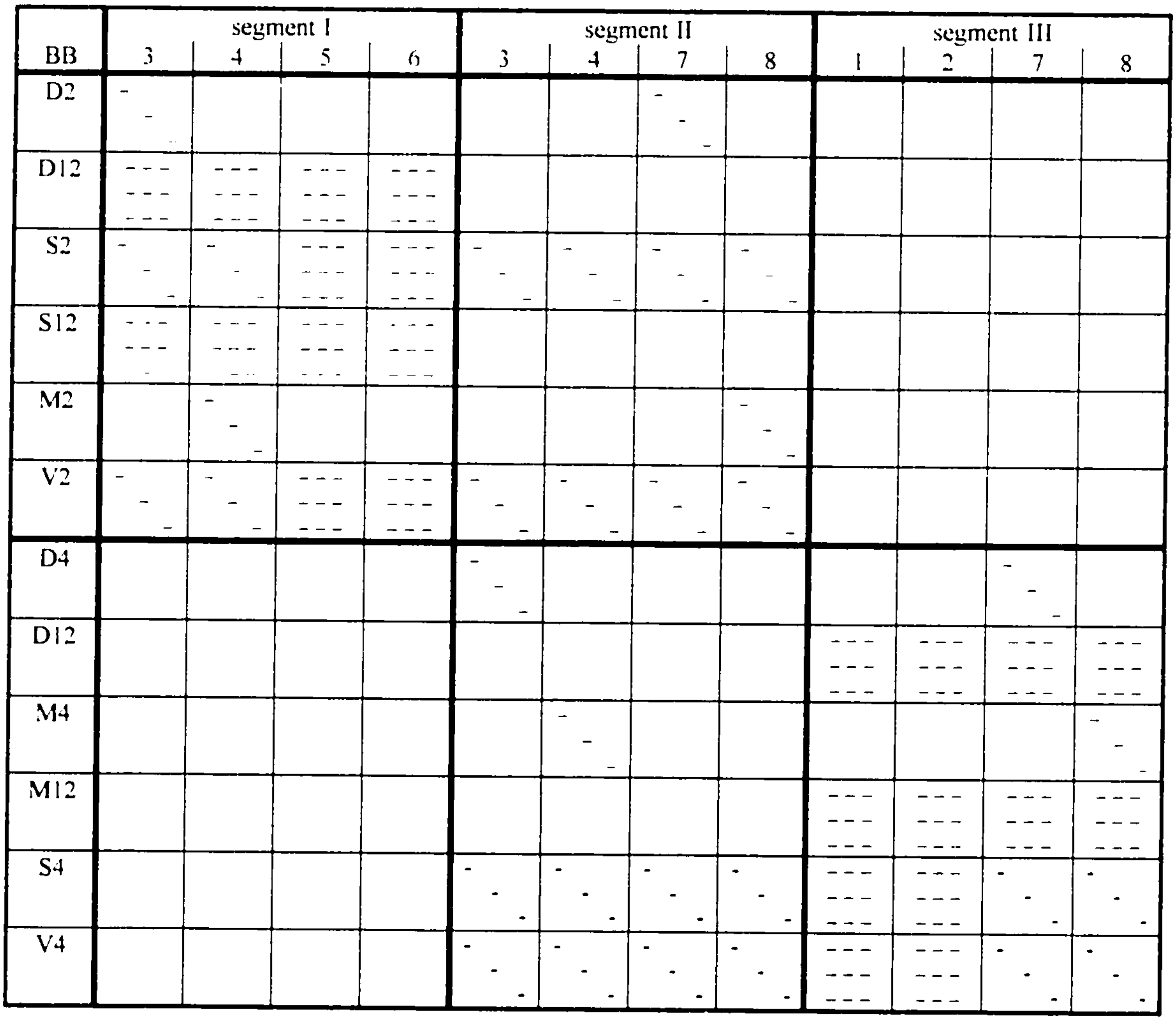


Figure 2-16 Eigenvalue matrix for the SSSC parallelogram plate, $\phi_t \leq 0.5$.

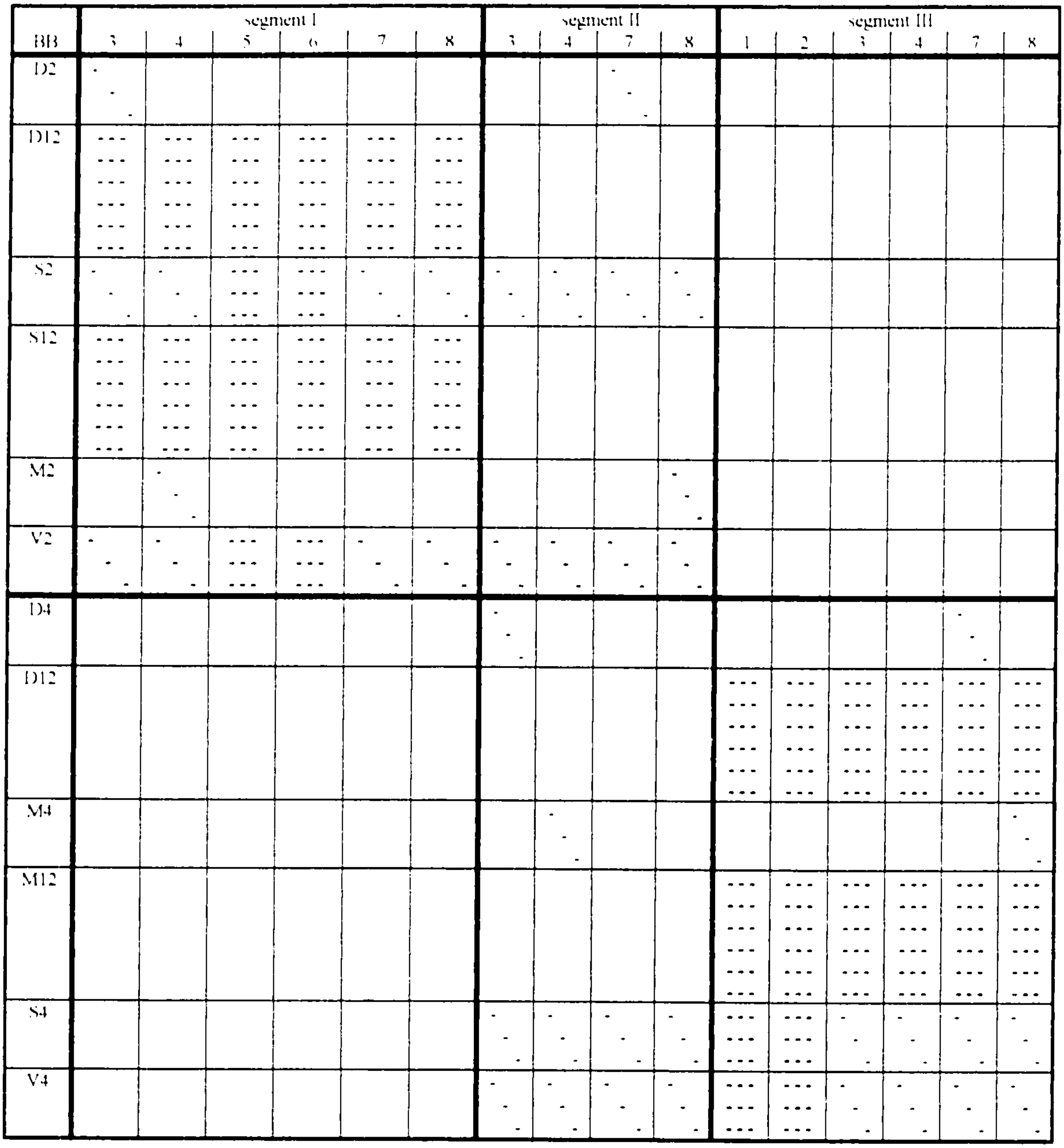


Figure 2-17 Eigenvalue matrix for the SSSC parallelogram plate, $0.5 < \phi_t < 2.0$.

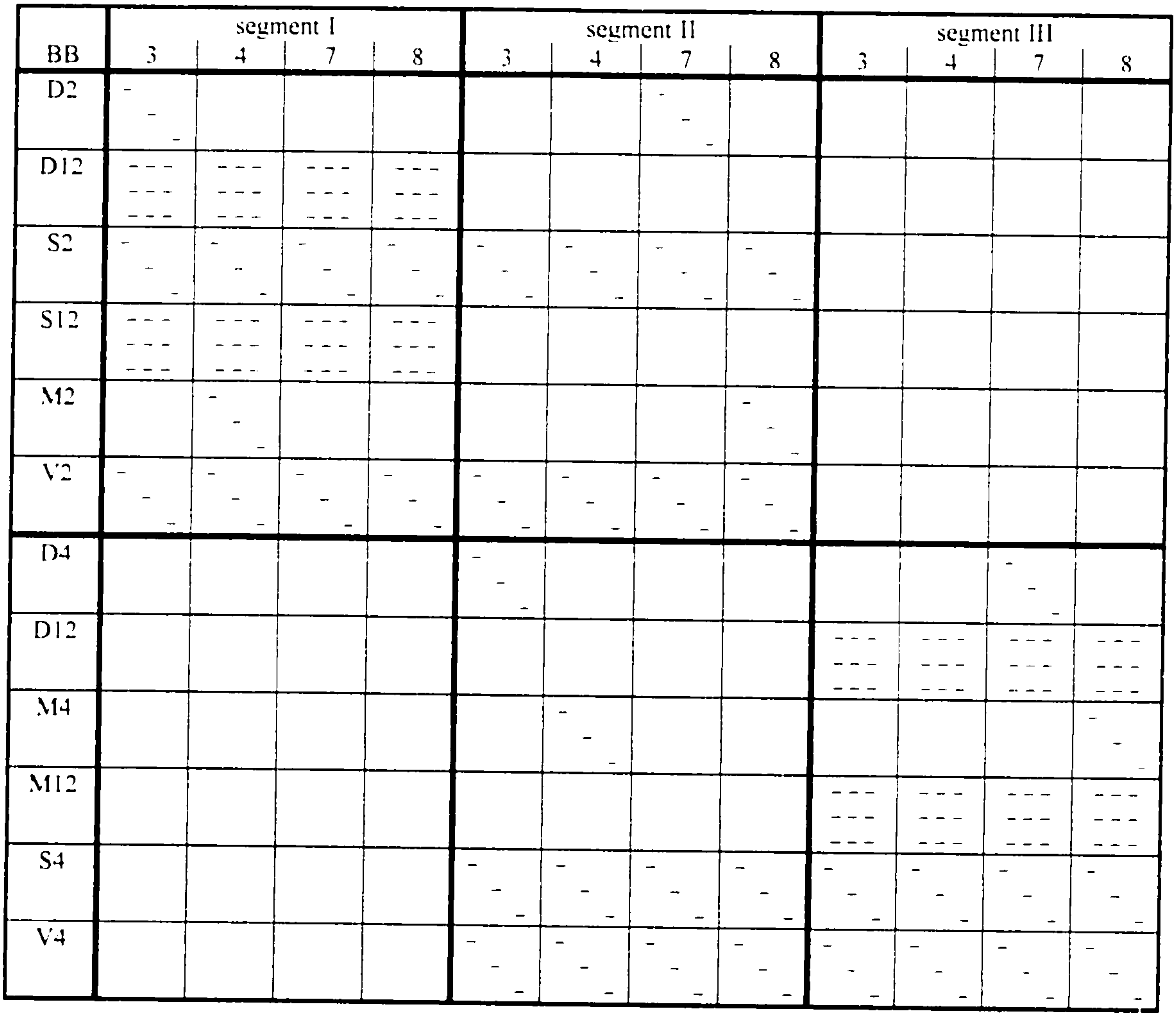


Figure 2-18 Eigenvalue matrix for the SSSC parallelogram plate, $\phi_t \geq 2.0$.

9.3 Eigenvalue Matrices for the SCSC Parallelogram Plate

For the case of the SCSC parallelogram plate, the combination of building blocks does not differ from the SSSS and SSSC cases; only the boundary conditions along the skew edges change.

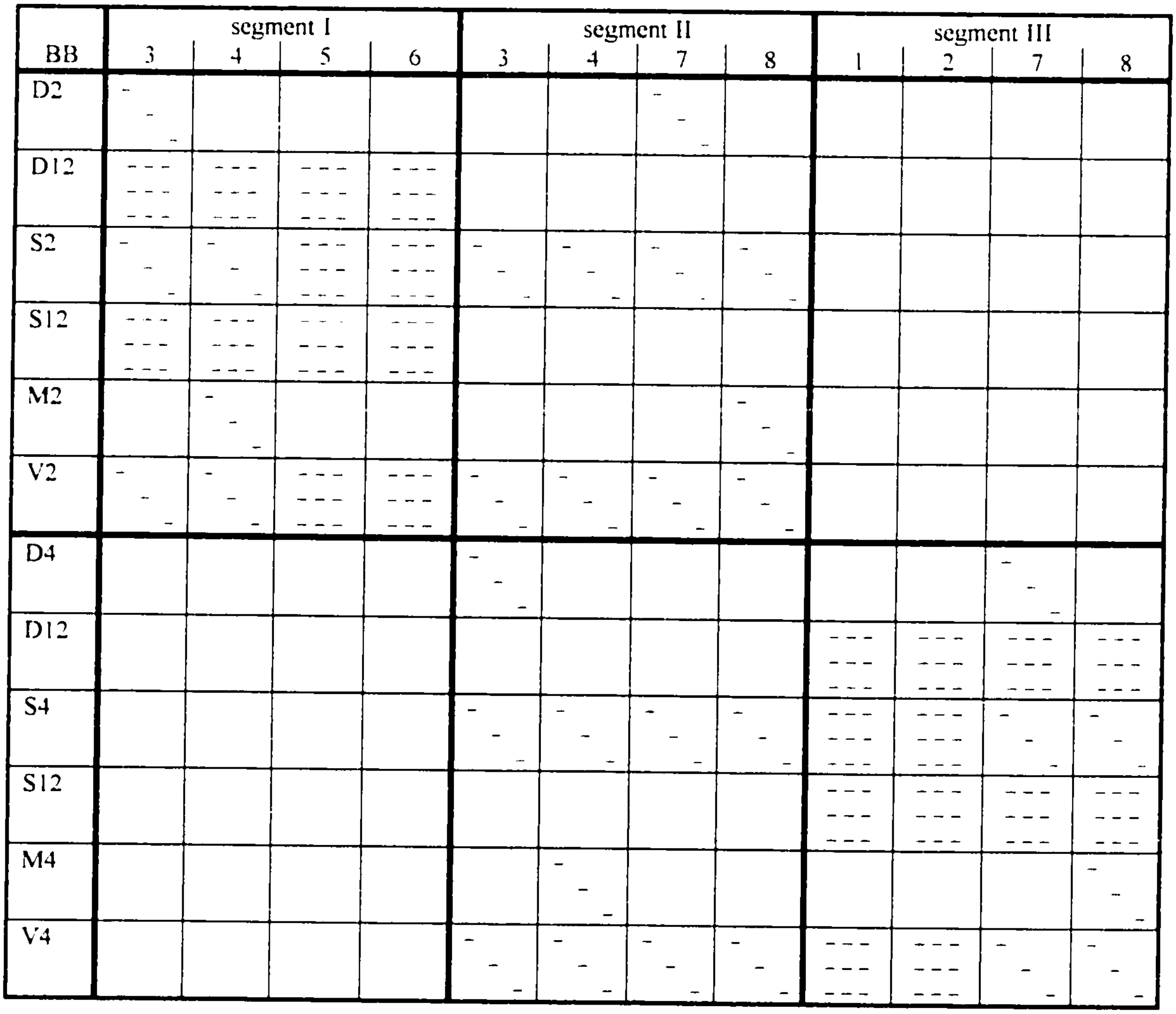


Figure 2-19 Eigenvalue matrix for the SCSCparallelogram plate, $\phi_t \leq 0.5$.

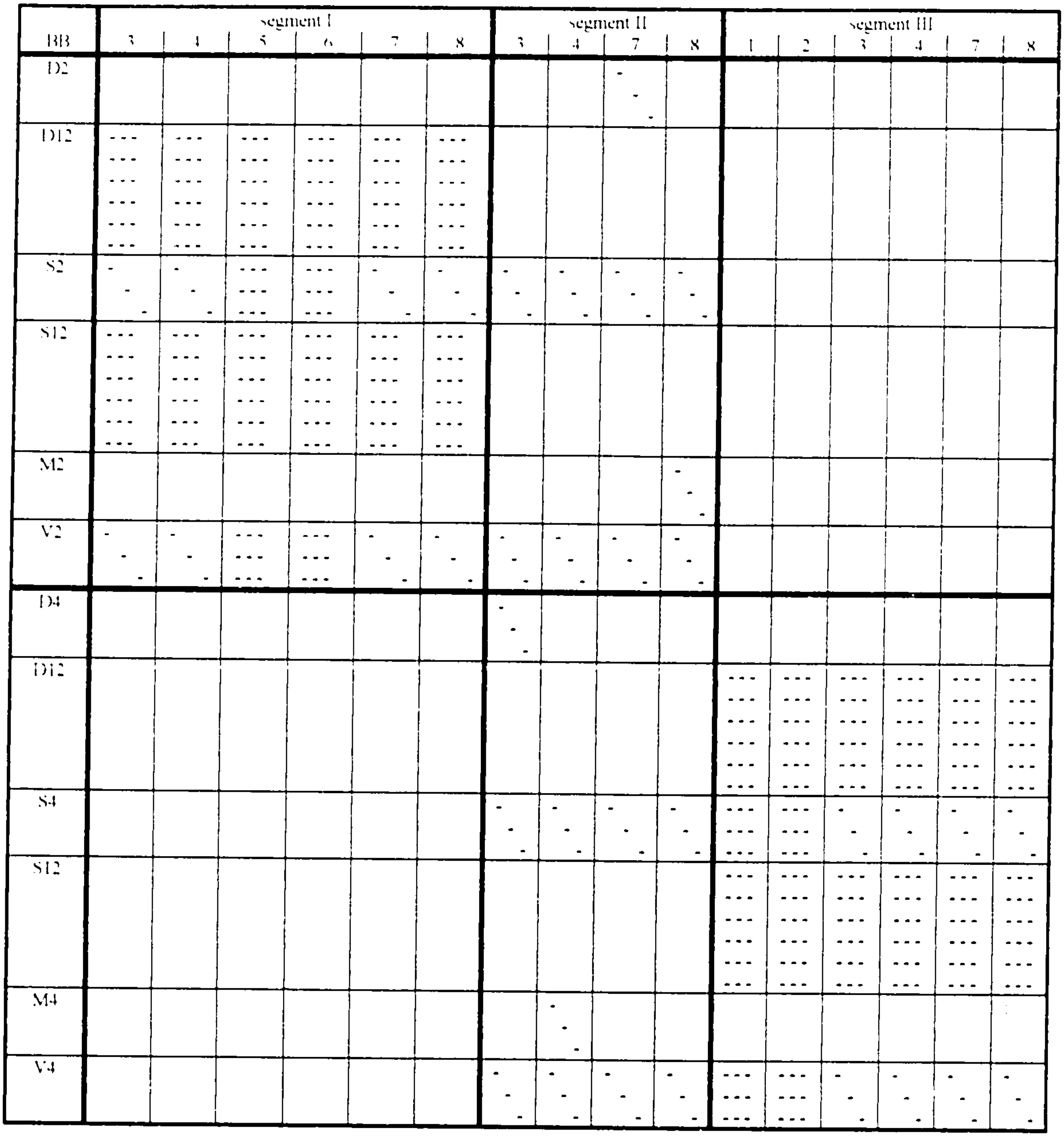


Figure 2-20 Eigenvalue matrix for the SCSC parallelogram plate, $0.5 < \phi_t < 2.0$.

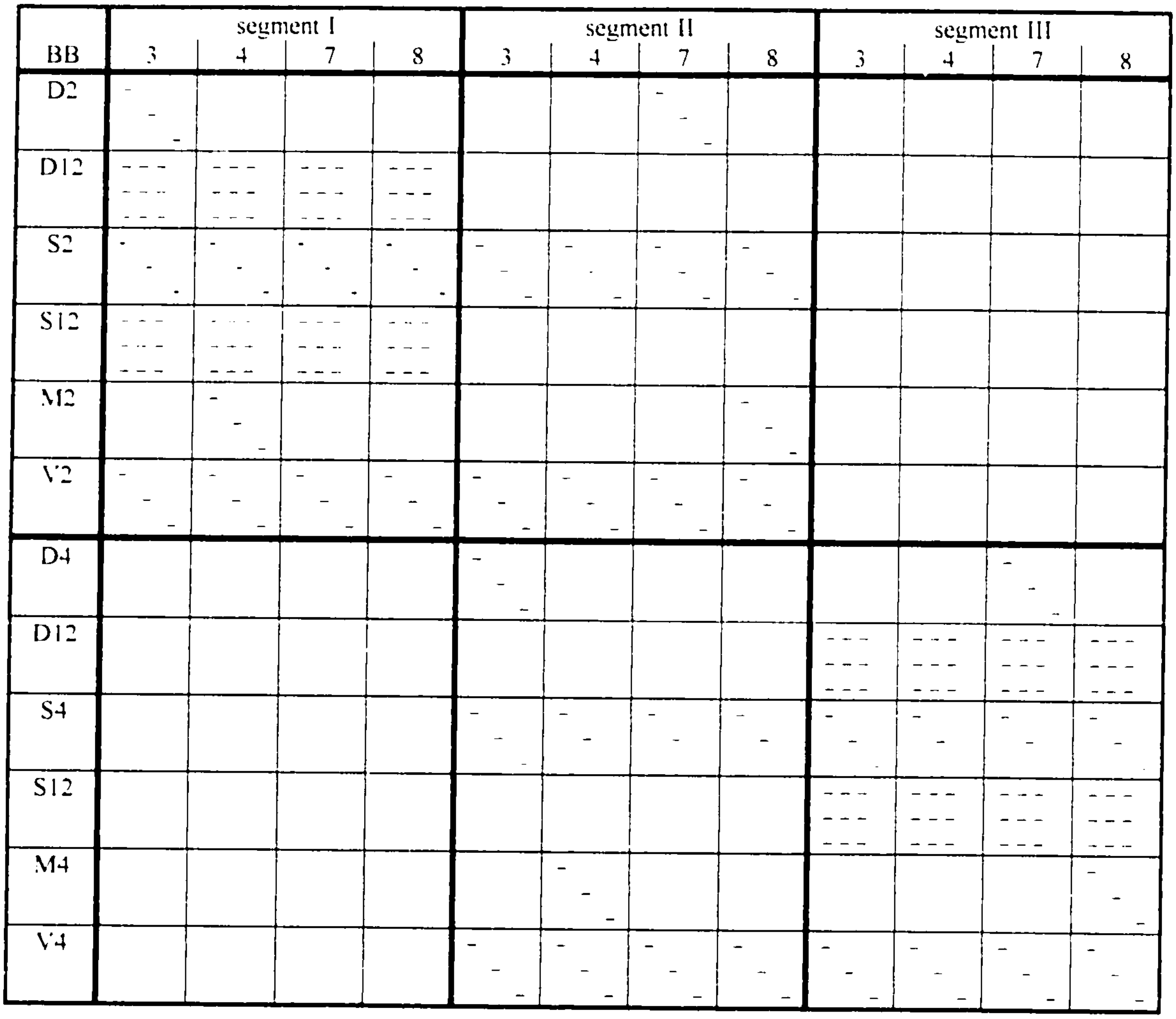


Figure 2-21 Eigenvalue matrix for the SCSC parallelogram plate, $\phi_t \geq 2.0$.

9.4 Eigenvalue Matrices for the CSSS Parallelogram Plate

To clamp the lower edge of the plate, two building blocks and two boundary conditions are added, as seen in Figures 2-22, 2-23 and 2-24.

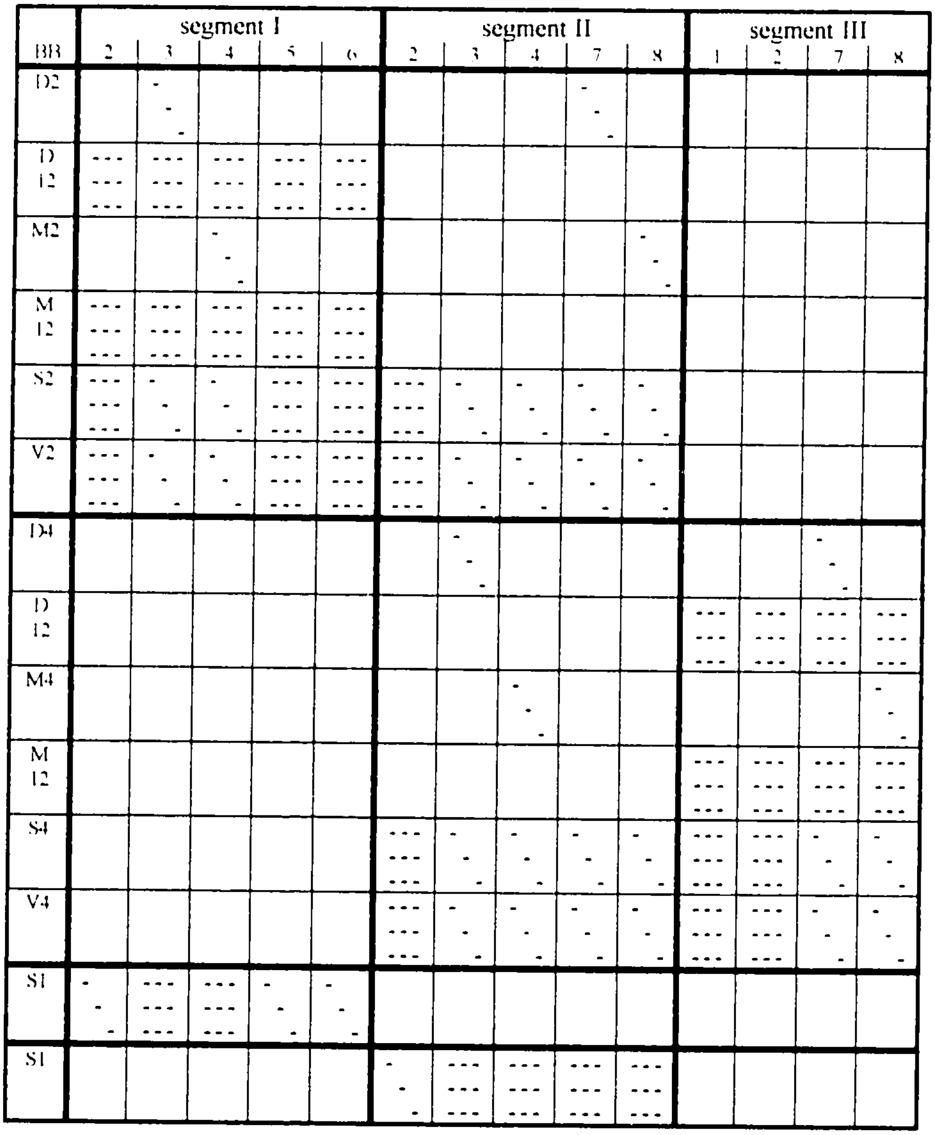


Figure 2-22 Eigenvalue matrix for the CSSS parallelogram plate, $\phi_t \leq 0.5$.

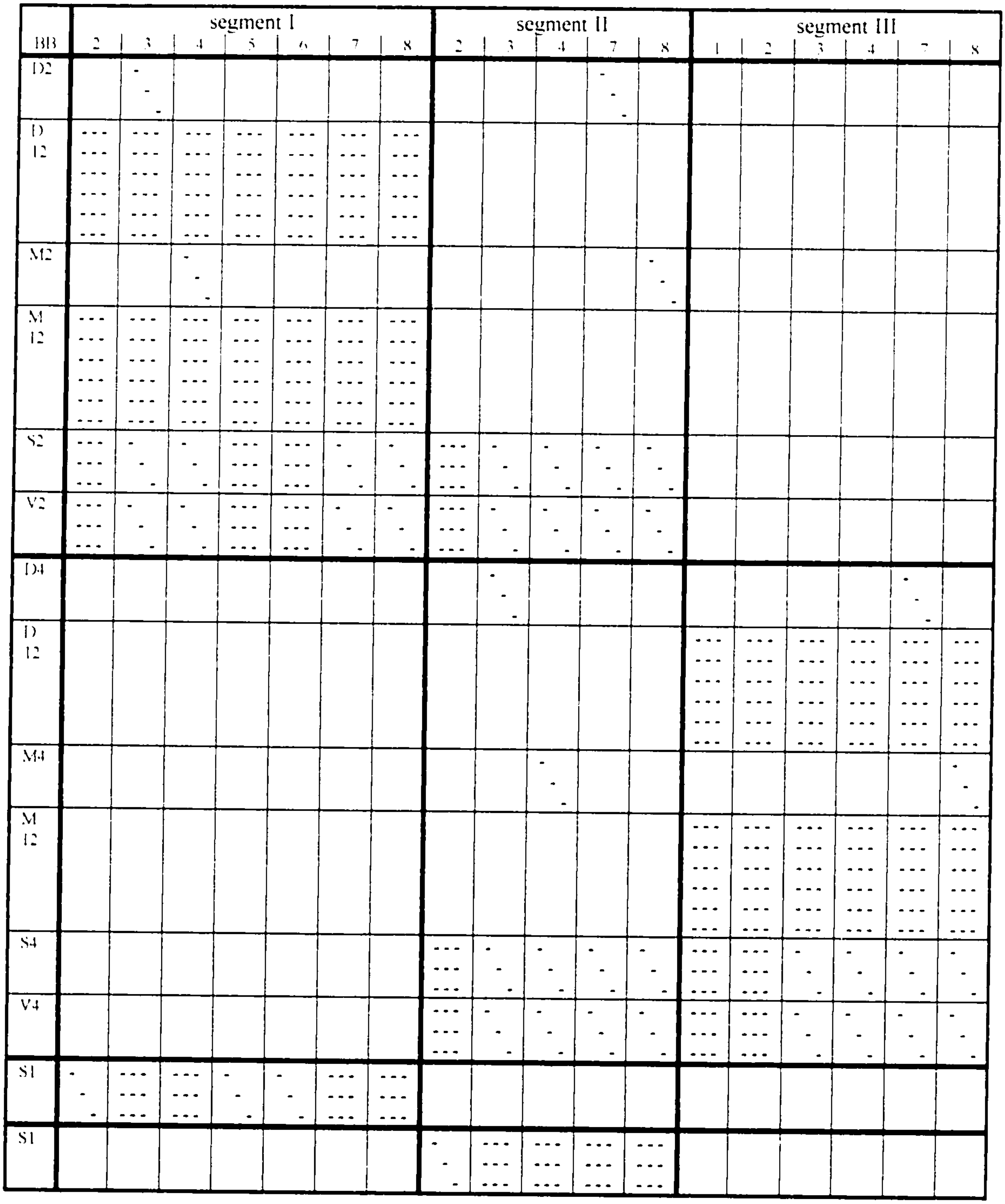


Figure 2-23 Eigenvalue matrix for the CSSS parallelogram plate, $0.5 < \phi_t < 2.0$.

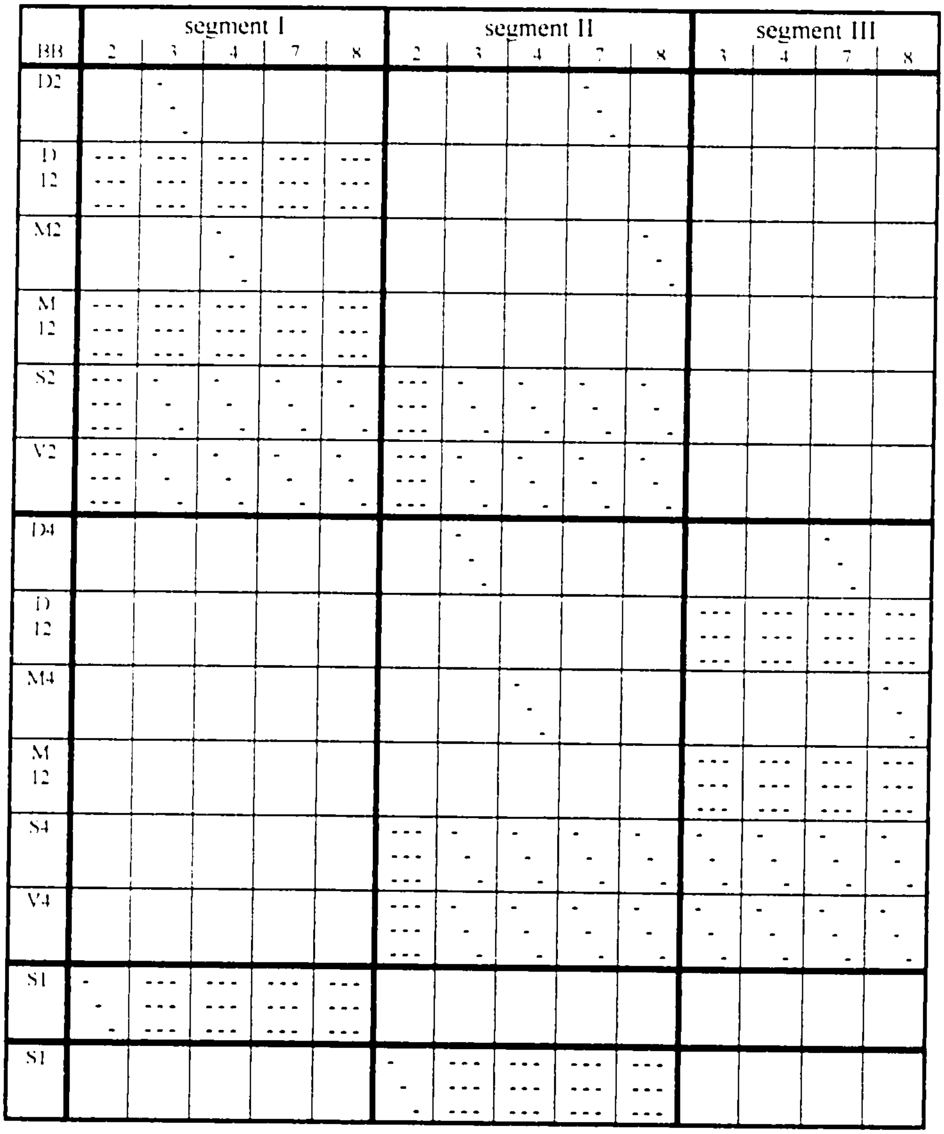


Figure 2-24 Eigenvalue matrix for the CSSS parallelogram plate, $\phi_t \geq 2.0$.

9.5 Eigenvalue Matrices for the CSSC Parallelogram Plate

Figures 2-25, 2-26 and 2-27 show the eigenvalue matrices for plates with the base and one skew edge clamped, and the remaining edges simply supported. The set of building blocks used is identical to that of the CSSS plate.

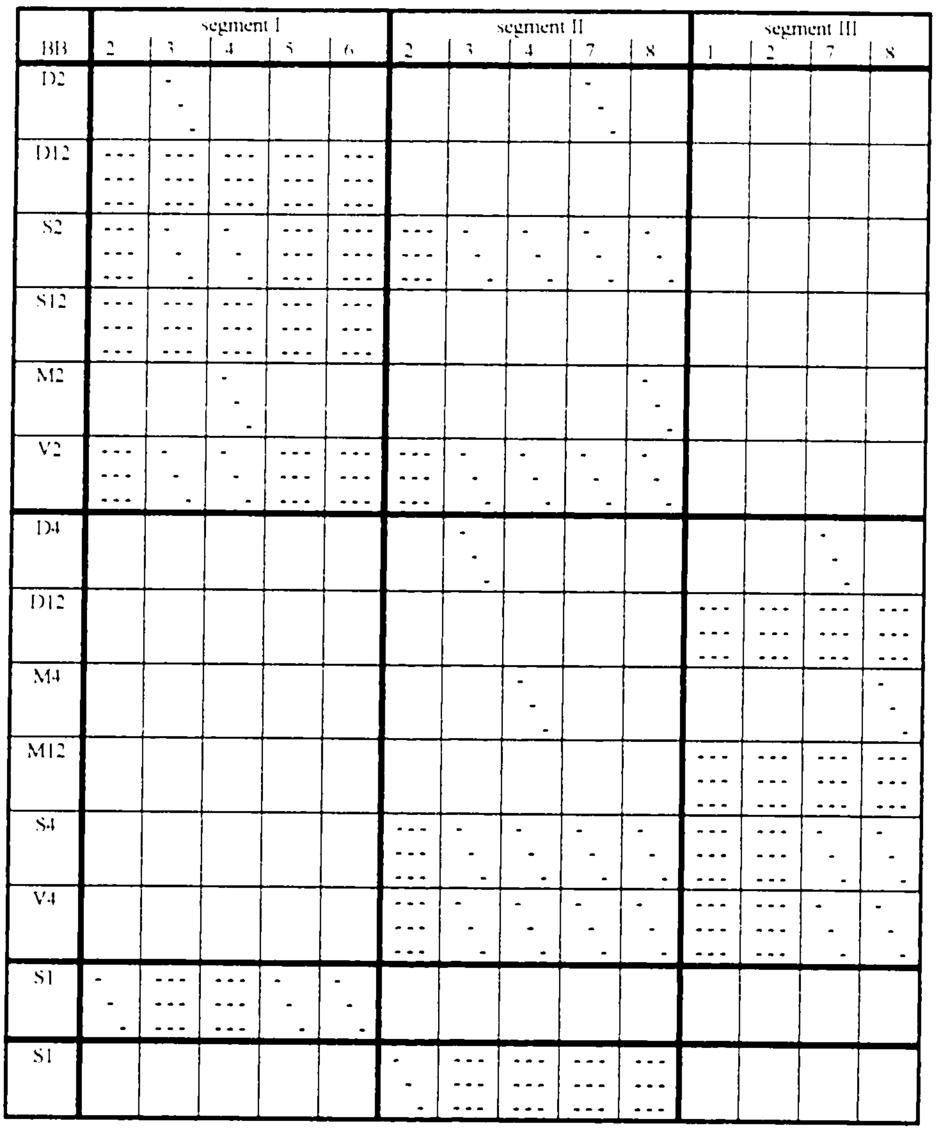


Figure 2-25 Eigenvalue matrix for the CSSC parallelogram plate, $\phi_t \leq 0.5$.

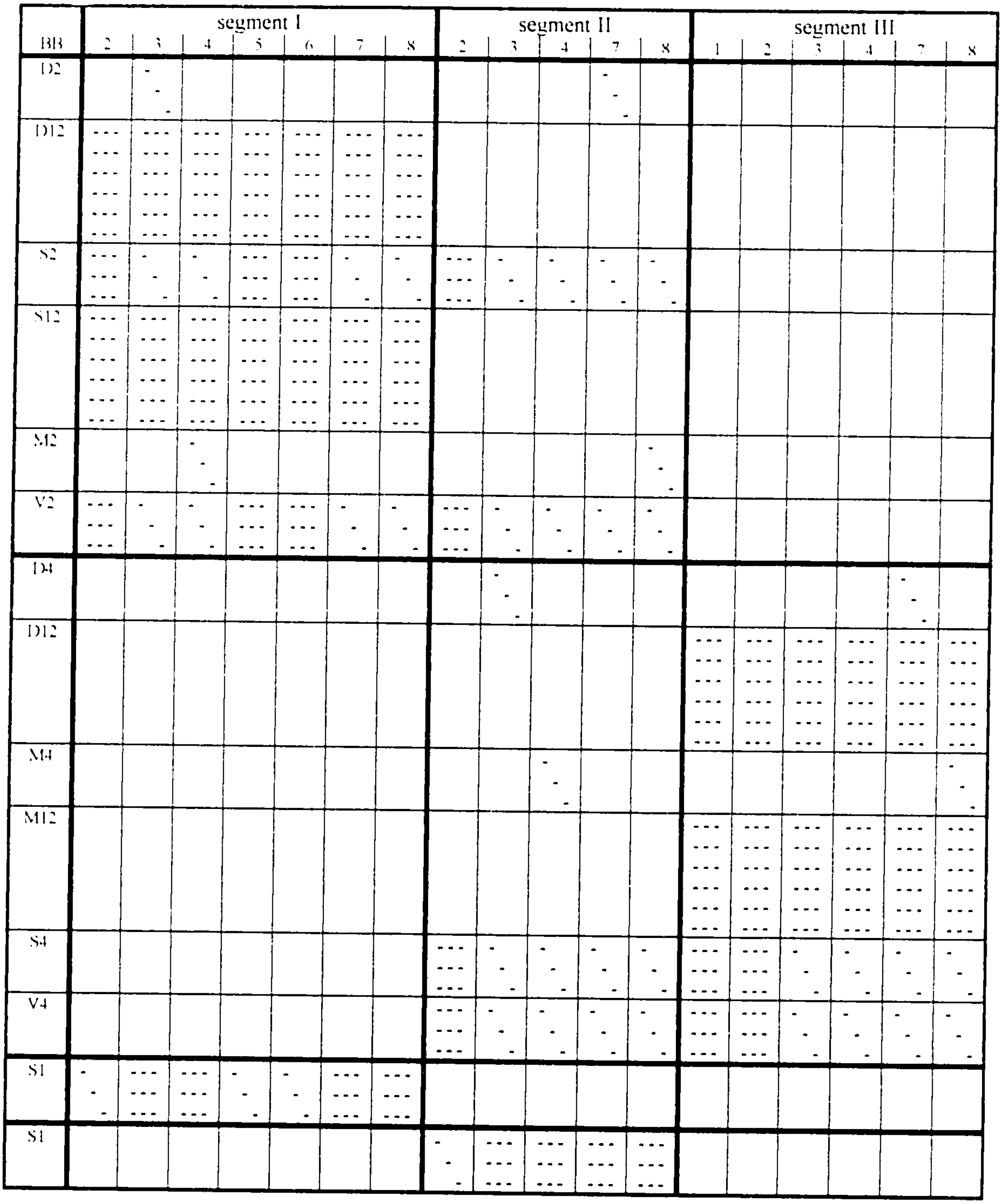


Figure 2-26 Eigenvalue matrix for the CSSC parallelogram plate, $0.5 < \phi_t < 2.0$.

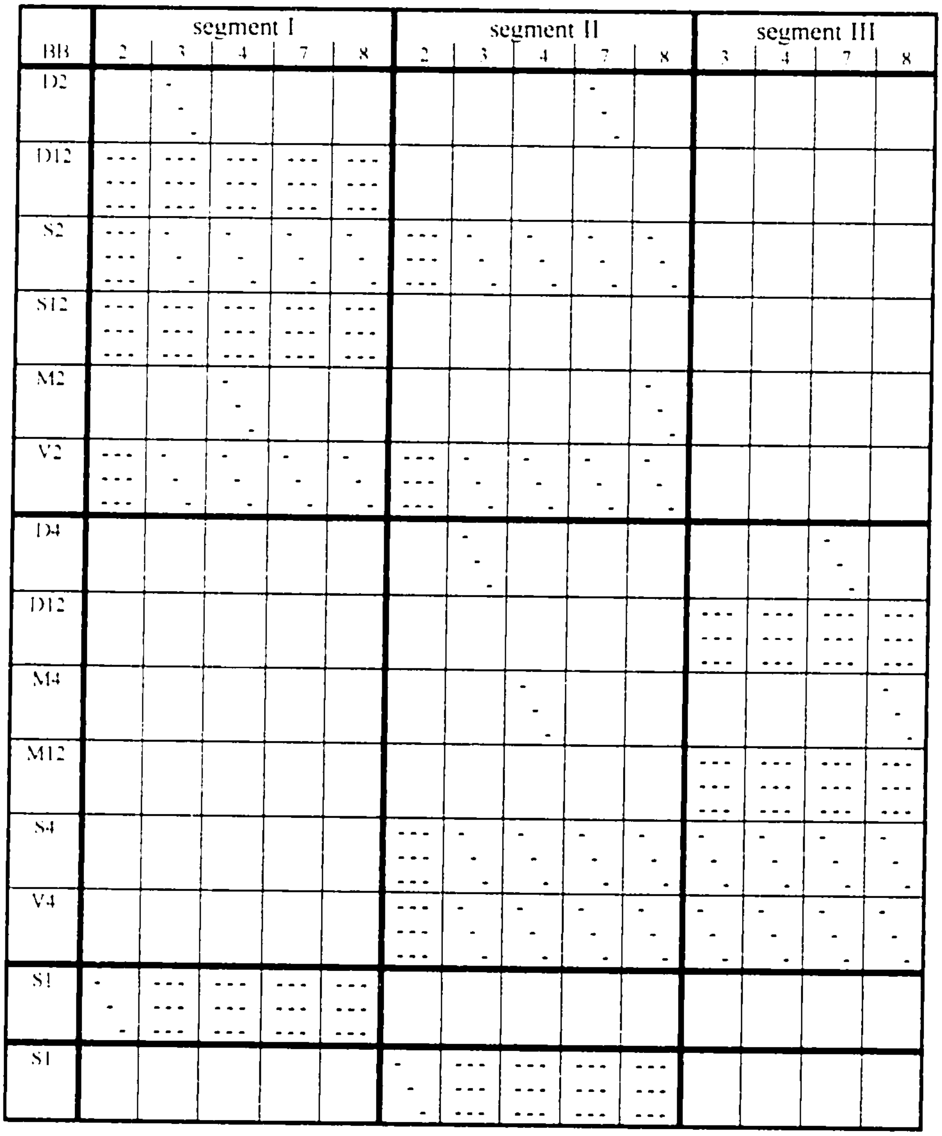


Figure 2-27 Eigenvalue matrix for the CSSC parallelogram plate, $\phi_t \geq 2.0$.

9.6 Eigenvalue Matrices for the CCSC Parallelogram Plate

The CCSC plate is clamped on three edges, with the top edge simply supported. Figures 2-28, 2-29 and 2-30 show the eigenvalue matrices for this plate.

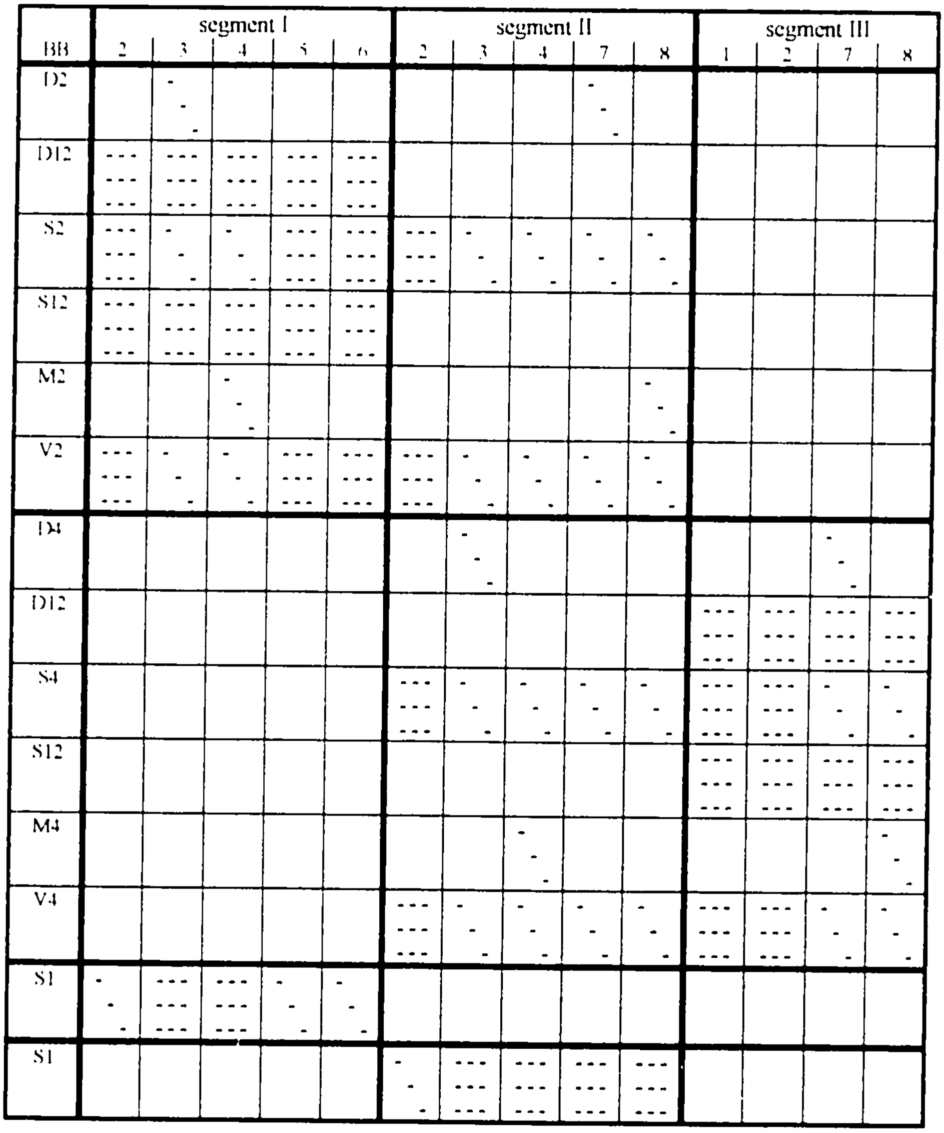


Figure 2-28 Eigenvalue matrix for the CCSC parallelogram plate, $\phi_t \leq 0.5$.

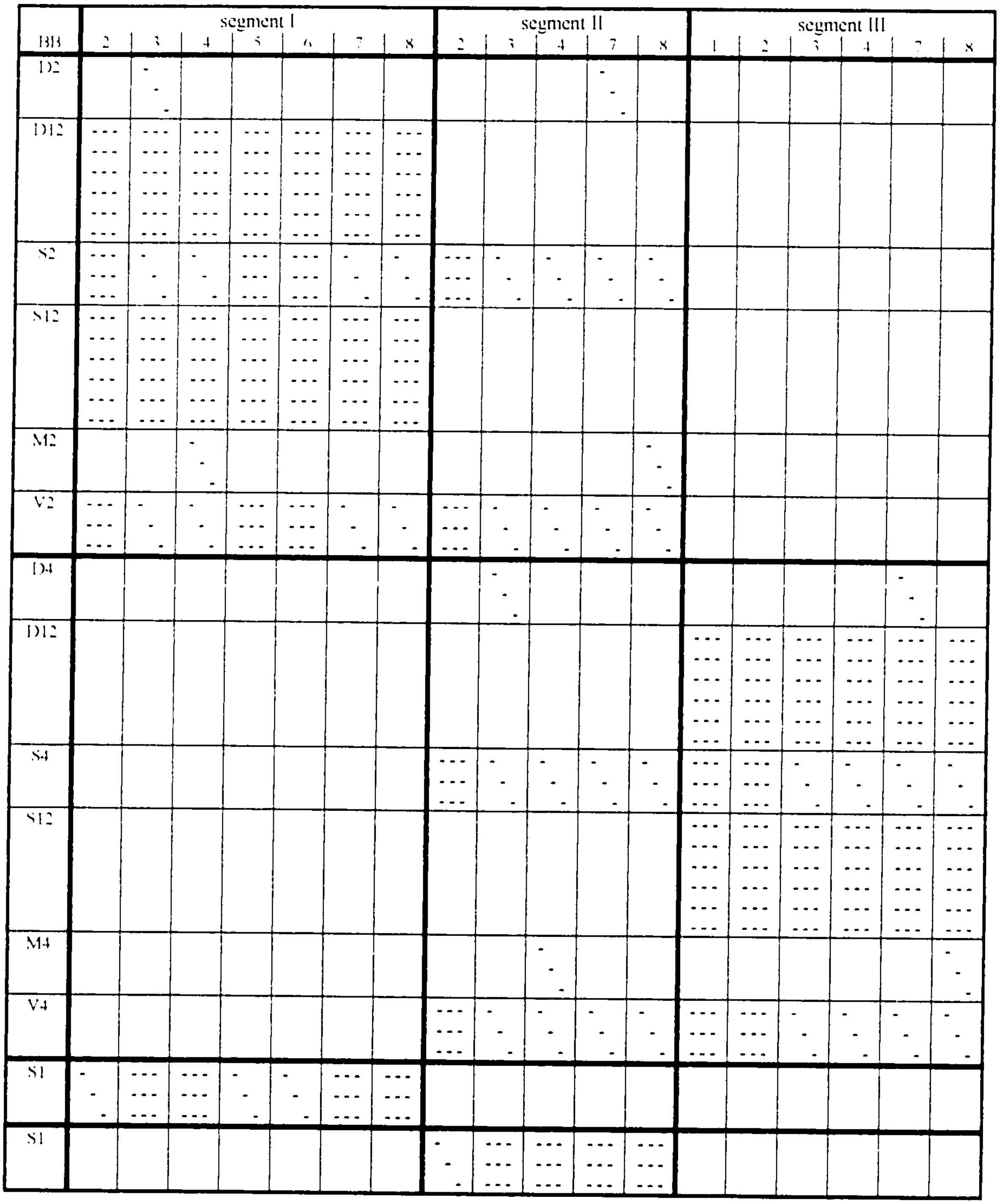


Figure 2-29. Eigenvalue matrix for the CCSC parallelogram plate, $0.5 < \phi_t < 2.0$.

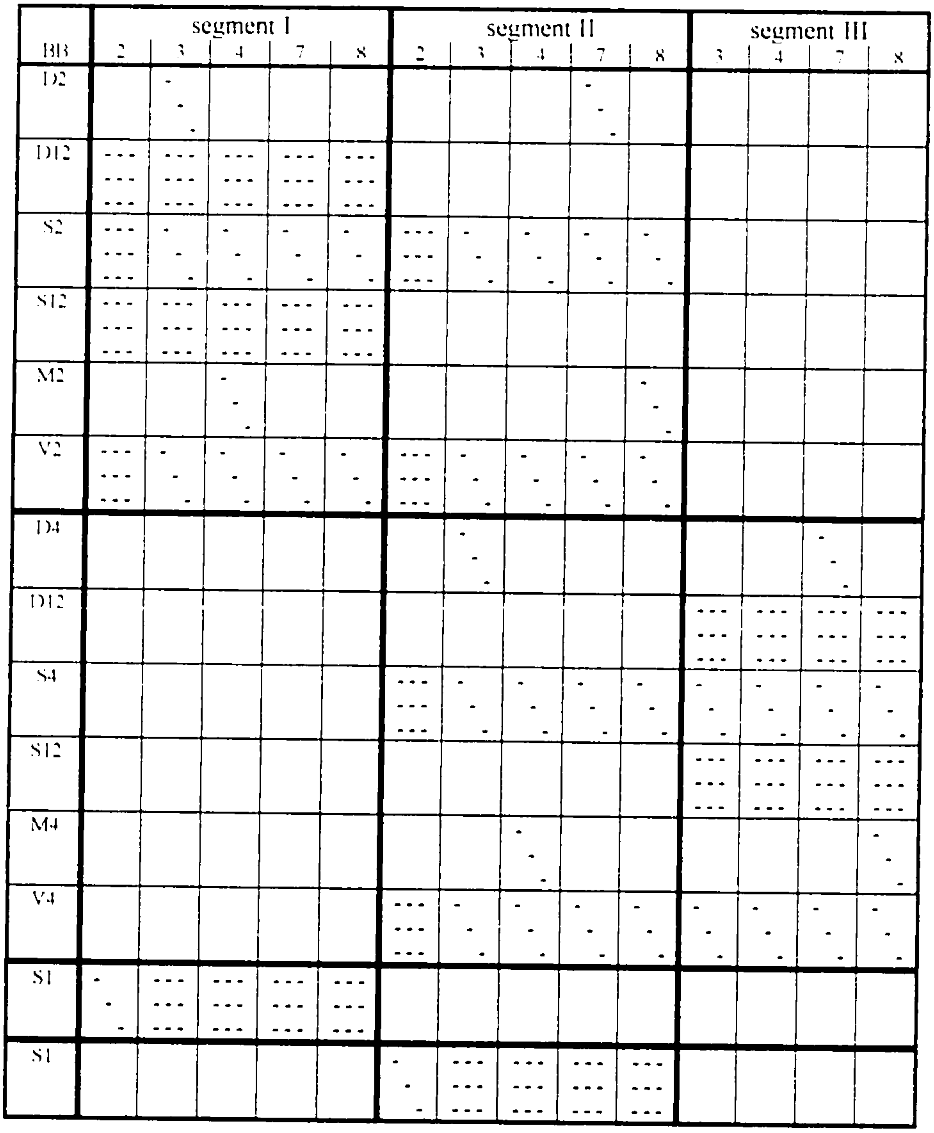


Figure 2-30 Eigenvalue matrix for the CCSC parallelogram plate, $\phi_t \geq 2.0$.

9.7 Eigenvalue Matrices for the CSCS Parallelogram Plate

Like the SCSC plate, the CSCS plate has two opposite edges clamped. However, the CSCS case requires four additional building blocks and four additional boundary conditions to produce the clamped support along the upper and lower edges. The resulting eigenvalue matrices are shown in Figures 2-31, 2-32 and 2-33.

	segment I					segment II						segment III				
BB	2	3	4	5	6	2	3	4	6	7	8	1	2	6	7	8
D2		⋱								⋱						
D 12	...	...	...	...	...											
M2			⋱								⋱					
M 12	...	...	...	...	...											
S2	...	⋱	⋱	...	...	...	⋱	⋱	...	⋱	⋱					
V2	...	⋱	⋱	...	...	...	⋱	⋱	...	⋱	⋱					
D4							⋱								⋱	
D 12												...	...	...	...	...
M4								⋱								⋱
M 12												...	...	...	...	...
S4						...	⋱	⋱	...	⋱	⋱	...	...	...	⋱	⋱
V4						...	⋱	⋱	...	⋱	⋱	...	...	...	⋱	⋱
S1 I	⋱	...	...	⋱	⋱											
S1 II						⋱	...	...	⋱	...	...					
S3 II						⋱	...	...	⋱	...	...					
S3 III												⋱	⋱	⋱	...	...

Figure2-31 Eigenvalue matrix for the CSCS parallelogram plate, $\phi_t \leq 0.5$.

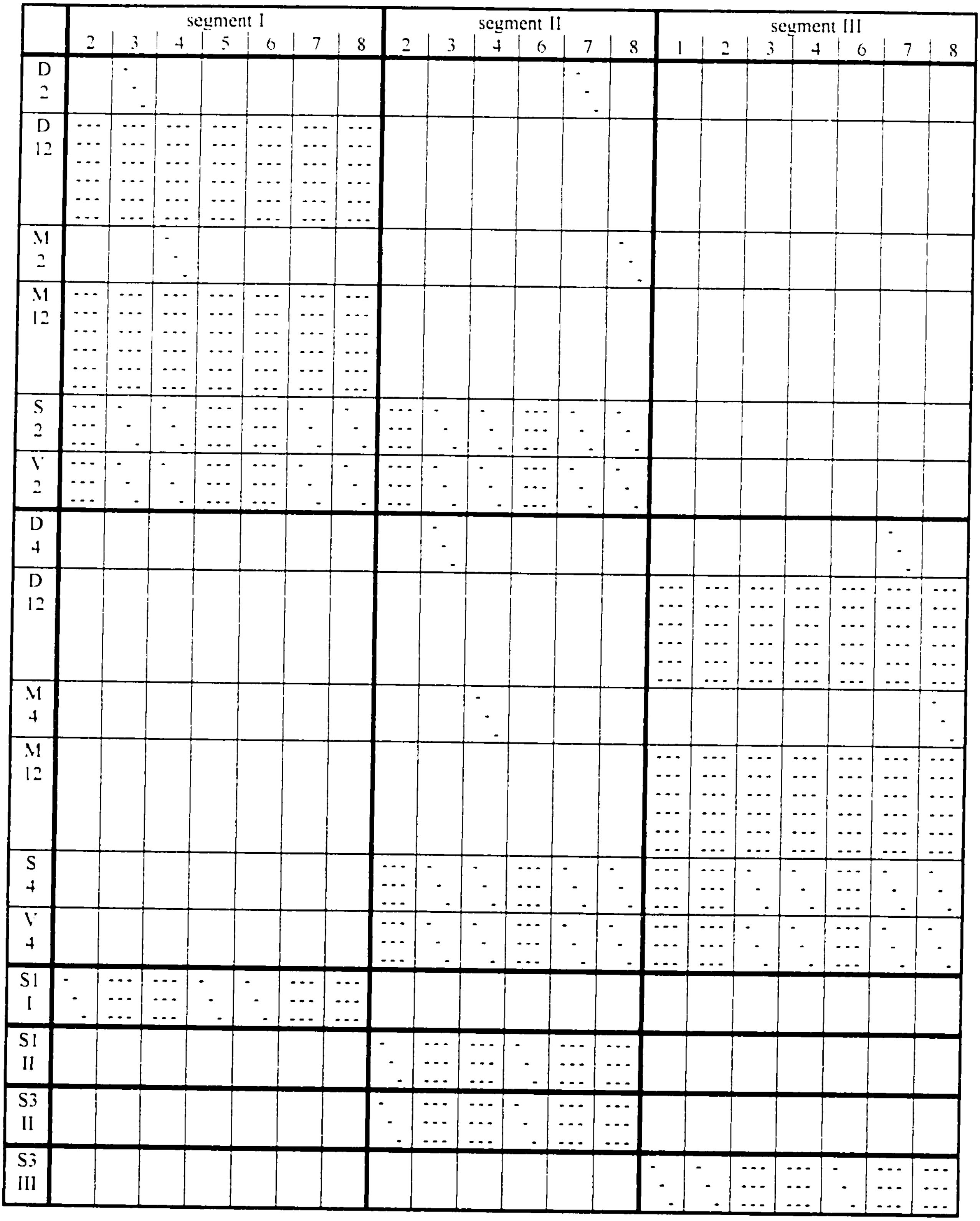


Figure 2-32 Eigenvalue matrix for the CSCS parallelogram plate, 0.5 < ϕ_t < 2.0.

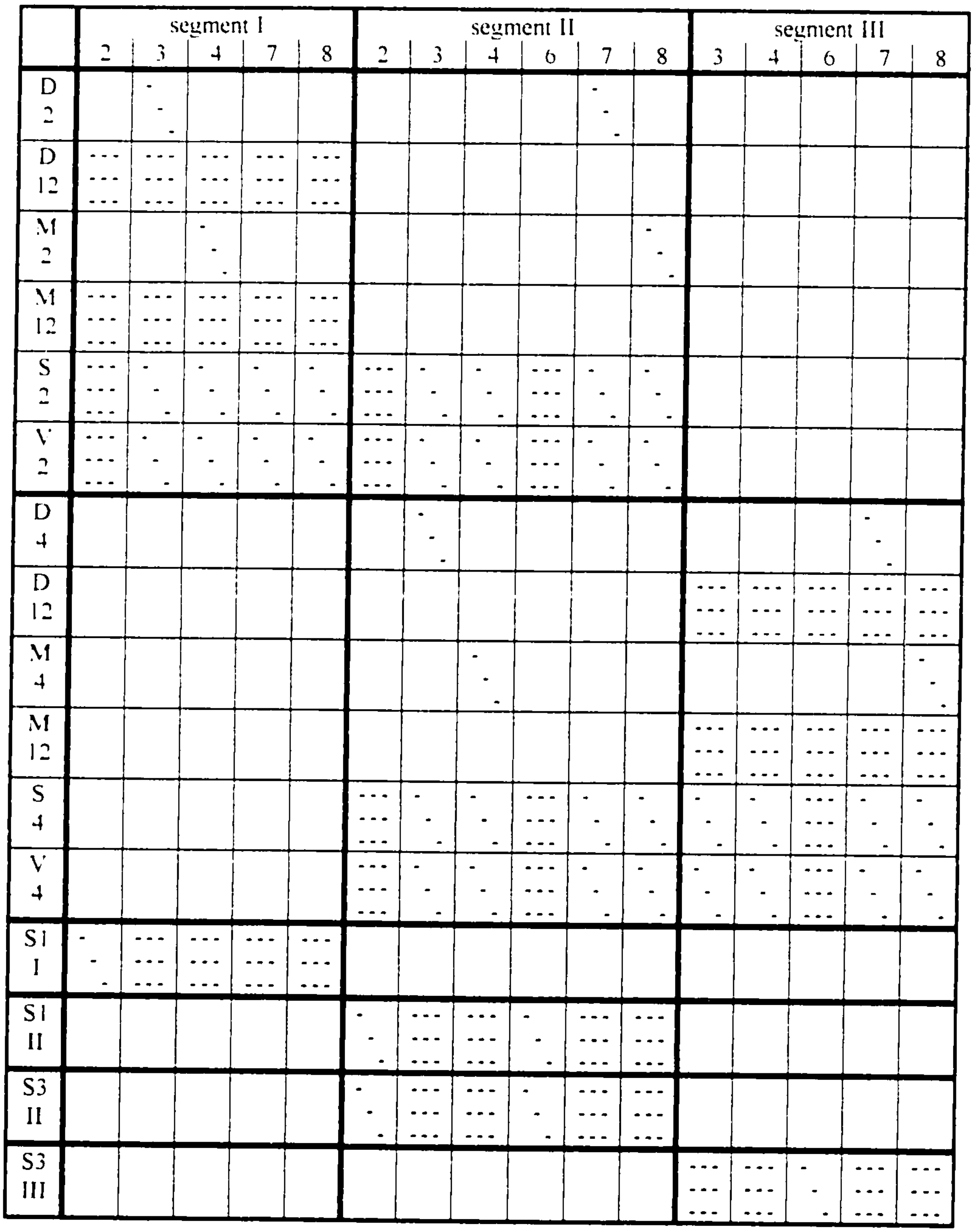


Figure 2-33 Eigenvalue matrix for the CSCS parallelogram plate, $\phi_t \geq 2.0$.

9.8 Eigenvalue Matrices for the CSCC Parallelogram Plate

Figures 2-34, 2-35 and 2-36 show the eigenvalue matrices for CSCC parallelogram plates.

	segment I					segment II						segment III				
BB	2	3	4	5	6	2	3	4	6	7	8	1	2	6	7	8
D 2		- - -								- - -						
D 12	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -											
S 2	- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - -	- - -					
S 12	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -											
M 2			- - -								- - -					
V 2	- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - -	- - -					
D 4							- - -								- - -	
D 12												- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -
M 4								- - -								- - -
M 12												- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -
S 4						- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - -	- - -
V 4						- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - -	- - -
S1 I	- - -	- - - - - - - - -	- - - - - - - - -	- - -	- - -											
S1 II						- - -	- - - - - - - - -	- - - - - - - - -	- - -	- - - - - - - - -	- - - - - - - - -					
S3 II						- - -	- - - - - - - - -	- - - - - - - - -	- - -	- - - - - - - - -	- - - - - - - - -					
S3 III												- - -	- - -	- - -	- - - - - - - - -	- - - - - - - - -

Figure 2-34 Eigenvalue matrix for the CSCC parallelogram plate, $\phi_t \leq 0.5$.

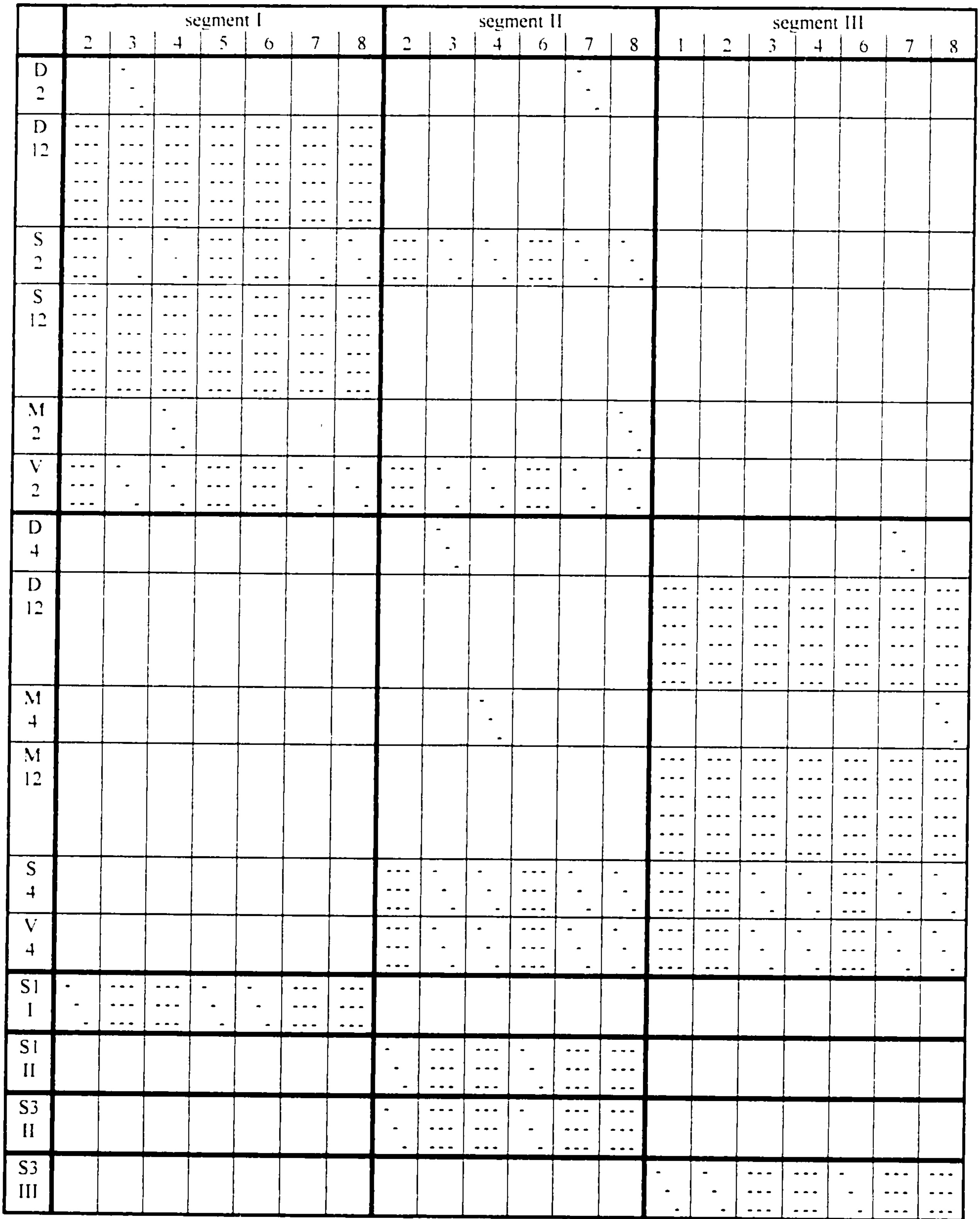


Figure 2-35 Eigenvalue matrix for the CSCC parallelogram plate, $0.5 < \phi_t < 2.0$.

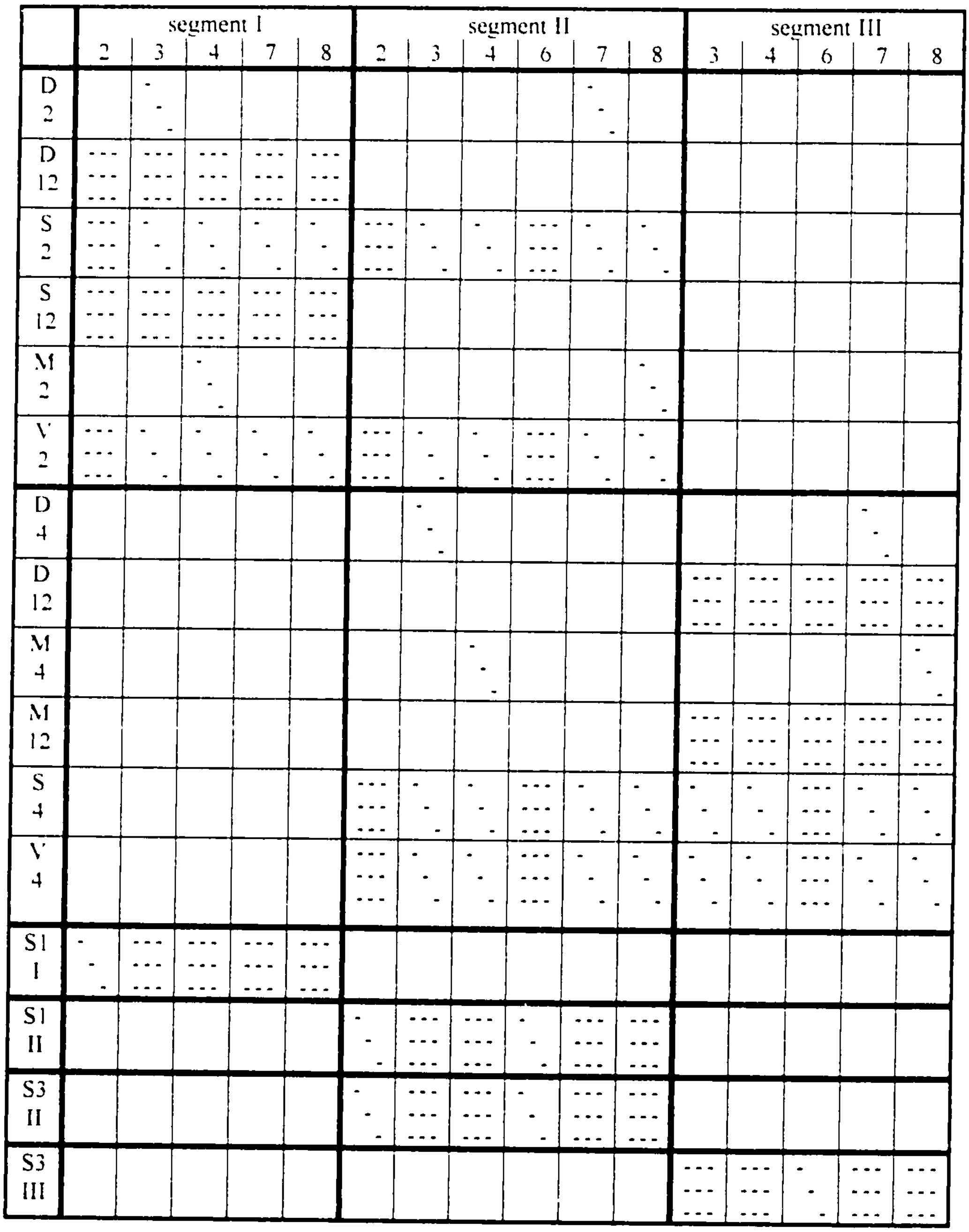


Figure 2-36 Eigenvalue matrix for the CSCC parallelogram plate, $\phi_t \geq 2.0$.

9.9 Eigenvalue Matrices for the CCCC Parallelogram Plate

The eigenvalue matrices for fully clamped parallelogram plates are shown in Figures 2-37, 2-38 and 2-39.

	segment I					segment II						segment III				
	2	3	4	5	6	2	3	4	6	7	8	1	2	6	7	8
D 2		- - -								- - -						
D 12	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -											
S 2	- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - -	- - -					
S 12	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -											
M 2			- - -								- - -					
V 2	- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - -	- - -					
D 4							- - -								- - -	
D 12												- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -
S 4						- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - -	- - -
S 12												- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -
M 4								- - -								- - -
V 4						- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - -	- - -	- - - - - - - - -	- - - - - - - - -	- - - - - - - - -	- - -	- - -
S1 I	- - -	- - - - - - - - -	- - - - - - - - -	- - -	- - -											
S1 II						- - -	- - - - - - - - -	- - - - - - - - -	- - -	- - - - - - - - -	- - - - - - - - -					
S3 II						- - -	- - - - - - - - -	- - - - - - - - -	- - -	- - - - - - - - -	- - - - - - - - -					
S3 III												- - -	- - -	- - -	- - - - - - - - -	- - - - - - - - -

Figure 2-37 Eigenvalue matrix for the CCCC parallelogram plate, $\phi_t \leq 0.5$.

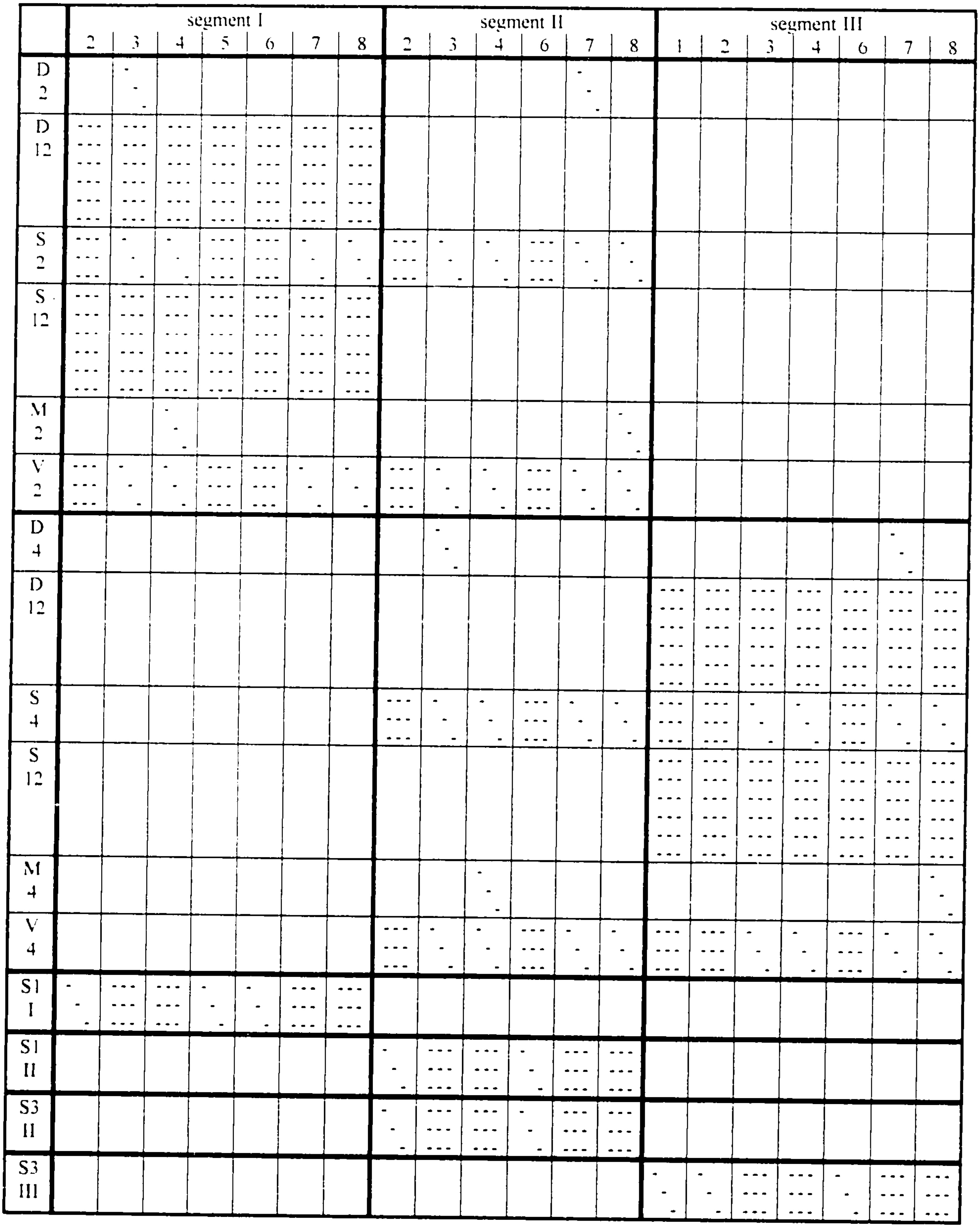


Figure 2-38 Eigenvalue matrix for the CCCC parallelogram plate, 0.5 < ϕ_t < 2.0.

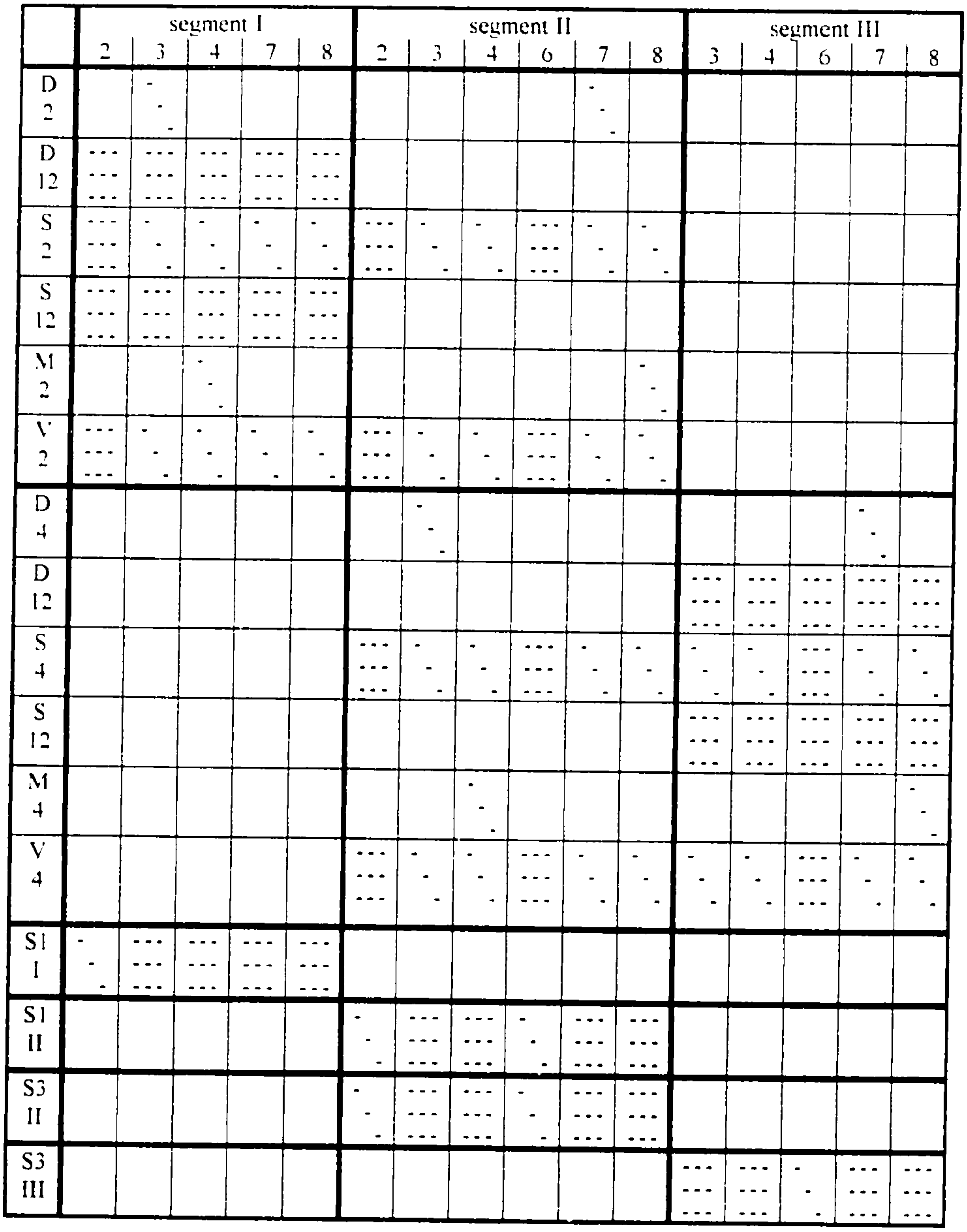


Figure 2-39 Eigenvalue matrix for the CCCC parallelogram plate, $\phi_t \geq 2.0$.

3 - Theoretical Results

Numerical results were obtained for a variety of edge combinations and plate sizes. Only a selection can be included in this limited space to demonstrate the validity of the method and to illustrate some observations. Comparison is made with some previously published data, and new data is presented as well.

1. Simply Supported Plates

Tables 3-1 to 3-5 offer a range of eigenvalues for SSSS parallelogram plates in a comparison with results that were obtained by Gorman [25] in a superposition solution that is different from the present work.

Table 3-1 Eigenvalues for the SSSS plate, ϕ_t = 2.50.

mode / ϕ_r	1.25	0.8333	0.625	0.5	0.4167	0.3572	0.3125
Present Work:							
1	2.771	2.238	1.996	1.866	1.789	1.739	1.706
2	6.031	4.166	3.232	2.722	2.415	2.216	2.081
3	7.811	7.009	5.230	4.126	3.450	3.008	2.706
4	9.925	7.103	6.716	6.013	4.874	4.106	3.576
5	13.15	9.282	7.899	6.641	6.532	5.490	4.681
6	14.80	10.63	8.028	7.414	6.602	6.472	6.000
7	15.63	13.37	10.30	8.503	7.248	6.972	6.452
8	19.15	14.50	11.02	8.911	8.084	7.101	6.810
Gorman [25]:							
1	2.768	2.235	1.995	1.866	1.789	1.738	1.704
2	6.029	4.163	3.229	2.720	2.413	2.216	2.080
3	7.806	7.008	5.227	4.125	3.448	3.006	2.704
4	9.926	7.102	6.715	6.011	4.872	4.104	3.574
5	13.15	9.278	7.898	6.640	6.531	5.490	4.696
6	14.80	10.63	8.027	7.413	6.602	6.472	5.998
7	15.62	13.37	10.296	8.501	7.246	6.971	6.451
8	19.14	14.50	11.02	8.910	8.083	7.101	6.810

Table 3-2 Eigenvalues for the SSSS plate, ϕ_t = 1.667.

Present Work:

mode / ϕ_r	1.25	0.8333	0.625	0.5	0.4167	0.3572	0.3125
1	5.661	4.789	4.363	4.124	3.977	3.880	3.813
2	11.42	8.389	6.763	5.828	5.246	4.861	4.594
3	16.89	13.76	10.59	8.606	7.336	6.484	5.889
4	18.51	15.50	14.72	12.28	10.18	8.721	7.685
5	26.11	19.90	15.87	14.86	13.60	11.51	9.950
6	26.96	20.23	17.48	15.93	14.65	14.41	12.62
7	34.21	27.52	21.51	17.56	16.11	14.85	14.49
8	37.98	27.60	22.04	19.19	17.09	15.56	15.09
Gorman [25]:							
1	5.670	4.795	4.363	4.126	3.978	3.881	3.812
2	11.42	8.395	6.768	5.832	5.249	4.864	4.594
3	16.92	13.76	10.59	8.611	7.340	6.487	5.890
4	18.52	15.51	14.72	12.28	10.18	8.723	7.686
5	26.13	19.93	15.87	14.86	13.60	11.51	9.950
6	26.96	20.23	17.50	15.93	14.65	14.41	12.62
7	26.96	27.52	21.51	17.57	16.12	14.85	14.49
8	37.98	27.62	22.06	19.21	17.10	15.56	14.99

Table 3-3 Eigenvalues for the SSSS plate, ϕ_t = 1.250.

Present Work:

mode / ϕ_r	1.25	0.8333	0.625	0.5	0.4167	0.3572	0.3125
1	9.418	8.214	7.594	7.234	7.007	6.855	6.749
2	17.85	13.72	11.38	9.972	9.075	8.470	8.044
3	28.27	21.77	17.32	14.40	12.47	11.14	10.19
4	29.27	27.07	24.35	20.17	17.03	14.19	13.16
5	40.40	31.59	27.14	26.32	22.44	19.27	16.86
6	42.83	34.07	29.95	26.35	25.94	24.24	21.17
7	53.84	42.29	34.48	30.13	28.27	25.95	25.67
8	60.70	45.94	37.40	32.36	28.40	27.40	25.74
Gorman [25]:							
1	9.440	8.230	7.603	7.238	7.008	6.854	6.752
2	17.86	13.74	11.39	9.984	9.088	8.480	8.045
3	28.27	21.77	17.32	14.41	12.48	11.15	10.20
4	29.32	27.09	24.35	20.17	17.04	14.80	13.16
5	40.40	31.58	27.14	26.32	22.44	19.27	16.86
6	42.87	34.11	29.98	26.36	25.94	24.24	21.17
7	53.84	42.29	34.48	30.14	28.27	25.96	25.67
8	60.70	45.96	37.43	32.38	28.41	27.41	25.74

Table 3-4 Eigenvalues for the SSSS plate, ϕ_t = 1.000.

Present Work:

mode / ϕ_r	1.25	0.8333	0.625	0.5	0.4167	0.3572	0.3125
1	14.03	12.50	11.68	11.19	10.87	10.66	10.51
2	25.29	20.10	17.02	15.12	13.87	13.02	12.42
3	39.04	31.00	25.30	21.43	18.78	16.92	15.58
4	44.80	40.95	35.08	29.50	25.30	22.21	19.92
5	54.98	45.21	41.98	38.25	32.90	28.62	25.30
6	63.12	51.23	44.32	41.00	40.03	35.68	31.46
7	72.51	59.03	50.64	46.32	41.61	40.45	37.99
8	86.84	67.53	55.31	47.36	44.04	42.01	40.18
Gorman [25]:							
1	14.07	12.52	11.69	11.19	10.88	10.66	10.51
2	25.31	20.13	17.05	15.14	13.89	13.04	12.43
3	39.04	31.00	25.31	21.45	18.80	16.94	15.59
4	44.87	40.98	35.08	39.51	25.31	22.23	19.94
5	54.98	45.22	41.99	38.25	32.90	28.62	25.30
6	63.19	51.30	44.35	41.01	40.04	35.68	31.46
7	72.51	59.03	50.65	46.35	41.61	40.45	37.90
8	86.84	67.56	55.36	47.38	44.07	42.03	40.18

Table 3-5 Eigenvalues for the SSSS plate, ϕ_t = 0.8333.

Present Work:

mode / ϕ_r	1.25	0.8333	0.625	0.5	0.4167	0.3572	0.3125
1	19.49	17.63	16.60	15.97	15.57	15.28	15.08
2	33.71	27.49	23.66	21.24	19.62	18.50	17.70
3	50.84	41.38	34.48	29.63	26.22	23.80	22.02
4	63.09	55.42	47.06	40.18	34.90	30.93	27.93
5	71.06	62.71	59.71	51.54	44.85	39.44	35.17
6	86.32	70.12	60.13	58.84	55.08	48.72	43.36
7	92.67	78.83	70.68	63.82	58.80	58.08	52.02
8	113.3	90.30	74.47	66.02	62.89	58.17	57.79
Gorman [25]:							
1	19.54	17.67	16.62	15.98	15.57	15.29	15.09
2	33.74	27.53	23.70	21.27	19.66	18.52	17.71
3	50.83	41.39	34.49	29.65	26.25	23.83	22.05
4	63.20	55.45	47.06	40.19	34.91	30.95	27.95
5	71.06	62.74	59.75	51.55	44.84	39.44	35.18
6	86.41	70.21	60.15	58.87	55.09	48.73	43.36
7	92.69	78.85	70.72	63.84	58.81	58.10	52.01
8	113.2	90.33	74.52	66.07	62.94	58.18	57.79

2. Clamped Plates

In Table 3-5 results obtained by the method presented in this work are compared to previously published results. The results by Durvasula [8] were obtained with the Galerkin method, while those by Lee and Pon [21] were obtained with characteristic orthogonal polynomials in the Rayleigh-Ritz method.

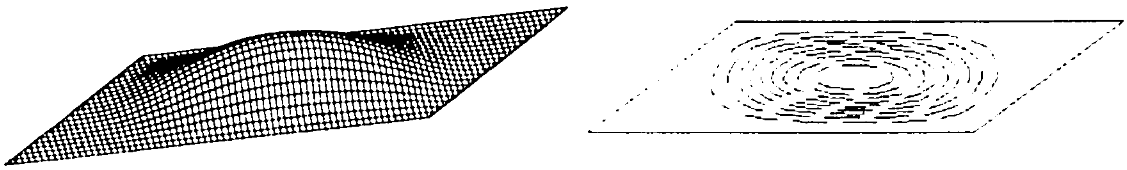
Table 3-6 Eigenvalues for the CCCC rhombic plate. $\lambda^2 = \omega a_r^2 \sqrt{\rho / D}$.

mode	source	skew angle 15°	30°	45°
1	present work	2.558	11.52	32.83
	Galerkin [8]	2.559	11.54	32.96
	Rayleigh-Ritz [21]	2.558	11.52	32.85
2	present work	4.886	20.40	53.26
	Galerkin [8]	4.884	20.42	53.30
	Rayleigh-Ritz [21]	4.883	20.44	53.25
3	present work	5.534	26.29	74.16
	Galerkin [8]	5.538	26.38	74.52
	Rayleigh-Ritz [21]	5.534	26.30	74.30
4	present work	7.345	29.81	78.64
	Galerkin [8]	7.352	29.88	79.45
	Rayleigh-Ritz [21]	7.341	29.83	78.88
5	present work	9.309	41.24	98.39
	Galerkin [8]	9.315	41.43	99.68
	Rayleigh-Ritz [21]	9.326	41.37	99.75
6	present work	9.725	41.32	114.8
	Galerkin [8]	9.732	41.45	116.0
	Rayleigh-Ritz [21]	9.739	41.47	116.2

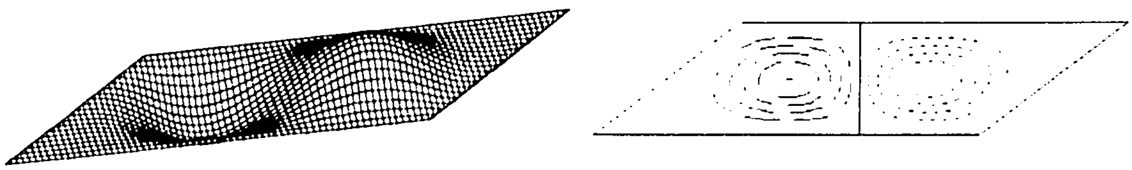
Table 3-7 contains eigenvalues for a range sizes of fully clamped parallelogram plates. The first five mode shapes for a series of these plates are shown in Figures 3-1 to 3-22 as mesh and contour plots. They demonstrate the effects of plate dimension on the vibration behaviour of skew plates.

Table 3-7 Eigenvalues $\lambda^2 = \omega a_r^2 \sqrt{\rho / D}$ for the CCCC parallelogram plate.

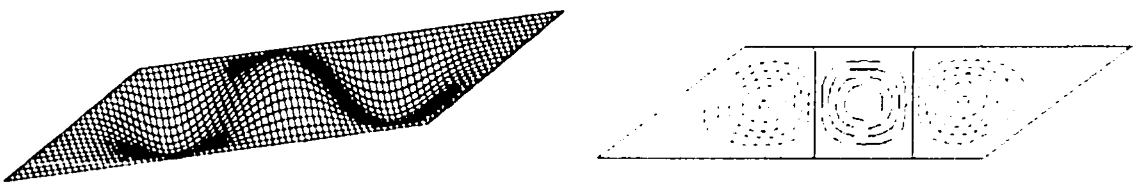
ϕ_r	0.50	0.75	1.00	1.25	1.50	2.00
ϕ_t = 0.75						
1	41.52	43.16	44.97	46.78	48.50	51.58
2	47.07	53.70	60.27	66.06	70.89	78.09
3	56.77	70.48	81.75	90.24	96.66	105.7
4	70.16	89.99	103.7	112.8	118.9	126.8
5	85.77	109.9	117.9	122.2	127.2	136.9
6	102.2	114.5	127.1	141.0	152.5	174.1
ϕ_t = 1.00						
1	23.47	24.59	25.89	27.25	28.59	31.06
2	26.98	31.55	36.34	40.73	44.48	50.09
3	33.18	42.80	51.24	57.75	62.73	69.90
4	41.87	56.06	64.05	67.06	69.54	74.56
5	52.19	64.90	69.74	77.03	83.29	92.77
6	62.13	68.28	77.91	86.62	94.76	107.7
ϕ_t = 1.25						
1	15.08	15.91	16.92	18.01	19.12	21.22
2	17.53	20.95	24.69	28.22	31.28	35.78
3	21.90	29.18	35.84	41.05	45.06	49.83
4	28.07	38.68	42.03	43.70	45.54	50.95
5	35.48	42.05	48.83	55.45	60.82	68.85
6	40.28	52.16	51.78	58.32	64.84	75.08
ϕ_t = 1.50						
1	10.51	11.16	11.98	12.89	13.83	15.68
2	12.33	15.02	18.05	20.99	23.55	27.17
3	15.60	21.35	26.80	30.75	32.29	36.24
4	20.24	28.06	29.50	31.07	34.36	39.32
5	25.84	30.02	36.82	42.33	46.99	53.84
6	28.04	32.53	36.97	42.43	47.85	55.78
ϕ_t = 3.00						
1	2.661	2.904	3.251	3.678	4.161	5.202
2	3.234	4.234	5.506	6.814	7.762	8.823
3	4.266	6.440	7.609	8.144	9.168	12.17
4	5.750	7.265	8.680	10.31	11.70	13.92
5	7.049	8.605	10.29	12.93	14.82	16.47
6	7.460	9.273	12.22	14.59	16.32	19.60



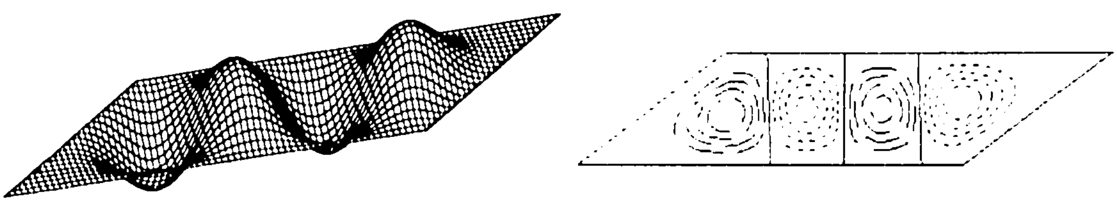
Mode 1 $\lambda^2 = 41.52$



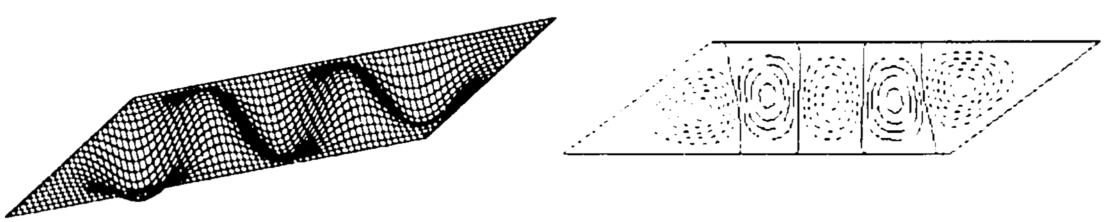
Mode 2 $\lambda^2 = 47.07$



Mode 3 $\lambda^2 = 56.77$

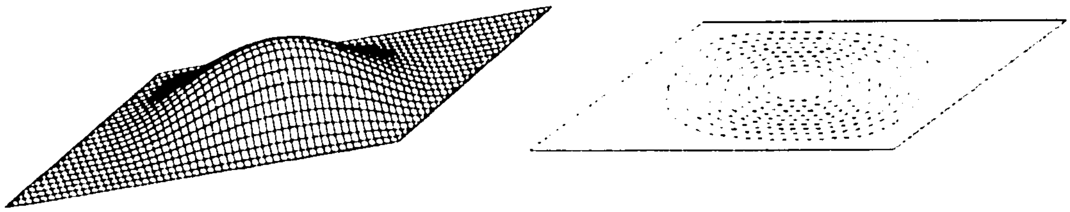


Mode 4 $\lambda^2 = 70.16$

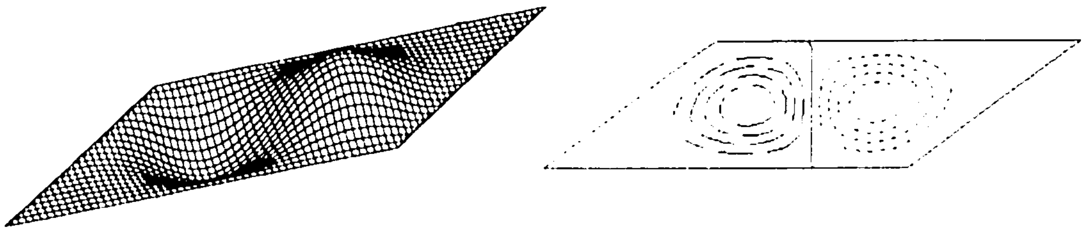


Mode 5 $\lambda^2 = 85.77$

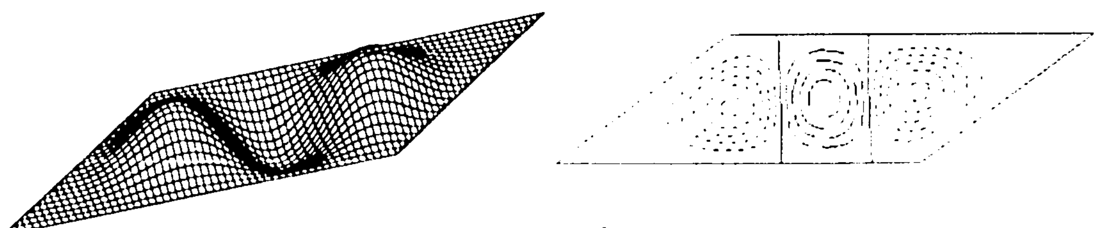
Figure 3-1 CCCC parallelogram plate, $\phi_t = 0.75$, $\phi_r = 0.50$.



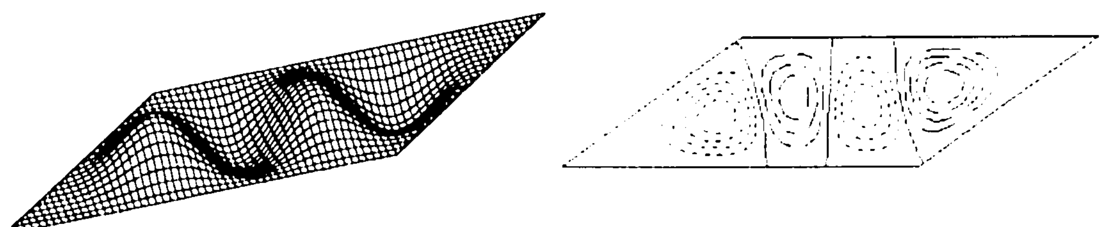
Mode 1 $\lambda^2 = 43.16$



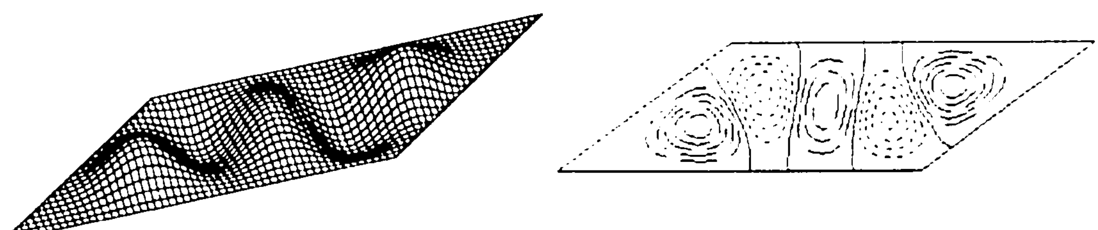
Mode 2 $\lambda^2 = 53.70$



Mode 3 $\lambda^2 = 70.44$

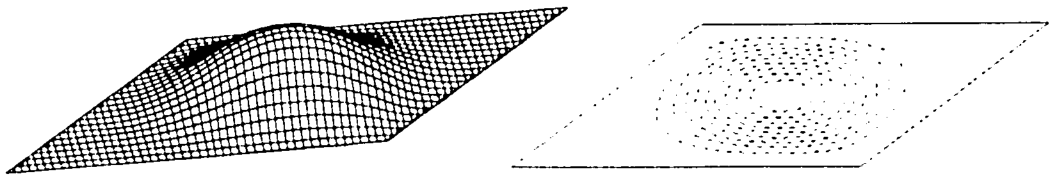


Mode 4 $\lambda^2 = 89.99$

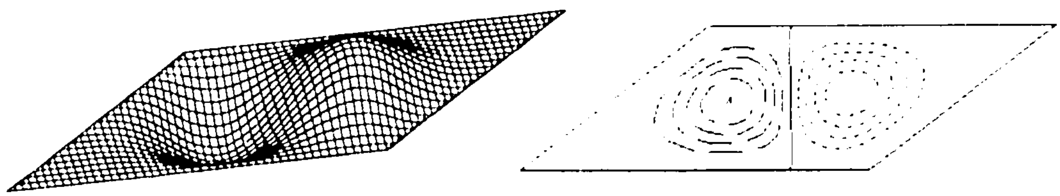


Mode 5 $\lambda^2 = 109.9$

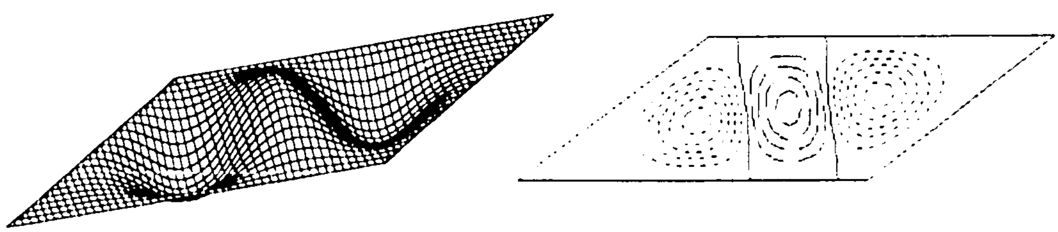
Figure 3-2 CCCC parallelogram plate, $\phi_t = 0.75$, $\phi_r = 0.75$.



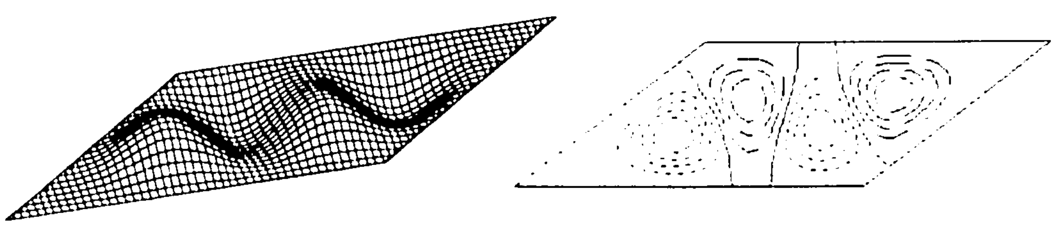
Mode 1 λ^2 = 44.97



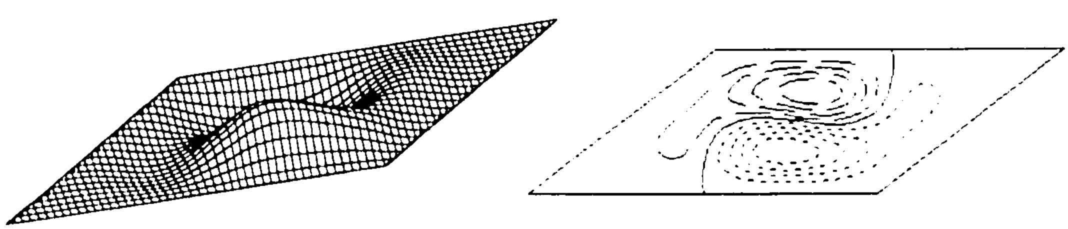
Mode 2 λ^2 = 60.57



Mode 3 λ^2 = 81.75

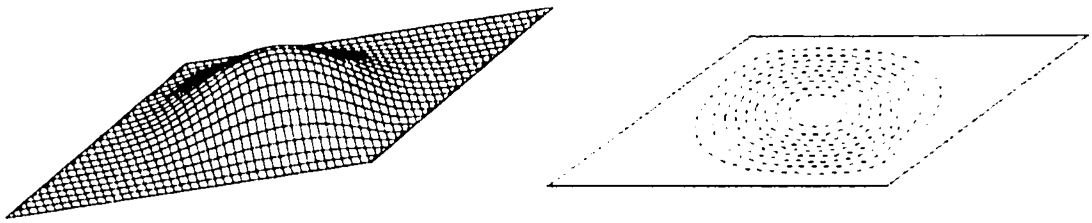


Mode 4 λ^2 = 103.7

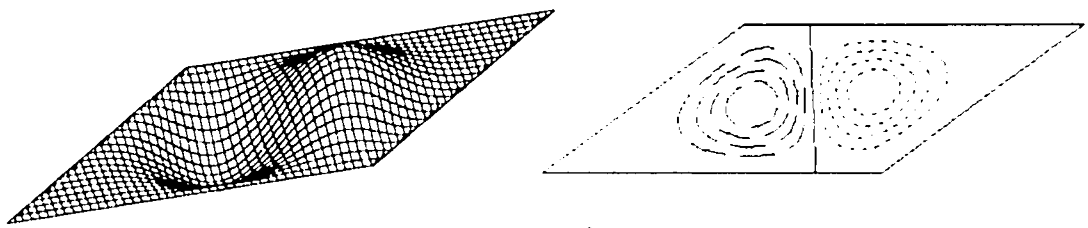


Mode 5 λ^2 = 117.9

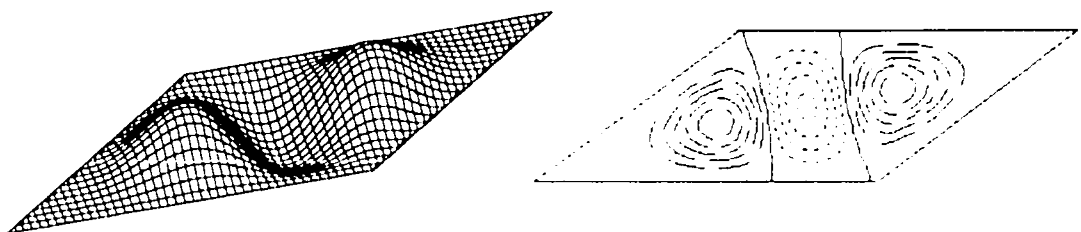
Figure 3-3 CCCC parallelogram plate, ϕ_t = 0.75, ϕ_r = 1.00.



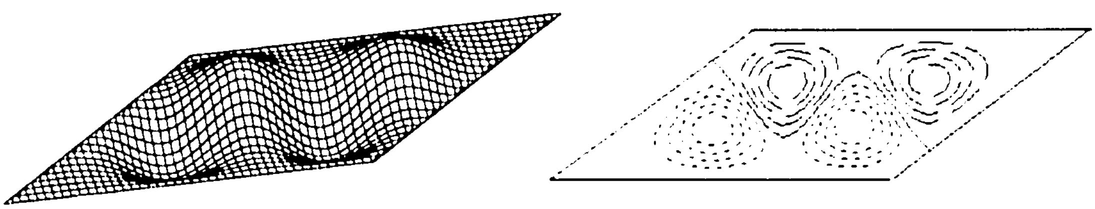
Mode 1 $\lambda^2 = 46.78$



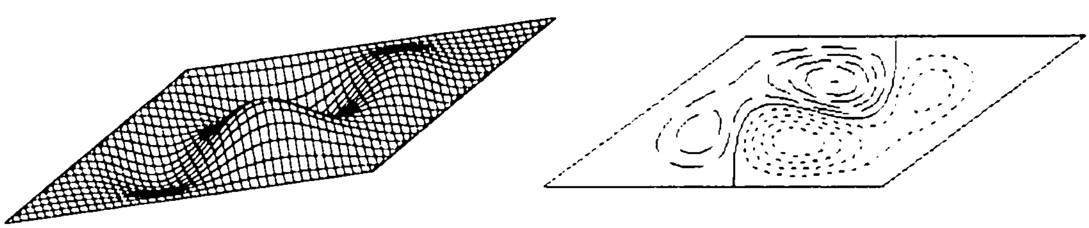
Mode 2 $\lambda^2 = 66.06$



Mode 3 $\lambda^2 = 90.24$



Mode 4 $\lambda^2 = 112.8$



Mode 5 $\lambda^2 = 122.2$

Figure 3-4 CCCC 0.75parallelogram plate, $\phi_t = 0.75$, $\phi_r = 1.25$.

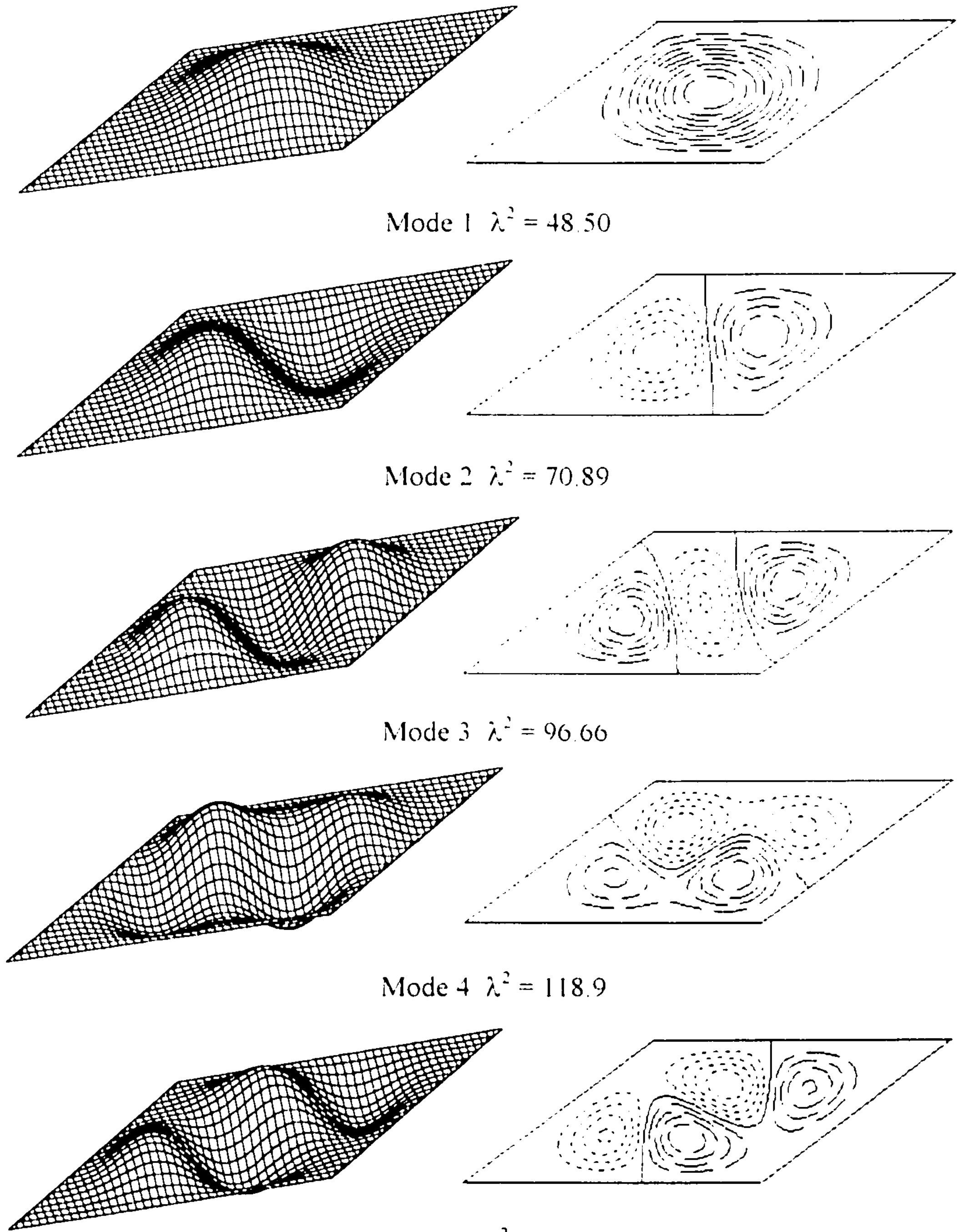
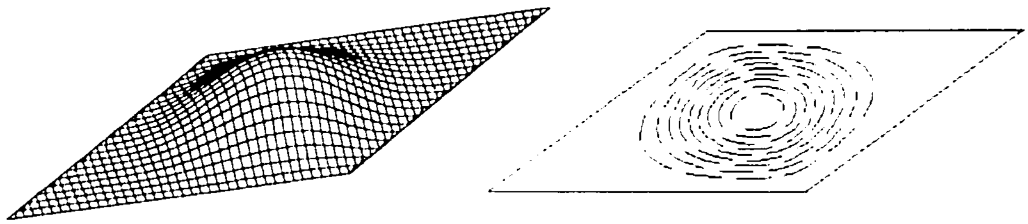
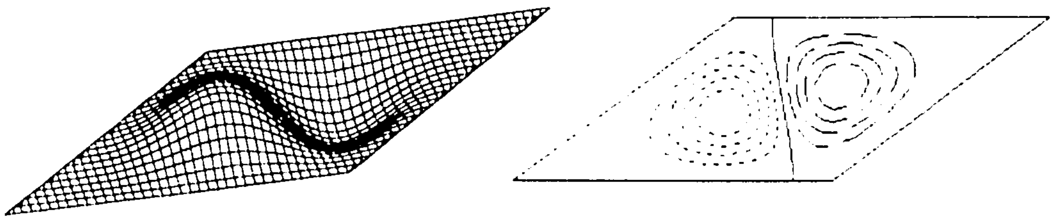


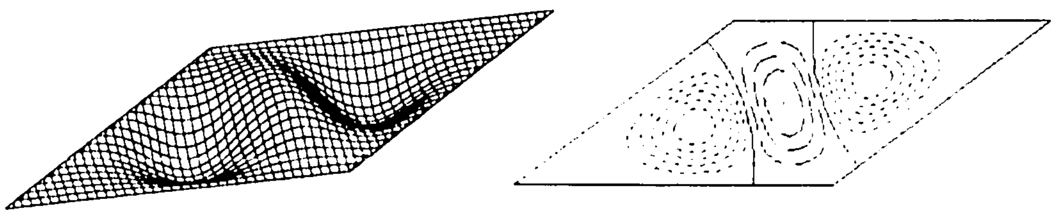
Figure 3-5 CCCC parallelogram plate, $\phi_t = 0.75$, $\phi_r = 1.50$.



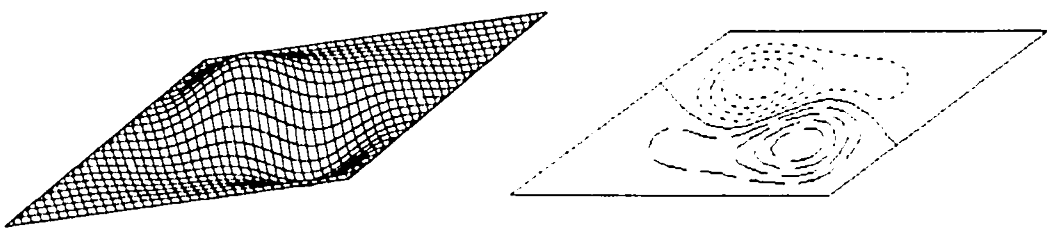
Mode 1 $\lambda^2 = 51.58$



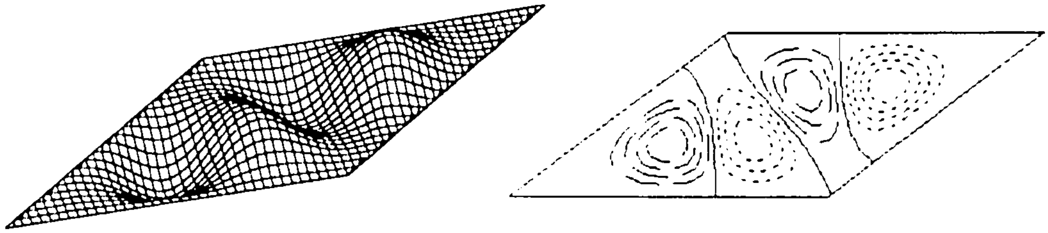
Mode 2 $\lambda^2 = 78.09$



Mode 3 $\lambda^2 = 105.7$

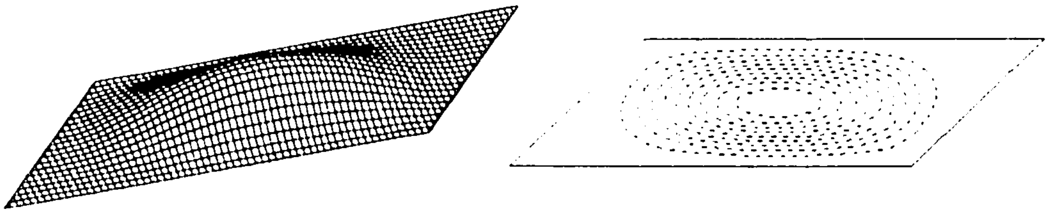


Mode 4 $\lambda^2 = 126.8$

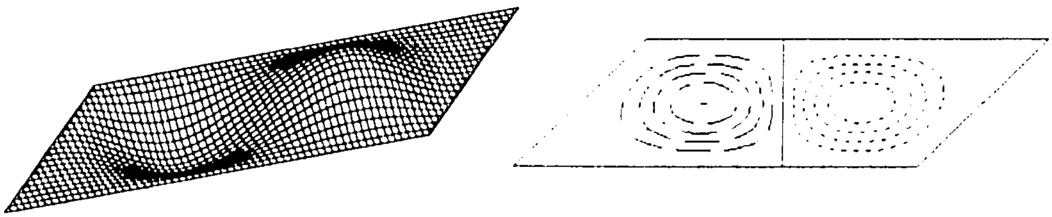


Mode 5 $\lambda^2 = 136.9$

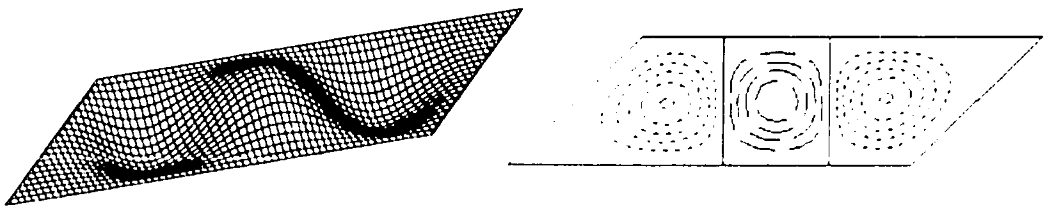
Figure 3-6 CCCC parallelogram plate, $\phi_t = 0.75$, $\phi_r = 2.00$.



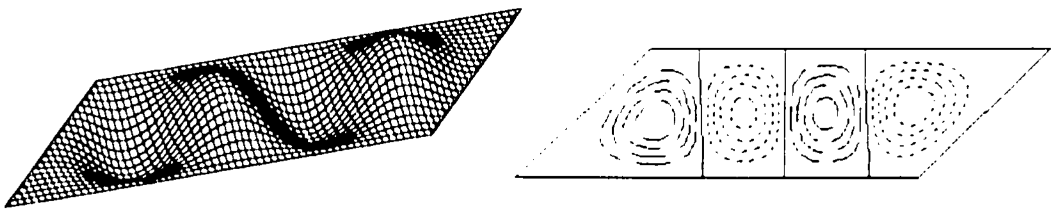
Mode 1 λ^2 = 23.47



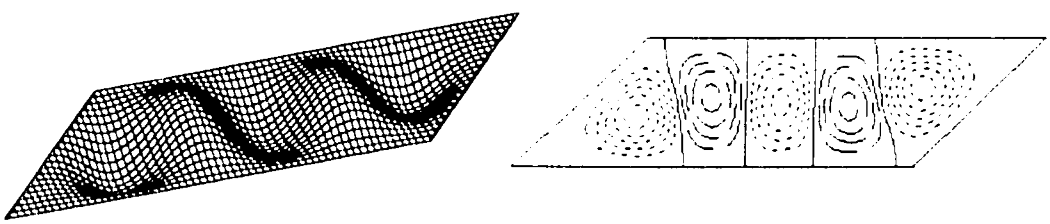
Mode 2 λ^2 = 26.98



Mode 3 λ^2 = 33.18



Mode 4 λ^2 = 41.87



Mode 5 λ^2 = 52.19

Figure 3-7 CCCC parallelogram plate, ϕ_t = 1.00, ϕ_r = 0.50.

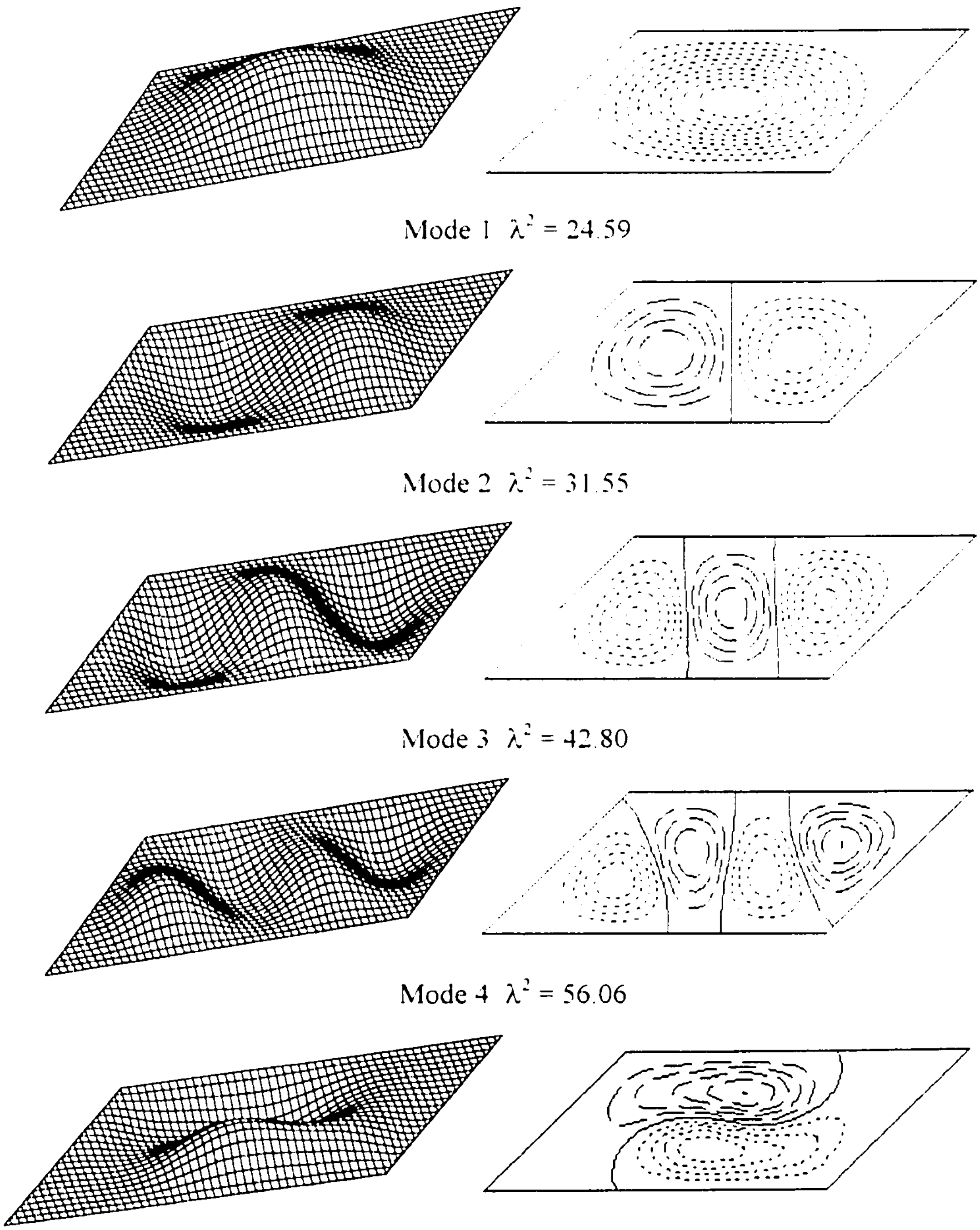
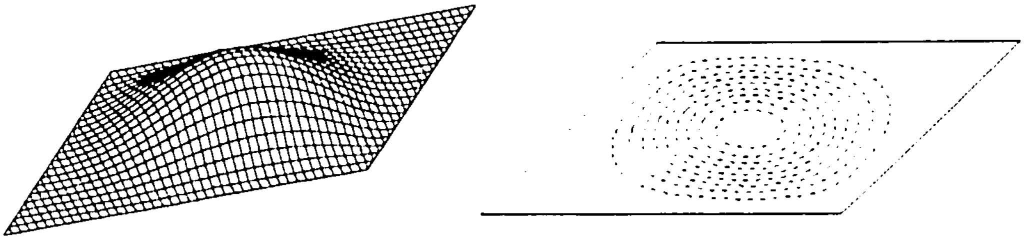
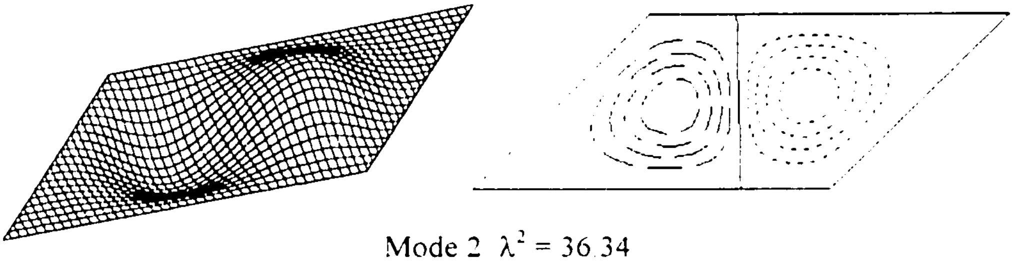


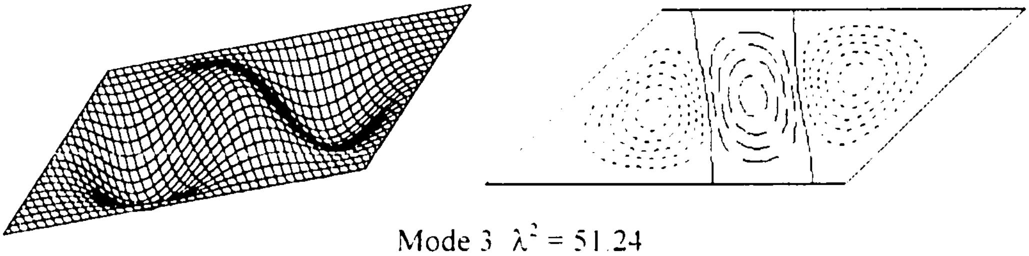
Figure 3-8 CCCC parallelogram plate, $\phi_t = 1.00$, $\phi_r = 0.75$.



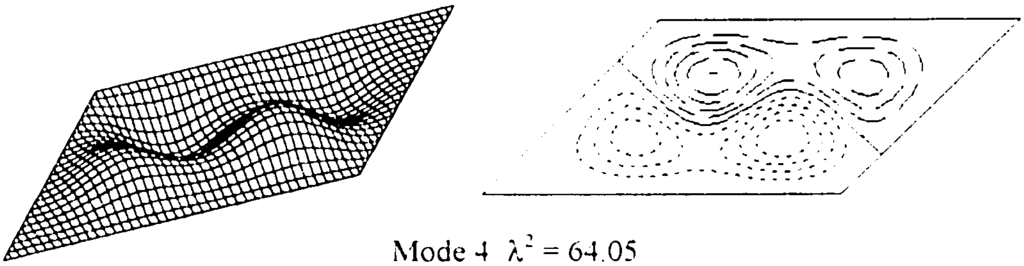
Mode 1 $\lambda^2 = 25.89$



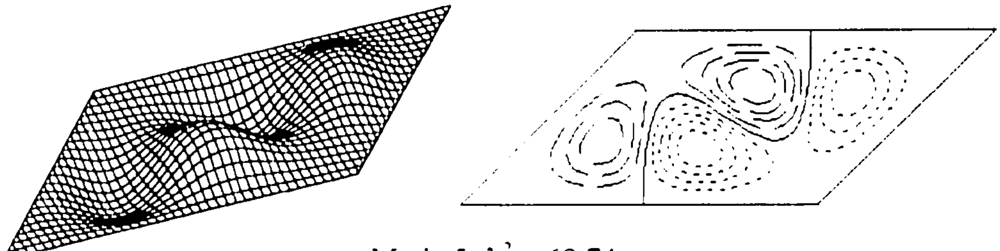
Mode 2 $\lambda^2 = 36.34$



Mode 3 $\lambda^2 = 51.24$

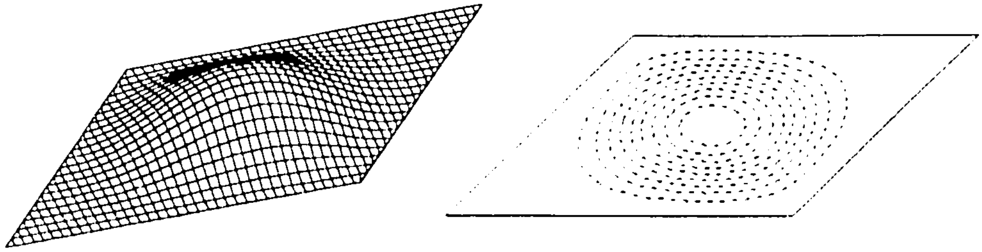


Mode 4 $\lambda^2 = 64.05$

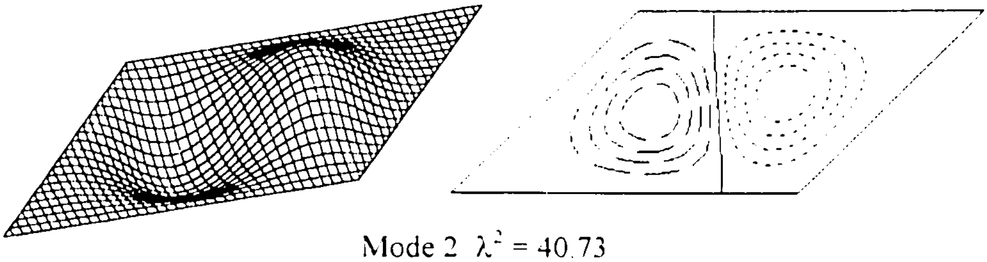


Mode 5 $\lambda^2 = 69.74$

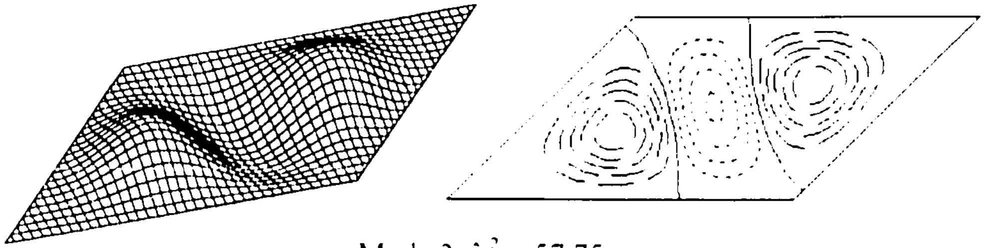
Figure 3-9 CCCC parallelogram plate, $\phi_t = 1.00$, $\phi_r = 1.00$.



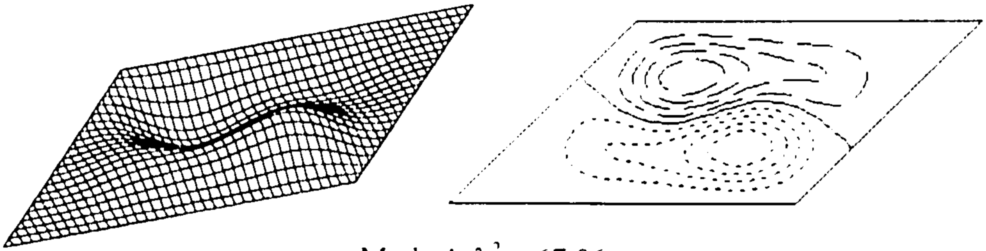
Mode 1 $\lambda^2 = 27.25$



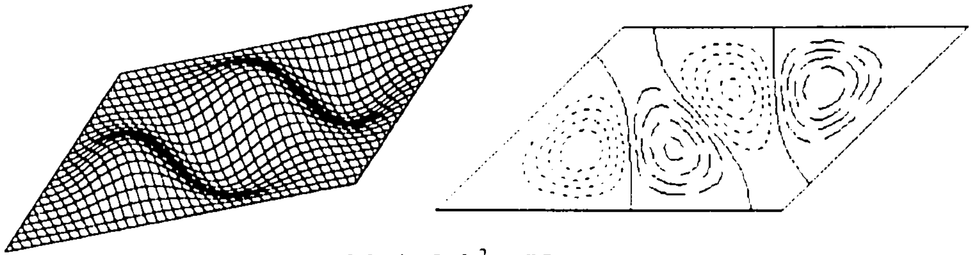
Mode 2 $\lambda^2 = 40.73$



Mode 3 $\lambda^2 = 57.75$

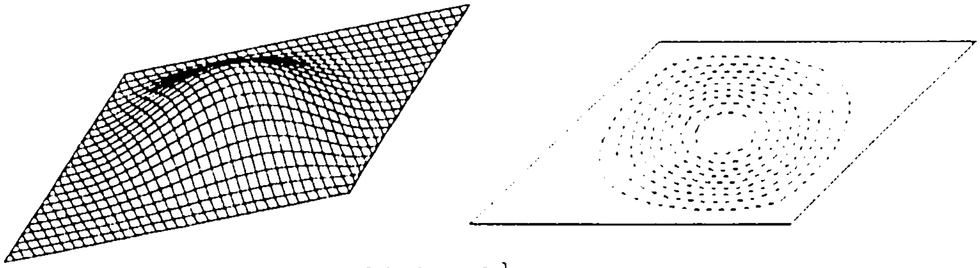


Mode 4 $\lambda^2 = 67.06$

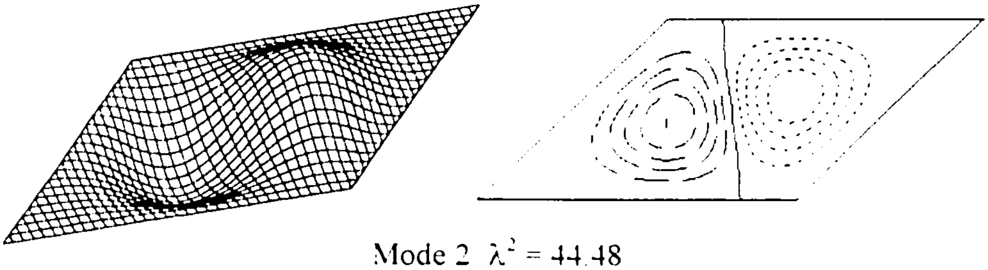


Mode 5 $\lambda^2 = 77.03$

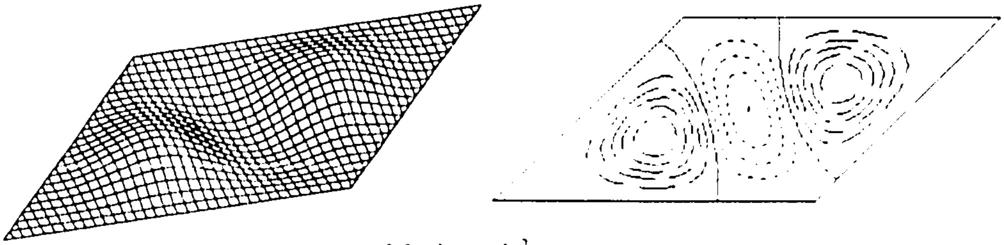
Figure 3-10 CCCC parallelogram plate, $\phi_t = 1.00$, $\phi_r = 1.25$.



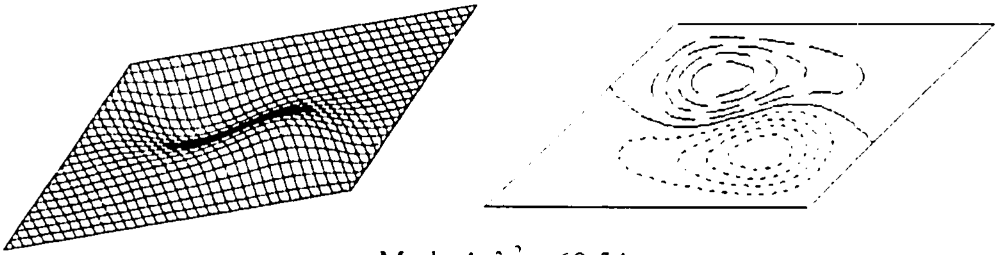
Mode 1 $\lambda^2 = 28.59$



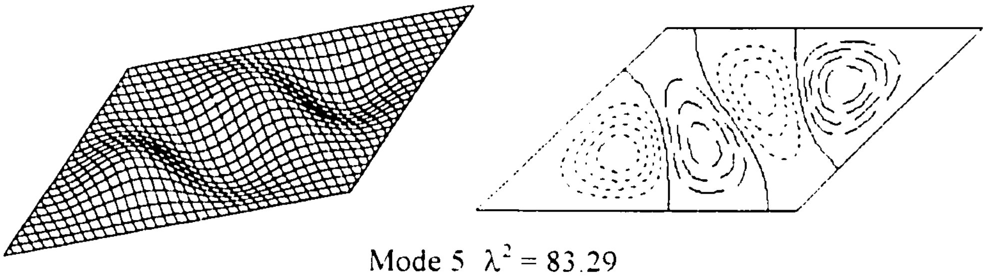
Mode 2 $\lambda^2 = 44.48$



Mode 3 $\lambda^2 = 62.73$



Mode 4 $\lambda^2 = 69.54$



Mode 5 $\lambda^2 = 83.29$

Figure 3-11 CCCC parallelogram plate, $\phi_t = 1.00$, $\phi_r = 1.50$.

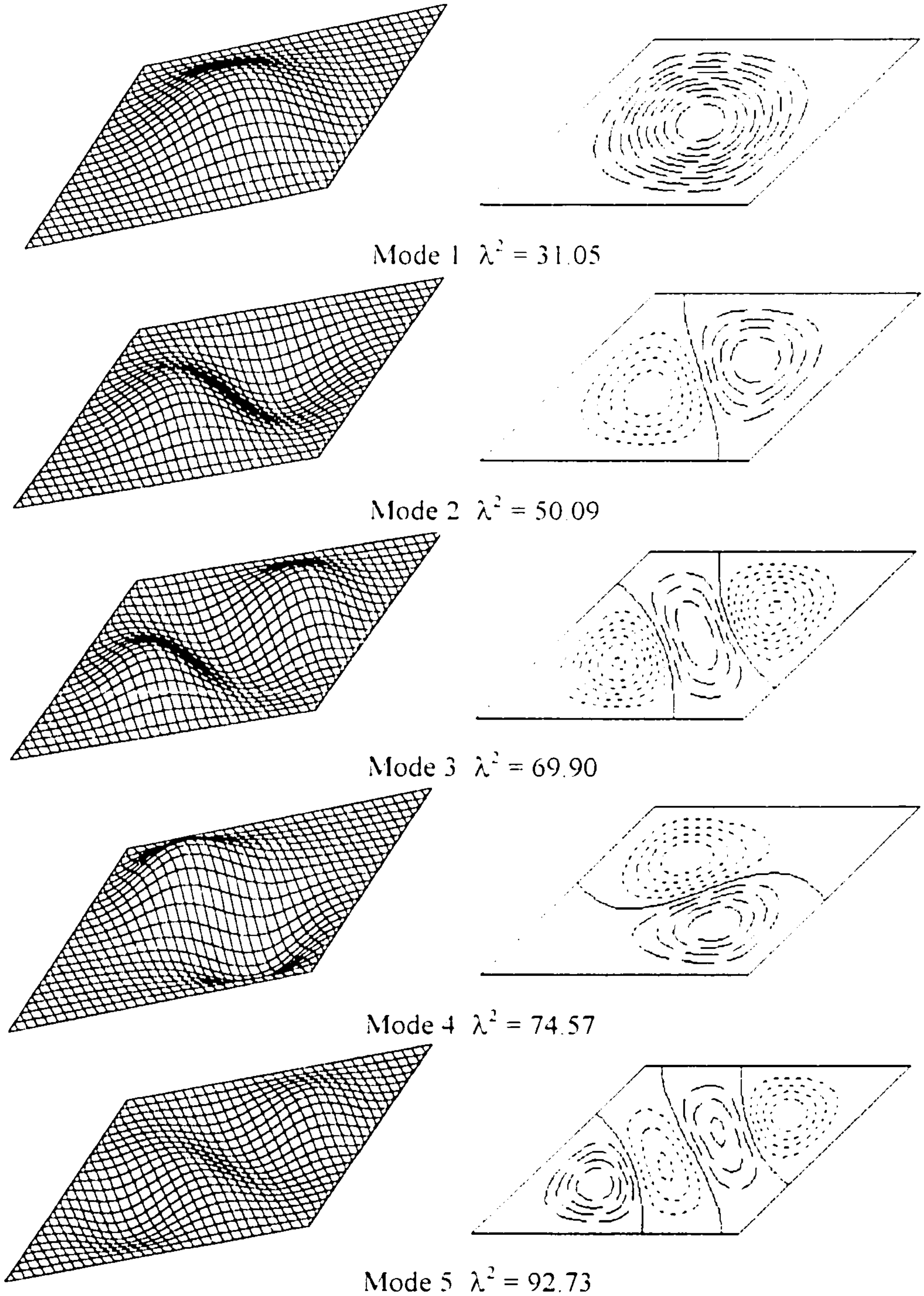
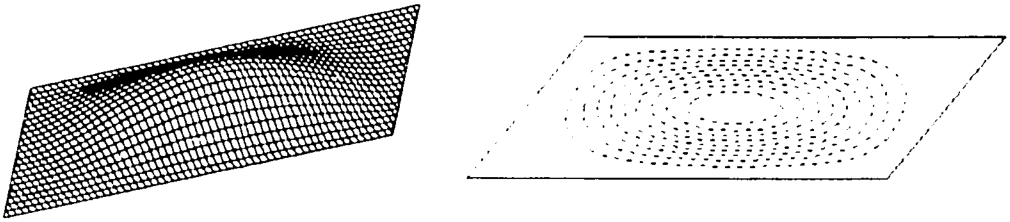
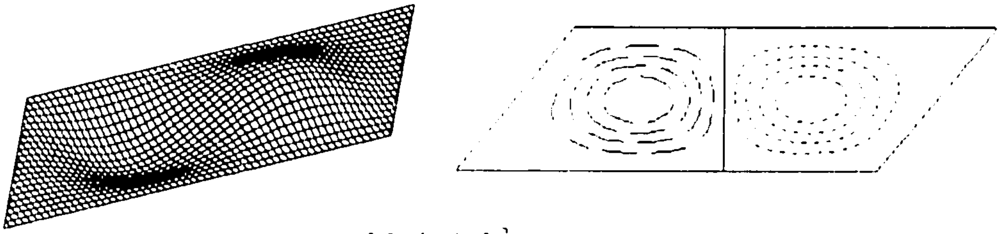


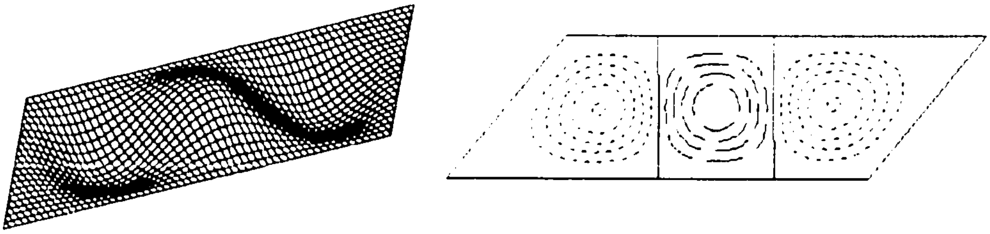
Figure 3-12 CCCC parallelogram plate, $\phi_t = 1.00$, $\phi_r = 2.00$.



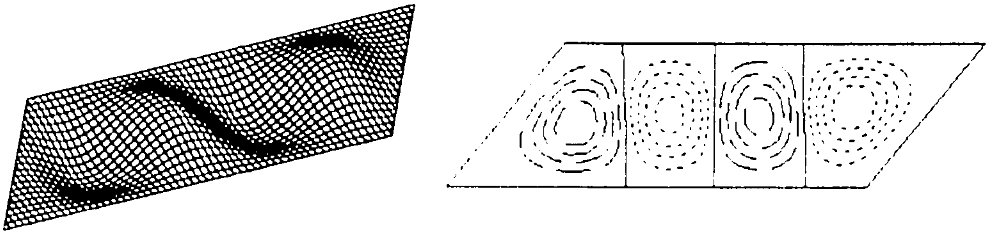
Mode 1 $\lambda^2 = 15.08$



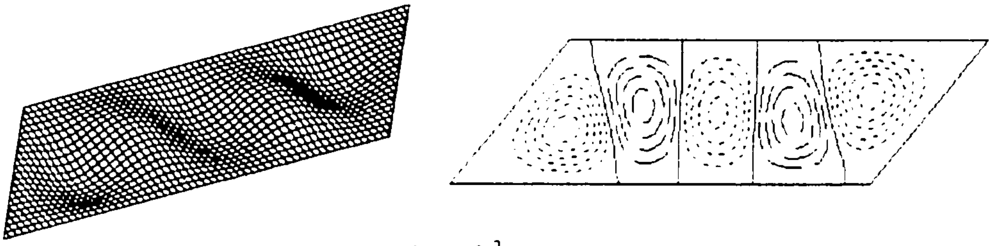
Mode 2 $\lambda^2 = 17.53$



Mode 3 $\lambda^2 = 21.90$

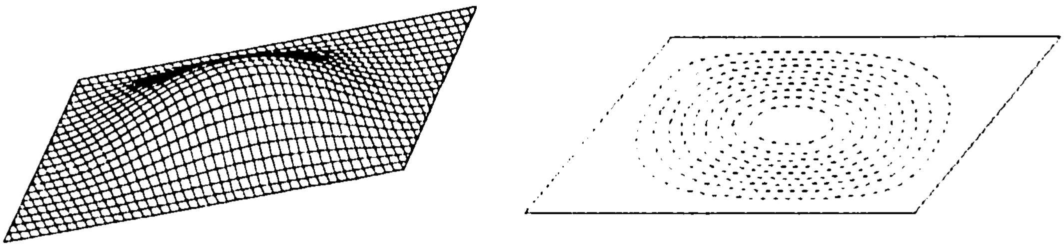


Mode 4 $\lambda^2 = 28.07$

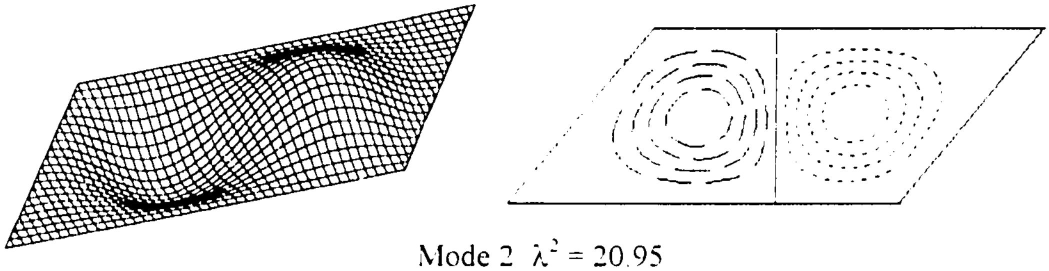


Mode 5 $\lambda^2 = 35.48$

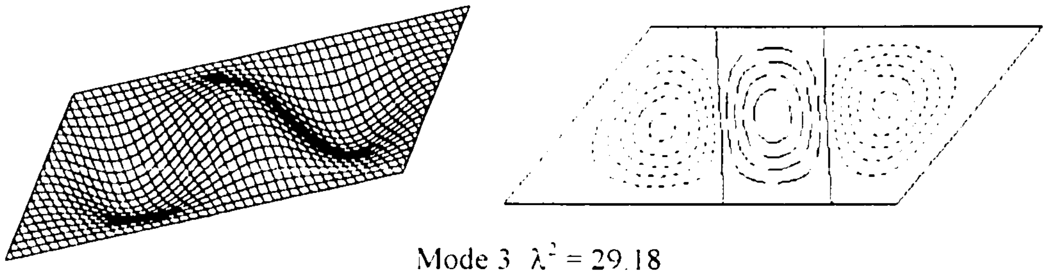
Figure 3-13 CCCC parallelogram plate, $\phi_t = 1.25$, $\phi_r = 0.50$.



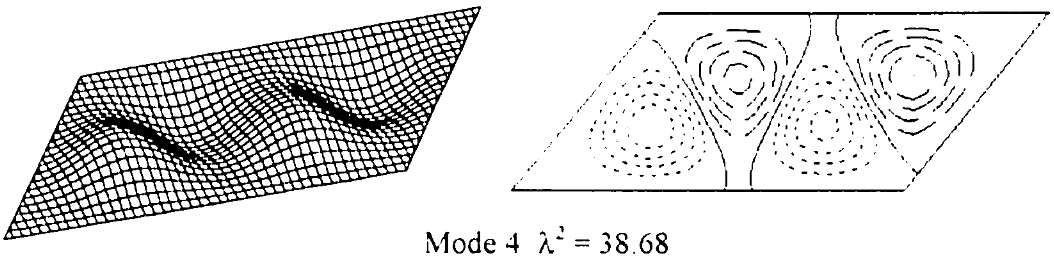
Mode 1 $\lambda^2 = 15.91$



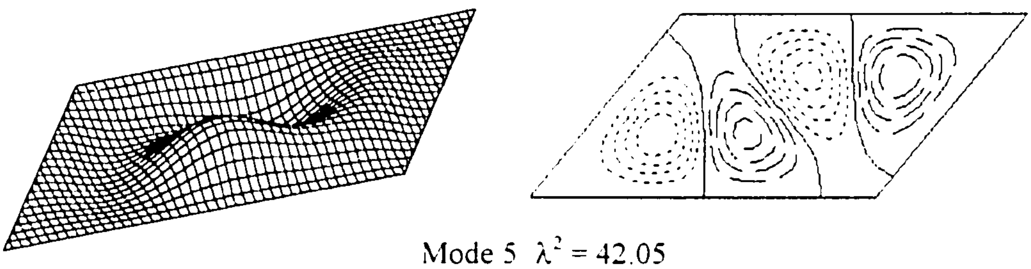
Mode 2 $\lambda^2 = 20.95$



Mode 3 $\lambda^2 = 29.18$



Mode 4 $\lambda^2 = 38.68$



Mode 5 $\lambda^2 = 42.05$

Figure 3-14 CCCC parallelogram plate, $\phi_t = 1.25$, $\phi_r = 0.75$.

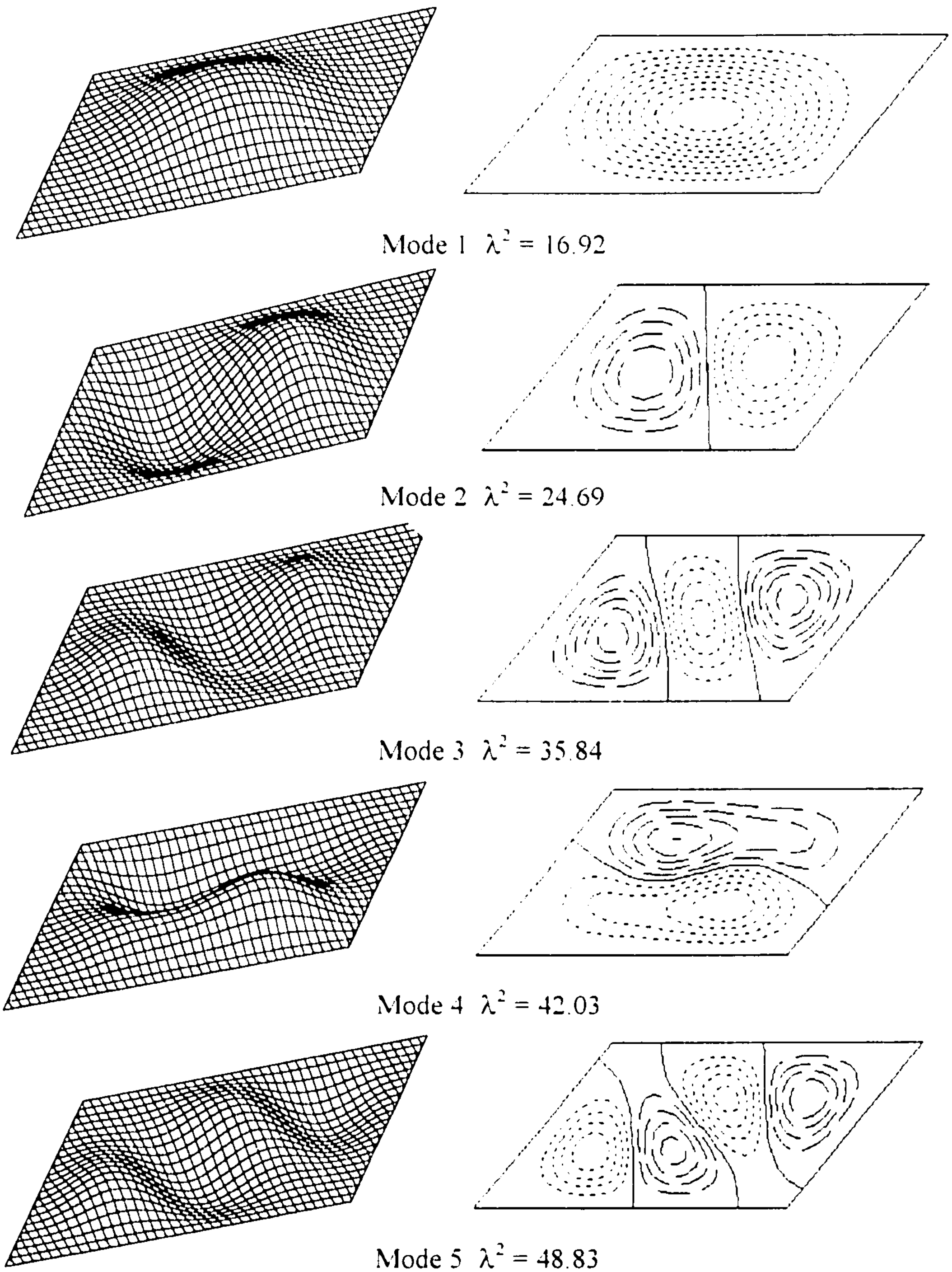
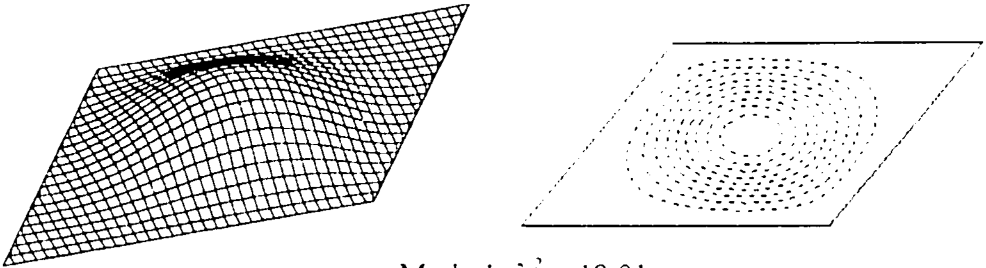
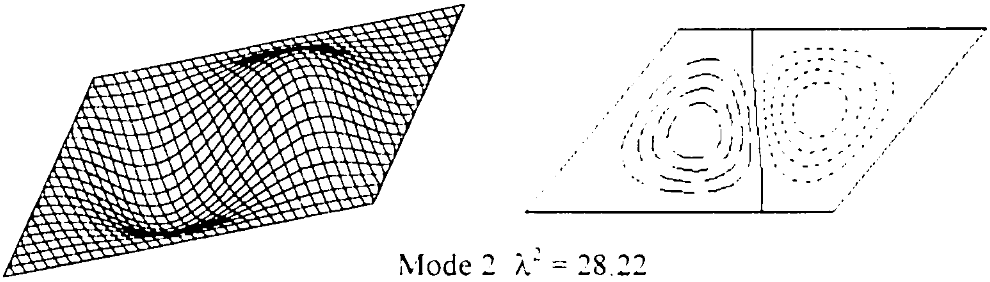


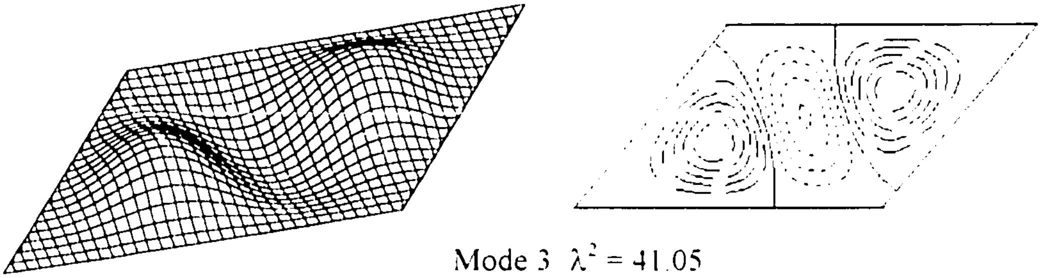
Figure 3-15 CCCC parallelogram plate, $\phi_t = 1.25$, $\phi_r = 1.00$.



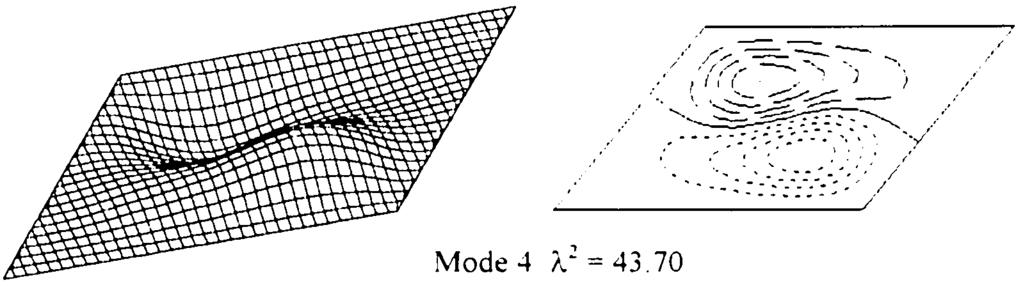
Mode 1 $\lambda^2 = 18.01$



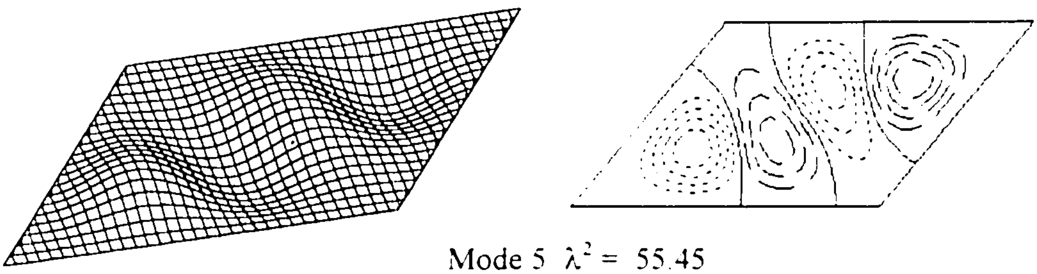
Mode 2 $\lambda^2 = 28.22$



Mode 3 $\lambda^2 = 41.05$



Mode 4 $\lambda^2 = 43.70$



Mode 5 $\lambda^2 = 55.45$

Figure 3-16 CCCC parallelogram plate. $\phi_t = 1.25$, $\phi_r = 1.25$.

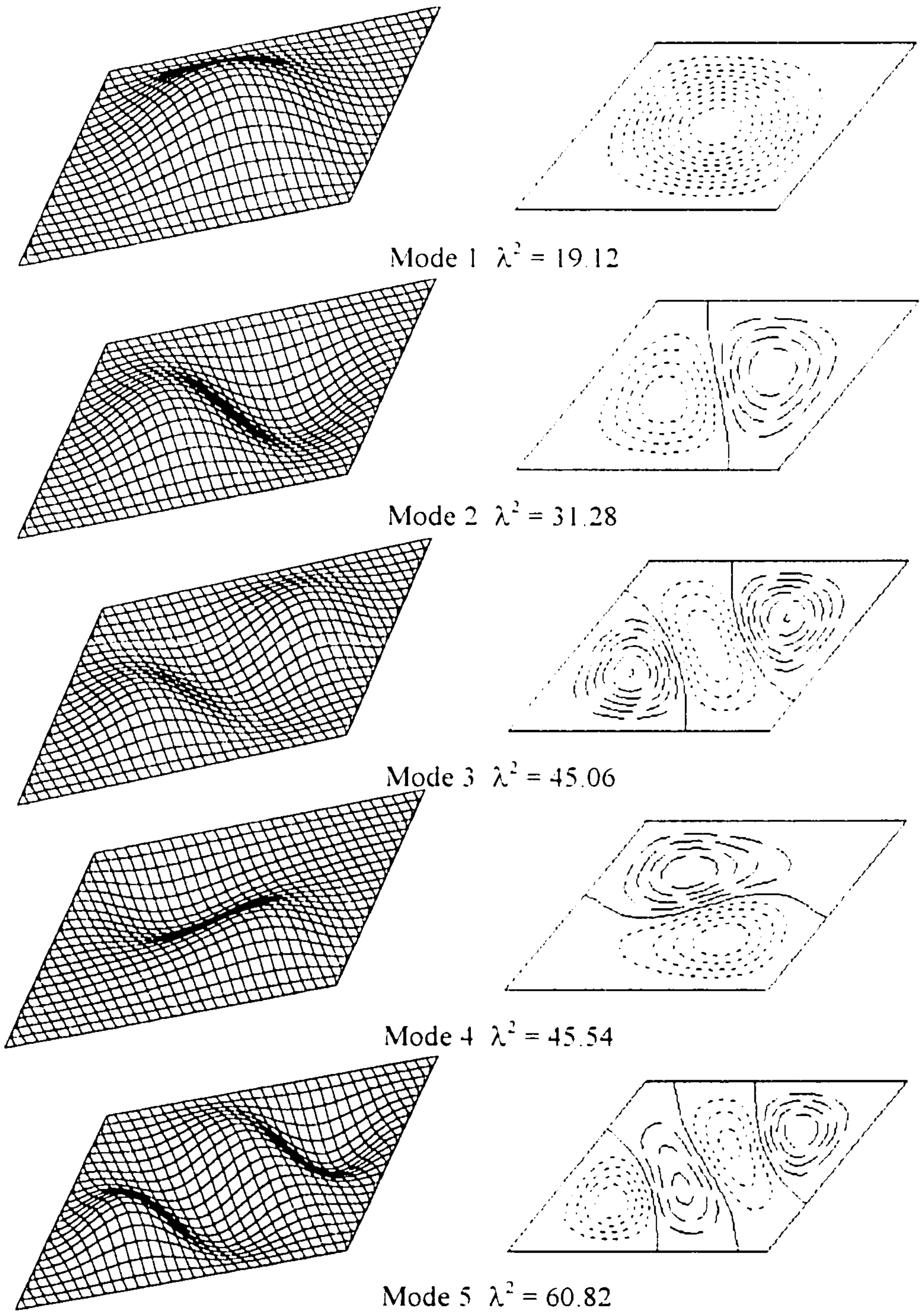


Figure 3-17 CCCC parallelogram plate, $\phi_t = 1.25$, $\phi_r = 1.50$.

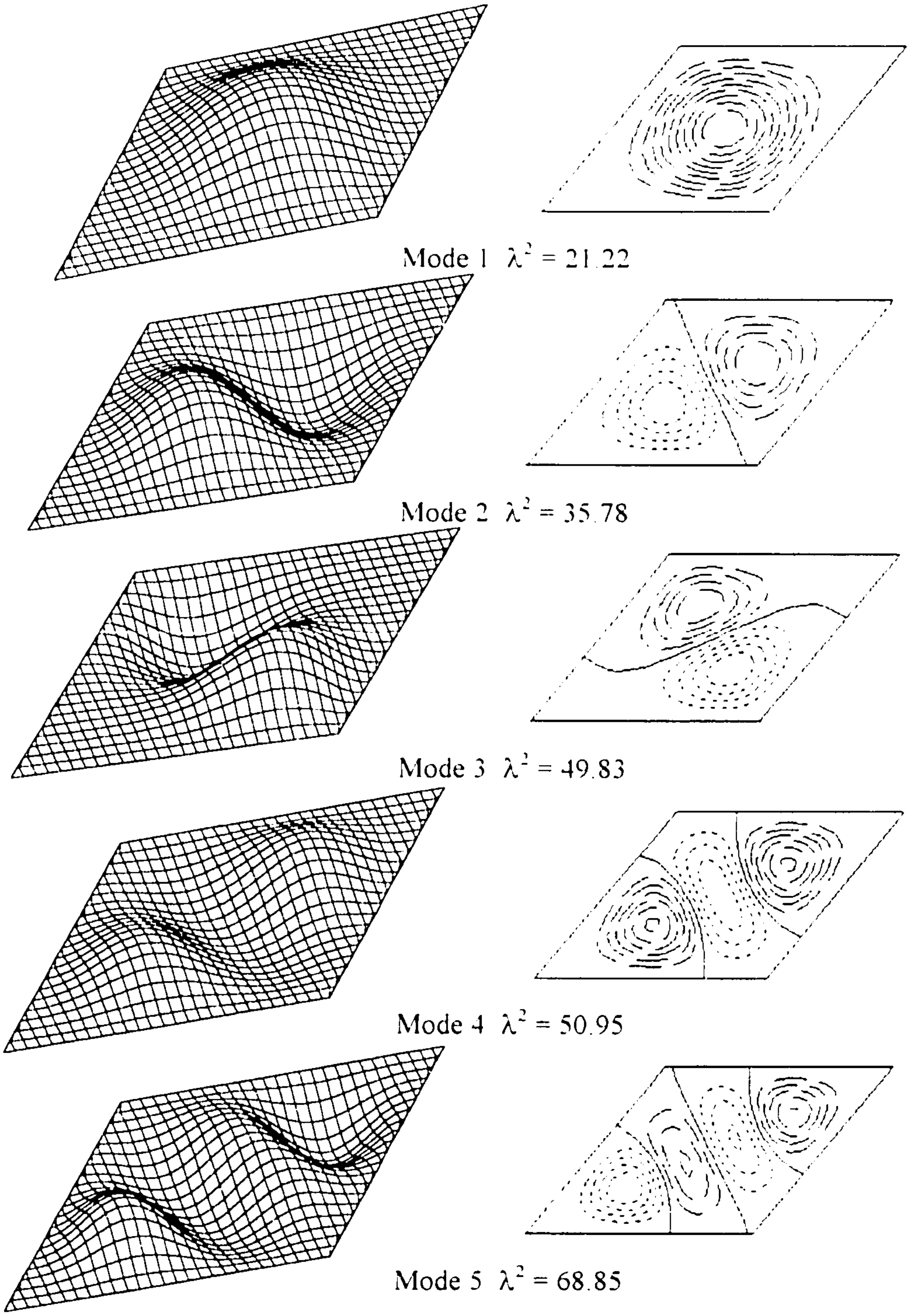
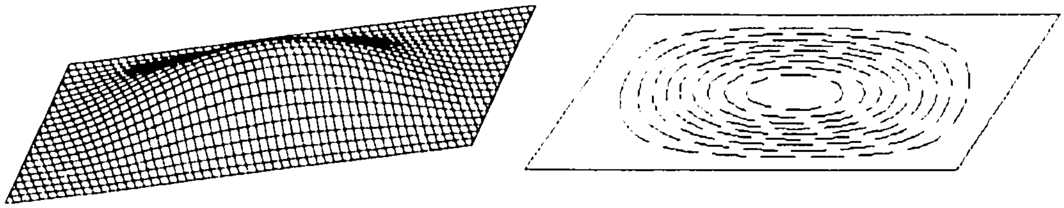
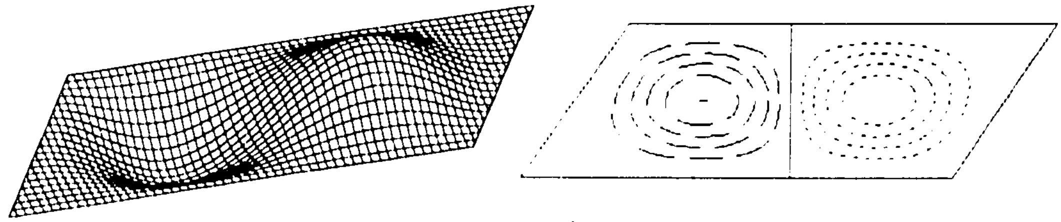


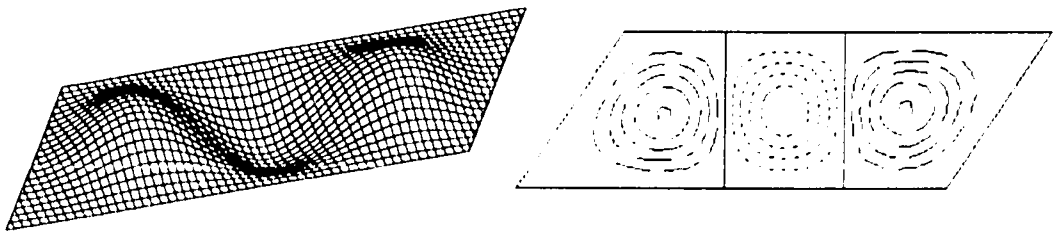
Figure 3-18 CCCC parallelogram plate, $\phi_t = 1.25$, $\phi_r = 2.00$.



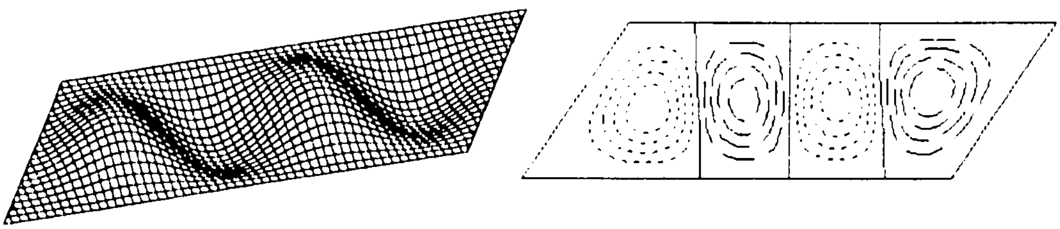
Mode 1 $\lambda^2 = 10.51$



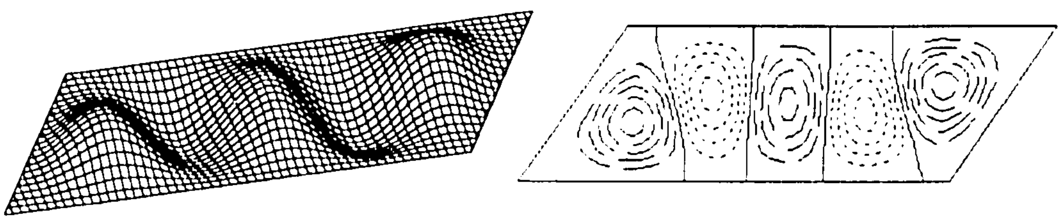
Mode 2 $\lambda^2 = 12.33$



Mode 3 $\lambda^2 = 15.60$

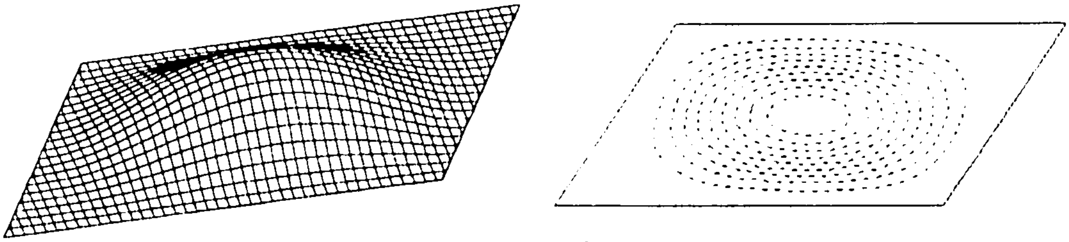


Mode 4 $\lambda^2 = 20.24$

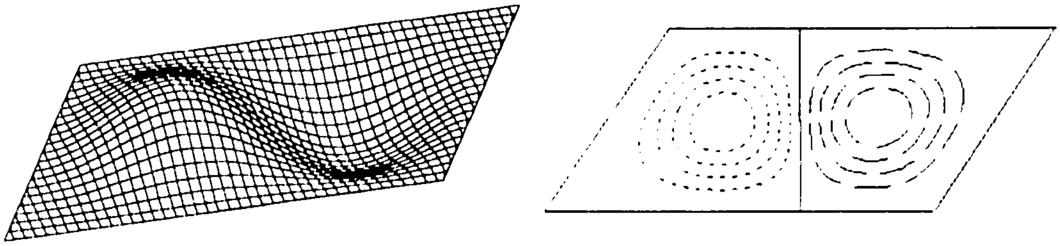


Mode 5 $\lambda^2 = 25.84$

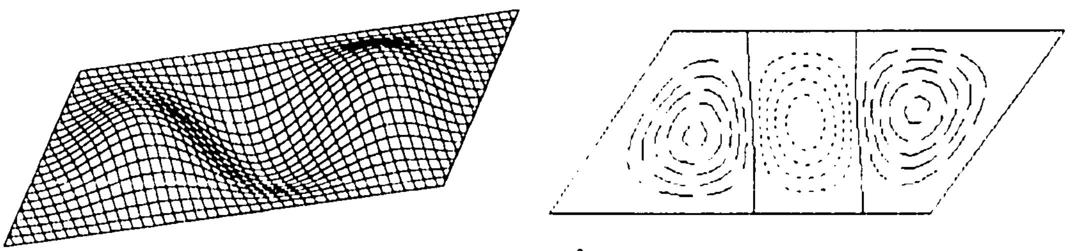
Figure 3-19 CCCC parallelogram plate, $\phi_t = 1.50$, $\phi_r = 0.50$.



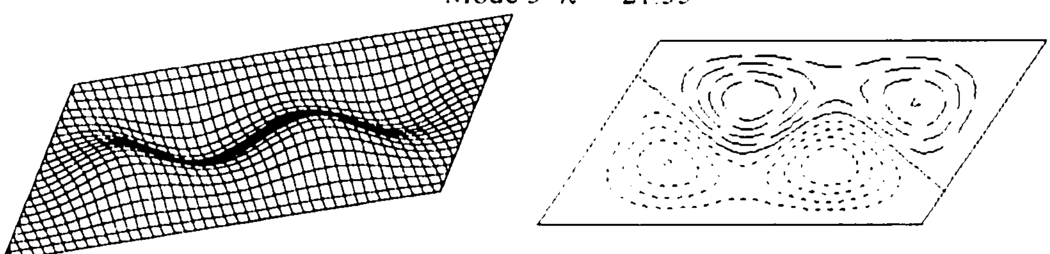
Mode 1 $\lambda^2 = 11.16$



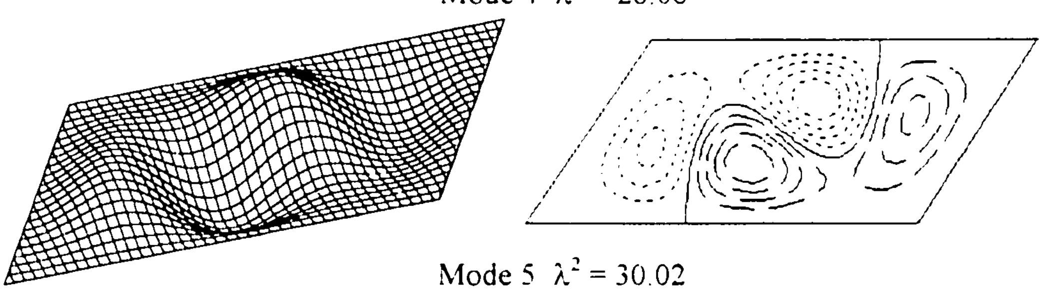
Mode 2 $\lambda^2 = 15.02$



Mode 3 $\lambda^2 = 21.35$

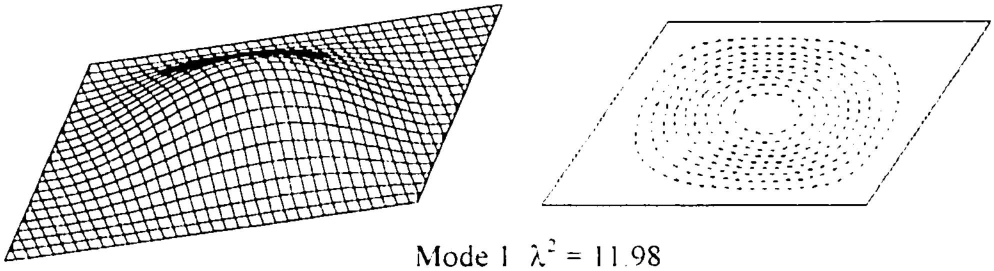


Mode 4 $\lambda^2 = 28.06$

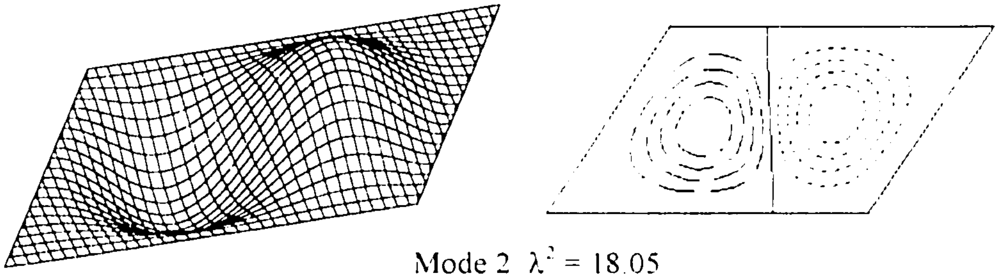


Mode 5 $\lambda^2 = 30.02$

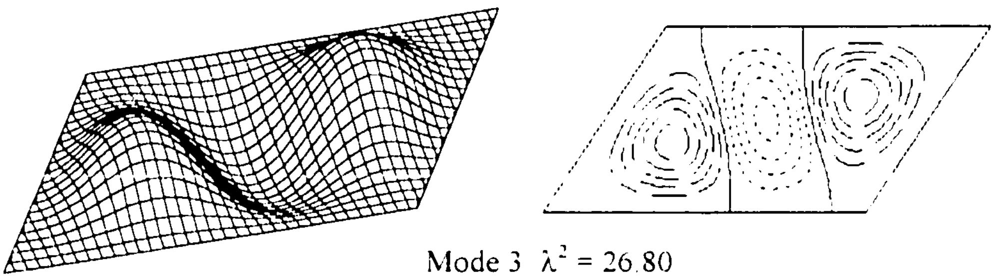
Figure 3-20 CCCC parallelogram plate, $\phi_t = 1.50$, $\phi_r = 0.75$.



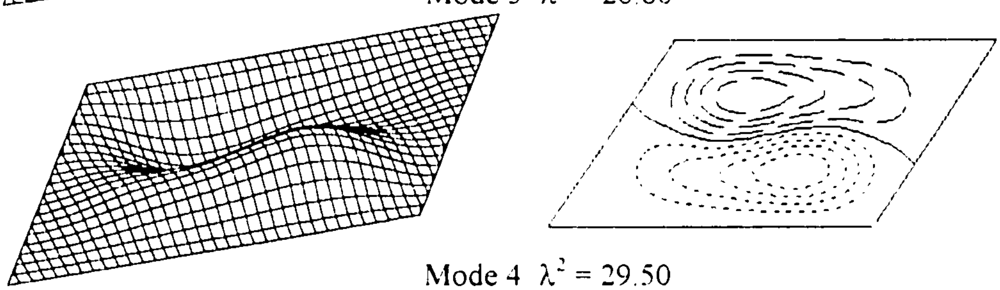
Mode 1 $\lambda^2 = 11.98$



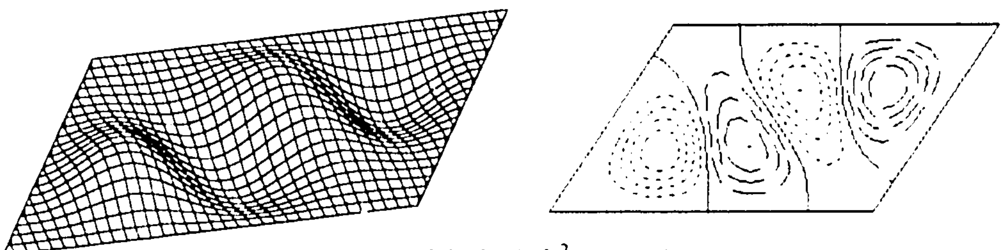
Mode 2 $\lambda^2 = 18.05$



Mode 3 $\lambda^2 = 26.80$



Mode 4 $\lambda^2 = 29.50$



Mode 5 $\lambda^2 = 36.82$

Figure 3-21 CCCC parallelogram plate, $\phi_t = 1.50$, $\phi_r = 1.00$.

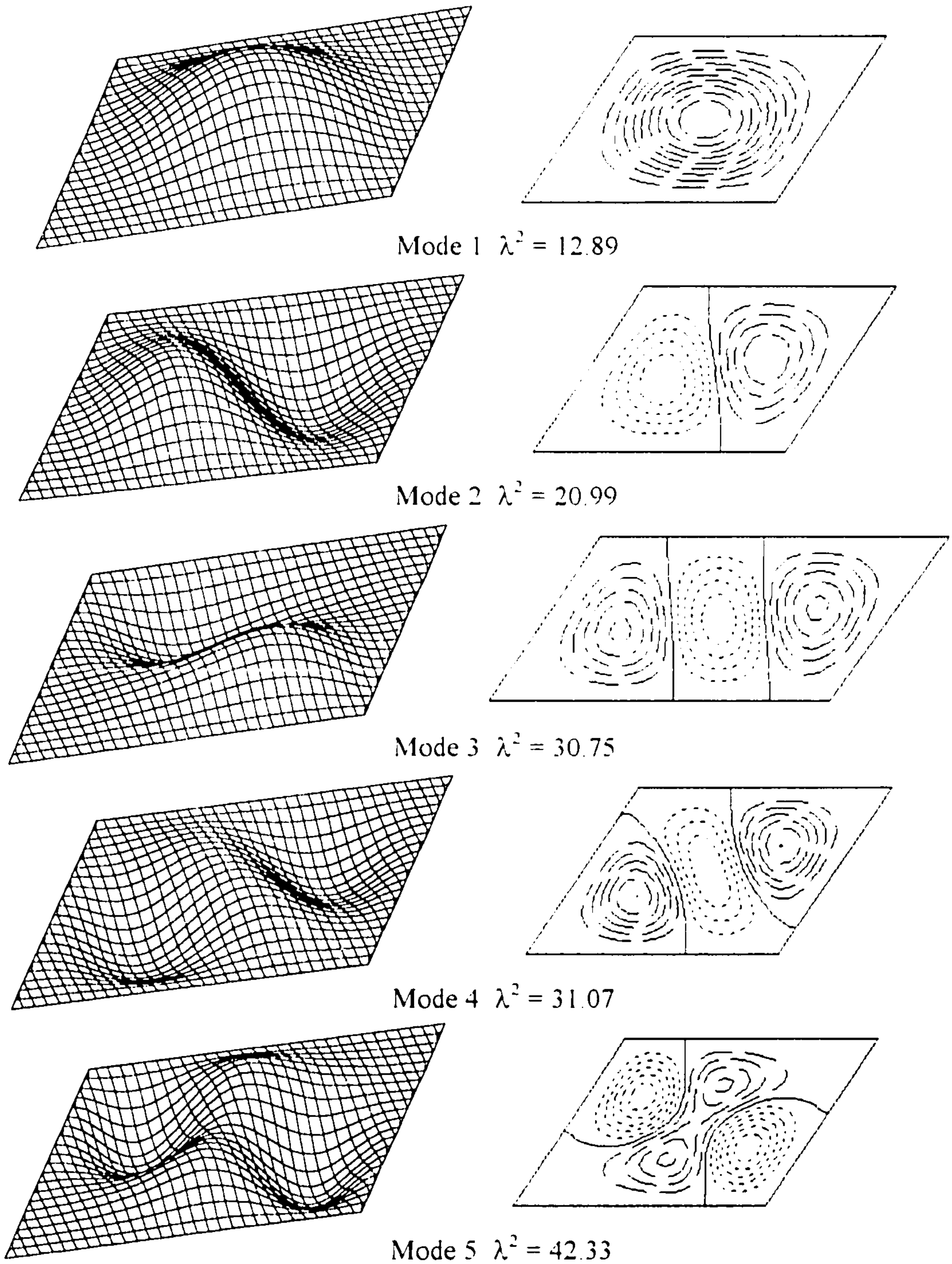


Figure 3-22 CCCC parallelogram plate, ϕ_t = 1.50, ϕ_r = 1.25.

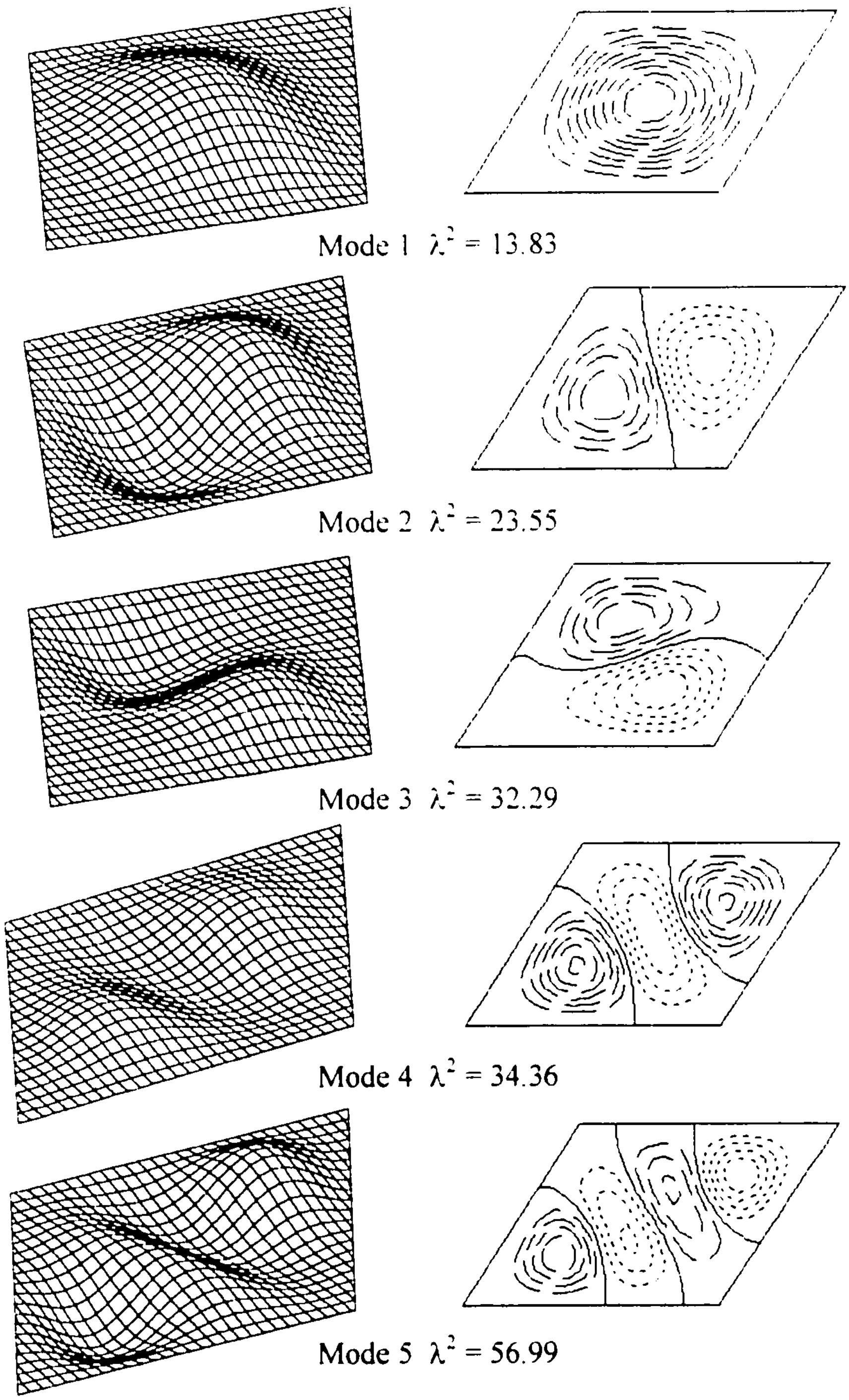
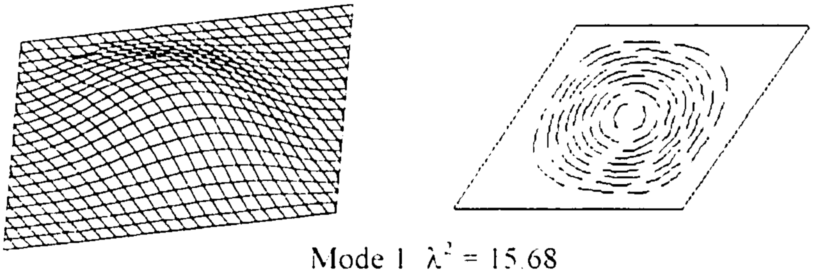
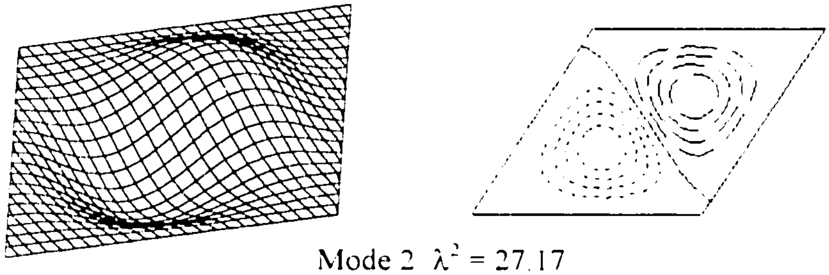


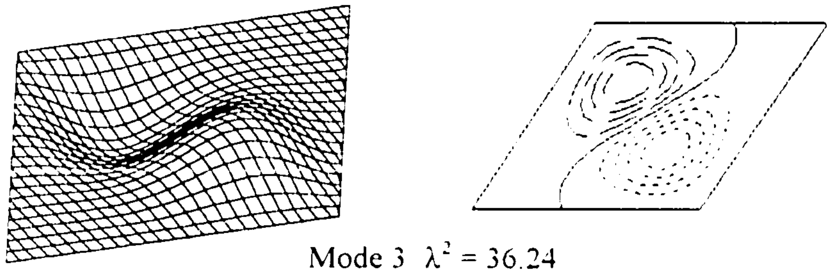
Figure 3-23 CCCC parallelogram plate, ϕ_t = 1.50, ϕ_r = 1.50.



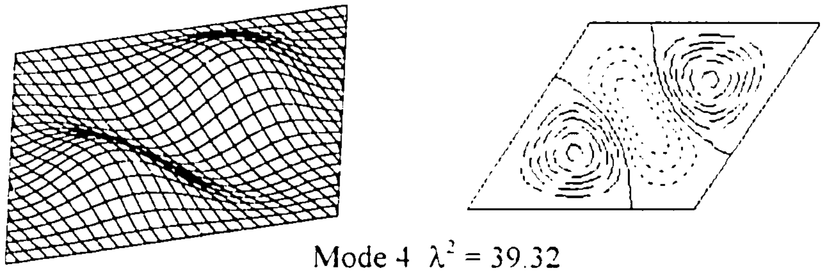
Mode 1 $\lambda^2 = 15.68$



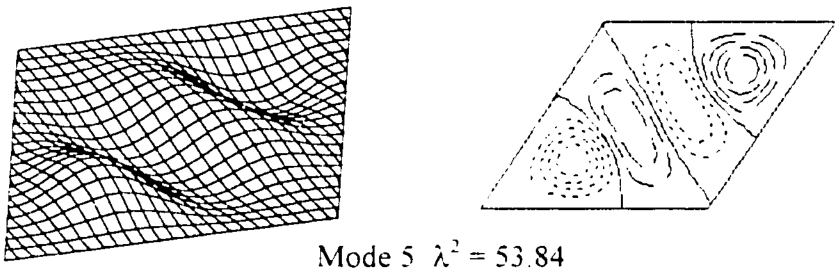
Mode 2 $\lambda^2 = 27.17$



Mode 3 $\lambda^2 = 36.24$

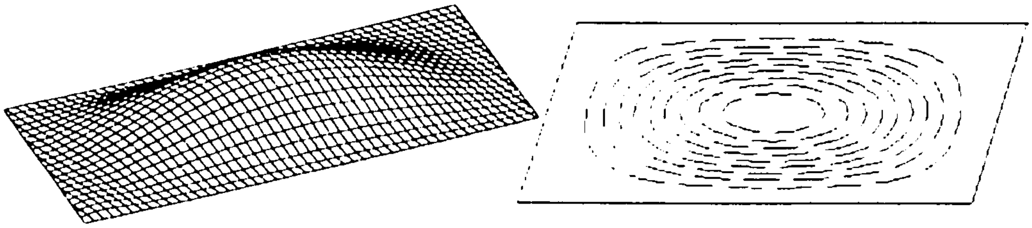


Mode 4 $\lambda^2 = 39.32$

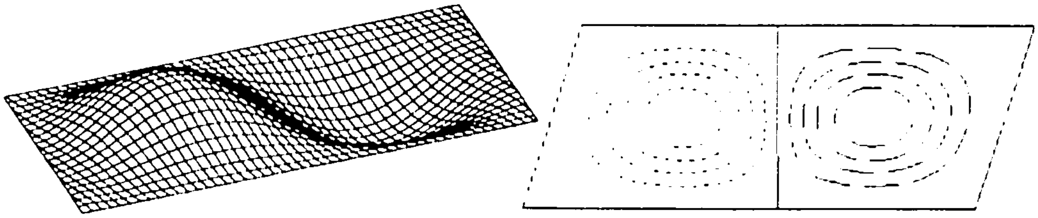


Mode 5 $\lambda^2 = 53.84$

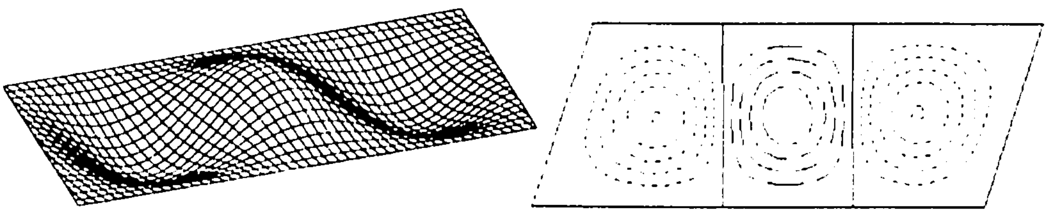
Figure 3-24 CCCC parallelogram plate, $\phi_t = 1.50$, $\phi_r = 2.00$.



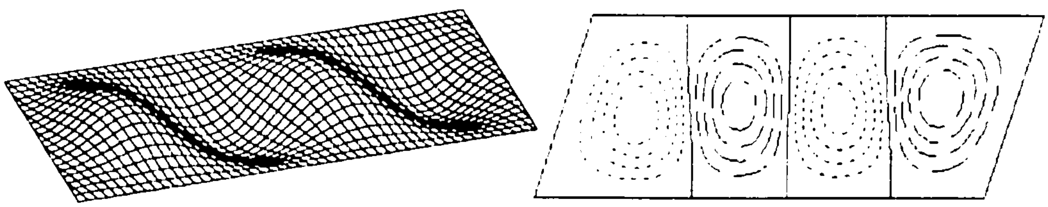
Mode 1 $\lambda^2 = 2.661$



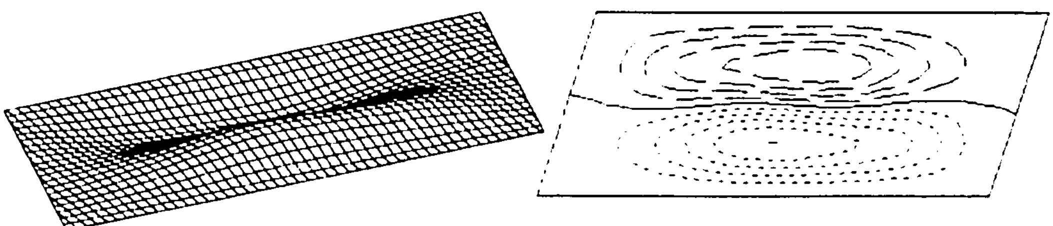
Mode 2 $\lambda^2 = 3.234$



Mode 3 $\lambda^2 = 4.266$

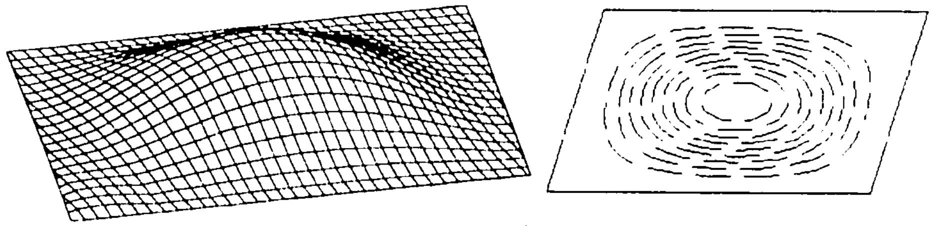


Mode 4 $\lambda^2 = 5.750$

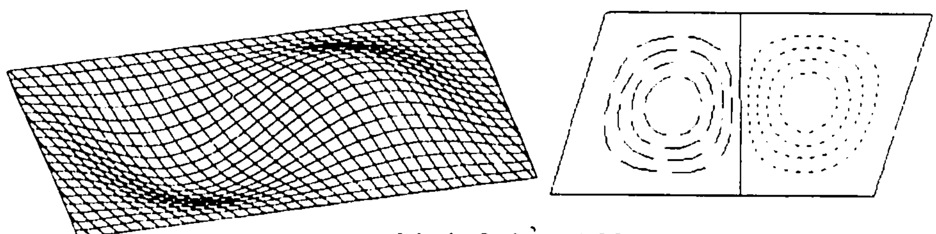


Mode 5 $\lambda^2 = 7.049$

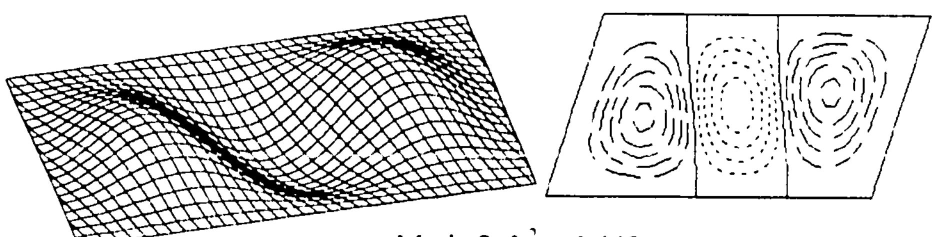
Figure 3-25 CCCC parallelogram plate, $\phi_t = 3.00$, $\phi_r = 0.50$.



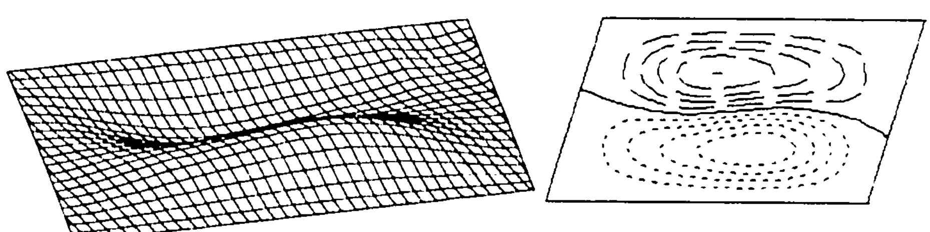
Mode 1 $\lambda^2 = 2.904$



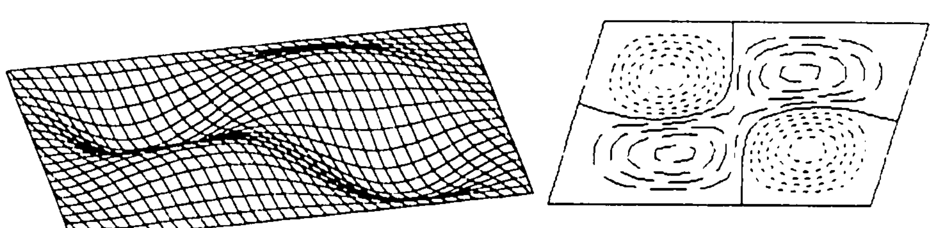
Mode 2 $\lambda^2 = 4.234$



Mode 3 $\lambda^2 = 6.440$



Mode 4 $\lambda^2 = 7.265$



Mode 5 $\lambda^2 = 8.605$

Figure 3-26 CCCC parallelogram plate, $\phi_t = 3.00$, $\phi_r = 0.75$.

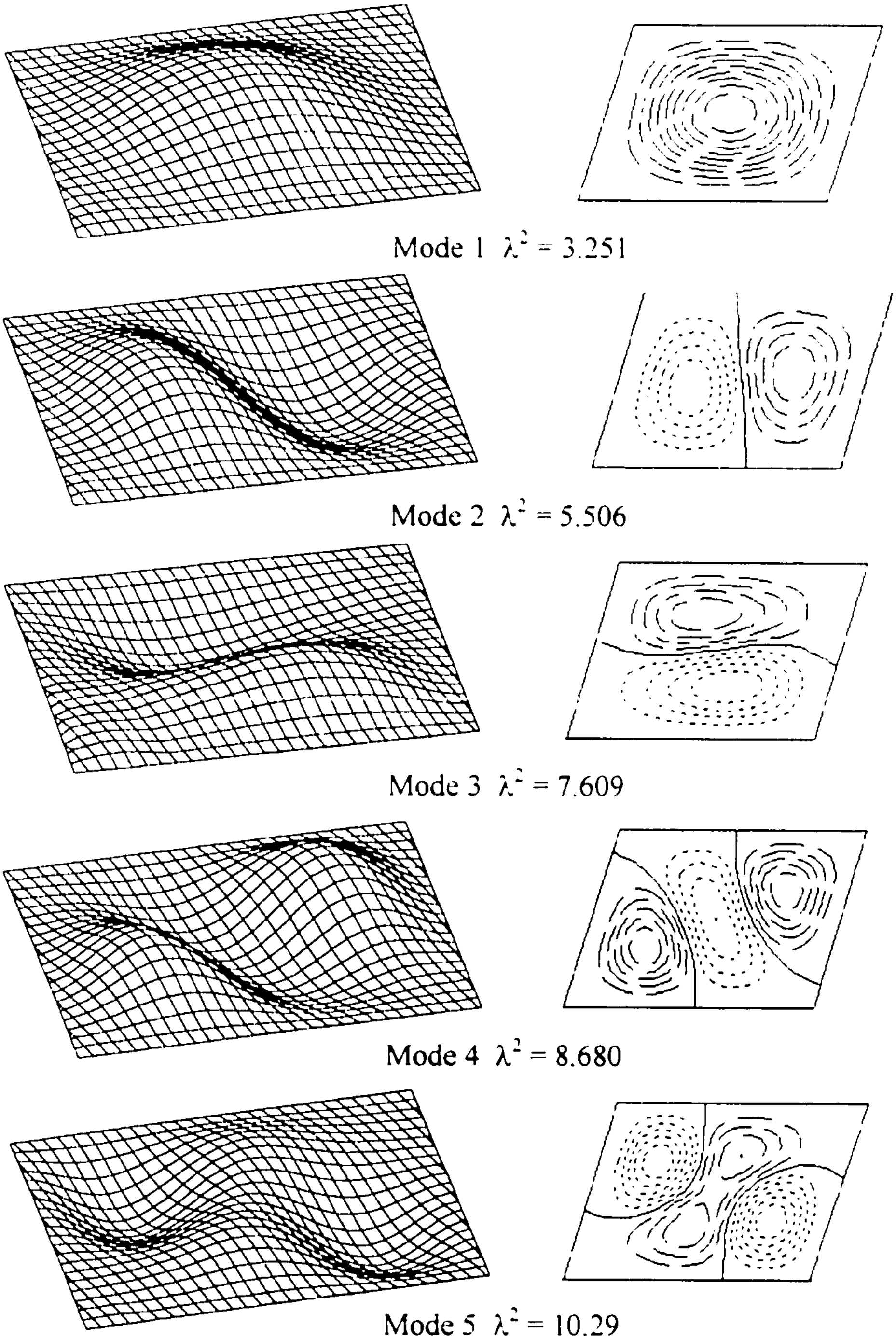
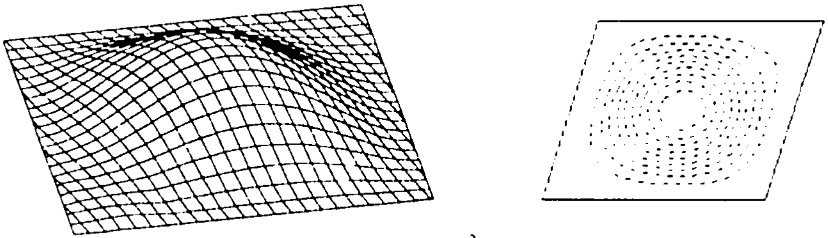
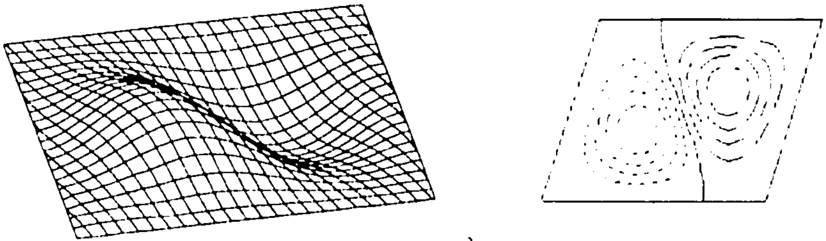


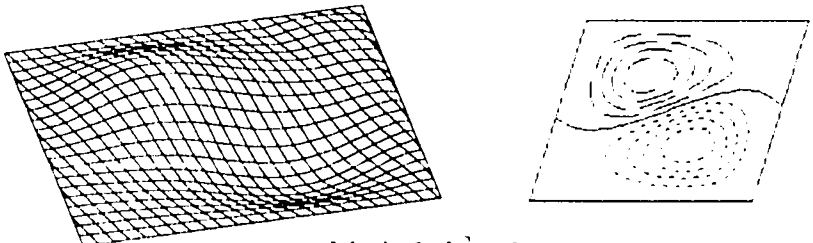
Figure 3-27 CCCC parallelogram plate, ϕ_t = 3.00, ϕ_r = 1.00.



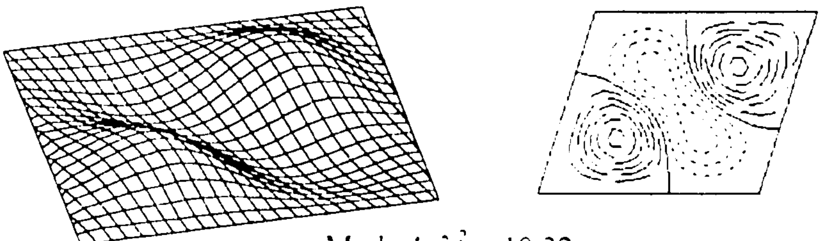
Mode 1 $\lambda^2 = 3.679$



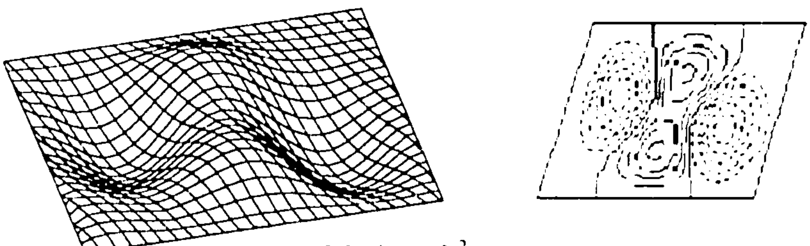
Mode 2 $\lambda^2 = 6.815$



Mode 3 $\lambda^2 = 8.144$



Mode 4 $\lambda^2 = 10.32$



Mode 5 $\lambda^2 = 12.93$

Figure 3-28 CCCC parallelogram plate, $\phi_t = 3.00$, $\phi_r = 1.25$.

The dependence of the eigenvalue on the relative dimensions of the plate, regardless of the mode shape, is shown in the graphs that follow. Figures 3-29 and 3-30 show the change of the eigenvalue for plates with the same aspect ratio ϕ_r, with changing aspect ratio ϕ_t. The increasing aspect ratio of the right triangular segment is equivalent to a decrease in the skew angle α. The effect of ϕ_t on the eigenvalue is illustrated in Figure 3-31 for ϕ_t = 0.75, for which the mode shapes are given in Figures 3-1 to 3-6. Each curve in Figure 3-31 represents a mode number and not a mode shape.

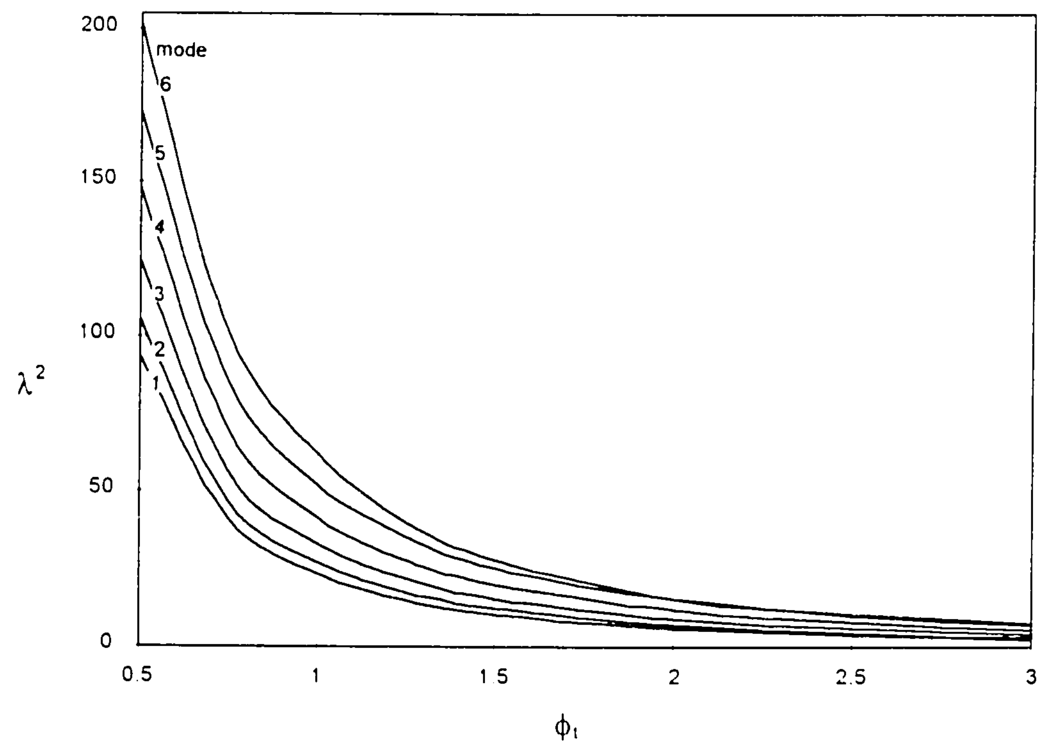


Figure 3-29 Effect of ϕ_t on the eigenvalue for CCCC parallelogram plates, ϕ_r = 0.50.

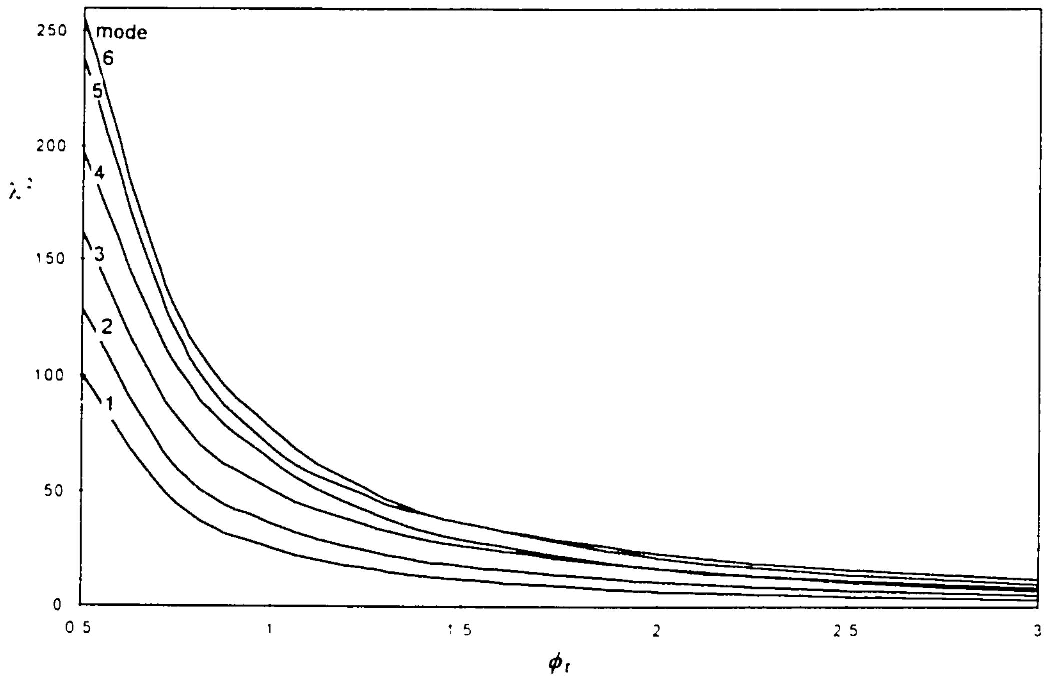


Figure 3-30 Effect of ϕ_t on the eigenvalue for CCCC parallelogram plates, ϕ_r = 1.00.

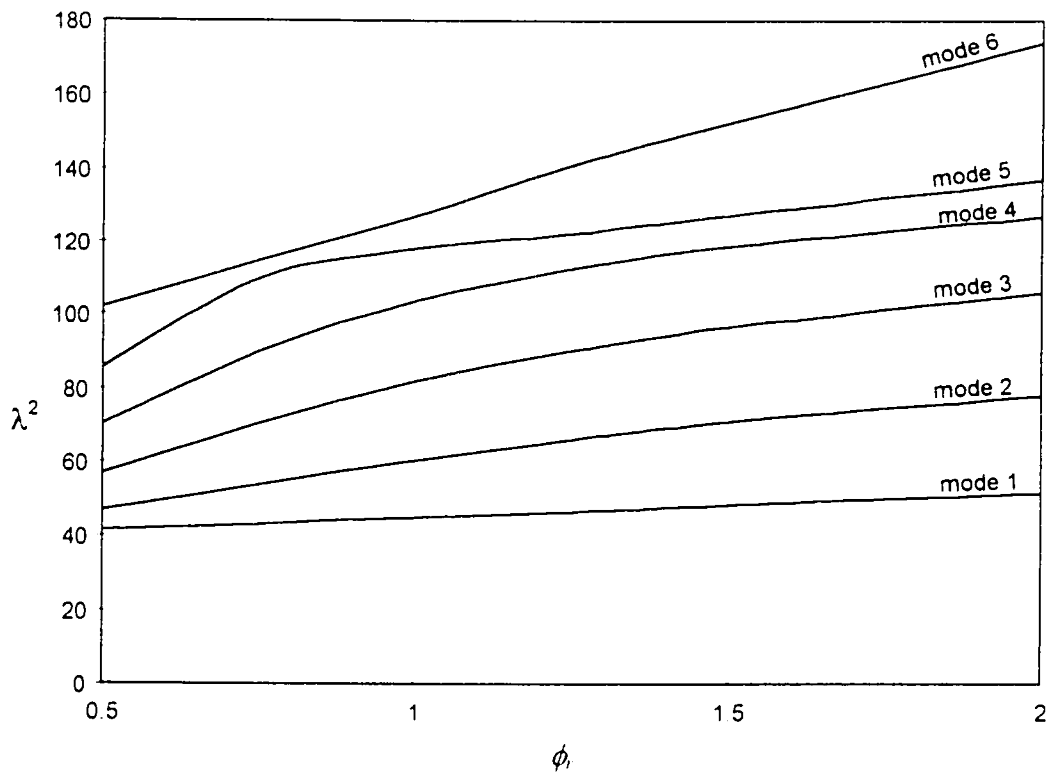


Figure 3-31 Effect of ϕ_r on the eigenvalue for CCCC parallelogram plates, ϕ_t = 0.75.

3. Combined Edge Supports

The eigenvalues for plates with combined simple and clamped supports are higher than those for the same size simply supported plate, but lower than those of the fully clamped plate for a given mode. The effects of plate dimension on the eigenvalue and the mode shape are the same as for simply supported and clamped supports. Figures 3-32 and 3-33 show the change of the eigenvalue λ^2 with aspect ratio ϕ_t, while Figure 3-34 is a close-up of Figure 3-33 showing where the switching of mode shapes occurs.

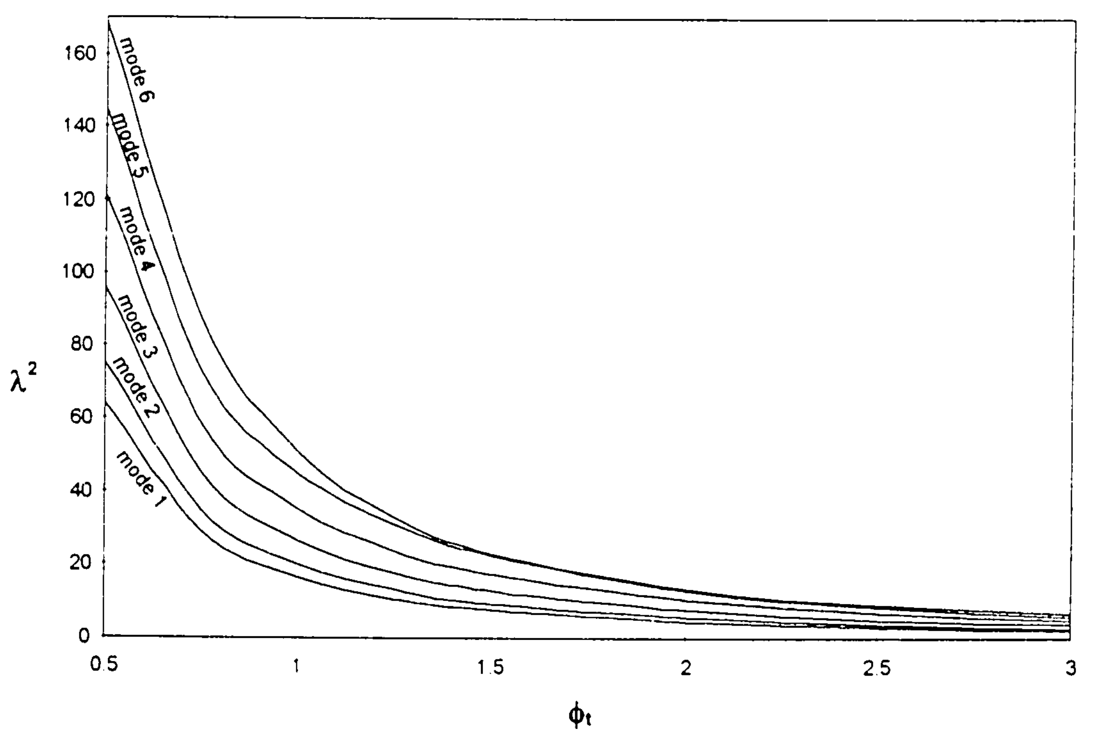


Figure 3-32 Effect of ϕ_t on the eigenvalue for CSSC parallelogram plates, ϕ_r = 0.50

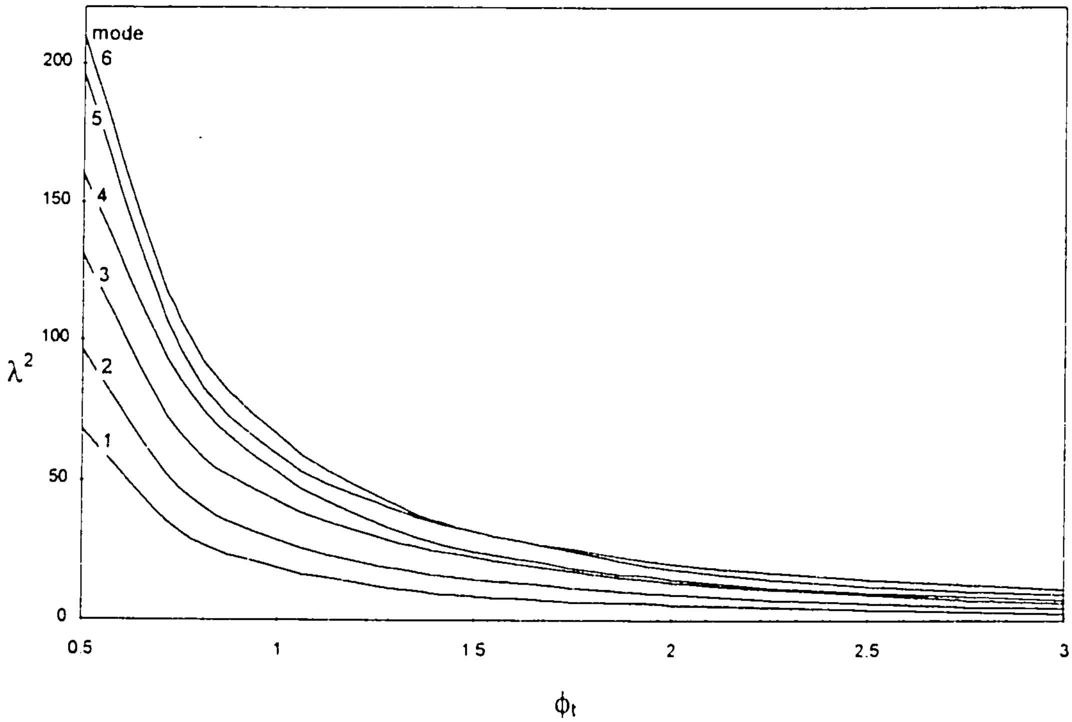


Figure 3-33 Effect of ϕ_t on the eigenvalue for CSSC parallelogram plates, ϕ_r = 1.00.

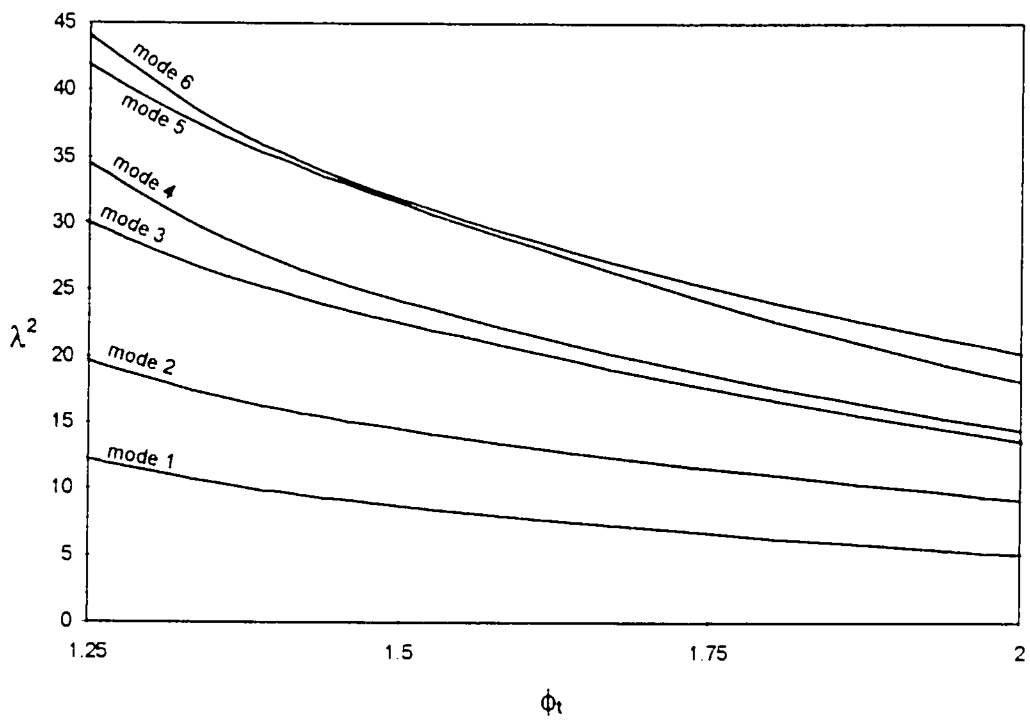


Figure 3-34 CSSC parallelogram plates, ϕ_r = 1.00: close-up.

4 - Experimental Analysis

In this work, six fully clamped plates were tested: three differently sized plates for each of two skew angles. The labeling of the dimensions is shown in Figure 1-1. Mode shapes and natural frequencies were obtained for the first three modes; natural frequencies only, for the next three modes.

1. Experimental Procedure

The experimental procedure is described in detail in Appendix 2. Six parallelogram plates, cut from 0.125 in. sheets of aluminum, were tested. The skew angles α were 15° and 30°; the aspect ratios of the rectangular segment were 1.00, 1.25 and 1.60. The plates were clamped around all four edges and subjected to a spectrum of white noise. The frequency response was measured with an accelerometer. For each plate, the first six eigenfrequencies as well as the first three mode shapes were obtained.

For the first three modes of vibration of each plate, the frequency response of each point on a grid marked on the plate was measured. From this, the resonance frequencies and the corresponding mode shapes were generated. Only the resonance frequencies were measured for the next three modes.

2. Experimental Results

The first three mode shapes were obtained by using the STAR Modal software package. The results are shown in Figures 4-1 and 4-2 along with the corresponding resonance frequencies. The theoretical contour and mesh plots as well as frequencies are shown alongside. Complete results for the first six modes are found in Tables 4-1 and 4-2.

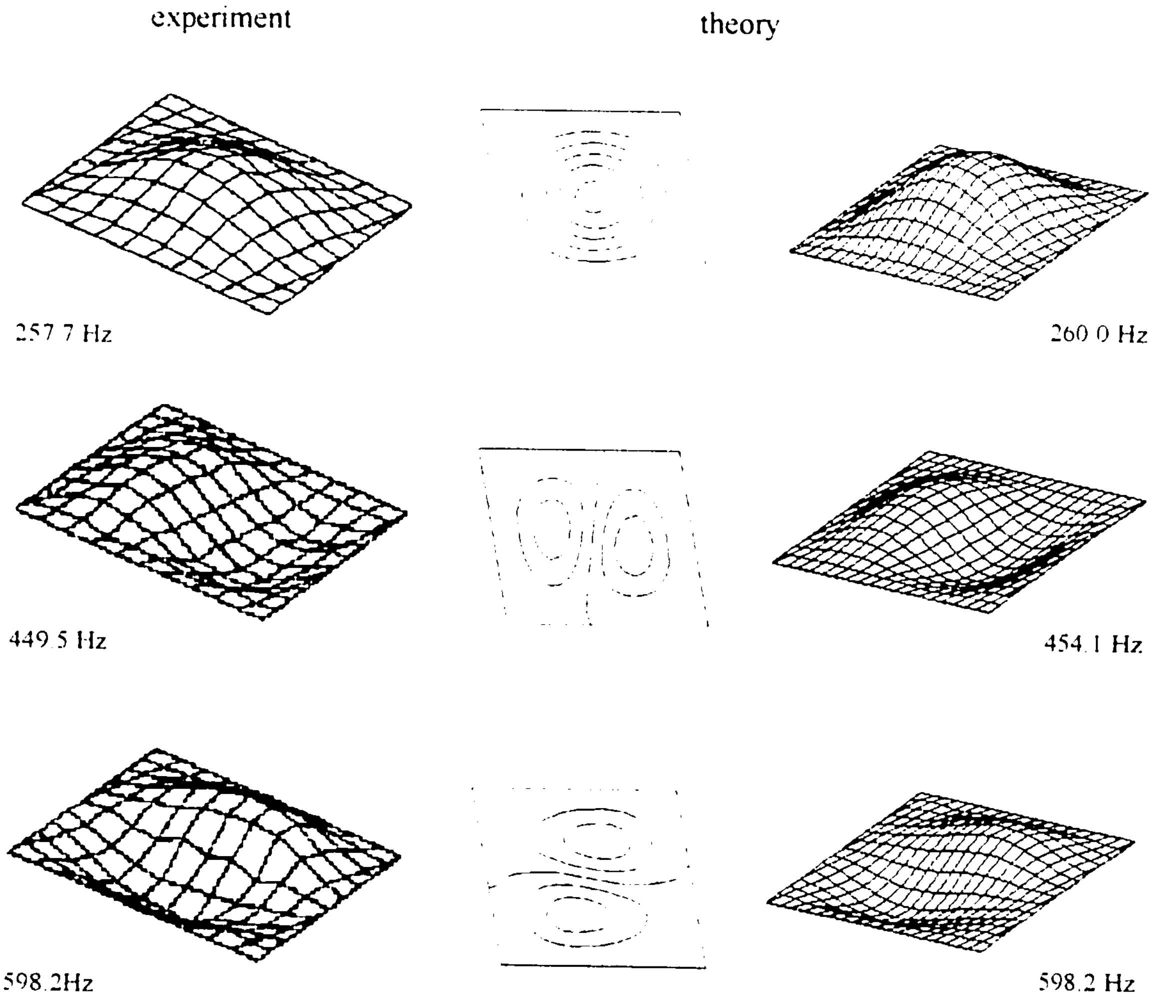


Figure 4-1 CCCC parallelogram plate, $\alpha = 15°$, $\phi_r = 1.00$.

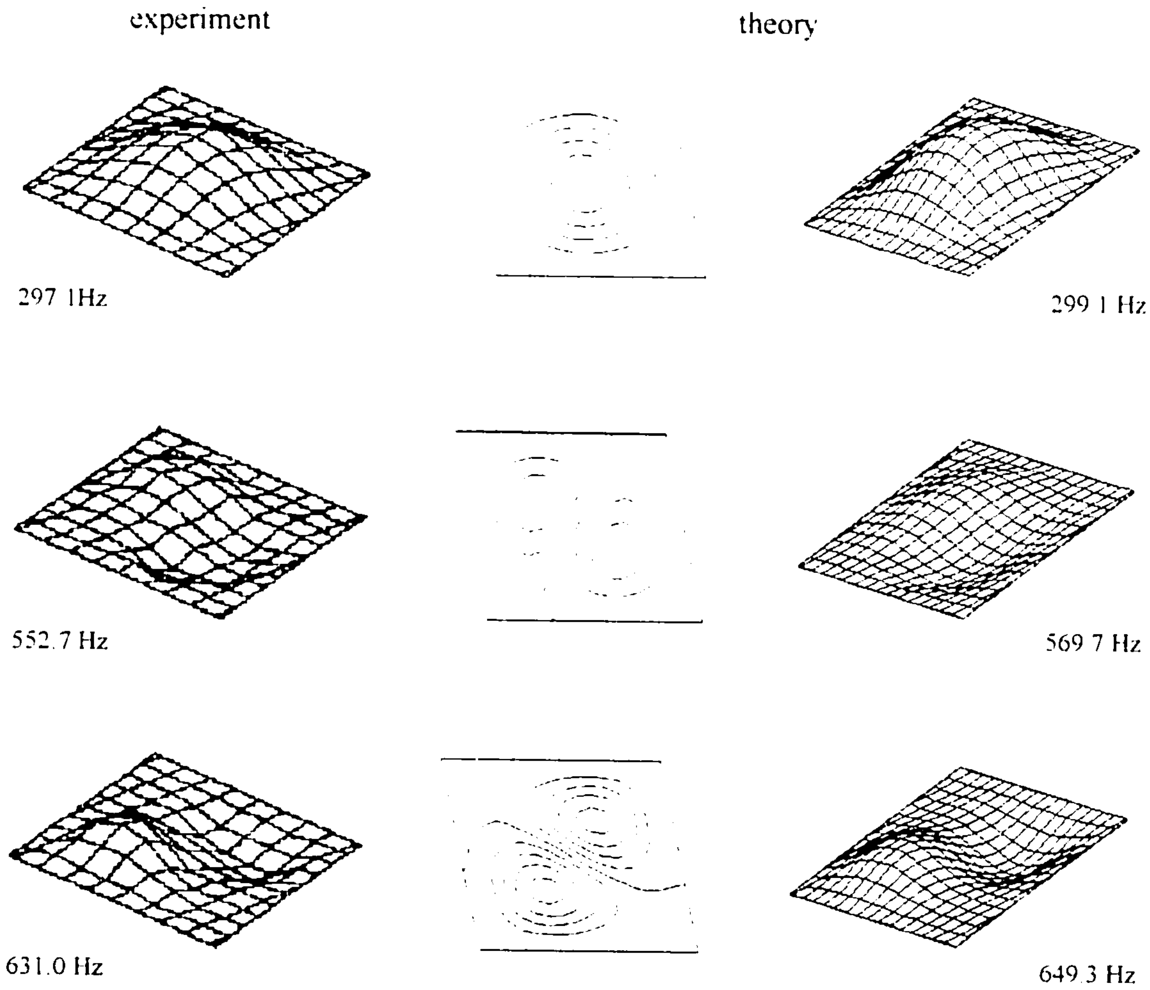


Figure 4-2 CCCC parallelogram plate, $\alpha = 15°$, $\phi_r = 1.25$.

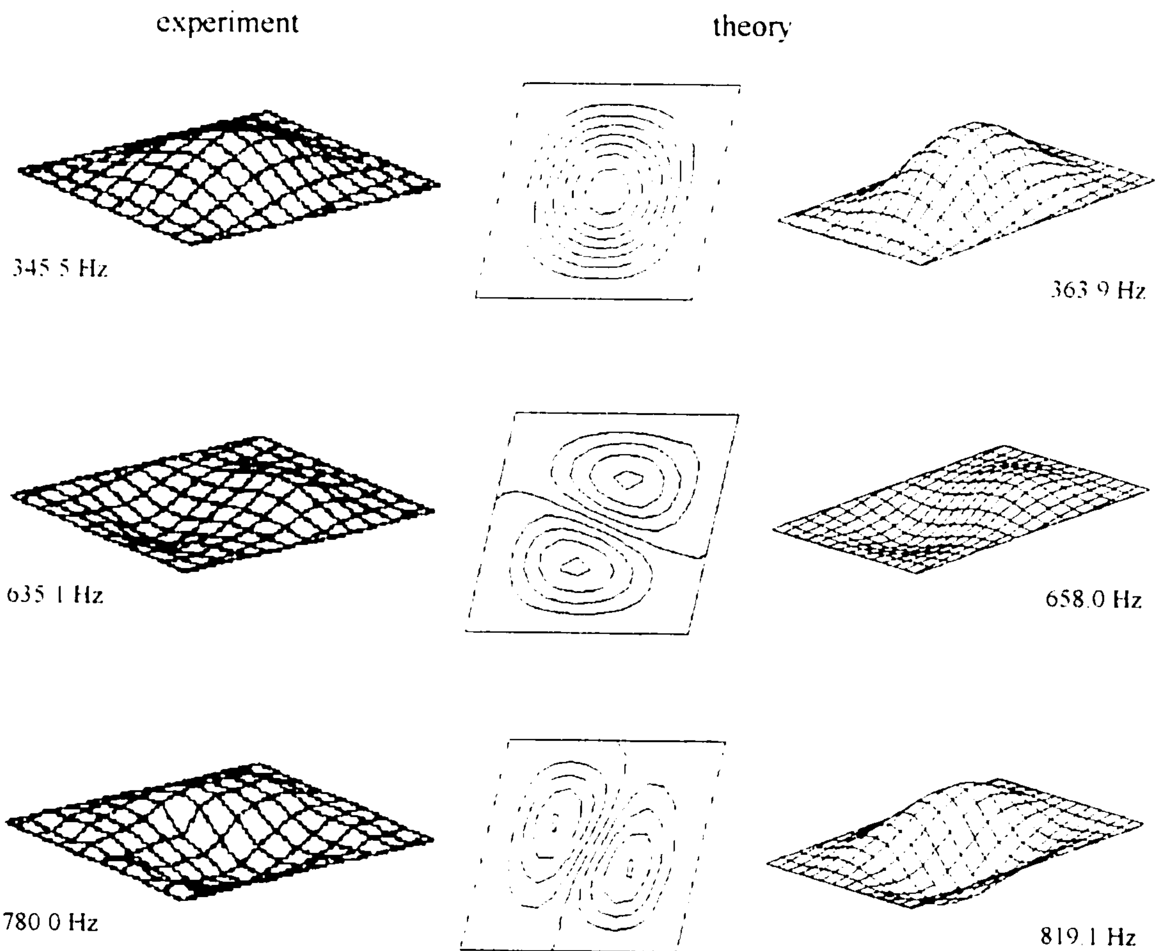


Figure 4-3 CCCC parallelogram plate, $\alpha = 15°$, $\phi_r = 1.60$.

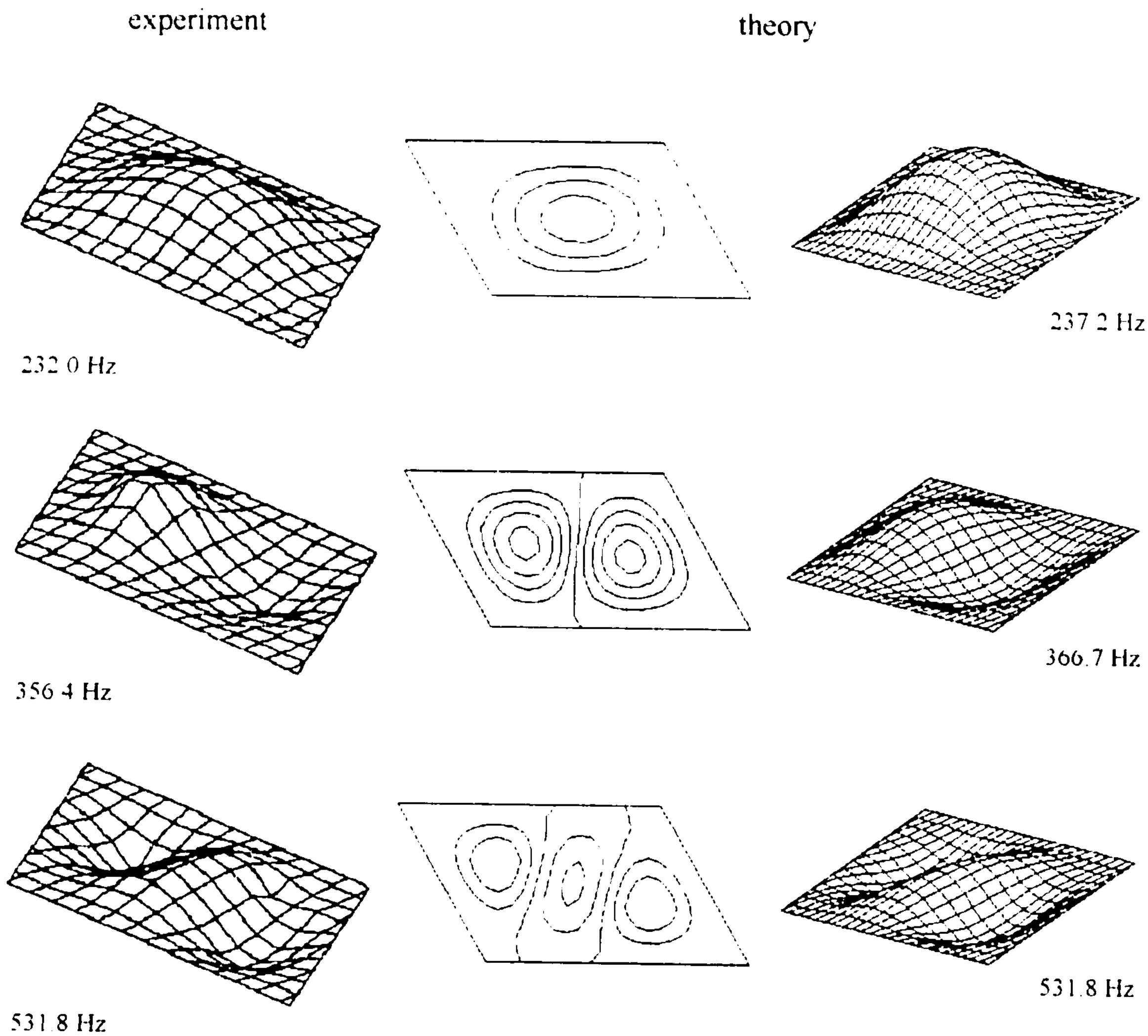


Figure 4-4 CCCC parallelogram plate, $\alpha = 30°$, $\phi_r = 1.00$.

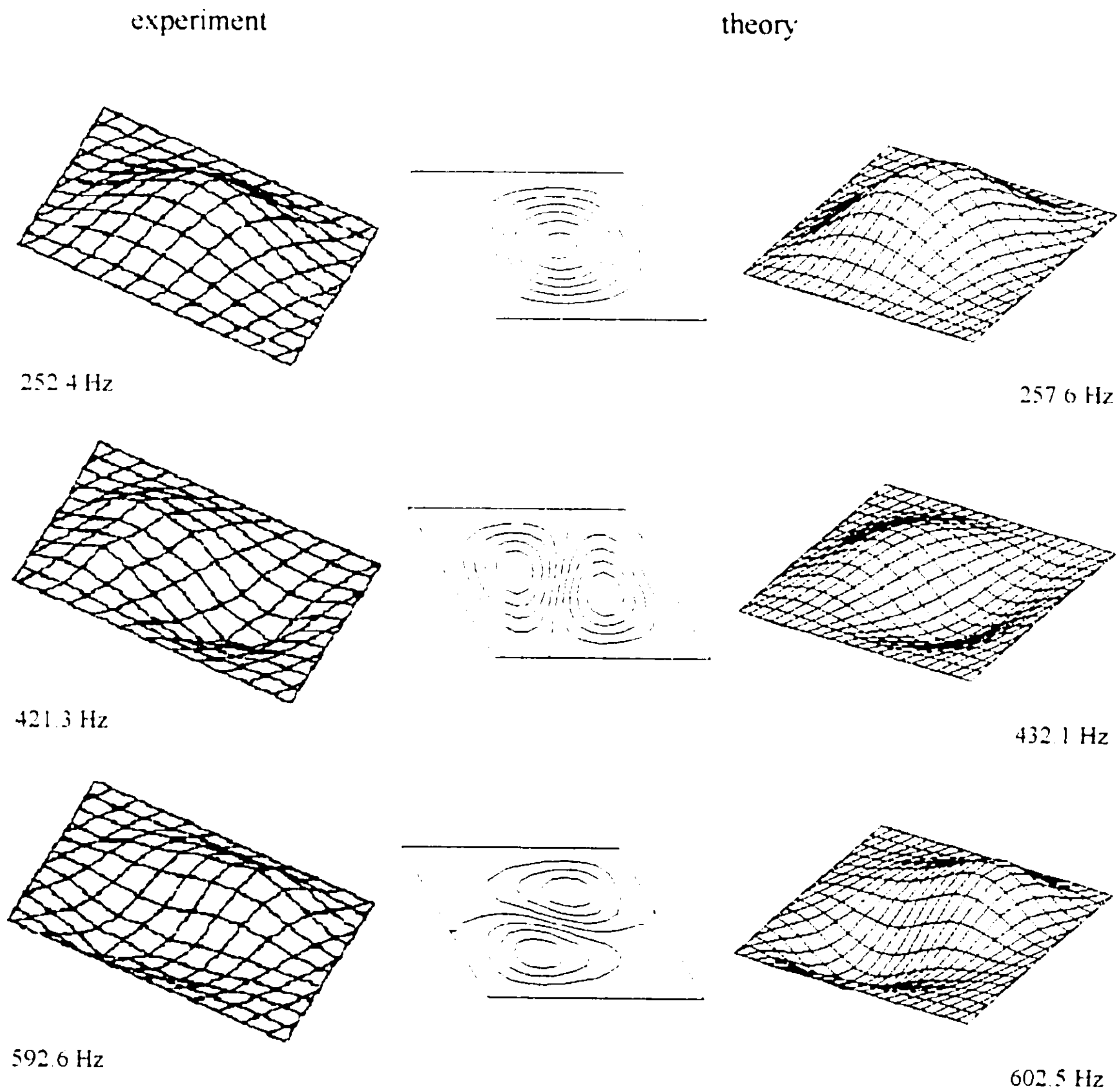


Figure 4-5 CCCC parallelogram plate, $\alpha = 30°$, $\phi_r = 1.25$

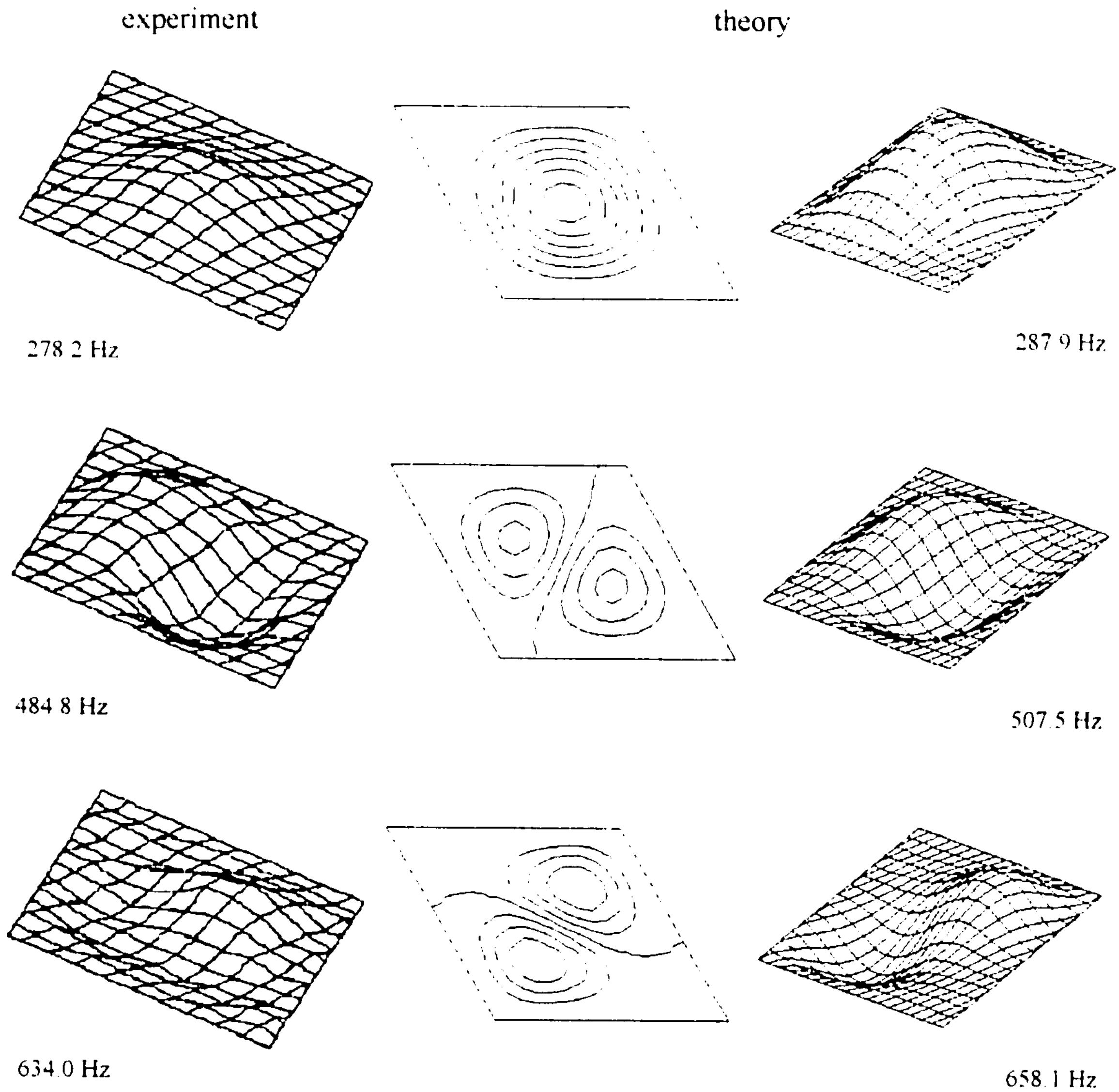


Figure 4-6 CCCC parallelogram plate, $\alpha = 30°$, $\phi_r = 1.60$

Table 4-1 CCCC parallelogram plates, α = 15°.

ϕ_t = 3.732		resonance frequency		
		experimental	theoretical	error
ϕ_r	mode	(Hz)	(Hz)	(%)
1.00	1	257.7	260.0	-0.90
	2	449.5	454.1	-1.01
	3	598.3	598.2	0.00
	4	712.1	721.6	-1.32
	5	818.7	831.4	-1.52
	6	1005	1020	-1.46
1.25	1	297.1	299.4	-0.77
	2	552.7	569.7	-2.98
	3	631.0	649.3	-2.82
	4	844.4	859.0	-1.70
	5	1047	1078	-2.93
	6	1129	1149	-1.75
1.60	1	345.5	363.9	-5.05
	2	635.1	657.9	-3.47
	3	780.0	819.1	-4.77
	4	1003	1029	-2.54
	5	1184	1206	-1.89
	6	1414	1461	-3.22

Table 4-2 CCCC parallelogram plates, α = 30°.

ϕ_t = 1.732		resonance frequency		
		experimental	theoretical	error
ϕ_r	mode	(Hz)	(Hz)	(%)
1.00	1	237.2	232.0	-2.18
	2	366.7	356.4	-2.81
	3	552.3	531.8	-3.72
	4	578.6	578.5	-0.02
	5	735.6	712.6	-3.12
	6	764.4	744.2	-2.63
1.25	1	257.6	252.4	-2.03
	2	432.1	421.3	-2.49
	3	605.5	592.6	-2.13
	4	645.2	622.5	-3.51
	5	855.9	829.5	-3.08
	6	889.5	873.4	-1.81
1.60	1	287.9	278.2	-3.36
	2	507.5	484.8	-4.48
	3	658.1	634.0	-3.65
	4	742.3	716.6	-3.46
	5	1027	985.3	-4.06
	6	1069	1130	-3.32

5 - Discussion

1. Comparison to Published Data

From the results presented in Table 3-6, it is clear that the results obtained by the superposition method agree with those obtained by other methods. In fact, since results of the superposition method can be considered analytical, the approximate methods can now be tested against analytically exact values. As researchers admit, solutions of the Rayleigh-Ritz type, and actually most others too, are upper bounds. Compared to the present results, this is confirmed for most of the values used in the comparison. Considering the analytical nature of the superposition results, the Rayleigh-Ritz results by Lee and Pon [8] are very close indeed.

A good check for any solution by the superposition method is another superposition solution. This can be a different combination of the building blocks at hand, a different segmentation of the plate in question, or the use of different building blocks. Gorman [25] segmented the simply supported plate differently from the present work, making use of the point symmetry. In addition, his solution used building blocks with cosine functions. In Tables 3-1 to 3-5, it is seen that the results are virtually identical to the present work.

2. Theoretical Analysis

The number of terms used for the expansions was $k = 12$ for most cases, and never more than 16, as no improvement in results was seen for a higher number of terms. In general, if more than twelve terms were needed, the aspect ratio ϕ_r was larger than unity, 1.5 or larger. The larger aspect ratios represent a shorter simple support along the base and the top edge for the rectangular segment. The rectangular segment tends to lend stability to the whole plate. This can be compared to the experimental results, where the smaller rectangular segments were associated with an increased experimental error.

The present method showed some problems with convergence at aspect ratios ϕ_r close to 0.5 and 2.0. These difficulties were the same, whether all six building blocks (see Figure 2-12) as used for $0.5 < \phi_r < 2.0$, or only the minimum four. This indicates the values of 0.5 and 2.0 are appropriate for changing the building blocks used. Well within the limits, and with aspect ratios much larger than 2.0, there were no problems with convergence. For the right triangular plate alone, the aspect ratio of 1.0 tends to present a problem with convergence because of symmetry. There appeared no problem with the parallelogram plate with an aspect ratio ϕ_t of unity.

Most CSSS plates can be analyzed as SSSC plates as well, or SCSC plates can be analyzed as CSCS plates. The decision on the approach depends on several factors. As mentioned, the size of the rectangular segment affects the convergence to a limited extent. The dimensions of the triangular segment appear to be more important in this author's experience. For aspect ratios $\phi_t = 1$ and larger, placing the clamping on the skew sides as opposed to the lower or upper edges makes little difference. For very small aspect ratios, representing high skew angles, the clamped skew edges show better convergence than the clamped base. The other consideration is of course computational effort. The computational effort required by simple support along the skew edge is the same as that for clamped support. The zero moment condition is changed to zero slope. To clamp the base or the top edge, two building blocks have to be added. For an expansion by $k = 12$ terms, and for $\phi_t < 0.5$, clamping the base increases the size of the eigenvalue matrix from 144 by 144 to 168 by 168.

With a high aspect ratio ϕ_r, the convergence for the eigenvalues was excellent. However the mode shapes were smoother for the smaller aspect ratios ϕ_r of the rectangular segment than for the larger ones, where the skew edges were longer than the base and top edges.

This is no problem for simply supported or clamped plates, where the plate can be rotated so that the former skew edge become the base

The plots of eigenvalues λ^2 versus the aspect ratios ϕ_t in Figures 3-29 and 3-30 for CCCC plate, as well as that for CSSC plate in Figures 3-32 and 3-33 for CSSC plates, reflect the characteristics of similar graphs for other plate sizes and support conditions. The eigenvalue for a specified mode number and aspect ratio ϕ_r decreases as the aspect ratio of the triangular segment ϕ_t increases. In other words, decreasing the skew angle increases the eigenvalue. The effect of an increasing aspect ratio of the rectangular segment ϕ_r for a constant size of the triangular segment is seen in Figure 3-31. The eigenvalue increases as the aspect ratio increases and the rectangular segment becomes smaller.

Close observation of the above mentioned graphs reveals places where the lines representing the higher modes appear to touch or even cross. Figure 3-34 zooms in on such an area for the CSSC plates of Figure 3-33. In these places the mode associated with the respective mode numbers switch, so that the shape belonging to mode 6, for example, is now associated with mode 5. Due to the changed dimensions of the plate, the new mode can now be accommodated before the mode shape previously associated with that mode number. A good example is found in Figures 3-26 and 3-27 for the CCCC plate, ϕ_t = 3.00: Modes 3 and 4 switch as the aspect ratio ϕ_t changes from 0.75 to 1.00. Nair and Durvasula [7] make the same observations, looking at the change of mode shapes with skew angle. They examine the mirror image of the problem, since the skew angle increases as ϕ_t decreases.

The mode shapes in Figures 3-1 to 3-28 are for fully clamped plates, however they are similar to those of plates with simple and combined supports for most cases except of course the slope along the edge. The zero-slope condition of the clamped support is obvious in mesh plots, as is the hinge-like action of the simple support. The eigenvalues for the simply supported and the fully clamped plate are lower and upper bounds respectively for plates of the same size with combined simple and clamped supports. This fact can be used to check results, and to reduce the eigenvalue search time.

3. Experimental Analysis

In the experimental setting, achieving the perfectly clamped condition, with absolutely no displacement and no rotation at the edge, was a critical element. The ideal condition can only be approached. Thus the experimental frequencies were expected to be lower than the theoretical ones, if they were not exactly the same. As such, they are lower bounds to the exact, analytical eigenfrequencies. Looking at the results of this work, it is indeed observed that the experimental values are consistently lower than the theoretical values. Previous approximate solutions to the vibration of thin flat quadrilateral plates are known to be upper bounds. Thus the results of these, along with the experimental results, support the accuracy of the superposition method, as applied to quadrilateral plates.

The mode shapes that were generated also correspond to those generated with the superposition method. Testing for each mode separately improved the resulting mode shapes over testing for several modes at one time. The results obtained for the corresponding resonance frequencies were unaffected. The results obtained by measuring the response at a few points were very close to those that would have been obtained by testing every grid point on the plate. The points chosen were in locations of high displacement as predicted by theory, and confirmed by the testing. The locations of nodal lines could be confirmed with spot checking as well. Checking only a few points saved a significant amount of testing time where the experimental mode shape was not of interest.

The maximum deviation of experimental resonance frequency from the theoretical counterpart was 4.77 %. The average error was 2.2 %. The trend was to have a higher deviation with an increasing aspect ratio ϕ_r, that is, with decreasing size of the rectangular segment. This phenomenon was to be expected, since the perimeter supported by the cross-supports relative to total perimeter of the plate was larger for the smaller plates. The cross-supports by their very nature represent a less rigid support than the concrete base halves that they span. Adding mass to the cross-supports would reduce this effect.

The error for each plate does not show a particular trend for most of the plates tested. The error for the $\alpha = 15°$, $\phi_r = 1.60$ plate decreases with increasing mode number. In general, a decreasing error indicates that the boundary conditions were not met very well. In the higher modes, the presence of an increased number of nodal lines reduces the role of the edge supports on the vibration behaviour of the plate, thus decreasing the error introduced in this case by loose clamping.

The plates were a big contributor to any error. Several plates showed some warping prior to mounting. When these plates were clamped, they were subjected to a bending moment which would then affect the vibration response of the plate. Any material defects incurred in the preparation of the plate or in the stock material were potential sources of error. Rolling the finished plates could have reduced some of the defects mentioned.

During the execution of the experiments, several potential sources of error were encountered, beyond those due to the plate supports. The plates were excited acoustically. It was impossible to filter out any noise other than the intended input from the speaker. Several nuisance sources of noise such as traffic, the ventilation system or a radio that could bias the input to the plate were present to some extent. Any effect on the outcome of the experiments was observed to be negligible, although the input amplitude of the noise was kept at or below that of conversational speech. Change in extraneous noise levels during the experiment would be more of a concern, as well as changes in the ambient temperature. Another challenge was the consistent mounting of the accelerometer as it has a very human component.

As mentioned earlier, adding more mass to the cross supports would increase the stiffness there. The alignment of the supports all around, an important factor, was excellent. The small gaps in support at the corners of the plate did not appear to induce any significant problems that would have become apparent in the mode shapes

The test method was very efficient, with excitation to the entire plate from the loudspeaker, rather than excitation of single points with an impact hammer. Mounting the plates on a static base rather than a shaker table allows for more rigid supports, and more variation in any support design.

6 - Conclusions

This work has applied the simplification that Saliba made to the building block superposition method originally developed by Gorman to an example of the thin quadrilateral plate with simple or clamped supports. It has been shown that the method presented provides a way of obtaining analytically exact eigenvalues for such plates. The parallelogram plate was chosen as example. No simplifying assumptions were made about the plate geometry or vibration behaviour in order to present a general solution. The eight building blocks used are obtained by rotating just two building blocks. The parallelogram plates are divided into right-triangular and rectangular segments. The building blocks for each segment are superimposed. Enforcing the boundary conditions along the plate perimeter and the conditions of continuity at the segment interfaces leads to a set of linear algebraic equations. Once the coefficient matrix is assembled, the eigenvalues are obtained quite easily and efficiently.

The method presented is easy to implement on a computer. It is user friendly and versatile. Various combinations of simple and clamped supports are easily accommodated. The method can be extended to more complicated quadrilateral plates that can be segmented into right-triangular and rectangular segments.

The experimental results presented serve as support for the results of the superposition method. Since results of the superposition method are considered analytically exact, they can also be used to say that the experimental analysis was successful, since the results are very close.

7 - References

[1] A.W. Leissa: *Vibration of Plates* National Aeronautics and Space Administration SP-160, 1969.

[2] A.W. Leissa: "Recent Research in Plate Vibration: Classical Theory", *Shock and Vibration Digest* **10**, 13-24 (1977).

[3] A.W. Leissa: "Plate Vibration Research, 1976-1980: Classical Theory", *Shock and Vibration Digest* **13**, 11-22 (1981).

[4] A.W. Leissa: "Recent Studies in Plate Vibrations: 1981-1965: Part I - Classical Theory", *Shock and Vibration Digest* **19**(2), 11-18 (1987).

[5] P.K. Kaul, E.V. Cadambe: "The Natural Frequencies of Thin Skew Plates", *Aeronautical Quarterly*, **74**(3), 337-352 (1956).

[6] D. Pnuelli: "Lower Bounds of the Gravest and All Higher Order Frequencies of Homogeneous Vibrating Plates of Arbitrary Shape", *Transactions of the ASME Journal of Applied Mechanics* 815-820 Dec. (1975).

[7] P.S. Nair, S. Durvasula: "Vibration of Skew Plates", *Journal of Sound and Vibration* **26**(1), 1-19 (1973).

[8] S. Durvasula: "Natural Frequencies and Modes of Clamped Skew Plates", *American Institute of Aeronautics and Astronautics Journal.* **7**(6), 1164-1167 (1969).

[9] S. Durvasula, P.S. Nair: "Application of the Partition Method to Vibration Problems of Plates", *Journal of Sound and Vibration* **37**(3), 429-445 (1974).

[10] T. Mizusawa, T. Kajita, M. Narvoka: "Vibration of Skew Plates by Using B-Spline Functions", *Journal of Sound and Vibration* **62**, 301-308 (1979).

[11] S.R. Kuttler, V.G. Sigillito: "Upper and Lower Bounds for Frequencies of Clamped Rhombical Plates", *Journal of Sound and Vibration* **68**(4), 597-607 (1980).

[12] K. Nagaya: "Vibration of Plate with Straight Line Boundaries", *Journal of Sound and Vibration* **68**, 35-43 (1980).

[13] T. Sakata: "Approximation Formulae for Natural Frequencies of Simply Supported Skew Plates", *Inernational Journal of Mechanical Sciences* **23**(11), 677-685 (1981).

[14] M.B. Subrahmanyam, T. Wah: "Vibration of Quadrilateral Plates", *Computer Methods in Applied Mechanics & Engineering* **48**(3), 315-323 (1984).

[15] G.N. Geannakas: "Vibration Analysis of Arbitrarily Shaped Plates Using Beam Characteristic Orthogonal Polynomials in the Semi-Analytical Finite Strip Method", *Journal of Sound and Vibration* **137**(2), 283-303 (1990).

[16] J.T. Katsikadelis: "A Boundary Element Solution to the Vibration Problem of Plates", *Journal of Sound and Vibration* **141**(2), 312-322 (1990).

[17] J.T. Katsikadelis, C.B. Kandilas: "A Flexibility Matrix Solution of the Vibration Problem of Plates based on the Boundary Element Method", *Acta Mechanica* **83**(1-2), 51-60 (1990).

[18] H. Matsuda, T. Sakiyama, C. Morita: "Fundamental Frequencies of Skew Plates with Mixed Boundary Conditions", *Journal of Sound and Vibration* **136**(3), 519-521 (1990).

[19] C.S. Hung, O.G. McGee, A.W. Leissa, J.W. Kim: "Accurate Vibration Analysis of Simply Supported Rhombic Plates by Considering Stress Singularities", *Transactions of the ASME Journal of Vibration, Acoustics, Stress and Reliability in Design* **117**(3A), 245-251 (1995).

[20] K.M. Liew, K.Y. Lam: "Application of 2-D Orthogonal Plate Function to Flexural Vibration of Skew Plates", *Journal of Sound and Vibration* **139**(2), 241-252 (1990).

[21] L.T. Lee, W.F. Pon: "Natural Frequencies of Parallelogrammic Plates using Characteristic Polynomials in Rayleigh-Ritz Method", *Computers in Engineering* **2**, 7-10 (1992).

[22] B. Singh, S. Chakraverty: "Flexural Vibration of Skew Plates Using Boundary Characteristic Orthogonal Polynomials in Two Variables", *Journal of Sound and Vibration* **173**(2), 157-178 (1994).

[23] X. Wang, A.G. Striz, C.W. Bert: "Buckling and Vibration Analysis of Skew Plates by the Differential Quadrature Method", *American Institute of Aeronautics and Astronautics Journal* **32**(4), 886-889 (1994).

[24] D.J. Gorman: "Accurate Free Vibration Analysis of Rhombic Plates with Simply Supported and Fully Clamped Edge Conditions", *Journal of Sound and Vibration* **125**(1), 281-290 (1988).

[25] D.J. Gorman: "Accurate Analytical Type Solutions For The Free Vibration Of Simply-Supported Parallelogram Plates", *Journal of Applied Mechanics* **58**, 203-208 (1991).

[26] D.J. Gorman: *Free Vibration Analysis of Rectangular Plates* New York: Elesevier North-Holland Inc. (1982).

[27] D.J. Gorman: "Free Vibration Analysis of Right Triangular Plates with Combinations of Clamped- Simply Supported Boundary Conditions". *Journal of Sound and Vibration* **106**, 419-431 (1986).

[28] D.J. Gorman: "A Modified Superposition Method for the Free Vibration Analysis of Right Triangular Plates", *Journal of Sound and Vibration* **112**, 173-176 (1987).

[29] H.T. Saliba: "Free Vibration Analysis of Simply Supported Symmetrical Trapezoidal Plates", *Journal of Sound and Vibration* **110**(1), 87-97 (1986).

[30] H.T. Saliba: "Transverse Free Vibration Analysis of Fully Clamped Symmetrical Trapezoidal Plates", *Journal of Sound and Vibration* **126**(2), 237-247 (1988).

[31] H.T. Saliba: *Free Vibration of Non-Rectangular Quadrilateral Plates*, Ph.D. Thesis, University of Ottawa (1986).

[32] H.T. Saliba: "Transverse Free Vibration Of Simply Supported Right Triangular Plates: A Highly Accurate Simplified Solution", *Journal of Sound and Vibration* **139**(2), 289-297 (1990).

[33] H.T. Saliba: "Vibration of Clamped Right Triangular Thin Plates: Accurate Simplified Solutions" *American Institute of Aeronautics and Astronautics Journal* **32**(12), 2486-2488 (1994).

[34] H.T. Saliba: "Transverse Free Vibration of Right Triangular Thin Plates with Combinations of Clamped and Simply Supported Boundary Conditions: A Highly Accurate Simplified Solution" *Journal of Sound and Vibration* **139**(2), 289-297 (1990).

[35] S. Timoshenko, S. Woinowsky-Krieger: *Theory of Plates and Shells*, McGraw-Hill Book Company, New York (1959).

APPENDICES

Appendix 1 - Frequently Used Integrals

The following integrals are commonly encountered in the Fourier expansions performed for the eight building blocks used in this work.

For the next three integrals, a ≠ b.

$$\int_0^1 \cos(ax)\cos(bx)\,dx = \frac{1}{2}\left[\frac{\sin(a-b)}{a-b}+\frac{\sin(a+b)}{a+b}\right]$$

$$\int_0^1 \sin(ax)\sin(bx)\,dx = \frac{1}{2}\left[\frac{\sin(a-b)}{a-b}-\frac{\sin(a+b)}{a+b}\right]$$

$$\int_0^1 \sin(ax)\cos(bx)\,dx = -\frac{1}{2}\left[\frac{\cos(a-b)}{a-b}+\frac{\cos(a+b)}{a+b}-\frac{2a}{a^2-b^2}\right]$$

$$\int_0^1 \sin(ax)\cosh(bx)\,dx = \frac{a(1-\cos a\cosh b)+b\sin a\sinh b}{a^2+b^2}$$

$$\int_0^1 \cos(ax)\sinh(bx)\,dx = \frac{b\cos a\cosh b+a\sin a\sinh b-b}{a^2+b^2}$$

$$\int_0^1 \cos(ax)\cosh(bx)\,dx = \frac{b\cos a\sinh b+a\sin a\cosh b}{a^2+b^2}$$

For the next three integrals, a ≠ c.

$$\int_0^1 \cos(ax+b)\cos(cx+d)\,dx = \frac{\sin(a-c+b-d)-\sin(b-d)}{2(a-c)} + \frac{\sin(a+b+c+d)-\sin(b+d)}{2(a+c)}$$

$$\int_0^1 \sin(ax+b)\sin(cx+d)\,dx = \frac{\sin(a+b-c-d)-\sin(b-d)}{2(a-c)} - \frac{\sin(a+b+c+d)-\sin(b+d)}{2(a+c)}$$

$$\int_0^1 \sin(ax+b)\cos(cx+d)\,dx = \frac{\cos(b-d)-\cos(a+b-c-d)}{2(a-c)} + \frac{\cos(b+d)-\cos(a+b+c+d)}{2(a+c)}$$

$$\int_0^1 \sinh(ax+b)\cos(cx+d)\,dx = a\,\frac{\cosh(a+b)\cos(c+d)-\cosh b\cos d}{a^2+c^2} + c\,\frac{\sinh(a+b)\sin(c+d)-\sinh b\sin d}{a^2+c^2}$$

$$\int_0^1 \cosh(ax+b)\sin(cx+d)\,dx = a\,\frac{\sinh(a+b)\sin(c+d)-\sinh b\sin d}{a^2+c^2} + c\,\frac{\cosh b\cos d-\cosh(a+b)\cos(c+d)}{a^2+c^2}$$

$$\int_0^1 \sinh(ax+b)\sin(cx+d)\,dx = a\,\frac{\cosh(a+b)\sin(c+d)-\cosh b\sin d}{a^2+c^2} + c\,\frac{\sinh b\cos d-\sinh(a+b)\sin(c+d)}{a^2+c^2}$$

$$\int_0^1 \cosh(ax+b)\cos(cx+d)\,dx = a\,\frac{\sinh(a+b)\cos(c+d)-\sinh b\cos d}{a^2+c^2} + c\,\frac{\cosh(a+b)\sin(c+d)-\cosh b\sin d}{a^2+c^2}$$

Appendix 2 - Experimental Investigation

1. Objective

The first six resonance frequencies were to be determined for fully clamped, thin parallelogram plates. For the first three modes of vibration the mode shapes were to be determined as well.

2. Set-up

The experimental set-up consists of a heavy concrete base, steel supports, broad spectrum speaker, measurement instrumentation, and the test plates. The basic arrangement is shown in Figure A2-1.

2.1 Test Base

The heavy, poured concrete base ensures that its own vibration response is insignificant in the frequency range in which the tests on the plates are carried out. Two blocks, each approximately 1 ft x 3 ft x 4 in, are connected by threaded steel rods at the bottom and a steel frame that can accommodate point supports at the top. A row of 1/2 in studs, 2 in on center, lines the top of each concrete block, along with a machined tool steel reference surface. The base halves are set at exactly 11.75 in apart. The maximum deviation of surface elevation along and between the reference surfaces is estimated to be 0.005 in. The base can be seen in Figures A2-3 and A2-5.

2.2 Longitudinal Supports

The longitudinal supports are 1.5 by 0.5 in bars which cover the length of the base. They have a round grove machined on one side to be used as simple support.

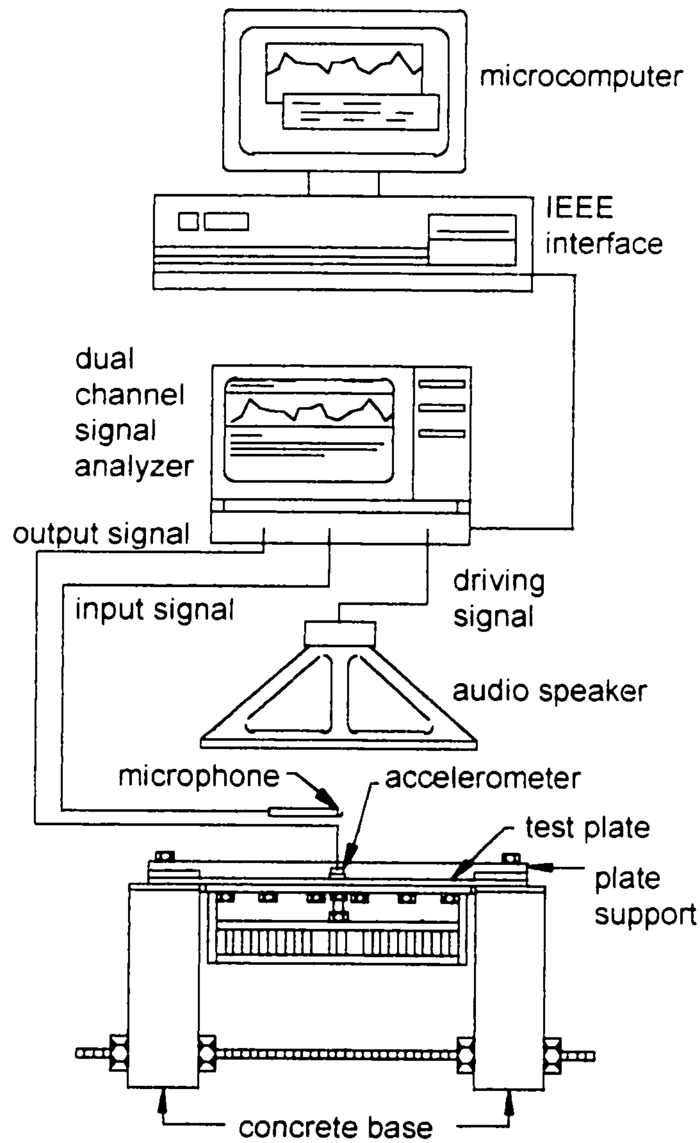


Figure A2-1 Overview of test set-up.

2.3 Cross Supports

The supports that span the base halves are made of 2 inch wide cold rolled steel stock. The surfaces of the 1.5 inch thick top and the 0.5 inch thick bottom that contact the clamped plate are machined. The lower half fastens to the top half by six 1/2 inch bolts. Alignment of the upper and lower halves for clamping is assured with two pins that were placed before the final machining of the side that constitutes the clamping edge. The cross supports also have a round grove machined into this face for testing simply supported plates.

2.4 Test Plates

The plates were cut from 0.125 in. thick aluminum 6061T stock. A two-inch perimeter was left beyond the dimensions of the area to be tested for clamping. Holes or slots were drilled to accommodate the studs on the base and the bolts and pins on the cross supports. The physical properties of the plate material that are relevant to the vibration response, $\sqrt{\rho / D}$, were tested for with cantilever plates cut from the same stock. This procedure is exlained detail in Appendix 3. The plates were marked with a rectangular grid on the top surface, as shown in Figure A2-2. The grid lines were between 0.75 and 1.5 inches apart.

Figure A2-2 Typical plate showing the test grid.

3. Mounting Procedure

The plate to be tested is placed onto the base and aligned with the base edges, as shown in Figure A2-3.

Figure A2-3 Test plate positioned on the base.

The longitudinal clamps are added next, as shown in Figure A2-4. Their alignment with the lower support on the base is critical to modelling the clamped support condition properly.

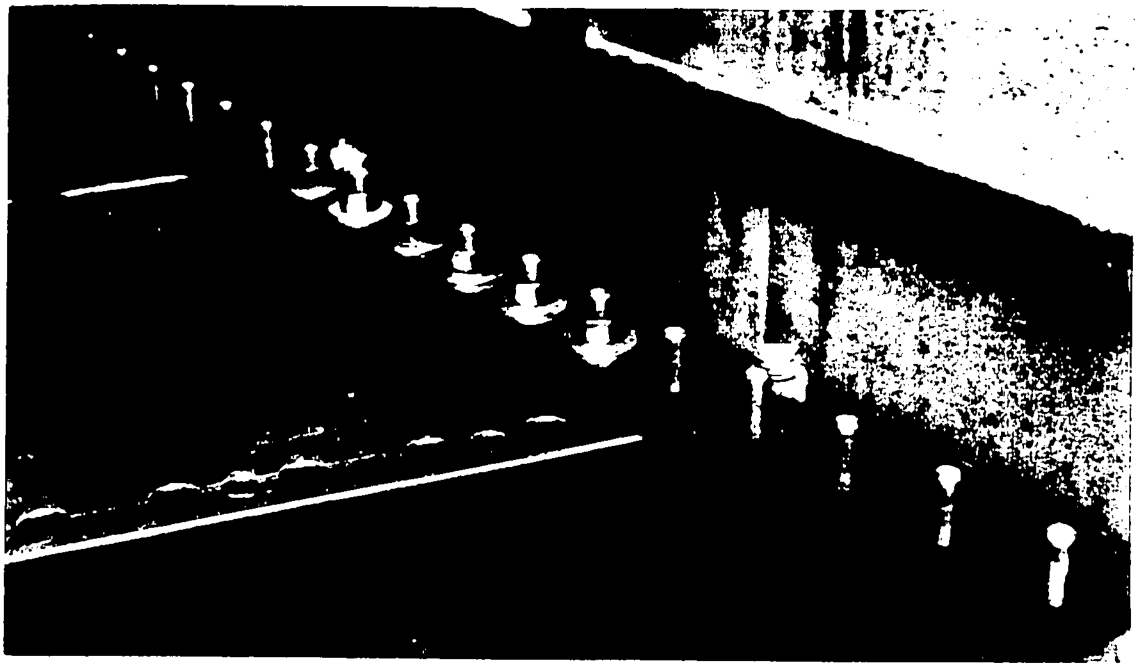


Figure A2-4 Longitudinal support in position.

Then the top section of the cross supports is added. Once this is aligned and tightened evenly, the bottom half is bolted on from the underside of the plate. Perfect alignment is guaranteed by two pins. Figure A2-5 shows a plate ready for testing.

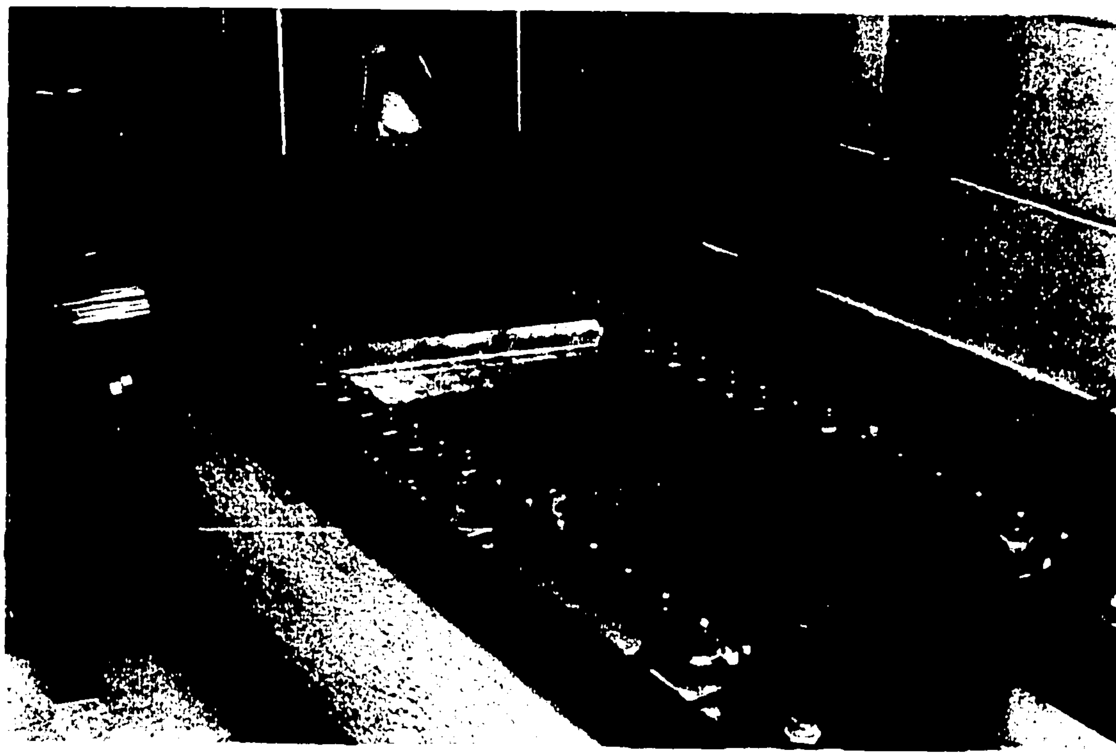


Figure A2-5 Fully mounted plate.

4. Instrumentation

The excitation to the test plate by a spectrum of white noise is provided with 15-inch audio speaker (Marsland Linear B 1500 VHP, 100 W maximum power output, operating range: 15-4000 Hz). The speaker is suspended above the plate, as seen in Figure A2-6, and is not moved during the test. In this way the entire plate is excited, eliminating the need to measure the vibration response to the excitation of various locations separately. It is also possible to test for several modes of vibration at one time. The input signal is

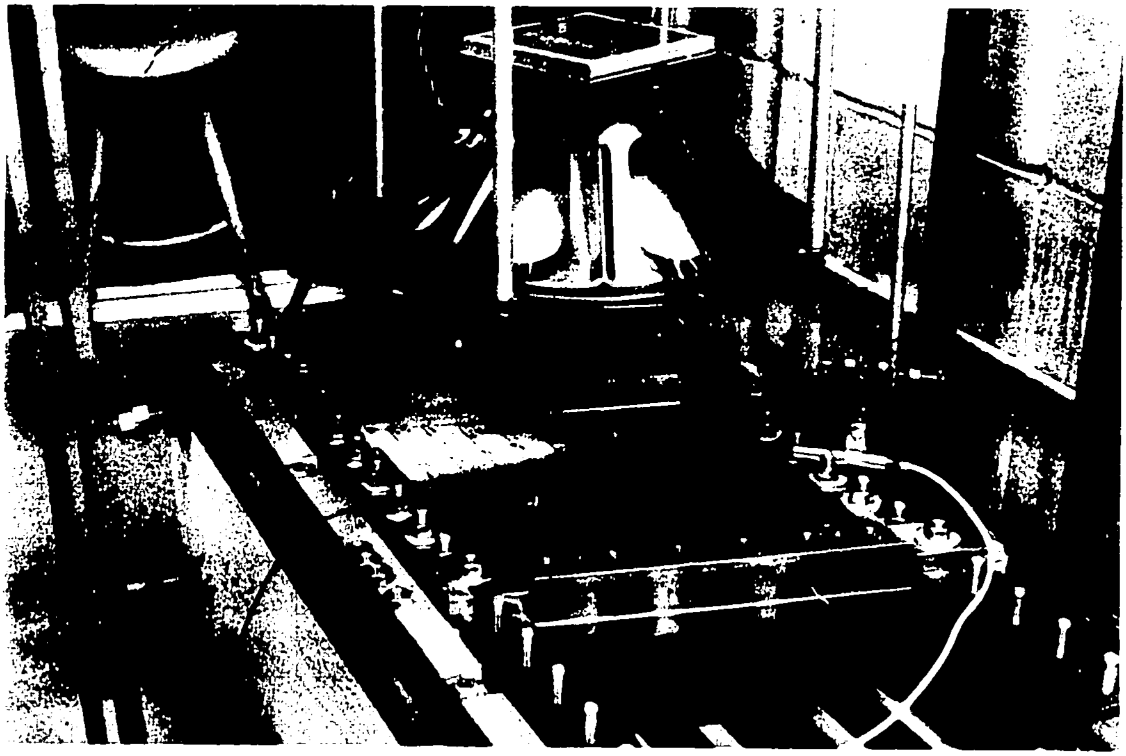


Figure A2-6 Speaker, microphone and accelerometer ready for testing, once the microphone is placed next to the accelerometer.

generated by the dual channel signal analyzer (B&K Type 2034), and routed through a power amplifier (Brüel & Kjær Type 2706). At the plate, the input signal is measured with a condenser microphone (B&K Type 2034). The vibration response of the plate is measured with an accelerometer (B&K Type 4393), amplified with a charge amplifier (B&K Type 2635). The accelerometer, seen in Figure A2-7, is wrung to the plate with a small amount of bees' wax. The microphone is positioned close to the accelerometer. Input and output signals are monitored with the dual signal channel analyzer (see Figure A2-8). The frequency response data can be transferred to a personal computer via an IEEE interface. The STAR Modal software package on the computer allows the generation of animated mode shapes. Figure A2-9 shows an example.

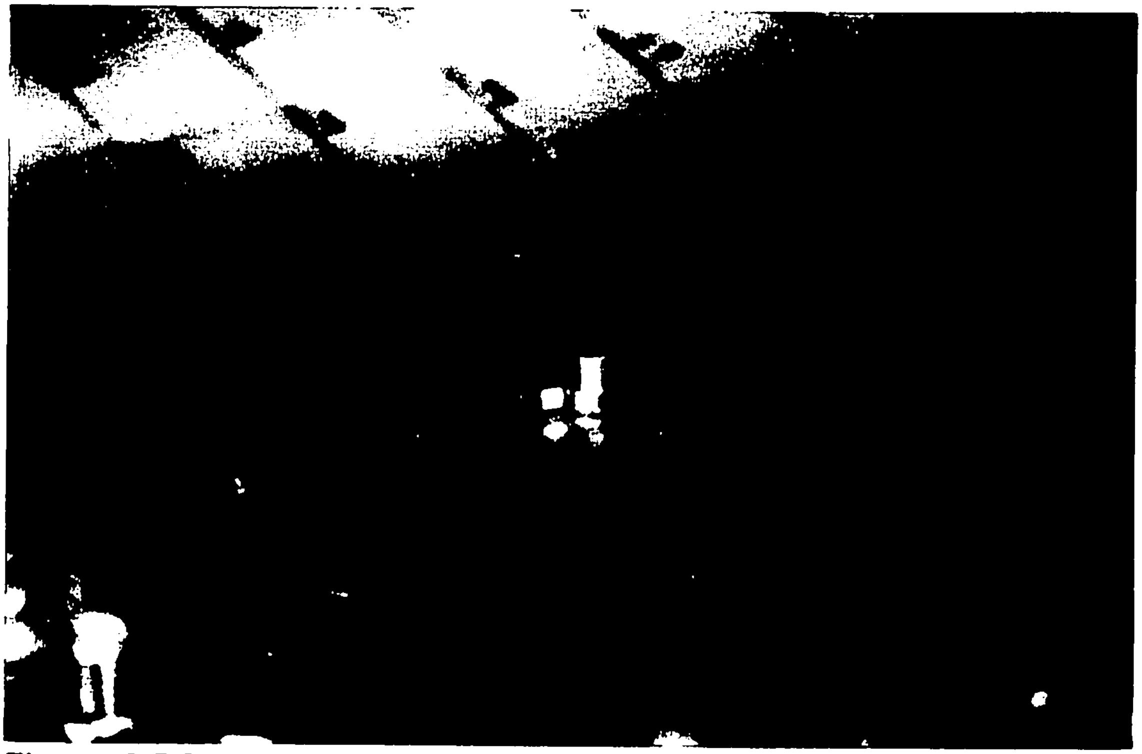


Figure A2-7 Close-up of the accelerometer.

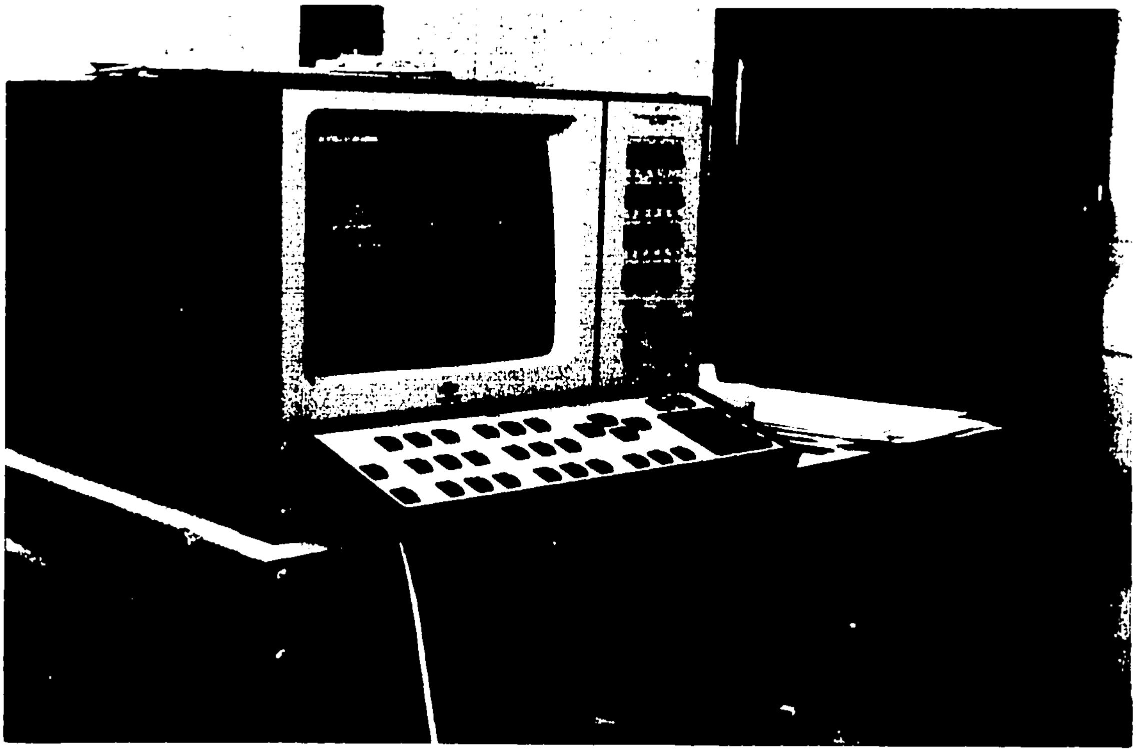


Figure A2-8 The dual channel signal analyzer.

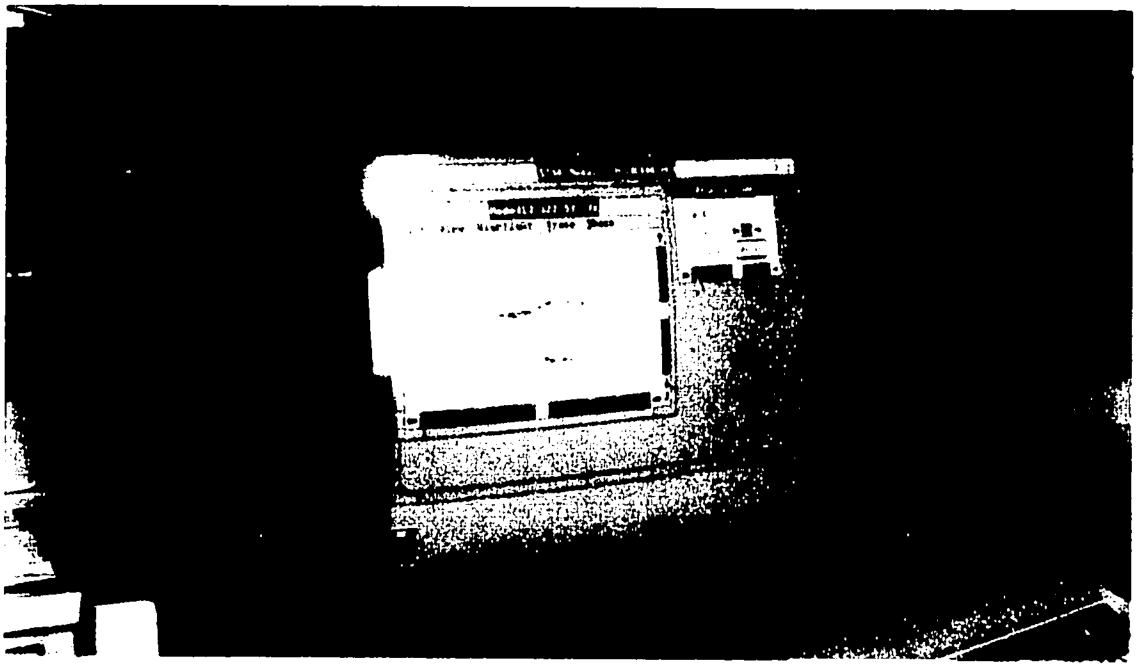


Figure A2-9 Animated mode shape produced by the STAR Modal software.

5. Experimental Procedure

The plates were prepared and mounted as described in sections 2 and 3. The instrumantation was put in place. The analyzer was set to linear averaging, Hanning weighting, and random noise generation. Each reading consisted of 100 samples. Each mode was tested for separately. The input frequency range was determined by taking some sample readings with a bandwidth of 200 to 800 Hz to esstimate the resonance frequencies. A smaller bandwidth of 25 to 100 Hz was then centered around the resonance frequency in question. The smaller bandwidth is desirable because of the finer resolution of the measurements. If two resonance frequencies were close together, the larger, 100 Hz range was used to capture results for both at the same time. The volume of the input signal was kept at an amplitude equivalent to or even less than that of conversational speech (about 50 dB). This varied slightly with the frequency. Once the frequency and amplitude were set, they were kept constant for that mode and plate.

Two sets of three plates each, one for a skew angle α = 15°, one for α = 30°, were tested. The reactangular segments had aspect ratios of ϕ_r = 1.00, 1.25 and 1.60 respectively. The first three modes of each plate were explored completely with mode shapes. For the fourth to sixth mode, only the frequencies were determined.

For the complete test that included the mode shape, a reading was taken at each point on the grid marked onto the plate. Each reading was transferred to the personal computer. The mode shape was then produced with the STAR Modal software.

For the frequency-only tests, readings were taken at a much smaller number of points and then averaged. These points were selected to be in the locations of maximum displacement for the mode in question. The locations of nodal lines were also confirmed with individual readings.

6. Results

The results are presented in Chapter 4.

Appendix 3 - Plate Property Check

1. Objective

The plate material property $\sqrt{\rho / D}$ was to be found to make a conversion between resonance frequencies and dimensionless eigenvalues possible.

2. Theory

The dimensionless eigenvalues are defined as $\lambda^2 = \omega a^2 \sqrt{\rho / D}$, where ρ and D are both properties of the plate material. Materials handbooks offer only ranges of values for material properties such as the density ρ and the modulus of elasticity E, and with reason: Due to variations in the manufacturing process, some of the material properties are changed slightly. By determining $\sqrt{\rho / D}$ directly, many steps and sources of accumulative error can be eliminated.

One plate configuration that is easily reproduced experimentally and for which theoretical eigenvalues are well established is the square cantilever plate. The associated mode shapes are also known. With the same test, the isotropy of the plate stock can be established by comparing two test plates cut in different orientations from the stock sheet. The cantilever plate has only one support, along one edge. This minimizes the error introduced by having a plate that is not perfectly flat, short of testing a free plate. The cantilever support is user friendly, and can be used with the same set-up as used for the actual tests.

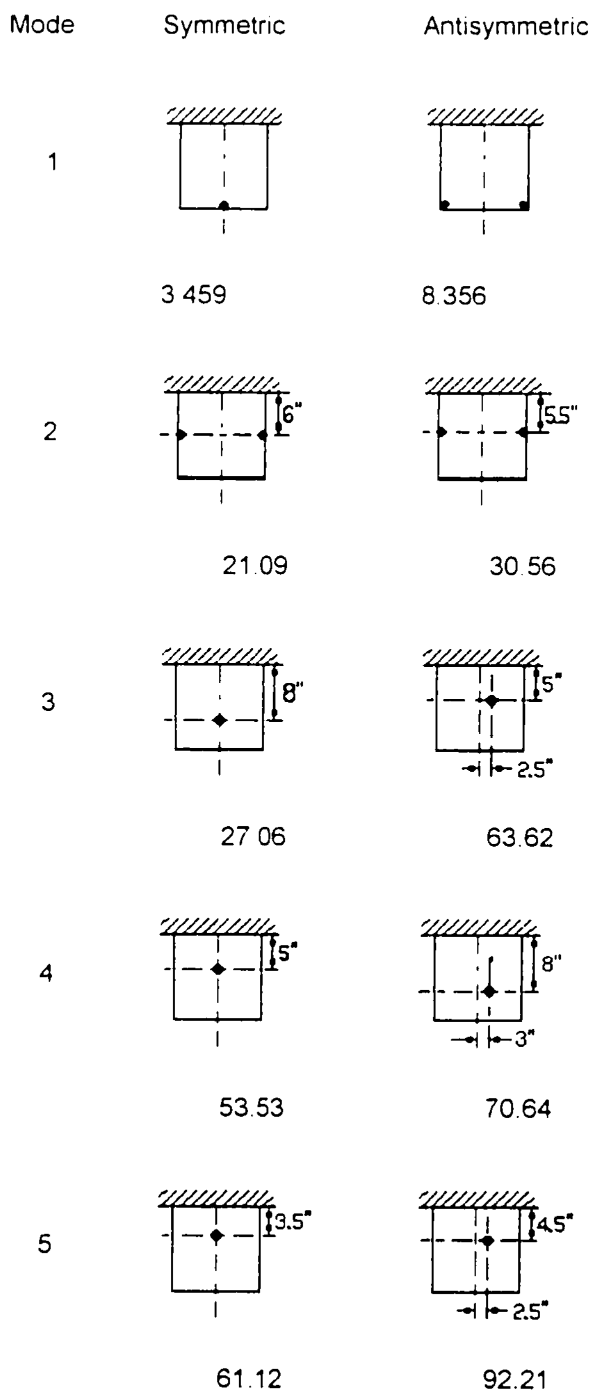


Figure A3-1 Optimum accelerometer positions for a 12-inch square cantilever plate with corresponding eigenvalues (from Reference [26]).

3. Set-up

The set-up is identical to that for the parallelogram plates, as explained in Appendix 2, except for the plate and the clamping. The plate is cut from the same 0.125 in thick aluminum 6061 T6 stock as the parallelogram plates were. The test area is 12 by 12 inches, with an extra 2-inch strip on one side to allow for the clamping. There are six holes in this strip to accommodate the bolts that hold the support halves together. The support consists of two halves that are bolted together, sandwiching the plate in-between. Each end of the of the cantilever support in turn is bolted onto one side of the concrete base. Positions of anticipated maximum deflection for the first five symmetric and the first five antisymmetric modes of vibration are marked onto the plate. The mounted cantilever plate is shown in Figure A3-2.

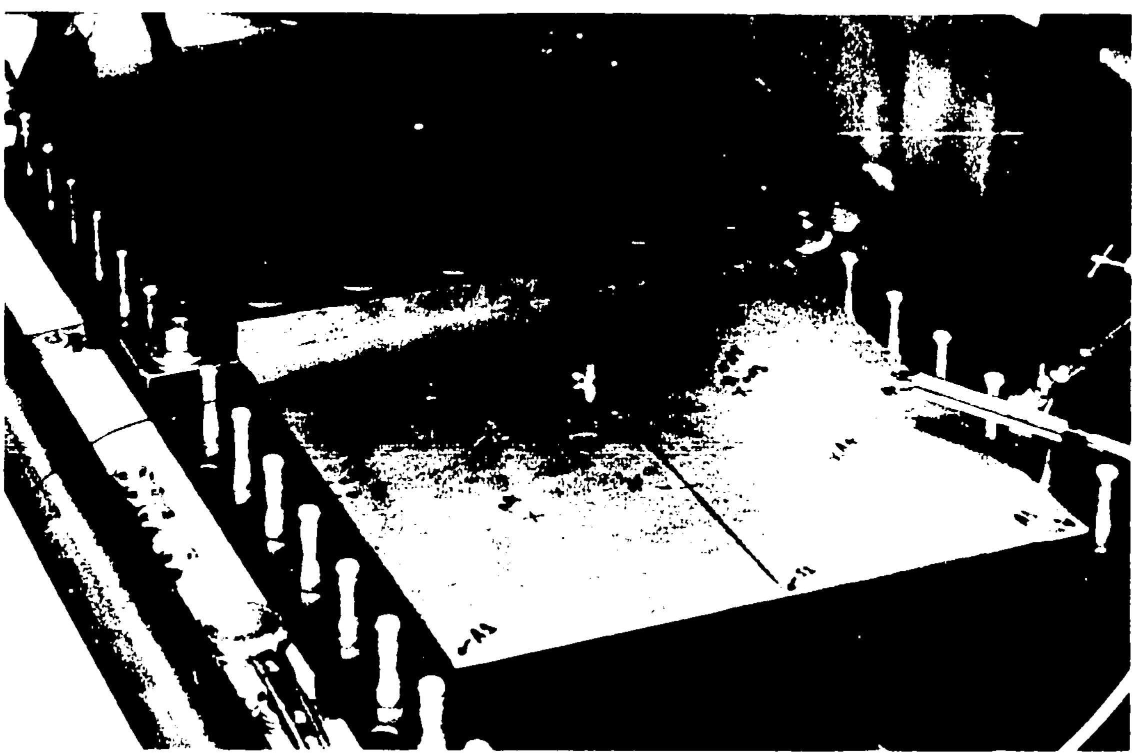


Figure A3-2 Square cantilever plate showing the location of the test points. 'S' indicates the location of the maximum amplitude of a symmetric mode; 'A', for an antisymmetric mode.

4. Procedure

The plate was mounted and marked as detailed in the previous section. The first five symmetric and the first five antisymmetric resonance frequencies were measured. The accelerometer was placed at the location of maximum amplitude for the mode. The plate was excited with a spectrum of random noise. After an initial reading, the bandwidth of the noise was narrowed to increase resolution, and centered close to the estimated resonance frequency. Then another reading was performed, and the frequency was recorded.

The value of $\sqrt{\rho / D}$ was calculated with the theoretical value for each mode. These were then averaged to come up with a final value.

5. Results

The plate property value was determined to be $1.3312*10^{-4}$s/in^2. Isotropy was a valid assumption for the plate stock.

Appendix 4 - Dimensioning of Parallelogram Plates

The choice for dimensioning a parallelogram plate depends on how convenient the convention is for the solution method. For the superposition method, as presented in this work, the choice appears obvious: the plate is labelled with the dimensions of the segments it is divided into. Even this method can lead to two possible descriptions, as shown later. For other methods, the overall dimensions are the more obvious choice. The rhombic plate, a special case of the parallelogram plate, can also be described in terms of its diagonals. The eigenvalues found by the various solutions usually have to be converted, before they can be compared, as they are non-dimensionalized with respect to different parts of the plate geometry. Some conversions are provided.

1. Alternate Dimensions

At first glance the choice for the segmentation of the parallelogram plate into one rectangular and two triangular segments appears obvious. However, many plates can be segmented in two ways, depending on which side is chosen as the base. Both possibilities are shown in Figure A4-1. The procedure for converting from one set of dimensions to the other follows its derivation.

The triangular aspect ratio ϕ_t remains the same, since the skew angle α remains the same as well:

$$\phi_t = \phi_t' \qquad \text{(A4-1)}$$

$$A + A_r = \frac{b}{\cos\alpha} \qquad \text{(a)}$$

$$B = (a + a_r)\cos\alpha \qquad \text{(b)}$$

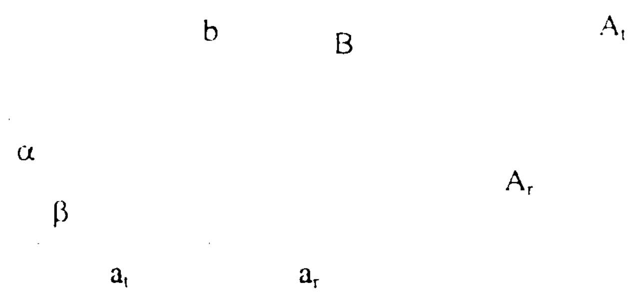


Figure A4-1 Parallelogram plate dimensioned in two ways.

Defining the aspect ratio of the alternate triangular segment as

$$\frac{B}{A_t} = \phi_t' \quad \Rightarrow \quad A_t = \frac{B}{\phi_t'} \quad \Rightarrow \quad \because \phi_t = \phi_t', \quad A_t = \frac{B}{\phi_t} \tag{c}$$

and that of the alternate rectangular segment as

$$\phi_r' = \frac{B}{A_r} \quad \Rightarrow \quad A_r = \frac{B}{\phi_r'}. \tag{d}$$

then substituting (c) and (d) into (a):

$$A_t + A_r = \frac{B}{\phi_t} + \frac{B}{\phi_r'} = \frac{b}{\cos\alpha}. \tag{e}$$

Substiting (b) into (e),

$$(a_t + a_r)\left(\frac{1}{\phi_t} + \frac{1}{\phi_r'}\right)\cos\alpha = \frac{b}{\cos\alpha}. \tag{f}$$

This can be simplified:

$$\frac{a_t + a_r}{b}\left(\frac{1}{\phi_t} + \frac{1}{\phi_r'}\right) = \frac{1}{\cos^2\alpha}$$

$$\left(\frac{a_t}{b} + \frac{a_r}{b}\right)\left(\frac{1}{\phi_t} + \frac{1}{\phi_r'}\right) = \frac{1}{\cos^2\alpha}$$

$$\left(\frac{1}{\phi_t} + \frac{1}{\phi_r}\right)\left(\frac{1}{\phi_t} + \frac{1}{\phi_r'}\right) = \frac{1}{\cos^2\alpha}$$

Then given ϕ_r', one can now solve for ϕ_r:

$$\frac{1}{\phi_r} = \frac{1}{\cos^2 \alpha}\left(\frac{1}{\phi_t} + \frac{1}{\phi_r'}\right)^{-1} - \frac{1}{\phi_t} \tag{A4-2}$$

One special case of the parallelogram plate is the plate that can be divided into two right triangular segments alone, without a rectangular segment. It can be analyzed as such, reducing the computational effort, or it can be looked at from the alternate perspective, as shown in Figure A4-2. In this case, $\phi_r' = \infty$. Equation (A4-1) reduces to

$$\frac{1}{\phi_r} = \frac{\phi_t}{\cos^2 \alpha} - \frac{1}{\phi_t} \tag{A4-3}$$

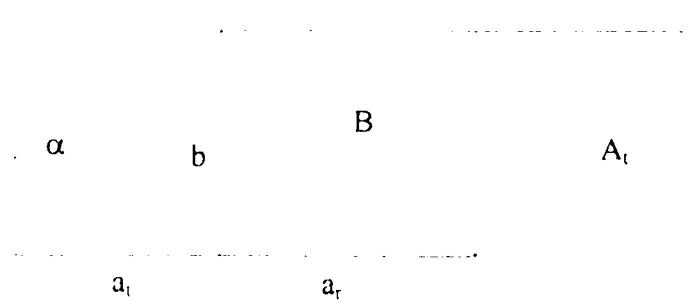


Figure A4-2 Parallelogram plate segmented into two right triangular plates.

2. Rhombic Plates

Rhombic plates are symmetric about both diagonals, and for the solution methods that make use of this, identifying the plates by the size of the part that is actually analyzed. Figure A4-3 shows how the rhombic plate is divided into right-triangular segments with aspect ratio $\phi^* = b^*/a^*$. From the figure, it can be seen that $b = c \cos \alpha$ and $a_t = c \sin \alpha$. The aspect ratio ϕ_t is then readily identified as

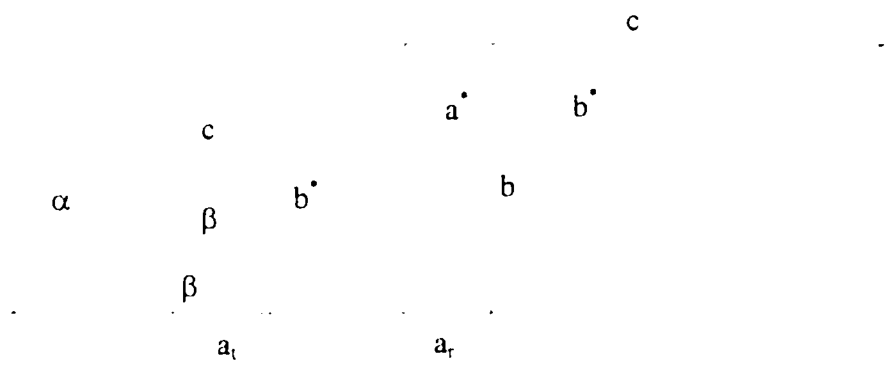


Figure A4-3 Dimensioning of rhombic plates.

$$\phi_t = \frac{h}{a_t} = \frac{1}{\tan\alpha}. \qquad \text{(A4-4)}$$

The next objective is to find an expression for ϕ_r. Since

$$a_r \quad c - a \quad c\,(1 - \sin\alpha),$$

$$\phi_r = \frac{h}{a_r} = \frac{c\cos\alpha}{c(1-\sin\alpha)}$$

and $$\phi_r = \frac{\cos\alpha}{1-\sin\alpha}. \qquad \text{(A4-5)}$$

It can be shown further by substituting $\frac{\pi}{2} - 2\beta$ for α in equation (A4-5) and simplifying, or by using similar triangles, that

$$\phi_r = \phi_R \qquad \text{(A4-6)}$$

Another useful relationship is

$$\phi_t = \frac{2\phi_R}{\phi_R^2 - 1}. \qquad \text{(A4-7)}$$

3. General Parallelogram Plates

The most common layout for parallelogram plates is that shown in Figure A4-4. The skew angle and the overall aspect ratio identify the plate. From the geometry it is seen that

$$b \quad B\cos\alpha \quad B\sin\beta,$$

$$a \quad B\sin\alpha \quad B\cos\beta.$$

and $$A \quad a_l \cdot a_r.$$

Then $$\phi_l = \frac{b}{a_l} = \frac{1}{\tan\alpha} = \tan\beta \qquad \text{(A4-8)}$$

and $$\frac{1}{\phi_r} = \frac{A}{B}\frac{1}{\cos\alpha} - \tan\alpha = \frac{1}{(B\ A)\cos\alpha} - \frac{1}{\phi_l} \qquad \text{(A4-9)}$$

When $(B\ \sin\alpha)$ becomes larger than A, the above conversion no longer works. Taking the B edge when this occurs eliminates the problem..

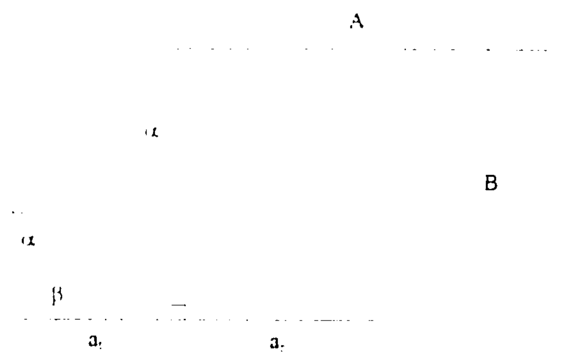


Figure A4-4 Common layout of the general parallelogram plate.

4. Eigenvalue Conversions

Almost every author has a different way of defining the eigenvalues. That may be convenient for their own work, but comparing results between authors becomes difficult. Finding a conversion factor requires a few steps. In general, the following works:

1. Identify the variables, especially the one for frequency.
2. Check the power of the frequencies: they must be the same. Square or take the square root of the eigenvalues, if necessary.
3. Divide one eigenvalue by the other.
4. Simplify with the help of geometry.

Some conversions are given, following a more detailed example for the derivation procedure.

For the case of alternate dimensions (see Section 1. of this appendix),

$$\frac{\lambda^2}{\lambda'^2} = \frac{\omega a_t^2 \sqrt{\rho / D}}{\omega A_t^2 \sqrt{\rho / D}}$$

$$= \frac{a_t^2}{A_t^2} = \frac{a_t^2 \cdot b^2}{A_t^2 \cdot b^2} = \frac{1}{\phi_t^2}\frac{b^2}{A_t^2}$$

$$= \frac{\cos^2\alpha}{\phi_t^2}\frac{(A_t + A_r)^2}{A_t^2}\frac{B^2}{B^2}$$

$$= \frac{\cos^2\alpha}{\phi_t^2}\frac{\left(\frac{1}{\phi_t} + \frac{1}{\phi_r}\right)^2}{\frac{1}{\phi_t'^2}}$$

$$\frac{\lambda^2}{\lambda'^2} = \left(\frac{1}{\phi_t} + \frac{1}{\phi_r}\right)^2 \cos^2\alpha \qquad \text{(A4-10)}$$

Alternatively,

$$\frac{\lambda^2}{\lambda'^2} = \left(\frac{1}{\phi_t} + \frac{1}{\phi_r}\right)^{-2} \cos^{-2}\alpha \qquad \text{(A4-11)}$$

Plate made of 2 triangular plates (see Section 1 , above)	$\frac{\lambda^2}{\lambda'^2} = \sin^2 \alpha$
Rhombic Plates [19,24]	$\frac{\lambda^2}{\lambda'^2} = \frac{2(1+\sin\alpha)}{\phi_r^2} = \frac{(1+\sin\alpha)(\phi_p^2-1)^2}{2\phi_p^2}$
Durvasula [7,8,9]	$\frac{\lambda^2}{k} = \frac{a_i^2}{A^2\ \pi^2} = \frac{\pi^2 A^2 \sin^2\alpha}{A^2} = \pi^2 \sin^2\alpha$
Kaul and Cadambe [5]	$\frac{\lambda^2}{\lambda'^2} = \frac{a_i^2}{A^2 \cos^2\alpha} = \frac{A^2\sin^2\alpha}{A^2\cos^2\alpha} = \tan^2\alpha = \frac{1}{\phi_t^2}$
Kuttler and Sigillito [11]	$\frac{\lambda^2}{\sqrt{\Omega}} = \frac{a_i^2}{1}$
Lee and Pon [21]	$\frac{\lambda^2}{\mu^2} = \frac{a_i^2}{A^2\ \pi^2} = \frac{\pi^2 A^2\sin^2\alpha}{A^2} = \pi^2\sin^2\alpha$
Subrahmanyam and Wah [14]	$\frac{\lambda^2}{\lambda'^2} = \frac{a_i^2}{A^2\sin^2\theta_a} = \frac{A^2\cos^2\beta}{A^2\sin^2\beta} = \frac{1}{\tan^2\beta} = \tan^2\alpha$

Appendix 5 - Program Listings

There are three main programs, each for different combinations of clamping for the base and the top of the parallelogram plate. They can easily be combined into a single program for ease of use. SBASE features a simply supported base and top; CBASE, a clamped base and simply supported top; CTOP, a clamped base and top. The support condition on the diagonal can be simple or clamped along either skew side with each of the three programs.

1. SBASE - Base And Top Are Simply Supported.

1.1 Main Program

```
C*********************************************************************
C   This is the main calling program for eigenvalue search and shape   *
C   data generation for parallelogram plates with base and top simply  *
C   supported.  Segments are arranged horizontally for this version.   *
C   Shape generation is included.                                      *
C             by S. Stangier                                           *
C*********************************************************************
C
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension a(320,320)
      dimension x(320),w(21,126),iclamp(2)
       poi = 0.3
      write(*,*) 'parashap.for: parallelogram plate - ssss,sssc, scsc'
      write(*,*) 'for left, right indicate clamping: 1=s, 2=c'
   10 write(*,20)
   20 format(' ',' phi     phir   k left right  almds    dlim     del',/)
      read(*,*) phi,phir,k,iclamp(1),iclamp(2),almds,dlim,del
       if (phi.lt.0.) goto 200
      write(*,30) phi,phir,k,iclamp(1),iclamp(2),almds,dlim,del
   30 format(' ',2(f7.5,1x),3(i2,1x),2(f9.5,1x),f7.5,/)
   32 c = 1.
      pi = 4.*datan(c)
      alpha = datan(1./phi)
      qlim = 360000000
   35 continue
       if (phi.le.0.5) then
             call phi_low (a,iclamp,nk)
       elseif (phi.ge.2.0) then
             call phi_hi (a,iclamp,nk)
       else
             call phi_mid (a,iclamp,nk)
       endif
c        do 62 ii = 1,nk
c   62   write(*,999) (a(ii,jj),jj=1,nk)
c        write(*,*) 'a is done'
       if (del.eq.0.) goto 190
```

```
c
c***********************************************************************
      call determ (a,nk,det)
c***********************************************************************
c
   61 write(*,65) almds,det
   65 format(' ',5x,f9.5,5x,e15.8)
      almds = almds+del
      if (almds.le.dlim) then
       goto 35
      else
       goto 10
      endif
  190 continue
       write(6,*) 'Number of grid points in eta-direction? (Max. 21)'
       read(5,*) ksl
       if (ksl.gt.21) ksl = 21
c         write(*,*) '    generating shape'
C
C***********************************************************************
      call detsol (a,nk,x)
C***********************************************************************
C
       x(nk) = -1
       if (phi.le.0.5) then
              call shape_lo (ksl,x,w)
       elseif (phi.ge.2.) then
              call shape_hi (ksl,x,w)
       else
              call shape_mi (ksl,x,w)
       endif
c        write(*,*) 'w:'
c        do 106 ii = 1,3*ksl-2
c  106    write(*,999) (w(ii,jj),jj=1,ksl)
      goto 10
  999 format(250(e10.3,1x))
  200 continue
      stop
      end
```

1.2 Subroutines For Assembling The Eigenvalue Matrix

```
c
c***********************************************************************
      subroutine phi_low (a,iclamp,nk)
c***********************************************************************
c
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension a(320,320), as(32,16), iclamp(2)
      tphi = phi
      talmds = almds
      nbb = 12
      ncon = 4
       kb = k
       ka = k
       ki = ka
       kj = ka
   35   iiii = 0
      do 63 iside = 1,2
      do 60 i = 1,ncon
          if (iclamp(iside).eq.1) then
           if (i.lt.3) then
             icon = 2*i-1
```

```
          nedge = 2
        else
          icon = i-(4-i)
          nedge = 1
        endif
       else
         icon = i
         if (i.lt.3) then
           nedge = 2
         else
           nedge = 1
         endif
       endif
    do 55 l = 1,nedge
      iedge = (12-2*iside)*l-(12-4*iside)
      jjjj = 0
      do 50 j = 1,nbb
            if ((iside.eq.2).and.(j.lt.5)) then
                ibb = 0
                call bb0 (as)
            elseif (j.lt.5) then
              if (j.lt.3) then
                ibb = j+2
                call bb34 (ibb,icon,as)
            else
                ibb = j+2
                call bb56 (ibb,icon,as)
              endif
            elseif (j.lt.9) then
c             rectangular segment
             almds = almds *phi*phi/(phir*phir)
            phi = phir
            if (iedge.ne.12) then
                  iedge = 6-iedge
                  if ((j.eq.5).or.(j.eq.6)) then
                   ibb = j-2
                   call bb34 (ibb,icon,as)
                else
                   ibb = j
                   call bb78 (ibb,icon,as)
                endif
                 call factor (as,icon,tphi)
                iedge = 6-iedge
            else
                  ibb = 0
                call bb0 (as)
            endif
            phi = tphi
               almds = talmds
             else
               if (iside.ne.2) then
                   ibb = 0
                   call bb0 (as)
               elseif (j.lt.11) then
                   ibb = j-8
                   call bb12 (ibb,icon,as)
               else
                   ibb = j-4
                   call bb78 (ibb,icon,as)
               endif
           endif

       do 40 ii = 1,ki
           do 40 jj = 1,kj
              iii = iiii+ii
              jjj = jjjj+jj
              a(iii,jjj) = as(ii,jj)
```

```
   40       continue
          jjjj = jjjj+kj
c      write(*,*) 'iedge = ',iedge,' iside = ',iside,' icon = ',icon,
c     1 ' ibb = ',ibb,'ki = ',ki,'kj = ',kj
c        do 41 ii = 1,ki
c   41  write(*,999) (as(ii,jj),jj=1,kj)
  990 format(8(i4,1x))
  999   format(70(e10.3,1x))
   50     continue
        iiii = iiii+ki
   55   continue
   60 continue
   63 continue
      nk = iiii
      return
      end


c
c***********************************************************************
       subroutine phi_mid (a,iclamp,nk)
c***********************************************************************
c
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension a(320,320), as(32,16), iclamp(2)
      tphi = phi
      talmds = almds
      nbb = 16
      ncon = 4
       kb = k
       ka = k
       ki = ka
       kj = ka
   35   iiii = 0
      do 63 iside = 1,2
      do 60 i = 1,ncon
          if (iclamp(iside).eq.1) then
           if (i.lt.3) then
             icon = 2*i-1
             nedge = 2
           else
             icon = i-(4-i)
             nedge = 1
           endif
          else
            icon = i
            if (i.lt.3) then
              nedge = 2
            else
              nedge = 1
            endif
          endif
        do 55 l = 1,nedge
         iedge = (12-2*iside)*l-(12-4*iside)
         if (iedge.eq.12) ki = 2*k
         jjjj = 0
          do 50 j = 1,nbb
               if ((iside.eq.2).and.(j.lt.7)) then
                  ibb = 0
                  call bb0 (as)
               elseif (j.lt.7) then
                 if (j.lt.3) then
                    ibb = j+2
                    call bb34 (ibb,icon,as)
                 elseif ((j.eq.3).or.(j.eq.4)) then
                   ibb = j+2
```

```
                  call bb56 (ibb,icon,as)
                 else
                     ibb = j+2
                     call bb78 (ibb,icon,as)
                 endif
               elseif (j.lt.11) then
c                rectangular segment
                almds = almds *phi*phi/(phir*phir)
               phi = phir
               if (iedge.ne.12) then
                     iedge = 6-iedge
                     if ((j.eq.7).or.(j.eq.8)) then
                      ibb = j-4
                      call bb34 (ibb,icon,as)
                     else
                      ibb = j-2
                      call bb78 (ibb,icon,as)
                     endif
                   call factor (as,icon,tphi)
                  iedge = 6-iedge
               else
                     ibb = 0
                  call bb0 (as)
               endif
               phi = tphi
                 almds = talmds
                else
                 if (iside.ne.2) then
                     ibb = 0
                     call bb0 (as)
                 elseif (j.lt.13) then
                     ibb = j-10
                     call bb12 (ibb,icon,as)
                 elseif (j.lt.15) then
                     ibb = j-10
                     call bb34 (ibb,icon,as)
                 else
                     ibb = j-8
                     call bb78 (ibb,icon,as)
                 endif
              endif
          do 40 ii = 1,ki
              do 40 jj = 1,kj
                 iii = iiii+ii
                 jjj = jjjj+jj
                 a(iii,jjj) = as(ii,jj)
   40     continue
          jjjj = jjjj+kj
c      write(*,*) 'iedge = ',iedge,' iside = ',iside,' icon = ',icon,
c     1 ' ibb = ',ibb,'ki = ',ki,'kj = ',kj
c       do 41 ii = 1,ki
c   41  write(*,999) (as(ii,jj),jj=1,kj)
  990 format(8(i4,1x),2(f10.3))
  999   format(70(e10.3,1x))
   50     continue
        iiii = iiii+ki
        if (iedge.eq.12) ki = k
   55   continue
   60 continue
   63 continue
      nk = iiii
      return
      end
```

```
c
c***********************************************************************
      subroutine phi_hi (a,iclamp,nk)
c***********************************************************************
c
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension a(320,320), as(32,16), iclamp(2)
      tphi = phi
      talmds = almds
      nbb = 12
      ncon = 4
       kb = k
       ka = k
       ki = ka
       kj = ka
   35  iiii = 0
c      write(*,*) 'phi > 2'
      do 63 iside = 1,2
        do 60 i = 1,ncon
          if (iclamp(iside).eq.1) then
           if (i.lt.3) then
             icon = 2*i-1
             nedge = 2
           else
             icon = i-(4-i)
             nedge = 1
           endif
          else
            icon = i
            if (i.lt.3) then
              nedge = 2
            else
              nedge = 1
            endif
          endif
         do 55 l = 1,nedge
           iedge = (12-2*iside)*l-(12-4*iside)
           jjjj = 0
           do 50 j = 1,nbb
                if ((iside.eq.2).and.(j.lt.5)) then
                    ibb = 0
                    call bb0 (as)
                elseif (j.lt.5) then
                  if (j.lt.3) then
                    ibb = j+2
                    call bb34 (ibb,icon,as)
                  else
                    ibb = j+4
                    call bb78 (ibb,icon,as)
                  endif
                elseif (j.lt.9) then
c                 rectangular segment
                 almds = almds *phi*phi/(phir*phir)
                phi = phir
                if (iedge.ne.12) then
                      iedge = 6-iedge
                      if ((j.eq.5).or.(j.eq.6)) then
                       ibb = j-2
                       call bb34 (ibb,icon,as)
                   else
                       ibb = j
                       call bb78 (ibb,icon,as)
                   endif
                    call factor (as,icon,tphi)
                   iedge = 6-iedge
                else
```

```
                    ibb = 0
                  call bb0 (as)
               endif
               phi = tphi
                 almds = talmds
               else
                 if (iside.ne.2) then
                    ibb = 0
                    call bb0 (as)
                 elseif (j.lt.11) then
                    ibb = j-6
                    call bb34 (ibb,icon,as)
                 else
                    ibb = j-4
                    call bb78 (ibb,icon,as)
                 endif
              endif
           do 40 ii = 1,ki
              do 40 jj = 1,kj
                 iii = iiii+ii
                 jjj = jjjj+jj
                  a(iii,jjj) = as(ii,jj)
 40         continue
            jjjj = jjjj+kj
990  format(8(i4,1x),2(f10.3))
999     format(70(e10.3,1x))
 50         continue
           iiii = iiii+ki
 55       continue
 60     continue
 63  continue
     nk = iiii
     return
     end
```

1.3 Subroutines For Shape Generation

```
cc*********************************************************************
       subroutine shape_lo (ks1,x,w)
c**********************************************************************
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension x(320),xi(126),eta(21),incode(21,126)
      dimension el3m(16),el4m(16),el5m(16),el6m(16)
      dimension em3m(16),em4m(16),em7m(16),em8m(16)
      dimension er1m(16),er2m(16),er7m(16),er8m(16)
      dimension w(21,126)
      character*14 outfile
c       write(*,*) ' entering shape_lo'
      ka = k
c       write(6,*)(x(i),i=1,ka)
      do 10 i = 1,ka
         el3m(i) = x(i)
         el4m(i) = x(ka+i)
         el5m(i) = x(2*ka+i)
         el6m(i) = x(3*ka+i)
         em3m(i) = x(4*ka+i)
         em4m(i) = x(5*ka+i)
         em7m(i) = x(6*ka+i)
         em8m(i) = x(7*ka+i)
         er1m(i) = x(8*ka+i)
         er2m(i) = x(9*ka+i)
         er7m(i) = x(10*ka+i)
         er8m(i) = x(11*ka+i)
 10  continue
```

```
      bigphi = phi*phir/(2.0*phir+phi)
      ksy = ksi
      ksx = int((ksy-1.0)/bigphi + 2.0)
      nt = int(bigphi*(ksx-1)/phi)+1
      nr = ksx - nt
      if (ksx.gt.126) ksx = 126
       do 15 i = 1, ksy
         do 15 j = 1, ksx
           w(i,j) = 0.0
   15  continue
cc*********************************************************************
       call grid (ksx,ksy,xi,eta,nt,nr,incode)
c**********************************************************************
       call shape34 (1,nt,ksy,ka,el3m,el4m,xi,eta,incode,w)
       call shape56 (1,nt,ksy,ka,el5m,el6m,xi,eta,incode,w)
       tphi = phi
       talmds = almds
       almds = almds*phi*phi/(phir*phir)
       phi = phir
       call shape34 (nt+1,nr,ksy,ka,em3m,em4m,xi,eta,incode,w)
       call shape78 (nt+1,nr,ksy,ka,em7m,em8m,xi,eta,incode,w)
       do 20 i = 1, ksy
         do 20 j = nt+1, nr
           w(i,j) = -1.0*w(i,j)
   20  continue
       almds = talmds
       phi = tphi
       call shape12 (nr+1,ksx,ksy,ka,er1m,er2m,xi,eta,incode,w)
       call shape78 (nr+1,ksx,ksy,ka,er7m,er8m,xi,eta,incode,w)
       write(6,*) 'Output file? (Use quotation marks)'
       read(5,*) outfile
       write(6,*) 'Writing displacement matrix to file ',outfile
       open(25,file=outfile,status='new')
       do 106 jj = 1,ksx
  106   write(25,999) (w(ii,jj),ii=1,ksy)
  999   format (21(e10.3,1x))
       close(25)
c        write(*,*) 'leaving shape_lo'
       return
       end



cc*********************************************************************
       subroutine shape_mi (ks1,x,w)
c**********************************************************************
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension x(320),xi(126),eta(21),incode(21,126)
      dimension el3m(16),el4m(16),el5m(16),el6m(16),el7m(16),el8m(16)
      dimension em3m(16),em4m(16),em7m(16),em8m(16)
      dimension er1m(16),er2m(16),er3m(16),er4m(16),er7m(16),er8m(16)
      dimension w(21,126)
      character*14 outfile
c       write(*,*) ' entering shape_mi'
      ka = k
c       write(6,*)(x(i),i=1,16*ka)
      do 10 i = 1,ka
         el3m(i) = x(i)
         el4m(i) = x(ka+i)
         el5m(i) = x(2*ka+i)
         el6m(i) = x(3*ka+i)
         el7m(i) = x(4*ka+i)
         el8m(i) = x(5*ka+i)
         em3m(i) = x(6*ka+i)
         em4m(i) = x(7*ka+i)
```

```
          em7m(i) = x(8*ka+i)
          em8m(i) = x(9*ka+i)
          er1m(i) = x(10*ka+i)
          er2m(i) = x(11*ka+i)
          er7m(i) = x(12*ka+i)
          er8m(i) = x(13*ka+i)
          er3m(i) = x(14*ka+i)
          er4m(i) = x(15*ka+i)
   10  continue
       bigphi = phi*phir/(2.0*phir+phi)
       ksy = ks1
       ksx = int((ksy-1.0)/bigphi + 2.0)
       nt = int(bigphi*(ksx-1)/phi)+1
       nr = ksx - nt
       if (ksx.gt.126) ksx = 126
        do 15 i = 1, ksy
          do 15 j = 1, ksx
            w(i,j) = 0.0
   15   continue
cc**********************************************************************
        call grid (ksx,ksy,xi,eta,nt,nr.incode)
c***********************************************************************
        call shape34 (1,nt,ksy,ka,el3m,el4m,xi,eta,incode,w)
        call shape56 (1,nt,ksy,ka,el5m,el6m,xi,eta,incode,w)
        call shape78 (1,nt,ksy,ka,el7m,el8m,xi,eta,incode,w)
c        write(6,*) 'in hshap_mi, done first segment'
        tphi = phi
        talmds = almds
        almds = almds*phi*phi/(phir*phir)
        phi = phir
        call shape34 (nt+1,nr,ksy,ka,em3m,em4m,xi,eta,incode,w)
        call shape78 (nt+1,nr,ksy,ka,em7m,em8m,xi,eta,incode,w)
        do 20 i = 1, ksy
          do 20 j = nt+1, nr
            w(i,j) = -1.0*w(i,j)
   20   continue
        almds = talmds
        phi = tphi
c        write(6,*) 'in hshap_mi, done 2nd segment'
        call shape12 (nr+1,ksx,ksy,ka,er1m,er2m,xi,eta,incode,w)
        call shape34 (nr+1,ksx.ksy,ka,er3m,er4m,xi,eta,incode,w)
        call shape78 (nr+1,ksx,ksy,ka,er7m,er8m,xi,eta,incode,w)
        write(6,*) 'Output file? (Use quotation marks)'
        read(5,*) outfile
        write(6,*) 'Writing displacement matrix to file ',outfile
        open(25,file=outfile,status='new')
        do 106 jj = 1,ksx
  106    write(25,999) (w(ii,jj),ii=1,ksy)
  999    format (21(e10.3,1x))
        close(25)
c        write(*,*) 'leaving shape_mi'
        return
        end



cc**********************************************************************
        subroutine shape_hi (ks1,x,w)
c***********************************************************************
       implicit real*8 (a-h,o-z)
       common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
       dimension x(320),xi(126),eta(21),incode(21,126)
       dimension el3m(16),el4m(16),el7m(16),el8m(16)
       dimension em3m(16),em4m(16),em7m(16),em8m(16)
       dimension er3m(16),er4m(16),er7m(16),er8m(16)
       dimension w(21,126)
```

```
      character*14 outfile
c       write(*,*) ' entering shape_hi'
      ka = k
c       write(6,*)(x(i),i=1,ka)
      do 10 i = 1,ka
         el3m(i) = x(i)
         el4m(i) = x(ka+i)
         el7m(i) = x(2*ka+i)
         el8m(i) = x(3*ka+i)
         em3m(i) = x(4*ka+i)
         em4m(i) = x(5*ka+i)
         em7m(i) = x(6*ka+i)
         em8m(i) = x(7*ka+i)
         er3m(i) = x(8*ka+i)
         er4m(i) = x(9*ka+i)
         er7m(i) = x(10*ka+i)
         er8m(i) = x(11*ka+i)
 10   continue
      bigphi = phi*phir/(2.0*phir+phi)
      ksy = ks1
      ksx = int((ksy-1.0)/bigphi + 2.0)
      nt = int(bigphi*(ksx-1)/phi)+1
      nr = ksx - nt
      if (ksx.gt.126) ksx = 126
      do 15 i = 1, ksy
        do 15 j = 1, ksx
          w(i,j) = 0.0
 15   continue
cc**********************************************************************
      call grid (ksx,ksy,xi,eta,nt,nr,incode)
c***********************************************************************
      call shape34 (1,nt,ksy,ka,el3m,el4m,xi,eta,incode,w)
      call shape78 (1,nt,ksy,ka,el7m,el8m,xi,eta,incode,w)
      tphi = phi
      talmds = almds
      almds = almds*phi*phi/(phir*phir)
      phi = phir
      call shape34 (nt+1,nr,ksy,ka,em3m,em4m,xi,eta,incode,w)
      call shape78 (nt+1,nr,ksy,ka,em7m,em8m,xi,eta,incode,w)
      do 20 i = 1, ksy
        do 20 j = nt+1, nr
          w(i,j) = -1.0*w(i,j)
 20   continue
      almds = talmds
      phi = tphi
      call shape34 (nr+1,ksx,ksy,ka,er3m,er4m,xi,eta,incode,w)
      call shape78 (nr+1,ksx,ksy,ka,er7m,er8m,xi,eta,incode,w)
      write(6,*) 'Output file? (Use quotation marks)'
      read(5,*) outfile
      write(6,*) 'Writing displacement matrix to file ',outfile
      open(25,file=outfile,status='new')
      do 106 jj = 1,ksx
 106  write(25,999) (w(ii,jj),ii=1,ksy)
 999  format (21(e10.3,1x))
      close(25)
c       write(*,*) 'leaving shape_hi'
      return
      end
```

2. CBASE - Base clamped, Top Edge Simply Supported

2.1 Main program

```
C***********************************************************************
C   This is the main calling program for eigenvalue search and shape   *
C   data generation for parallelogram plates with clamped base and     *
C   simply supported top edge.                                         *
C   Segments are arranged horizontally for this version. Shape         *
C   generation is included.                                            *
C              by S. Stangier                                          *
C***********************************************************************
C
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension a(320,320)
      dimension x(320),w(21,126),iclamp(2)
       poi = 0.3
      write(*,*) 'parashap.for: parallelogram plate - csss,cssc, ccsc'
      write(*,*) 'for left, right indicate clamping: 1=s, 2=c'
   10 write(*,20)
   20 format(' ',' phi     phir    k left right  almds     dlim     del',/)
      read(*,*) phi,phir,k,iclamp(1),iclamp(2),almds,dlim,del
       if (phi.lt.0.) goto 200
      write(*,30) phi,phir,k,iclamp(1),iclamp(2),almds,dlim,del
   30 format(' ',2(f7.5,1x),3(i2,1x),2(f9.5,1x),f7.5,/)
   32 c = 1.
      pi = 4.*datan(c)
      alpha = datan(1./phi)
      qlim = 360000000
   35 continue
       if (phi.le.0.5) then
              call bphi_low (a,iclamp,nk)
       elseif (phi.ge.2.0) then
              call bphi_hi (a,iclamp,nk)
       else
              call bphi_mid (a,iclamp,nk)
       endif
c        do 62 ii = 1,nk
c   62   write(*,999) (a(ii,jj),jj=1,nk)
c        write(*,*) 'a is done'
       if (del.eq.0.) goto 190
c
c***********************************************************************
      call determ (a,nk,det)
c***********************************************************************
c
   61 write(*,65) almds,det
   65 format(' ',5x,f9.5,5x,e15.8)
      almds = almds+del
      if (almds.le.dlim) then
       goto 35
      else
       goto 10
      endif
  190 continue
       write(6,*) 'Number of grid points in eta-direction? (Max. 21)'
       read(5,*) ks1
       if (ks1.gt.21) ks1 = 21
c        write(*,*) '    generating shape'
C
C***********************************************************************
      call detsol (a,nk,x)
C***********************************************************************
```

```
c
        x(nk) = -1
        if (phi.le.0.5) then
              call bshap_lo (ks1,x,w)
        elseif (phi.ge.2.) then
              call bshap_hi (ks1,x,w)
        else
              call bshap_mi (ks1,x,w)
        endif
c        write(*,*) 'w:'
c        do 106 ii = 1,3*ks1-2
c  106    write(*,999) (w(ii,jj),jj=1,ks1)
        goto 10
  999 format(250(e10.3,1x))
  200 continue
      stop
      end
```

2.2 Subroutines for matrix assembly with CBASE

```
c*********************************************************************
       subroutine bphi_low (a,iclamp,nk)
c*********************************************************************
c
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension a(320,320), as(32,16), iclamp(2)
      tphi = phi
      talmds = almds
      nbb = 14
       kb = k
       ka = k
       ki = ka
       kj = ka
   35   iiii = 0
c     write(*,*) 'phi > 2'
      do 63 iside = 1,4
         if (iside.lt.3) then
            ncon = 4
         else
            ncon = 1
         endif
      do 60 i = 1,ncon
        if (iside.gt.2) then
            icon = 2
            nedge = 1
        else
          if (iclamp(iside).eq.1) then
           if (i.lt.3) then
             icon = 2*i-1
             nedge = 2
           else
             icon = i-(4-i)
             nedge = 1
           endif
          else
            icon = i
            if (i.lt.3) then
              nedge = 2
            else
              nedge = 1
            endif
          endif
```

```
      endif
      do 55 i = 1,nedge
           if (iside.lt.3) then
               iedge = (12-2*iside)*1-(12-4*iside)
           else
               iedge = 1
           endif
           jjjj = 0
         do 50 j = 1,nbb
             if (((iside.eq.2).or.(iside.eq.4)).and.(j.lt.6)) then
                 ibb = 0
                 call bb0 (as)
             elseif (j.lt.6) then
               if (j.eq.1) then
                 ibb = 2
                 call bb12 (ibb,icon,as)
               elseif (j.lt.4) then
                 ibb = j+1
                 call bb34 (ibb,icon,as)
               else
                 ibb = j+1
                 call bb56 (ibb,icon,as)
               endif
             elseif (j.lt.11) then
c              rectangular segment
              almds = almds *phi*phi/(phir*phir)
            phi = phir
            if ((iedge.ne.12).and.(iside.ne.3)) then
                   if (iedge.ne.1) iedge = 6-iedge
                   if (j.eq.6) then
                         ibb = 2
                         call bb12 (ibb,icon,as)
                   elseif ((j.eq.7).or.(j.eq.8)) then
                    ibb = j-4
                    call bb34 (ibb,icon,as)
               else
                    ibb = j-2
                    call bb78 (ibb,icon,as)
               endif
                 call factor (as,icon,tphi)
               if (iedge.ne.1) iedge = 6-iedge
            else
                  ibb = 0
               call bb0 (as)
            endif
            phi = tphi
               almds = talmds
             else
               if (iside.ne.2) then
                   ibb = 0
                   call bb0 (as)
               elseif (j.lt.13) then
                   ibb = j-10
                   call bb12 (ibb,icon,as)
               else
                   ibb = j-6
                   call bb78 (ibb,icon,as)
               endif
            endif

           do 40 ii = 1,ki
           do 40 jj = 1,kj
               iii = iiii+ii
               jjj = jjjj+jj
               a(iii,jjj) = as(ii,jj)
40           continue
           jjjj = jjjj+kj
```

```
c         write(*,*) 'iedge = ',iedge,' iside = ',iside,' icon = ',icon,
c     1 ' ibb = ',ibb,'ki = ',ki,'kj = ',kj
c                do 41 ii = 1,ki
c   41           write(*,999) (as(ii,jj),jj=1,kj)
  990            format(8(i4,1x),2(f10.3))
  999            format(70(e10.3,1x))
   50        continue
            iiii = iiii+ki
   55    continue
   60 continue
   63 continue
      nk = iiii
      return
      end



c
c**********************************************************************
       subroutine bphi_mid (a,iclamp,nk)
c**********************************************************************
c
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension a(320,320), as(32,16), iclamp(2)
      tphi = phi
      talmds = almds
      nbb = 18
       kb = k
       ka = k
       ki = ka
       kj = ka
   35   iiii = 0
      do 63 iside = 1,4
         if (iside.lt.3) then
            ncon = 4
         else
            ncon = 1
         endif
      do 60 i = 1,ncon
        if (iside.gt.2) then
            icon = 2
            nedge = 1
        else
          if (iclamp(iside).eq.1) then
           if (i.lt.3) then
             icon = 2*i-1
             nedge = 2
           else
             icon = i-(4-i)
             nedge = 1
           endif
          else
            icon = i
            if (i.lt.3) then
              nedge = 2
            else
              nedge = 1
            endif
          endif
      endif
       do 55 l = 1,nedge
           if (iside.lt.3) then
               iedge = (12-2*iside)*l-(12-4*iside)
           else
               iedge = 1
```

```
      endif
      if (iedge.eq.12) ki = 2*k
      jjjj = 0
       do 50 j = 1,nbb
            if (((iside.eq.2).or.(iside.eq.4)).and.(j.lt.8)) then
                ibb = 0
                call bb0 (as)
            elseif (j.lt.8) then
              if (j.eq.1) then
                  ibb = 2
                  call bb12 (ibb,icon,as)
              elseif (j.lt.4) then
                  ibb = j+1
                  call bb34 (ibb,icon,as)
              elseif ((j.eq.4).or.(j.eq.5)) then
                ibb = j+1
                call bb56 (ibb,icon,as)
              else
                  ibb = j+1
                  call bb78 (ibb,icon,as)
              endif
            elseif (j.lt.13) then
c             rectangular segment
             almds = almds *phi*phi/(phir*phir)
           phi = phir
           if ((iedge.ne.12).and.(iside.ne.3)) then
                  if (iedge.ne.1) iedge = 6-iedge
                  if (j.eq.8) then
                       ibb = 2
                       call bb12 (ibb,icon,as)
                  elseif ((j.eq.9).or.(j.eq.10)) then
                   ibb = j-6
                   call bb34 (ibb,icon,as)
               else
                   ibb = j-4
                   call bb78 (ibb,icon,as)
               endif
                 call factor (as,icon,tphi)
                  if (iedge.ne.1) iedge = 6-iedge
          else
                 ibb = 0
              call bb0 (as)
          endif
          phi = tphi
             almds = talmds
           else
             if (iside.ne.2) then
                 ibb = 0
                 call bb0 (as)
             elseif (j.lt.15) then
                 ibb = j-12
                 call bb12 (ibb,icon,as)
             elseif (j.lt.17) then
                 ibb = j-12
                 call bb34 (ibb,icon,as)
             else
                 ibb = j-10
                 call bb78 (ibb,icon,as)
             endif
         endif
       do 40 ii = 1,ki
           do 40 jj = 1,kj
              iii = iiii+ii
              jjj = jjjj+jj
              a(iii,jjj) = as(ii,jj)
40      continue
       jjjj = jjjj+kj
```

```
990 format(8(i4,1x),2(f10.3))
  999   format(70(e10.3,1x))
   50     continue
         iiii = iiii+ki
         if (iedge.eq.12) ki = k
   55   continue
   60 continue
   63 continue
      nk = iiii
      return
      end


c
c**********************************************************************
       subroutine bphi_hi (a,iclamp,nk)
c**********************************************************************
c
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension a(320,320), as(32,16), iclamp(2)
      tphi = phi
      talmds = almds
      nbb = 14
       kb = k
       ka = k
       ki = ka
       kj = ka
   35   iiii = 0
      do 63 iside = 1,4
         if (iside.lt.3) then
            ncon = 4
         else
            ncon = 1
         endif
      do 60 i = 1,ncon
        if (iside.gt.2) then
            icon = 2
            nedge = 1
        else
          if (iclamp(iside).eq.1) then
           if (i.lt.3) then
             icon = 2*i-1
             nedge = 2
           else
             icon = i-(4-i)
             nedge = 1
           endif
          else
            icon = i
            if (i.lt.3) then
              nedge = 2
            else
              nedge = 1
            endif
          endif
      endif
      do 55 l = 1,nedge
           if (iside.lt.3) then
               iedge = (12-2*iside)*l-(12-4*iside)
           else
               iedge = 1
           endif
           jjjj = 0
         do 50 j = 1,nbb
             if (((iside.eq.2).or.(iside.eq.4)).and.(j.lt.6)) then
                ibb = 0
```

```
                 call bb0 (as)
              elseif (j.lt.6) then
                if (j.eq.1) then
                  ibb = 2
                  call bb12 (ibb,icon,as)
                elseif (j.lt.4) then
                  ibb = j+1
                  call bb34 (ibb,icon,as)
                else
                  ibb = j+3
                  call bb78 (ibb,icon,as)
                endif
              elseif (j.lt.11) then
c               rectangular segment
               almds = almds *phi*phi/(phir*phir)
              phi = phir
              if ((iedge.ne.12).and.(iside.ne.3)) then
                    if (iedge.ne.1) iedge = 6-iedge
                    if (j.eq.6) then
                         ibb = 2
                         call bb12 (ibb,icon,as)
                    elseif ((j.eq.7).or.(j.eq.8)) then
                     ibb = j-4
                     call bb34 (ibb,icon,as)
                 else
                     ibb = j-2
                     call bb78 (ibb,icon,as)
                 endif
                  call factor (as,icon,tphi)
                 if (iedge.ne.1) iedge = 6-iedge
             else
                    ibb = 0
                 call bb0 (as)
             endif
             phi = tphi
               almds = talmds
              else
                if (iside.ne.2) then
                    ibb = 0
                    call bb0 (as)
                elseif (j.lt.13) then
                    ibb = j-8
                    call bb34 (ibb,icon,as)
                else
                    ibb = j-6
                    call bb78 (ibb,icon,as)
                endif
             endif

             do 40 ii = 1,ki
             do 40 jj = 1,kj
                 iii = iiii+ii
                 jjj = jjjj+jj
                 a(iii,jjj) = as(ii,jj)
 40            continue
             jjjj = jjjj+kj
990            format(8(i4,1x),2(f10.3))
999            format(70(e10.3,1x))
 50        continue
          iiii = iiii+ki
 55    continue
 60 continue
 63 continue
    nk = iiii
    return
    end
```

2.3 Subroutines for shape generation

```
cc*******************************************************************
      subroutine bshap_lo (ks1,x,w)
c********************************************************************
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension x(320),xi(126),eta(21),incode(21,126)
      dimension el2m(16),el3m(16),el4m(16),el5m(16),el6m(16)
      dimension em2m(16),em3m(16),em4m(16),em7m(16),em8m(16)
      dimension er1m(16),er2m(16),er7m(16),er8m(16),dummy(16)
      dimension w(21,126)
      character*14 outfile
c       write(*,*) ' entering bshap_lo'
      ka = k
c       write(6,*)(x(i),i=1,ka)
      do 10 i = 1,ka
         dummy(i) = 0.0
         el2m(i) = x(i)
         el3m(i) = x(ka+i)
         el4m(i) = x(2*ka+i)
         el5m(i) = x(3*ka+i)
         el6m(i) = x(4*ka+i)
         em2m(i) = x(5*ka+i)
         em3m(i) = x(6*ka+i)
         em4m(i) = x(7*ka+i)
         em7m(i) = x(8*ka+i)
         em8m(i) = x(9*ka+i)
         er1m(i) = x(10*ka+i)
         er2m(i) = x(11*ka+i)
         er7m(i) = x(12*ka+i)
         er8m(i) = x(13*ka+i)
   10 continue
      bigphi = phi*phir/(2.0*phir+phi)
      ksy = ks1
      ksx = int((ksy-1.0)/bigphi + 2.0)
      nt = int(bigphi*(ksx-1)/phi)+1
      nr = ksx - nt
      if (ksx.gt.126) ksx = 126
       do 15 i = 1, ksy
         do 15 j = 1, ksx
           w(i,j) = 0.0
   15  continue
cc*******************************************************************
      call grid (ksx,ksy,xi,eta,nt,nr,incode)
c********************************************************************
      call shape12 (1,nt,ksy,ka,dummy,el2m,xi,eta,incode,w)
      call shape34 (1,nt,ksy,ka,el3m,el4m,xi,eta,incode,w)
      call shape56 (1,nt,ksy,ka,el5m,el6m,xi,eta,incode,w)
      tphi = phi
      talmds = almds
      almds = almds*phi*phi/(phir*phir)
      phi = phir
      call shape12 (nt+1,nr,ksy,ka,dummy,em2m,xi,eta,incode,w)
      call shape34 (nt+1,nr,ksy,ka,em3m,em4m,xi,eta,incode,w)
      call shape78 (nt+1,nr,ksy,ka,em7m,em8m,xi,eta,incode,w)
      do 20 i = 1, ksy
        do 20 j = nt+1, nr
          w(i,j) = -1.0*w(i,j)
   20 continue
      almds = talmds
      phi = tphi
      call shape12 (nr+1,ksx,ksy,ka,er1m,er2m,xi,eta,incode,w)
      call shape78 (nr+1,ksx,ksy,ka,er7m,er8m,xi,eta,incode,w)
```

```
      write(6,*) 'Output file? (Use quotation marks)'
      read(5,*) outfile
      write(6,*) 'Writing displacement matrix to file ',outfile
      open(25,file=outfile,status='new')
      do 106 jj = 1,ksx
  106  write(25,999) (w(ii,jj),ii=1,ksy)
  999  format (21(e10.3,1x))
      close(25)
c      write(*,*) 'leaving bshap_lo'
      return
      end




cc*********************************************************************
      subroutine bshap_mi (ks1,x,w)
c**********************************************************************
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension x(320),xi(126),eta(21),incode(21,126)
      dimension el2m(16),el3m(16),el4m(16),el5m(16),el6m(16)
      dimension el7m(16),el8m(16)
      dimension em2m(16),em3m(16),em4m(16),em7m(16),em8m(16)
      dimension er1m(16),er2m(16),er3m(16),er4m(16)
      dimension er7m(16),er8m(16), dummy(16)
      dimension w(21,126)
      character*14 outfile
c       write(*,*) ' entering bshap_mi'
      ka = k
c       write(6,*)(x(i),i=1,16*ka)
      do 10 i = 1,ka
         dummy(i) = 0.0
         el2m(i) = x(i)
         el3m(i) = x(ka+i)
         el4m(i) = x(2*ka+i)
         el5m(i) = x(3*ka+i)
         el6m(i) = x(4*ka+i)
         el7m(i) = x(5*ka+i)
         el8m(i) = x(6*ka+i)
         em2m(i) = x(7*ka+i)
         em3m(i) = x(8*ka+i)
         em4m(i) = x(9*ka+i)
         em7m(i) = x(10*ka+i)
         em8m(i) = x(11*ka+i)
         er1m(i) = x(12*ka+i)
         er2m(i) = x(13*ka+i)
         er3m(i) = x(14*ka+i)
         er4m(i) = x(15*ka+i)
         er7m(i) = x(16*ka+i)
         er8m(i) = x(17*ka+i)
   10 continue
      bigphi = phi*phir/(2.0*phir+phi)
      ksy = ks1
      ksx = int((ksy-1.0)/bigphi + 2.0)
      nt = int(bigphi*(ksx-1)/phi)+1
      nr = ksx - nt
      if (ksx.gt.126) ksx = 126
       do 15 i = 1, ksy
         do 15 j = 1, ksx
           w(i,j) = 0.0
   15  continue
cc*********************************************************************
      call grid (ksx,ksy,xi,eta,nt,nr,incode)
c**********************************************************************
```

```
      call shape12 (1,nt,ksy,ka,dummy,el2m,xi,eta,incode,w)
      call shape34 (1,nt,ksy,ka,el3m,el4m,xi,eta,incode,w)
      call shape56 (1,nt,ksy,ka,el5m,el6m,xi,eta,incode,w)
      call shape78 (1,nt,ksy,ka,el7m,el8m,xi,eta,incode,w)
c       write(6,*) 'in bshap_mi, done first segment'
c     do 100 jj = 1,ksx
c 100   write(6,999) (w(ii,jj),ii=1,ksy)
      tphi = phi
      talmds = almds
      almds = almds*phi*phi/(phir*phir)
      phi = phir
      call shape12 (nt+1,nr,ksy,ka,dummy,em2m,xi,eta,incode,w)
      call shape34 (nt+1,nr,ksy,ka,em3m,em4m,xi,eta,incode,w)
      call shape78 (nt+1,nr,ksy,ka,em7m,em8m,xi,eta,incode,w)
      do 20 i = 1, ksy
        do 20 j = nt+1, nr
          w(i,j) = -1.0*w(i,j)
   20   continue
c     do 101 jj = 1,ksx
c 101   write(6,999) (w(ii,jj),ii=1,ksy)
      almds = talmds
      phi = tphi
c       write(6,*) 'in bshap_mi, done 2nd segment'
      call shape12 (nr+1,ksx,ksy,ka,er1m,er2m,xi,eta,incode,w)
      call shape34 (nr+1,ksx,ksy,ka,er3m,er4m,xi,eta,incode,w)
      call shape78 (nr+1,ksx,ksy,ka,er7m,er8m,xi,eta,incode,w)
c     do 102 jj = 1,ksx
c 102   write(6,999) (w(ii,jj),ii=1,ksy)

      write(6,*) 'Output file? (Use quotation marks)'
      read(5,*) outfile
      write(6,*) 'Writing displacement matrix to file ',outfile
      open(25,file=outfile,status='new')
      do 106 jj = 1,ksx
  106   write(25,999) (w(ii,jj),ii=1,ksy)
  999   format (21(e10.3,1x))
      close(25)
c       write(*,*) 'leaving bshap_mi'
      return
      end




cc*********************************************************************
      subroutine bshap_hi (ks1,x,w)
c**********************************************************************
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension x(320),xi(126),eta(21),incode(21,126)
      dimension el2m(16),el3m(16),el4m(16),el7m(16),el8m(16)
      dimension em2m(16),em3m(16),em4m(16),em7m(16),em8m(16)
      dimension er3m(16),er4m(16),er7m(16),er8m(16),dummy(16)
      dimension w(21,126)
      character*14 outfile
c       write(*,*) ' entering bshap_hi'
      ka = k
c       write(6,*)(x(i),i=1,ka)
      do 10 i = 1,ka
        dummy(i) = 0.0
        el2m(i) = x(i)
        el3m(i) = x(ka+i)
        el4m(i) = x(2*ka+i)
```

```
        el7m(i) = x(3*ka+i)
        el8m(i) = x(4*ka+i)
        em2m(i) = x(5*ka+i)
        em3m(i) = x(6*ka+i)
        em4m(i) = x(7*ka+i)
        em7m(i) = x(8*ka+i)
        em8m(i) = x(9*ka+i)
        er3m(i) = x(10*ka+i)
        er4m(i) = x(11*ka+i)
        er7m(i) = x(12*ka+i)
        er8m(i) = x(13*ka+i)
   10 continue
      bigphi = phi*phir/(2.0*phir+phi)
      ksy = ks1
      ksx = int((ksy-1.0)/bigphi + 2.0)
      nt = int(bigphi*(ksx-1)/phi)+1
      nr = ksx - nt
      if (ksx.gt.126) ksx = 126
       do 15 i = 1, ksy
         do 15 j = 1, ksx
           w(i,j) = 0.0
   15  continue
cc*****************************************************************
       call grid (ksx,ksy,xi,eta,nt,nr,incode)
c******************************************************************
       call shape12 (1,nt,ksy,ka,dummy,el2m,xi,eta,incode,w)
       call shape34 (1,nt,ksy,ka,el3m,el4m,xi,eta,incode,w)
       call shape78 (1,nt,ksy,ka,el7m,el8m,xi,eta,incode,w)
       tphi = phi
       talmds = almds
       almds = almds*phi*phi/(phir*phir)
       phi = phir
       call shape12 (nt+1,nr,ksy,ka,dummy,em2m,xi,eta,incode,w)
       call shape34 (nt+1,nr,ksy,ka,em3m,em4m,xi,eta,incode,w)
       call shape78 (nt+1,nr,ksy,ka,em7m,em8m,xi,eta,incode,w)
       do 20 i = 1, ksy
         do 20 j = nt+1, nr
           w(i,j) = -1.0*w(i,j)
   20  continue
       almds = talmds
       phi = tphi
       call shape34 (nr+1,ksx,ksy,ka,er3m,er4m,xi,eta,incode,w)
       call shape78 (nr+1,ksx,ksy,ka,er7m,er8m,xi,eta,incode,w)
       write(6,*) 'Output file? (Use quotation marks)'
       read(5,*) outfile
       write(6,*) 'Writing displacement matrix to file ',outfile
       open(25,file=outfile,status='new')
       do 106 jj = 1,ksx
  106   write(25,999) (w(ii,jj),ii=1,ksy)
  999   format (21(e10.3,1x))
       close(25)
c       write(*,*) 'leaving bshap_hi'
       return
       end
```

3. CTOP - Both Base and Top Edge are Clamped

3.1 Main Program

```
C*********************************************************************
C   This is the main calling program for eigenvalue search and shape  *
C   data generation for parallelogram plates with clamped base and    *
C   clamped top edge.                                                  *
C   Segments are arranged horizontally for this version. Shape        *
C   generation is included.                                            *
C               by S. Stangier                                         *
C*********************************************************************
C
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension a(320,320)
      dimension x(320),w(21,126),iclamp(2)
       poi = 0.3
      write(*,*) 'parashap.for: parallelogram plate - cscs,cscc, cccc'
      write(*,*) 'for left, right indicate clamping: 1=s, 2=c'
   10 write(*,20)
   20 format(' ',' phi     phir   k left right  almds    dlim    del',/)
      read(*,*) phi,phir,k,iclamp(1),iclamp(2),almds,dlim,del
       if (phi.lt.0.) goto 200
      write(*,30) phi,phir,k,iclamp(1),iclamp(2),almds,dlim,del
   30 format(' ',2(f7.5,1x),3(i2,1x),2(f9.5,1x),f7.5,/)
   32 c = 1.
      pi = 4.*datan(c)
      alpha = datan(1./phi)
      qlim = 360000000
   35 continue
       if (phi.lt.0.5) then
             call tphi_low (a,iclamp,nk)
       elseif (phi.ge.2.0) then
             call tphi_hi (a,iclamp,nk)
       else
             call tphi_mid (a,iclamp,nk)
       endif
c        do 62 ii = 1,nk
c   62   write(*,999) (a(ii,jj),jj=1,nk)
c        write(*,*) 'a is done'
       if (del.eq.0.) goto 190
c
c*********************************************************************
      call determ (a,nk,det)
c*********************************************************************
c
   61 write(*,65) almds,det
   65 format(' ',5x,f9.5,5x,e15.8)
      almds = almds+del
      if (almds.le.dlim) then
       goto 35
      else
       goto 10
      endif
  190 continue
      write(6,*) 'Number of grid points in eta-direction? (Max. 21)'
      read(5,*) ks1
      if (ks1.gt.21) ks1 = 21
c         write(*,*) '    generating shape'
C
```

```
C***********************************************************************
      call detsol (a,nk,x)
C***********************************************************************
C
       x(nk) = -1
       if (phi.le.0.5) then
              call tshap_lo (ks1,x,w)
       elseif (phi.ge.2.) then
              call tshap_hi (ks1,x,w)
       else
              call tshap_mi (ks1,x,w)
       endif
c        write(*,*) 'w:'
c        do 106 ii = 1,3*ks1-2
c  106     write(*,999) (w(ii,jj),jj=1,ks1)
       goto 10
  999 format(250(e10.3,1x))
  200 continue
      stop
      end
```

3.2 Subroutines for matrix assembly

```
c
c***********************************************************************
       subroutine tphi_low (a,iclamp,nk)
c***********************************************************************
c
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension a(320,320), as(32,16), iclamp(2)
      tphi = phi
      talmds = almds
      nbb = 16
       kb = k
       ka = k
       ki = ka
       kj = ka
   35   iiii = 0
      do 63 iside = 1,6
        if (iside.lt.3) then
           ncon = 4
        else
           ncon = 1
        endif
      do 60 i = 1,ncon
       if (iside.gt.2) then
           icon = 2
           nedge = 1
       else
          if (iclamp(iside).eq.1) then
           if (i.lt.3) then
             icon = 2*i-1
             nedge = 2
           else
             icon = i-(4-i)
             nedge = 1
           endif
          else
            icon = i
            if (i.lt.3) then
```

```
              nedge = 2
            else
              nedge = 1
            endif
          endif
        endif
        do 55 l = 1,nedge
            if (iside.lt.3) then
               iedge = (12-2*iside)*l-(12-4*iside)
            elseif (iside.lt.5) then
               iedge = 1
            else
               iedge = 3
            endif
            jjjj = 0
            do 50 j = 1,nbb
               if (((iside.eq.2).or.(iside.gt.3)).and.(j.lt.6)) then
                  ibb = 0
                  call bb0 (as)
               elseif (j.lt.6) then
                if (j.eq.1) then
                  ibb = 2
                  call bb12 (ibb,icon,as)
                elseif (j.lt.4) then
                  ibb = j+1
                  call bb34 (ibb,icon,as)
                else
                  ibb = j+1
                  call bb56 (ibb,icon,as)
                endif
               elseif (j.lt.12) then
c                 rectangular segment
                almds = almds *phi*phi/(phir*phir)
                phi = phir
                if ((iedge.ne.12).and.
     1                    ((iside.ne.3).and.(iside.ne.6))) then
                    if (iedge.ne.1) iedge = 6-iedge
                    if (j.eq.6) then
                        ibb = 2
                        call bb12 (ibb,icon,as)
                    elseif ((j.eq.7).or.(j.eq.8)) then
                        ibb = j-4
                        call bb34 (ibb,icon,as)
                    elseif (j.eq.9) then
                        ibb = 6
                        call bb56 (ibb,icon,as)
                    else
                        ibb = j-3
                        call bb78 (ibb,icon,as)
                    endif
                    call factor (as,icon,tphi)
                    if (iedge.ne.1) iedge = 6-iedge
               else
                    ibb = 0
                    call bb0 (as)
               endif
               phi = tphi
               almds = talmds
              else
               if ((iside.ne.2).and.(iside.ne.6)) then
                    ibb = 0
                    call bb0 (as)
               elseif (j.lt.14) then
                    ibb = j-11
                    call bb12 (ibb,icon,as)
               elseif (j.lt.15) then
                    ibb = 6
```

```
                    call bb56 (ibb,icon,as)
                 else
                    ibb = j-8
                    call bb78 (ibb,icon,as)
                 endif
                endif

                do 40 ii = 1,ki
                do 40 jj = 1,kj
                    iii = iiii+ii
                    jjj = jjjj+jj
                    a(iii,jjj) = as(ii,jj)
 40               continue
                jjjj = jjjj+kj
990               format(8(i4,1x),2(f10.3))
999               format(70(e10.3,1x))
 50           continue
            iiii = iiii+ki
 55     continue
 60   continue
 63   continue
      nk = iiii
      return
      end




c
c************************************************************************
       subroutine tphi_mid (a,iclamp,nk)
c************************************************************************
c
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension a(320,320), as(32,16), iclamp(2)
      tphi = phi
      talmds = almds
      nbb = 20
       kb = k
       ka = k
       ki = ka
       kj = ka
 35     iiii = 0
c       write(*,*) 'phi > 2'
c       write(*,*) 'iside iedge icon ibb ki  kj  iiii jjjj phi almds'
      do 63 iside = 1,6
        if (iside.lt.3) then
           ncon = 4
        else
           ncon = 1
        endif
      do 60 i = 1,ncon
       if (iside.gt.2) then
           icon = 2
           nedge = 1
       else
          if (iclamp(iside).eq.1) then
           if (i.lt.3) then
             icon = 2*i-1
             nedge = 2
           else
             icon = i-(4-i)
             nedge = 1
```

```
          endif
         else
           icon = i
           if (i.lt.3) then
             nedge = 2
           else
             nedge = 1
           endif
         endif
      endif
      do 55 l = 1,nedge
          if (iside.lt.3) then
             iedge = (12-2*iside)*l-(12-4*iside)
          elseif (iside.lt.5) then
             iedge = 1
          else
             iedge = 3
          endif
          if (iedge.eq.12) ki = 2*k
          jjjj = 0
          do 50 j = 1,nbb
             if (((iside.eq.2).or.(iside.gt.3)).and.(j.lt.8)) then
                ibb = 0
                call bb0 (as)
             elseif (j.lt.8) then
              if (j.eq.1) then
                ibb = 2
                call bb12 (ibb,icon,as)
              elseif (j.lt.4) then
                ibb = j+1
                call bb34 (ibb,icon,as)
              elseif (j.lt.6) then
                ibb = j+1
                call bb56 (ibb,icon,as)
              else
                ibb = j+1
                call bb78 (ibb,icon,as)
              endif
             elseif (j.lt.14) then
c                 rectangular segment
              almds = almds *phi*phi/(phir*phir)
              phi = phir
              if ((iedge.ne.12).and.
     1                     ((iside.ne.3).and.(iside.ne.6))) then
                  if (iedge.ne.1) iedge = 6-iedge
                  if (j.eq.8) then
                      ibb = 2
                      call bb12 (ibb,icon,as)
                  elseif ((j.eq.9).or.(j.eq.10)) then
                      ibb = j-6
                      call bb34 (ibb,icon,as)
                  elseif (j.eq.11) then
                      ibb = 6
                      call bb56 (ibb,icon,as)
                  else
                      ibb = j-5
                      call bb78 (ibb,icon,as)
                  endif
                  call factor (as,icon,tphi)
                  if (iedge.ne.1) iedge = 6-iedge
             else
                  ibb = 0
                  call bb0 (as)
             endif
             phi = tphi
             almds = talmds
            else
```

```
              if ((iside.ne.2).and.(iside.ne.6)) then
                  ibb = 0
                  call bb0 (as)
              elseif (j.lt.16) then
                  ibb = j-13
                  call bb12 (ibb,icon,as)
              elseif (j.lt.18) then
                  ibb = j-13
                  call bb34 (ibb,icon,as)
              elseif (j.lt.19) then
                  ibb = 6
                  call bb56 (ibb,icon,as)
              else
                  ibb = j-12
                  call bb78 (ibb,icon,as)
              endif
             endif

             do 40 ii = 1,ki
             do 40 jj = 1,kj
                 iii = iiii+ii
                 jjj = jjjj+jj
                 a(iii,jjj) = as(ii,jj)
   40           continue
             jjjj = jjjj+kj
c         write(*,*) 'iedge = ',iedge,' iside = ',iside,' icon = ',icon,
c     1 ' ibb = ',ibb,'ki = ',ki,'kj = ',kj
c               do 41 ii = 1,ki
c    41         write(*,999) (as(ii,jj),jj=1,kj)
  990           format(8(i4,1x),2(f10.3))
  999           format(70(e10.3,1x))
   50        continue
          iiii = iiii+ki
          if (iedge.eq.12) ki = k
   55   continue
   60 continue
   63 continue
      nk = iiii
      return
      end


c
c*********************************************************************
       subroutine tphi_hi (a,iclamp,nk)
c*********************************************************************
c
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension a(320,320), as(32,16), iclamp(2)
      tphi = phi
      talmds = almds
      nbb = 16
       kb = k
       ka = k
       ki = ka
       kj = ka
   35   iiii = 0
c       write(*,*) 'phi > 2'
c       write(*,*) 'iside iedge icon ibb ki  kj  iiii jjjj phi almds'
      do 63 iside = 1,6
        if (iside.lt.3) then
           ncon = 4
        else
           ncon = 1
        endif
```

```
      do 60 i = 1,ncon
       if (iside.gt.2) then
           icon = 2
           nedge = 1
       else
          if (iclamp(iside).eq.1) then
           if (i.lt.3) then
             icon = 2*i-1
             nedge = 2
           else
             icon = i-(4-i)
             nedge = 1
           endif
          else
            icon = i
            if (i.lt.3) then
              nedge = 2
            else
              nedge = 1
            endif
          endif
      endif
      do 55 l = 1,nedge
          if (iside.lt.3) then
             iedge = (12-2*iside)*l-(12-4*iside)
          elseif (iside.lt.5) then
             iedge = 1
          else
             iedge = 3
          endif
          jjjj = 0
          do 50 j = 1,nbb
             if (((iside.eq.2).or.(iside.gt.3)).and.(j.lt.6)) then
                ibb = 0
                call bb0 (as)
             elseif (j.lt.6) then
              if (j.eq.1) then
                ibb = 2
                call bb12 (ibb,icon,as)
              elseif (j.lt.4) then
                ibb = j+1
                call bb34 (ibb,icon,as)
              else
                ibb = j+3
                call bb78 (ibb,icon,as)
              endif
             elseif (j.lt.12) then
c                rectangular segment
              almds = almds *phi*phi/(phir*phir)
              phi = phir
              if ((iedge.ne.12).and.
     1                    ((iside.ne.3).and.(iside.ne.6))) then
                 if (iedge.ne.1) iedge = 6-iedge
                 if (j.eq.6) then
                     ibb = 2
                     call bb12 (ibb,icon,as)
                 elseif ((j.eq.7).or.(j.eq.8)) then
                     ibb = j-4
                     call bb34 (ibb,icon,as)
                 elseif (j.eq.9) then
                     ibb = 6
                     call bb56 (ibb,icon,as)
                 else
                     ibb = j-3
                     call bb78 (ibb,icon,as)
                 endif
                 call factor (as,icon,tphi)
```

```
                if (iedge.ne.1) iedge = 6-iedge
            else
                ibb = 0
                call bb0 (as)
            endif
            phi = tphi
            almds = talmds
           else
            if ((iside.ne.2).and.(iside.ne.6)) then
                ibb = 0
                call bb0 (as)
            elseif (j.lt.14) then
                ibb = j-9
                call bb34 (ibb,icon,as)
            elseif (j.lt.15) then
                ibb = 6
                call bb56 (ibb,icon,as)
            else
                ibb = j-8
                call bb78 (ibb,icon,as)
c      write(*,990) iside,iedge,icon,ibb,ki,kj,iiii,jjjj,phi,almds
            endif
           endif

           do 40 ii = 1,ki
           do 40 jj = 1,kj
              iii = iiii+ii
              jjj = jjjj+jj
              a(iii,jjj) = as(ii,jj)
   40         continue
           jjjj = jjjj+kj
c         write(*,*) 'iedge = ',iedge,' iside = ',iside,' icon = ',icon,
c    1 ' ibb = ',ibb,'ki = ',ki,'kj = ',kj
c             do 41 ii = 1,ki
c   41        write(*,999) (as(ii,jj),jj=1,kj)
  990         format(8(i4,1x),2(f10.3))
  999         format(70(e10.3,1x))
   50      continue
          iiii = iiii+ki
   55   continue
   60 continue
   63 continue
      nk = iiii
      return
      end
```

3.3 Soubroutines for shape generation

```
cc*********************************************************************
       subroutine tshap_lo (ks1,x,w)
c**********************************************************************
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension x(320),xi(126),eta(21),incode(21,126)
      dimension el2m(16),el3m(16),el4m(16),el5m(16),el6m(16)
      dimension em2m(16),em3m(16),em4m(16),em6m(16),em7m(16),em8m(16)
      dimension er1m(16),er2m(16),er6m(16),er7m(16),er8m(16),dummy(16)
      dimension w(21,126)
      character*14 outfile
c       write(*,*) ' entering tshap_lo'
```

```
      ka = k
c       write(6,*)(x(i),i=1,ka)
      do 10 i = 1,ka
         dummy(i) = 0.0
         el2m(i) = x(i)
         el3m(i) = x(ka+i)
         el4m(i) = x(2*ka+i)
         el5m(i) = x(3*ka+i)
         el6m(i) = x(4*ka+i)
         em2m(i) = x(5*ka+i)
         em3m(i) = x(6*ka+i)
         em4m(i) = x(7*ka+i)
         em6m(i) = x(8*ka+i)
         em7m(i) = x(9*ka+i)
         em8m(i) = x(10*ka+i)
         er1m(i) = x(11*ka+i)
         er2m(i) = x(12*ka+i)
         er6m(i) = x(13*ka+i)
         er7m(i) = x(14*ka+i)
         er8m(i) = x(15*ka+i)
   10 continue
      bigphi = phi*phir/(2.0*phir+phi)
      ksy = ks1
      ksx = int((ksy-1.0)/bigphi + 2.0)
      nt = int(bigphi*(ksx-1)/phi)+1
      nr = ksx - nt
      if (ksx.gt.126) ksx = 126
      do 15 i = 1, ksy
        do 15 j = 1, ksx
          w(i,j) = 0.0
   15  continue
cc*************************************************************************
      call grid (ksx,ksy,xi,eta,nt,nr,incode)
c**************************************************************************
      call shape12 (1,nt,ksy,ka,dummy,el2m,xi,eta,incode,w)
      call shape34 (1,nt,ksy,ka,el3m,el4m,xi,eta,incode,w)
      call shape56 (1,nt,ksy,ka,el5m,el6m,xi,eta,incode,w)
      tphi = phi
      talmds = almds
      almds = almds*phi*phi/(phir*phir)
      phi = phir
      call shape12 (nt+1,nr,ksy,ka,dummy,em2m,xi,eta,incode,w)
      call shape34 (nt+1,nr,ksy,ka,em3m,em4m,xi,eta,incode,w)
      call shape56 (nt+1,nr,ksy,ka,dummy,em6m,xi,eta,incode,w)
      call shape78 (nt+1,nr,ksy,ka,em7m,em8m,xi,eta,incode,w)
      do 20 i = 1, ksy
        do 20 j = nt+1, nr
          w(i,j) = -1.0*w(i,j)
   20  continue
      almds = talmds
      phi = tphi
      call shape12 (nr+1,ksx,ksy,ka,er1m,er2m,xi,eta,incode,w)
      call shape56 (nr+1,ksx,ksy,ka,dummy,er6m,xi,eta,incode,w)
      call shape78 (nr+1,ksx,ksy,ka,er7m,er8m,xi,eta,incode,w)
      write(6,*) 'Output file? (Use quotation marks)'
      read(5,*) outfile
      write(6,*) 'Writing displacement matrix to file ',outfile
      open(25,file=outfile,status='new')
      do 106 jj = 1,ksx
  106  write(25,999) (w(ii,jj),ii=1,ksy)
  999  format (21(e10.3,1x))
      close(25)
c       write(*,*) 'leaving tshap_lc'
      return
      end
```

```
cc*********************************************************************
       subroutine tshap_mi (ks1,x,w)
c**********************************************************************
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension x(320),xi(126),eta(21),incode(21,126)
      dimension el2m(16),el3m(16),el4m(16),el5m(16),el6m(16)
      dimension el7m(16),el8m(16)
      dimension em2m(16),em3m(16),em4m(16),em6m(16),em7m(16),em8m(16)
      dimension er1m(16),er2m(16),er3m(16),er4m(16),er6m(16)
      dimension er7m(16),er8m(16), dummy(16)
      dimension w(21,126)
      character*14 outfile
c       write(*,*) ' entering tshap_mi'
      ka = k
c       write(6,*)(x(i),i=1,16*ka)
      do 10 i = 1,ka
         dummy(i) = 0.0
         el2m(i) = x(i)
         el3m(i) = x(ka+i)
         el4m(i) = x(2*ka+i)
         el5m(i) = x(3*ka+i)
         el6m(i) = x(4*ka+i)
         el7m(i) = x(5*ka+i)
         el8m(i) = x(6*ka+i)
         em2m(i) = x(7*ka+i)
         em3m(i) = x(8*ka+i)
         em4m(i) = x(9*ka+i)
         em6m(i) = x(10*ka+i)
         em7m(i) = x(11*ka+i)
         em8m(i) = x(12*ka+i)
         er1m(i) = x(13*ka+i)
         er2m(i) = x(14*ka+i)
         er3m(i) = x(15*ka+i)
         er4m(i) = x(16*ka+i)
         er6m(i) = x(17*ka+i)
         er7m(i) = x(18*ka+i)
         er8m(i) = x(19*ka+i)
   10 continue
      bigphi = phi*phir/(2.0*phir+phi)
      ksy = ks1
      ksx = int((ksy-1.0)/bigphi + 2.0)
      nt = int(bigphi*(ksx-1)/phi)+1
      nr = ksx - nt
      if (ksx.gt.126) ksx = 126
      do 15 i = 1, ksy
        do 15 j = 1, ksx
          w(i,j) = 0.0
   15  continue
cc*********************************************************************
      call grid (ksx,ksy,xi,eta,nt,nr,incode)
c**********************************************************************
      call shape12 (1,nt,ksy,ka,dummy,el2m,xi,eta,incode,w)
      call shape34 (1,nt,ksy,ka,el3m,el4m,xi,eta,incode,w)
      call shape56 (1,nt,ksy,ka,el5m,el6m,xi,eta,incode,w)
      call shape78 (1,nt,ksy,ka,el7m,el8m,xi,eta,incode,w)
c      write(6,*) 'in tshap_mi, done first segment'
c     do 100 jj = 1,ksx
c 100   write(6,999) (w(ii,jj),ii=1,ksy)
      tphi = phi
      talmds = almds
      almds = almds*phi*phi/(phir*phir)
      phi = phir
      call shape12 (nt+1,nr,ksy,ka,dummy,em2m,xi,eta,incode,w)
      call shape34 (nt+1,nr,ksy,ka,em3m,em4m,xi,eta,incode,w)
      call shape56 (nt+1,nr,ksy,ka,dummy,em6m,xi,eta,incode,w)
      call shape78 (nt+1,nr,ksy,ka,em7m,em8m,xi,eta,incode,w)
```

```
        do 20 i = 1, ksy
          do 20 j = nt+1, nr
            w(i,j) = -1.0*w(i,j)
   20   continue
c       do 101 jj = 1,ksx
c  101    write(6,999) (w(ii,jj),ii=1,ksy)

        almds = talmds
        phi = tphi
c         write(6,*) 'in tshap_mi, done 2nd segment'
        call shape12 (nr+1,ksx,ksy,ka,er1m,er2m,xi,eta,incode,w)
        call shape34 (nr+1,ksx,ksy,ka,er3m,er4m,xi,eta,incode,w)
        call shape56 (nr+1,ksx,ksy,ka,dummy,er6m,xi,eta,incode,w)
        call shape78 (nr+1,ksx,ksy,ka,er7m,er8m,xi,eta,incode,w)
c       do 102 jj = 1,ksx
c  102    write(6,999) (w(ii,jj),ii=1,ksy)

        write(6,*) 'Output file? (Use quotation marks)'
        read(5,*) outfile
        write(6,*) 'Writing displacement matrix to file ',outfile
        open(25,file=outfile,status='new')
        do 106 jj = 1,ksx
  106     write(25,999) (w(ii,jj),ii=1,ksy)
  999     format (21(e10.3,1x))
        close(25)
        return
        end




cc*********************************************************************
        subroutine tshap_hi (ks1,x,w)
c*********************************************************************
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension x(320),xi(126),eta(21),incode(21,126)
      dimension el2m(16),el3m(16),el4m(16),el7m(16),el8m(16)
      dimension em2m(16),em3m(16),em4m(16),em6m(16),em7m(16),em8m(16)
      dimension er3m(16),er4m(16),er6m(16),er7m(16),er8m(16),dummy(16)
      dimension w(21,126)
      character*14 outfile
c       write(*,*) ' entering tshap_hi'
      ka = k
c       write(6,*)(x(i),i=1,ka)
      do 10 i = 1,ka
        dummy(i) = 0.0
        el2m(i) = x(i)
        el3m(i) = x(ka+i)
        el4m(i) = x(2*ka+i)
        el7m(i) = x(3*ka+i)
        el8m(i) = x(4*ka+i)
        em2m(i) = x(5*ka+i)
        em3m(i) = x(6*ka+i)
        em4m(i) = x(7*ka+i)
        em6m(i) = x(8*ka+i)
        em7m(i) = x(9*ka+i)
        em8m(i) = x(10*ka+i)
        er3m(i) = x(11*ka+i)
        er4m(i) = x(12*ka+i)
        er6m(i) = x(13*ka+i)
        er7m(i) = x(14*ka+i)
        er8m(i) = x(15*ka+i)
   10 continue
      bigphi = phi*phir/(2.0*phir+phi)
```

```
      ksy = ks1
      ksx = int((ksy-1.0)/bigphi + 2.0)
      nt = int(bigphi*(ksx-1)/phi)+1
      nr = ksx - nt
      if (ksx.gt.126) ksx = 126
       do 15 i = 1, ksy
         do 15 j = 1, ksx
           w(i,j) = 0.0
   15  continue
cc*******************************************************************
       call grid (ksx,ksy,xi,eta,nt,nr,incode)
c********************************************************************
       call shape12 (1,nt,ksy,ka,dummy,el2m,xi,eta,incode,w)
       call shape34 (1,nt,ksy,ka,el3m,el4m,xi,eta,incode,w)
       call shape78 (1,nt,ksy,ka,el7m,el8m,xi,eta,incode,w)
       tphi = phi
       talmds = almds
       almds = almds*phi*phi/(phir*phir)
       phi = phir
       call shape12 (nt+1,nr,ksy,ka,dummy,em2m,xi,eta,incode,w)
       call shape34 (nt+1,nr,ksy,ka,em3m,em4m,xi,eta,incode,w)
       call shape56 (nt+1,nr,ksy,ka,dummy,em6m,xi,eta,incode,w)
       call shape78 (nt+1,nr,ksy,ka,em7m,em8m,xi,eta,incode,w)
       do 20 i = 1, ksy
         do 20 j = nt+1, nr
           w(i,j) = -1.0*w(i,j)
   20  continue
       almds = talmds
       phi = tphi
       call shape34 (nr+1,ksx,ksy,ka,er3m,er4m,xi,eta,incode,w)
       call shape56 (nr+1,ksx,ksy,ka,dummy,em6m,xi,eta,incode,w)
       call shape78 (nr+1,ksx,ksy,ka,er7m,er8m,xi,eta,incode,w)
       write(6,*) 'Output file? (Use quotation marks)'
       read(5,*) outfile
       write(6,*) 'Writing displacement matrix to file ',outfile
       open(25,file=outfile,status='new')
       do 106 jj = 1,ksx
  106   write(25,999) (w(ii,jj)),ii=1,ksy)
  999  format (21(e10.3,1x))
       close(25)
       return
       end
```

4. Common Subroutines

```
c
c*******************************************************************
      subroutine bb0 (as)
c*******************************************************************
c
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension as(32,16)
       do 200 i = 1,ki
       do 200 j = 1,kj
           as(i,j) = 0.
  200  continue
      return
      end
```

```
C***********************************************************************
      subroutine bb12 (ibb,icon,as)
C***********************************************************************
       implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension as(32,16)
      phis = phi*phi
      do 200 j = 1,kj
       emp = j*pi
       emps = emp*emp
       bms = phis*(almds+emps)
       bm  = dsqrt(bms)
       testm = almds-emps
       gms = phis*testm
       if (gms.lt.0.0) gms = -gms
       gm  = dsqrt(gms)
       if (ibb.eq.1) then
          if (testm.lt.0.0) then
             if (bms.gt.qlim) then
              tdm  = -(bms-poi*phis*emps)
     1                 /(gms-poi*phis*emps)
              tddm = 1.0/(1.0+tdm)
             else
              tdm  = -(bms-poi*phis*emps)*dsinh(bm)
     1                 /((gms-poi*phis*emps)*dsinh(gm))
              tddm = 1.0/(dsinh(bm)+tdm*dsinh(gm))
             endif
          else
              tdm  = (bms-poi*phis*emps)*dsinh(bm)
     1                 /((gms+poi*phis*emps)*dsin(gm))
              tddm = 1.0/(dsinh(bm)+tdm*dsin(gm))
          endif
       else
          if (testm.lt.0.0) then
             if (bms.gt.qlim) then
              tdm  = -1.0
              tddm = -1.0/(bms+tdm*gms)
             else
              tdm  = -dsinh(bm)/dsinh(gm)
              tddm = -1.0/(bms*dsinh(bm)+tdm*gms*dsinh(gm))
             endif
          else
              tdm  = -dsinh(bm)/dsin(gm)
              tddm = -1.0/(bms*dsinh(bm)-tdm*gms*dsin(gm))
          endif
       endif
       do 200 i = 1,ki
      enp = i*pi
      enps = enp*enp
      if (iedge.eq.1) then
          as(i,j) = 0.0
          if ((icon.eq.1).and.(ibb.eq.1)) then
             as(j,j) = 1.0
          elseif (icon.eq.2) then
             if (testm.lt.0.0) then
               if (bms.gt.qlim) then
                 as(j,j) = tddm*(bm+tdm*gm)
               else
                 as(j,j) = tddm*(bm*dcosh(bm)+tdm*gm*dcosh(gm))
               endif
             else
                 as(j,j) = tddm*(bm*dcosh(bm)+tdm*gm*dcos(gm))
             endif
          elseif ((icon.eq.3).and.(ibb.eq.2)) then
             as(j,j) = 1.0
          elseif (icon.eq.4) then
             if (testm.lt.0.0) then
```

```
            if (bms.gt.qlim) then
              a1n = tddm*(bms*bm+tdm*gms*gm)
              a2n = -tddm*emps*(bm+tdm*gm)
            else
              a1n = tddm*(bms*bm*dcosh(bm)
1                      +tdm*gms*gm*dcosh(gm))
              a2n = -tddm*emps*(bm*dcosh(bm)+tdm*gm*dcosh(gm))
            endif
          else
              a1n = tddm*(bms*bm*dcosh(bm)
1                      -tdm*gms*gm*dcos(gm))
              a2n = -tddm*emps*(bm*dcosh(bm)+tdm*gm*dcos(gm))
          endif
          as(j,j) = -(a1n+(2.0-poi)*phis*a2n)
        endif
      elseif ((iedge.eq.2).or.(iedge.eq.4)) then
        if ((icon.eq.1).or.(icon.eq.3)) then
          as(i,j) = 0.0
        else
          if ((testm.lt.0.0).and.(bms.gt.qlim)) then
           ex1 = -2.0*enp*dcos(enp)/(bms+enps)
           ex2 = -2.0*enp*dcos(enp)/(gms+enps)
          else
           ex1 = -2.0*enp*dcos(enp)*dsinh(bm)/(bms+enps)
           if (testm.lt.0.0) then
               ex2 = -2.0*enp*dcos(enp)*dsinh(gm)/(gms+enps)
           else
               ex2 = 2.0*enp*dcos(enp)*dsin(gm)/(gms-enps)
           endif
          endif
          if (icon.eq.2) then
             as(i,j) = tddm*emp*(ex1+tdm*ex2)
          else
             a1n = -tddm*emps*emp*(ex1+tdm*ex2)
             if (testm.lt.0.0) ex2 = -1.0*ex2
             a2n = tddm*emp*(bms*ex1-tdm*gms*ex2)
             as(i,j) = -(a1n+(2.0-poi)*a2n/phis)
          endif
          if (iedge.eq.2) as(i,j) = as(i,j)*dcos(emp)
        endif
      elseif (iedge.eq.3) then
        as(i,j) = 0.0
        if (icon.eq.2) then
          if ((testm.lt.0.0).and.(gms.gt.qlim)) then
              as(j,j) = 0.0
          elseif ((testm.lt.0.0).and.(bms.gt.qlim)) then
              as(j,j) = tddm*tdm*gm/dsinh(gm)
          else
              as(j,j) = tddm*(bm+tdm*gm)
          endif
        elseif (icon.eq.4) then
          if (testm.lt.0.0) then
            if (bms.gt.qlim) then
               if (gms.gt.qlim) then
              a1n = 0.0
              a2n = 0.0
               else
              a1n = tddm*tdm*gms*gm/dsinh(gm)
              a2n = -tddm*emps*tdm*gm/dsinh(gm)
               endif
            else
              a1n = tddm*(bms*bm+tdm*gms*gm)
              a2n = -tddm*emps*(bm+tdm*gm)
            endif
          else
            a1n = tddm*(bms*bm-tdm*gms*gm)
            a2n = -tddm*emps*(bm+tdm*gm)
```

```
          endif
          as(j,j) = -(a1n+(2.0-poi)*phis*a2n)
        endif
      elseif (iedge.eq.12) then
        if ((icon.eq.1).or.(icon.eq.3)) then
          if ((testm.lt.0.0).and.(bms.gt.qlim)) then
          ex1 = bm/(bms+(emp-enp)*(emp-enp))
          ex2 = -bm/(bms+(emp+enp)*(emp+enp))
          ex3 = gm/(gms+(emp-enp)*(emp-enp))
          ex4 = -gm/(gms+(emp+enp)*(emp+enp))
          else
          ex1 = bm*(dcosh(bm)-dcos(emp-enp))
     1               /(bms+(emp-enp)*(emp-enp))
          ex2 = bm*(dcos(emp+enp)-dcosh(bm))
     1               /(bms+(emp+enp)*(emp+enp))
          if (testm.lt.0.0) then
              ex3 = gm*(dcosh(gm)-dcos(emp-enp))
     1                   /(gms+(emp-enp)*(emp-enp))
              ex4 = gm*(dcos(emp+enp)-dcosh(gm))
     1                    /(gms+(emp+enp)*(emp+enp))
          else
              if (gms.eq.((emp-enp)*(emp-enp))) then
                ex3 = dsin(gm)/2.0
              else
                ex3 = gm*(dcos(emp-enp)-dcos(gm))
     1                     /(gms-(emp-enp)*(emp-enp))
              endif
              if (gms.eq.((emp+enp)*(emp+enp))) then
                ex4 = -dsin(gm)/2.0
              else
                ex4 = gm*(dcos(gm)-dcos(emp+enp))
     1                     /(gms-(emp+enp)*(emp+enp))
              endif
          endif
         endif
         if (icon.eq.1) then
           as(i,j) = tddm*(ex1+ex2+tdm*(ex3+ex4))
         else
           if ((testm.lt.0.0).and.(bms.gt.qlim)) then
             y1 = (emp+enp)/(bms+(emp+enp)*(emp+enp))
             y2 = -(emp-enp)/(bms+(emp-enp)*(emp-enp))
             y3 = (emp+enp)/(gms+(emp+enp)*(emp+enp))
             y4 = -(emp-enp)/(gms+(emp-enp)*(emp-enp))
             a2n = tddm*(bms*(ex1+ex2)+tdm*gms*(ex3+ex4))
           else
             y1 = (emp+enp)*(dcosh(bm)-dcos(emp+enp))
     1                         /(bms+(emp+enp)*(emp+enp))
             y2 = (emp-enp)*(dcos(emp-enp)-dcosh(bm))
     1                         /(bms+(emp-enp)*(emp-enp))
             if (testm.lt.0.) then
               y3 = (emp+enp)*(dcosh(gm)-dcos(emp+enp))
     1                         /(gms+(emp+enp)*(emp+enp))
               y4 = (emp-enp)*(dcos(emp-enp)-dcosh(gm))
     1                         /(gms+(emp-enp)*(emp-enp))
               a2n = tddm*(bms*(ex1+ex2)+tdm*gms*(ex3+ex4))
             else
               if (gms.eq.((emp+enp)*(emp+enp))) then
                   y3 = dsin(gm)/2.0
               else
                   y3 = (emp+enp)*(dcos(gm)-dcos(emp+enp))
     1                             /((emp+enp)*(emp+enp)-gms)
               endif
               if (gms.eq.((emp-enp)*(emp-enp))) then
                   y4 = -dsin(gm)/2.0
               else
                   y4 = (emp-enp)*(dcos(gm)-dcos(emp-enp))
     1                             /(gms-(emp-enp)*(emp-enp))
```

```
                    endif
                    a2n = tddm*(bms*(ex1+ex2)-tdm*gms*(ex3+ex4))
                 endif
              endif
              a1n = -tddm*emps*(ex1+ex2+tdm*(ex3+ex4))
              a3n = tddm*emp*(bm*(y1+y2)+tdm*gm*(y3+y4))
              td1 = dcos(alpha)*dcos(alpha)
     1                             +poi*dsin(alpha)*dsin(alpha)
              td2 = dsin(alpha)*dsin(alpha)
     1                             +poi*dcos(alpha)*dcos(alpha)
              td3 = (1.0-poi)*dsin(2.0*alpha)
              as(i,j) = -(td1*phis*a1n+td2*a2n+td3*phi*a3n)
            endif
          elseif (icon.eq.2) then
            if ((testm.lt.0.0).and.(bms.gt.qlim)) then
            ex1 = (emp+enp)/(bms+(emp+enp)*(emp+enp))
            ex2 = -(emp-enp)/(bms+(emp-enp)*(emp-enp))
            ex3 = (emp+enp)/(gms+(emp+enp)*(emp+enp))
            ex4 = -(emp-enp)/(gms+(emp-enp)*(emp-enp))
            y1 = bm/(bms+(emp-enp)*(emp-enp))
            y2 = -bm/(bms+(emp+enp)*(emp+enp))
            y3 = gm/(gms+(emp-enp)*(emp-enp))
            y4 = -gm/(gms+(emp+enp)*(emp+enp))
           else
            ex1 = (emp+enp)*dsinh(bm)/(bms+(emp+enp)*(emp+enp))
            ex2 = -(emp-enp)*dsinh(bm)/(bms+(emp-enp)*(emp-enp))
            y1 = bm*dsinh(bm)/(bms+(emp-enp)*(emp-enp))
            y2 = -bm*dsinh(bm)/(bms+(emp+enp)*(emp+enp))
            if (testm.lt.0.0) then
                ex3 = (emp+enp)*dsinh(gm)
     1                   /(gms+(emp+enp)*(emp+enp))
                ex4 = -(emp-enp)*dsinh(gm)
     1                   /(gms+(emp-enp)*(emp-enp))
                y3 = gm*dsinh(gm)
     1                   /(gms+(emp-enp)*(emp-enp))
                y4 = -gm*dsinh(gm)
     1                   /(gms+(emp+enp)*(emp+enp))
            else
                if (gms.eq.((emp+enp)*(emp+enp))) then
                   ex3 = dsin(gm)/2.0/gm-dcos(gm)/2.0
                   y4 = ex3
                else
                   ex3 = (emp+enp)*dsin(gm)
     1                         /((emp+enp)*(emp+enp)-gms)
                   y4 = gm*dsin(gm)/((emp+enp)*(emp+enp)-gms)
                endif
                if (gm.eq.(emp-enp)) then
                   ex4 = dcos(gm)/2.0-dsin(gm)/2.0/gm
                   y3 = ex4
                elseif (gms.eq.(enp-emp)) then
                   ex4 = dsin(gm)/2.0/gm-dcos(gm)/2.0
                   y3 = -ex4
                else
                   ex4 = (emp-enp)*dsin(gm)
     1                      /(gms-(emp-enp)*(emp-enp))
                   y3 = gm*dsin(gm)/(gms-(emp-enp)*(emp-enp))
                endif
           endif
          endif
          a1n = tddm*emp*(ex1+ex2+tdm*(ex3+ex4))
          a2n = tddm*(bm*(y1+y2)+tdm*gm*(y3+y4))
          as(i,j) = a1n*dcos(alpha)+(a2n*dsin(alpha)/phi)
        endif
      endif
200 continue
      return
      end
```

```
C*********************************************************************
      subroutine bb34 (ibb,icon,as)
C*********************************************************************
C      building blocks 3 and 4
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension as(32,16)
      phis = phi*phi
      do 200 j = 1,kj
       emp = j*pi
       emps = emp*emp
       bms = (1.0/phis)*(phis*almds+emps)
       bm  = dsqrt(bms)
       testm = phis*almds-emps
       gms = (1.0/phis)*testm
       if (gms.lt.0.0) gms = -gms
       gm  = dsqrt(gms)
       if (ibb.eq.3) then
          if (testm.lt.0.) then
             if (bms.gt.qlim) then
              tdm   = -(bms-poi*(1.0/phis)*emps)
     1                  /((gms-poi*(1.0/phis)*emps))
              tddm = 1.0/(1.0+tdm)
             else
              tdm   = -(bms-poi*(1.0/phis)*emps)*dsinh(bm)
     1                  /((gms-poi*(1.0/phis)*emps)*dsinh(gm))
              tddm = 1.0/(dsinh(bm)+tdm*dsinh(gm))
             endif
          else
              tdm   = (bms-poi*(1.0/phis)*emps)*dsinh(bm)
     1                  /((gms+poi*(1.0/phis)*emps)*dsin(gm))
              tddm = 1.0/(dsinh(bm)+tdm*dsin(gm))
          endif
      else
          if (testm.lt.0.0) then
             if (bms.gt.qlim) then
              tdm   = -1.0
              tddm = -1.0/(bms+tdm*gms)
             else
              tdm   = -dsinh(bm)/dsinh(gm)
              tddm = -1.0/(bms*dsinh(bm)+tdm*gms*dsinh(gm))
             endif
          else
              tdm   = -dsinh(bm)/dsin(gm)
              tddm = -1.0/(bms*dsinh(bm)-tdm*gms*dsin(gm))
          endif
      endif
      do 200 i = 1,ki
      enp = i*pi
      enps = enp*enp
      if ((iedge.eq.1).or.(iedge.eq.3)) then
          if ((icon.eq.1).or.(icon.eq.3)) then
             as(i,j) = 0.0
          else
            if ((testm.lt.0.0).and.(bms.gt.qlim)) then
             ex1 = -2.0*enp*dcos(enp)/(bms+enps)
             ex2 = -2.0*enp*dcos(enp)/(gms+enps)
            else
             ex1 = -2.0*enp*dcos(enp)*dsinh(bm)/(bms+enps)
             if (testm.lt.0.0) then
                  ex2 = -2.0*enp*dcos(enp)*dsinh(gm)/(gms+enps)
             else
                  ex2 = 2.0*enp*dcos(enp)*dsin(gm)/(gms-enps)
             endif
            endif
            if (icon.eq.2) then
```

```
            as(i,j) = tddm*emp*(ex1+tdm*ex2)
         else
            a1n = -tddm*emps*emp*(ex1+tdm*ex2)
            if (testm.lt.0.) ex2 = -ex2
            a2n = tddm*emp*(bms*ex1-tdm*gms*ex2)
            as(i,j) = -(a1n+(2.0-poi)*phis*a2n)
         endif
       endif
       if (iedge.eq.1) as(i,j) = as(i,j)*dcos(emp)
  elseif (iedge.eq.2) then
       as(i,j) = 0.0
       if ((icon.eq.1).and.(ibb.eq.3)) then
          as(j,j) = 1.0
       elseif (icon.eq.2) then
          if ((testm.lt.0.0).and.(bms.gt.qlim)) then
             as(j,j) = tddm*(bm+tdm*gm)
          elseif (testm.lt.0.0) then
             as(j,j) = tddm*(bm*dcosh(bm)+tdm*gm*dcosh(gm))
          else
             as(j,j) = tddm*(bm*dcosh(bm)+tdm*gm*dcos(gm))
          endif
       elseif ((icon.eq.3).and.(ibb.eq.4)) then
          as(j,j) = 1.0
       elseif (icon.eq.4) then
          if (testm.lt.0.0) then
            if (bms.gt.qlim) then
             a1n = tddm*(bms*bm+tdm*gms*gm)
             a2n = -tddm*emps*(bm+tdm*gm)
            else
             a1n = tddm*(bms*bm*dcosh(bm)+tdm*gms*gm*dcosh(gm))
             a2n = -tddm*emps*(bm*dcosh(bm)+tdm*gm*dcosh(gm))
            endif
          else
             a1n = tddm*(bms*bm*dcosh(bm)-tdm*gms*gm*dcos(gm))
             a2n = -tddm*emps*(bm*dcosh(bm)+tdm*gm*dcos(gm))
          endif
          as(j,j) = -(a1n+(2.0-poi)*a2n/phis)
       endif
  elseif (iedge.eq.4) then
       as(i,j) = 0.0
       if (icon.eq.2) then
         if ((testm.lt.0.0).and.(gms.gt.qlim)) then
            as(j,j) = 0.0
         elseif ((testm.lt.0.0).and.(bms.gt.qlim)) then
            as(j,j) = tddm*tdm*gm/dsinh(gm)
         else
            as(j,j) = tddm*(bm+tdm*gm)
         endif
       elseif (icon.eq.4) then
         if (testm.lt.0.0) then
             if (gms.gt.qlim) then
                a1n = 0.0
                a2n = 0.0
             elseif (bms.gt.qlim) then
                a1n = tddm*tdm*gms*gm/dsinh(gm)
                a2n = -tddm*tdm*gm/dsinh(gm)
             else
                a1n = tddm*(bms*bm+tdm*gms*gm)
                a2n = -tddm*emps*(bm+tdm*gm)
             endif
         else
             a1n = tddm*(bms*bm-tdm*gms*gm)
             a2n = -tddm*emps*(bm+tdm*gm)
         endif
         as(j,j) = -(a1n+(2.0-poi)*a2n/phis)
       endif
  elseif (iedge.eq.12) then
```

```
      if ((icon.eq.1).or.(icon.eq.3)) then
         if ((testm.lt.0.0).and.(bms.gt.qlim)) then
            ex1 = bm*dcos(enp)
     1                       /(bms+(emp+enp)*(emp+enp))
            ex2 = -bm*dcos(enp)
     1                       /(bms+(emp-enp)*(emp-enp))
            ex3 = gm*dcos(enp)
     1                       /(gms+(emp+enp)*(emp+enp))
            ex4 = -gm*dcos(enp)
     1                       /(gms+(emp-enp)*(emp-enp))
         else
           ex1 = bm*(dcosh(bm)*dcos(enp)-dcos(emp))
     1                    /(bms+(emp+enp)*(emp+enp))
           ex2 = bm*(dcos(emp)-dcosh(bm)*dcos(enp))
     1                    /(bms+(emp-enp)*(emp-enp))
           if (testm.lt.0.0) then
             ex3 = gm*(dcosh(gm)*dcos(enp)-dcos(emp))
     1                      /(gms+(emp+enp)*(emp+enp))
             ex4 = gm*(dcos(emp)-dcosh(gm)*dcos(enp))
     1                      /(gms+(emp-enp)*(emp-enp))
           else
             ex3 = gm*(dcos(emp)-dcos(gm+enp))
     1                /(gms-(emp+enp)*(emp+enp))
             ex4 = gm*(dcos(gm+enp)-dcos(emp))
     1                /(gms-(emp-enp)*(emp-enp))
           endif
         endif
         if (icon.eq.1) then
             as(i,j) = tddm*(ex1+ex2+tdm*(ex3+ex4))
         else
           if ((testm.lt.0.0).and.(bms.gt.qlim)) then
             y1 = (emp-enp)*dcos(enp)
     1                           /(bms+(emp-enp)*(emp-enp))
             y2 = -(emp+enp)*dcos(enp)
     1                           /(bms+(emp+enp)*(emp+enp))
             y3 = (emp-enp)*dcos(enp)
     1                           /(gms+(emp-enp)*(emp-enp))
             y4 = -(emp+enp)*dcos(enp)
     1                           /(gms+(emp+enp)*(emp+enp))
             a1n = tddm*(bms*(ex1+ex2)+tdm*gms*(ex3+ex4))
           else
             y1 = (emp-enp)*(dcosh(bm)*dcos(enp)-dcos(emp))
     1                           /(bms+(emp-enp)*(emp-enp))
             y2 = (emp+enp)*(dcos(emp)-dcosh(bm)*dcos(enp))
     1                           /(bms+(emp+enp)*(emp+enp))
             if (testm.lt.0.0) then
               y3 = (emp-enp)*(dcosh(gm)*dcos(enp)-dcos(emp))
     1                               /(gms+(emp-enp)*(emp-enp))
               y4 = (emp+enp)*(dcos(emp)-dcosh(gm)*dcos(enp))
     1                               /(gms+(emp+enp)*(emp+enp))
               a1n = tddm*(bms*(ex1+ex2)+tdm*gms*(ex3+ex4))
             else
               y3 = (emp-enp)*(dcos(emp)-dcos(gm+enp))
     1                      /(gms-(emp-enp)*(emp-enp))
               y4 = (emp+enp)*(dcos(gm+enp)-dcos(emp))
     1                      /(gms-(emp+enp)*(emp+enp))
               a1n = tddm*(bms*(ex1+ex2)-tdm*gms*(ex3+ex4))
             endif
           endif
           a2n = -tddm*emps*(ex1+ex2+tdm*(ex3+ex4))
           a3n = tddm*emp*(bm*(y1+y2)+tdm*gm*(y3+y4))
           td1 = dcos(alpha)*dcos(alpha)
     1                         +poi*dsin(alpha)*dsin(alpha)
           td2 = dsin(alpha)*dsin(alpha)
     1                         +poi*dcos(alpha)*dcos(alpha)
           td3 = (1.0-poi)*dsin(2.0*alpha)
           as(i,j) = -(td1*phis*a1n+td2*a2n+td3*phi*a3n)
```

```
            endif
          elseif (icon.eq.2) then
            if ((testm.lt.0.0).and.(bms.gt.qlim)) then
             ex1 = bm*dcos(enp)/(bms+(emp+enp)*(emp+enp))
             ex2 = -bm*dcos(enp)/(bms+(emp-enp)*(emp-enp))
             ex3 = gm*dcos(enp)/(gms+(emp+enp)*(emp+enp))
             ex4 = -gm*dcos(enp)/(gms+(emp-enp)*(emp-enp))
             y1  = (emp-enp)*dcos(enp)/(bms+(emp-enp)*(emp-enp))
             y2  = -(emp+enp)*dcos(enp)/(bms+(emp+enp)*(emp+enp))
             y3 = (emp-enp)*dcos(enp)/(gms+(emp-enp)*(emp-enp))
             y4 = -(emp+enp)*dcos(enp)/(gms+(emp+enp)*(emp+enp))
            else
             ex1 = bm*dcos(enp)*dsinh(bm)/(bms+(emp+enp)*(emp+enp))
             ex2 = -bm*dcos(enp)*dsinh(bm)/(bms+(emp-enp)*(emp-enp))
             y1  = (emp-enp)*dcos(enp)*dsinh(bm)
     1                 (bms+(emp-enp)*(emp-enp))
             y2  = -(emp+enp)*dcos(enp)*dsinh(bm)
     1                  (bms+(emp+enp)*(emp+enp))
             if (testm.lt.0.0) then
                 ex3 = gm*dcos(enp)*dsinh(gm)
     1                   /(gms+(emp+enp)*(emp+enp))
                 ex4 = -gm*dcos(enp)*dsinh(gm)
     1                   (gms+(emp-enp)*(emp-enp))
                 y3 = (emp-enp)*dcos(enp)*dsinh(gm)
     1                     (gms+(emp-enp)*(emp-enp))
                 y4 = -(emp+enp)*dcos(enp)*dsinh(gm)
     1                      (gms+(emp+enp)*(emp+enp))
             else
                 ex3 = gm*dsin(gm+enp)/(gms-(emp+enp)*(emp+enp))
                 ex4 = -gm*dsin(gm+enp)/(gms-(emp-enp)*(emp-enp))
                 y3  = (emp-enp)*dsin(gm+enp)
     1                  /((emp-enp)*(emp-enp)-gms)
                 y4  = (emp+enp)*dsin(gm-enp)
     1                  /(gms-(emp+enp)*(emp+enp))
             endif
            endif
            a1n = tddm*(bm*(ex1+ex2)+tdm*gm*(ex3+ex4))
            a2n = tddm*emp*(y1+y2+tdm*(y3+y4))
            as(i,j) = a1n*dcos(alpha)+(a2n*dsin(alpha)/phi)
          else
c              icon = 4 yet to be added
            as(i,j) = 0.0
          endif
      endif
  200 continue
      return
      end


c**********************************************************************
      subroutine bb56 (ibb,icon,as)
c**********************************************************************
c      building blocks 5 and 6
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension as(32,16)
      phis = phi*phi
      do 200 j = 1,kj
       emp = j*pi
       emps = emp*emp
       bms = phis*(almds+emps)
       bm  = dsqrt(bms)
       testm = almds-emps
       gms = phis*testm
       if (gms.lt.0.0) gms = -gms
       gm  = dsqrt(gms)
```

```
      if (ibb.eq.5) then
         if (testm.lt.0.0) then
            if (bms.gt.qlim) then
             tdm   = -(bms-poi*phis*emps)
     1                 /(gms-poi*phis*emps)
             tddm = 1.0/(1.0+tdm)
            else
             tdm   = -(bms-poi*phis*emps)*dsinh(bm)
     1                 /((gms-poi*phis*emps)*dsinh(gm))
             tddm = 1.0/(dsinh(bm)+tdm*dsinh(gm))
            endif
         else
             tdm   = (bms-poi*phis*emps)*dsinh(bm)
     1                 /((gms+poi*phis*emps)*dsin(gm))
             tddm = 1.0/(dsinh(bm)+tdm*dsin(gm))
         endif
      else
         if (testm.lt.0.0) then
            if (bms.gt.qlim) then
             tdm   = -1.0
             tddm = -1.0/(bms+tdm*gms)
            else
             tdm   = -dsinh(bm)/dsinh(gm)
             tddm = -1.0/(bms*dsinh(bm)+tdm*gms*dsinh(gm))
            endif
         else
             tdm   = -dsinh(bm)/dsin(gm)
             tddm = -1.0/(bms*dsinh(bm)-tdm*gms*dsin(gm))
         endif
      endif
      do 200 i = 1,ki
      enp = i*pi
      enps = enp*enp
      if (iedge.eq.1) then
         as(i,j) = 0.0
         if (icon.eq.2) then
           if ((testm.lt.0.0).and.(gms.gt.qlim)) then
              as(j,j) = 0.0
           elseif ((testm.lt.0.0).and.(bms.gt.qlim)) then
              as(j,j) = -tddm*tdm*gm/dsinh(gm)
           else
              as(j,j) = -tddm*(bm+tdm*gm)
           endif
         elseif (icon.eq.4) then
           if (testm.lt.0.0) then
              if (gms.gt.qlim) then
                 a1n = 0.0
                 a2n = 0.0
              elseif (bms.gt.qlim) then
                 a1n = -tddm*tdm*gms*gm/dsinh(gm)
                 a2n = tddm*emps*tdm*gm/dsinh(gm)
              else
                 a1n = -tddm*(bms*bm+tdm*gms*gm)
                 a2n = tddm*emps*(bm+tdm*gm)
              endif
           else
                 a1n = -tddm*(bms*bm-tdm*gms*gm)
                 a2n = tddm*emps*(bm+tdm*gm)
           endif
           as(j,j) = -(a1n+(2.0-poi)*phis*a2n)
         endif
      elseif ((iedge.eq.2).or.(iedge.eq.4)) then
         if ((icon.eq.1).or.(icon.eq.3)) then
           as(i,j) = 0.0
         else
           if ((testm.lt.0.0).and.(bms.gt.qlim)) then
               ex1 = 2.0*enp/(bms+enps)
```

```
                ex2 = 2.0*enp/(gms+enps)
              else
                ex1 = 2.0*enp*dsinh(bm)/(bms+enps)
                if (testm.lt.0.0) then
                    ex2 = 2.0*enp*dsinh(gm)/(gms+enps)
                else
                    ex2 = 2.0*enp*dsin(gm)/(enps-gms)
                endif
              endif
              if (icon.eq.2) then
                  as(i,j) = tddm*emp*(ex1+tdm*ex2)
              else
                  a1n = -tddm*emps*emp*(ex1+tdm*ex2)
                  if (testm.lt.0.0) then
                     a2n = tddm*emp*(bms*ex1+tdm*gms*ex2)
                  else
                     a2n = tddm*emp*(bms*ex1-tdm*gms*ex2)
                  endif
                  as(i,j) = -(a1n+(2.0-poi)*a2n/phis)
              endif
              if (iedge.eq.2) as(i,j) = as(i,j)*dcos(emp)
            endif
        elseif (iedge.eq.3) then
            as(i,j) = 0.0
            if ((icon.eq.1).and.(ibb.eq.5)) then
              as(j,j) = 1.0
            elseif (icon.eq.2) then
              if (testm.lt.0.) then
                if (bms.gt.qlim) then
                  as(j,j) = -tddm*(bm+tdm*gm)
                else
                  as(j,j) = -tddm*(bm*dcosh(bm)+tdm*gm*dcosh(gm))
                endif
              else
                  as(j,j) = -tddm*(bm*dcosh(bm)+tdm*gm*dcos(gm))
              endif
            elseif ((icon.eq.3).and.(ibb.eq.6)) then
                  as(j,j) = 1.0
            elseif (icon.eq.4) then
              if (testm.lt.0.0) then
                if (bms.gt.qlim) then
                  a1n = -tddm*(bms*bm+tdm*gms*gm)
                  a2n = tddm*emps*(bm+tdm*gm)
                else
                  a1n = -tddm*(bms*bm*dcosh(bm)
     1                     +tdm*gms*gm*dcosh(gm))
                  a2n = tddm*emps*(bm*dcosh(bm)
     1                          +tdm*gm*dcosh(gm))
                endif
              else
                  a1n = -tddm*(bms*bm*dcosh(bm)
     1                      -tdm*gms*gm*dcos(gm))
                  a2n = tddm*emps*(bm*dcosh(bm)
     1                          +tdm*gm*dcos(gm))
              endif
              as(j,j) = -(a1n+(2.0-poi)*phis*a2n)
            endif
        elseif (iedge.eq.12) then
            if ((icon.eq.1).or.(icon.eq.3)) then
              if ((testm.lt.0.0).and.(bms.gt.qlim)) then
              ex1 = bm*dcos(emp-enp)/(bms+(emp-enp)*(emp-enp))
              ex2 = -bm*dcos(emp+enp)/(bms+(emp+enp)*(emp+enp))
              ex3 = gm*dcos(emp-enp)/(gms+(emp-enp)*(emp-enp))
              ex4 = -gm*dcos(emp+enp)/(gms+(emp+enp)*(emp+enp))
              else
              ex1 = bm*(dcosh(bm)*dcos(emp-enp)-1.0)
     1                 /(bms+(emp-enp)*(emp-enp))
```

```
          ex2 = bm*(1.0-dcosh(bm)*dcos(emp+enp))
     1                 /(bms+(emp+enp)*(emp+enp))
          if (testm.lt.0.0) then
              ex3 = gm*(dcosh(gm)*dcos(emp-enp)-1.0)
     1                 /(gms+(emp-enp)*(emp-enp))
              ex4 = gm*(1.0-dcosh(gm)*dcos(emp+enp))
     1                 /(gms+(emp+enp)*(emp+enp))
          else
              if (gms.eq.((emp-enp)*(emp-enp))) then
                 ex3 = dsin(gm)*dsin(gm)/2.0/gm
              else
                 ex3 = gm*(1.0-dcos(gm)*dcos(emp-enp))
     1                   /(gms-(emp-enp)*(emp-enp))
              endif
              if (gms.eq.((emp+enp)*(emp+enp))) then
                 ex4 = -dsin(gm)*dsin(gm)/2.0/gm
              else
                 ex4 = gm*(dcos(gm)*dcos(emp+enp)-1.0)
     1                   /(gms-(emp+enp)*(emp+enp))
              endif
          endif
        endif
        if (icon.eq.1) then
            as(i,j) = tddm*(ex1+ex2+tdm*(ex3+ex4))
        else
           if ((testm.lt.0.0).and.(bms.gt.qlim)) then
             y1 =  -(emp+enp)*dcos(emp+enp)
     1                /(bms+(emp+enp)*(emp+enp))
             y2 =   (emp-enp)*dcos(emp-enp)
     1                /(bms+(emp-enp)*(emp-enp))
             y3 =  -(emp+enp)*dcos(emp+enp)
     1                /(gms+(emp+enp)*(emp+enp))
             y4 =  (emp-enp)*dcos(emp-enp)
     1                /(gms+(emp-enp)*(emp-enp))
             a2n = tddm*(bms*(ex1+ex2)+tdm*gms*(ex3+ex4))
           else
             y1 = (emp+enp)*(1.0-dcos(emp+enp)*dcosh(bm))
     1                      /(bms+(emp+enp)*(emp+enp))
             y2 = (emp-enp)*(dcos(emp-enp)*dcosh(bm)-1.0)
     1                      /(bms+(emp-enp)*(emp-enp))
             if (testm.lt.0.0) then
               y3  = (emp+enp)*(1.0-dcos(emp+enp)*dcosh(gm))
     1                          /(gms+(emp+enp)*(emp+enp))
               y4  = (emp-enp)*(dcos(emp-enp)*dcosh(gm)-1.0)
     1                          /(gms+(emp-enp)*(emp-enp))
               a2n = tddm*(bms*(ex1+ex2)+tdm*gms*(ex3+ex4))
             else
               if (gms.eq.((emp+enp)*(emp+enp))) then
                   y3 = dsin(gm)*dsin(gm)/2.0/gm
               else
                   y3 = (emp+enp)*(1.0-dcos(emp+enp)*dcos(gm))
     1                            /((emp+enp)*(emp+enp)-gms)
               endif
               if (gms.eq.(emp-enp)) then
                   y4 = -dsin(gm)*dsin(gm)/2.0/gm
               elseif (gms.eq.(enp-emp)) then
                   y4 = dsin(gm)*dsin(gm)/2.0/gm
               else
                   y4 = (emp-enp)*(dcos(emp-enp)*dcos(gm)-1.0)
     1                            /((emp-enp)*(emp-enp)-gms)
               endif
               a2n = tddm*(bms*(ex1+ex2)-tdm*gms*(ex3+ex4))
             endif
          endif
          a1n = -tddm*emps*(ex1+ex2+tdm*(ex3+ex4))
          a3n = -tddm*emp*(bm*(y1+y2)+tdm*gm*(y3+y4))
          td1 = dcos(alpha)*dcos(alpha)
```

```
     1                                  +poi*dsin(alpha)*dsin(alpha)
               td2 = dsin(alpha)*dsin(alpha)
     1                                  +poi*dcos(alpha)*dcos(alpha)
               td3 = (1.0-poi)*dsin(2.0*alpha)
               as(i,j) = -(td1*phis*a1n+td2*a2n+td3*phi*a3n)
             endif
           elseif (icon.eq.2) then
             if ((testm.lt.0.0).and.(bms.gt.qlim)) then
              ex1 =-(emp+enp)*dcos(emp+enp)/(bms+(emp+enp)*(emp+enp))
              ex2 = (emp-enp)*dcos(emp-enp)/(bms+(emp-enp)*(emp-enp))
              ex3 =-(emp+enp)*dcos(emp+enp)/(gms+(emp+enp)*(emp+enp))
              ex4 = (emp-enp)*dcos(emp-enp)/(gms+(emp-enp)*(emp-enp))
              y1  = bm*dcos(emp-enp)/(bms+(emp-enp)*(emp-enp))
              y2  = -bm*dcos(emp+enp)/(bms+(emp+enp)*(emp+enp))
              y3  = gm*dcos(emp-enp)/(gms+(emp-enp)*(emp-enp))
              y4  = -gm*dcos(emp+enp)/(gms+(emp+enp)*(emp+enp))
             else
              ex1 = -(emp+enp)*dcos(emp+enp)*dsinh(bm)
     1                         /(bms+(emp+enp)*(emp+enp))
              ex2 = (emp-enp)*dcos(emp-enp)*dsinh(bm)
     1                        /(bms+(emp-enp)*(emp-enp))
              y1  = bm*dcos(emp-enp)*dsinh(bm)
     1                 /(bms+(emp-enp)*(emp-enp))
              y2  = -bm*dcos(emp+enp)*dsinh(bm)
     1                 /(bms+(emp+enp)*(emp+enp))
              if (testm.lt.0.0) then
                  ex3 = -(emp+enp)*dcos(emp+enp)*dsinh(gm)
     1                            /(gms+(emp+enp)*(emp+enp))
                  ex4 = (emp-enp)*dcos(emp-enp)*dsinh(gm)
     1                           /(gms+(emp-enp)*(emp-enp))
                  y3  = gm*dcos(emp-enp)*dsinh(gm)
     1                      /(gms+(emp-enp)*(emp-enp))
                  y4  = -gm*dcos(emp+enp)*dsinh(gm)
     1                      /(gms+(emp+enp)*(emp+enp))
              else
                  ex3 = (emp+enp)*dcos(emp+enp)*dsin(gm)
     1                           /(gms-(emp+enp)*(emp+enp))
                  y4  = gm*dsin(gm)*dcos(emp+enp)
     1                     /((emp+enp)*(emp+enp)-gms)
                  ex4 = (emp-enp)*dcos(emp-enp)*dsin(gm)
     1                     /((emp-enp)*(emp-enp)-gms)
                  y3  = gm*dsin(gm)*dcos(emp-enp)
     1                     /(gms-(emp-enp)*(emp-enp))
              endif
             endif
             a1n = tddm*emp*(ex1+ex2+tdm*(ex3+ex4))
             a2n = -tddm*(bm*(y1+y2)+tdm*gm*(y3+y4))
             as(i,j) = a1n*dcos(alpha)+(a2n*dsin(alpha)/phi)
           endif
        endif
  200 continue
        return
        end




C**********************************************************************
      subroutine bb78 (ibb,icon,as)
C**********************************************************************
C
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension as(32,16)
      phis = phi*phi
```

```
      do 200 j = 1,kj
       emp = j*pi
       emps = emp*emp
       bms = (1.0/phis)*(phis*almds+emps)
       bm  = dsqrt(bms)
       testm = phis*almds-emps
       gms = (1.0/phis)*testm
       if (gms.lt.0.) gms = -gms
       gm  = dsqrt(gms)
       if (ibb.eq.7) then
          if (testm.lt.0.0) then
             if (bms.gt.qlim) then
              tdm   = -(bms-poi*(1.0/phis)*emps)
     1                  /(gms-poi*(1.0/phis)*emps)
              tddm = 1.0/(1.0+tdm)
             else
              tdm   = -(bms-poi*(1.0/phis)*emps)*dsinh(bm)
     1                  /((gms-poi*(1.0/phis)*emps)*dsinh(gm))
              tddm = 1.0/(dsinh(bm)+tdm*dsinh(gm))
             endif
          else
              tdm   = (bms-poi*(1.0/phis)*emps)*dsinh(bm)
     1                  /((gms+poi*(1.0/phis)*emps)*dsin(gm))
              tddm = 1.0/(dsinh(bm)+tdm*dsin(gm))
          endif
       else
          if (testm.lt.0.0) then
             if (bms.gt.qlim) then
              tdm   = -1.0
              tddm = -1.0/(bms+tdm*gms)
             else
              tdm   = -dsinh(bm)/dsinh(gm)
              tddm = -1.0/(bms*dsinh(bm)+tdm*gms*dsinh(gm))
             endif
          else
              tdm   = -dsinh(bm)/dsin(gm)
              tddm = -1.0/(bms*dsinh(bm)-tdm*gms*dsin(gm))
          endif
      endif
      do 200 i = 1,ki
      enp = i*pi
      enps = enp*enp
      if ((iedge.eq.1).or.(iedge.eq.3)) then
          if ((icon.eq.1).or.(icon.eq.3)) then
             as(i,j) = 0.0
          else
             if ((testm.lt.0.0).and.(bms.gt.qlim)) then
                   ex1 = 2.0*enp/(enps+bms)
                   ex2 = 2.0*enp/(enps+gms)
             else
                 ex1 = 2.0*enp*dsinh(bm)/(enps+bms)
                 if (testm.lt.0.0) then
                    ex2 = 2.0*enp*dsinh(gm)/(enps+gms)
                 else
                    ex2 = 2.0*enp*dsin(gm)/(enps-gms)
                 endif
             endif
             if (icon.eq.2) then
                 as(i,j) = tddm*emp*(ex1+tdm*ex2)
             else
                 a1n = -tddm*emps*emp*(ex1+tdm*ex2)
                 if (testm.lt.0.0) ex2 = -ex2
                 a2n = tddm*emp*(bms*ex1-tdm*gms*ex2)
                 as(i,j) = -(a1n+(2.0-poi)*phis*a2n)
             endif
             if (iedge.eq.1) as(i,j) = as(i,j)*dcos(emp)
          endif
```

```
 elseif (iedge.eq.2) then
     as(i,j) = 0.0
     if (icon.eq.2) then
          if ((testm.lt.0.0).and.(gms.gt.qlim)) then
               as(j,j) = 0.0
          elseif ((testm.lt.0.0).and.(bms.gt.qlim)) then
               as(j,j) = -tddm*tdm*gm/dsinh(gm)
          else
           as(j,j) = -tddm*(bm+tdm*gm)
          endif
     elseif (icon.eq.4) then
        if (testm.lt.0.0) then
             if (gms.gt.qlim) then
               a1n = 0.0
               a2n = 0.0
             elseif (bms.eq.qlim) then
               a1n = -tddm*tdm*gms*gm/dsinh(gm)
               a2n = tddm*tdm*emps*gm/dsinh(gm)
             else
            a1n = -tddm*(bms*bm+tdm*gms*gm)
               a2n = tddm*emps*(bm+tdm*gm)
             endif
        else
            a1n = -tddm*(bms*bm-tdm*gms*gm)
            a2n = tddm*emps*(bm+tdm*gm)
          endif
        as(j,j) = -(a1n+(2.0-poi)*a2n/phis)
     endif
 elseif (iedge.eq.4) then
     as(i,j) = 0.0
     if ((icon.eq.1).and.(ibb.eq.7)) then
        as(j,j) = 1.0
     elseif (icon.eq.2) then
        if (testm.lt.0.0) then
            if (bms.gt.qlim) then
              as(j,j) = -tddm*(bm+tdm*gm)
            else
              as(j,j) = -tddm*(bm*dcosh(bm)+tdm*gm*dcosh(gm))
            endif
        else
            as(j,j) = -tddm*(bm*dcosh(bm)+tdm*gm*dcos(gm))
        endif
     elseif ((icon.eq.3).and.(ibb.eq.8)) then
        as(j,j) = 1.0
     elseif (icon.eq.4) then
        if (testm.lt.0.0) then
             if (bms.gt.qlim) then
            a1n = -tddm*(bms*bm+tdm*gms*gm)
            a2n = tddm*emps*(bm+gm*tdm)
          else
            a1n = -tddm*(bms*bm*dcosh(bm)+tdm*gms*gm*dcosh(gm))
            a2n = tddm*emps*(bm*dcosh(bm)+gm*tdm*dcosh(gm))
             endif
        else
                 a1n = -tddm*(bms*bm*dcosh(bm)-tdm*gms*gm*dcos(gm))
              a2n = tddm*emps*(bm*dcosh(bm)+tdm*gm*dcos(gm))
        endif
        as(j,j) = -(a1n+(2.0-poi)*a2n/phis)
     endif
 elseif (iedge.eq.12) then
     if ((icon.eq.1).or.(icon.eq.3)) then
        if ((testm.lt.0.0).and.(bms.gt.qlim)) then
            ex1 =  bm*dcos(emp)/(bms+(emp+enp)*(emp+enp))
            ex2 = -bm*dcos(emp)/(bms+(emp-enp)*(emp-enp))
            ex3 =  gm*dcos(emp)/(gms+(emp+enp)*(emp+enp))
            ex4 = -gm*dcos(emp)/(gms+(emp-enp)*(emp-enp))
        else
```

```
            ex1 = bm*(dcos(emp)*dcosh(bm)-dcos(enp))
     1                         /(bms+(emp+enp)*(emp+enp))
            ex2 = bm*(dcos(enp)-dcos(emp)*dcosh(bm))
     1                         /(bms+(emp-enp)*(emp-enp))
            if (testm.lt.0.0) then
               ex3 = gm*(dcos(emp)*dcosh(gm)-dcos(enp))
     1                            /(gms+(emp+enp)*(emp+enp))
               ex4 = gm*(dcos(enp)-dcos(emp)*dcosh(gm))
     1                            /(gms+(emp-enp)*(emp-enp))
            else
               ex3 = gm*(dcos(enp)-dcos(emp)*dcos(gm))
     1                           /(gms-(emp+enp)*(emp+enp))
               ex4 = gm*(dcos(emp)*dcos(gm)-dcos(enp))
     1                          /(gms-(emp-enp)*(emp-enp))
            endif
         endif
         if (icon.eq.1) then
            as(i,j) = tddm*(ex1+ex2+tdm*(ex3+ex4))
         else
            if ((testm.lt.0.0).and.(bms.gt.qlim)) then
              y1 = (emp+enp)*dcos(emp)/(bms+(emp+enp)*(emp+enp))
              y2 = -(emp-enp)*dcos(emp)/(bms+(emp-enp)*(emp-enp))
              y3 = (emp+enp)*dcos(emp)/(gms+(emp+enp)*(emp+enp))
              y4 = -(emp-enp)*dcos(emp)/(gms+(emp-enp)*(emp-enp))
            else
              y1 = (emp+enp)*(dcos(emp)*dcosh(bm)-dcos(enp))
     1                                /(bms+(emp+enp)*(emp+enp))
              y2 = (emp-enp)*(dcos(enp)-dcos(emp)*dcosh(bm))
     1                                /(bms+(emp-enp)*(emp-enp))
              if (testm.lt.0.0) then
                y3 = (emp+enp)*(dcos(emp)*dcosh(gm)-dcos(enp))
     1                                  /(gms+(emp+enp)*(emp+enp))
                y4 = (emp-enp)*(dcos(enp)-dcos(emp)*dcosh(gm))
     1                                  /(gms+(emp-enp)*(emp-enp))
                a1n = tddm*(bms*(ex1+ex2)+tdm*gms*(ex3+ex4))
              else
                y3 = (emp+enp)*(dcos(emp)*dcos(gm)-dcos(enp))
     1                                 /((emp+enp)*(emp+enp)-gms)
                y4 = (emp-enp)*(dcos(enp)-dcos(emp)*dcos(gm))
     1                                 /((emp-enp)*(emp-enp)-gms)
                a1n = tddm*(bms*(ex1+ex2)-tdm*gms*(ex3+ex4))
              endif
            endif
            a2n = -tddm*emps*(ex1+ex2+tdm*(ex3+ex4))
            a3n = -tddm*emp*(bm*(y1+y2)+tdm*gm*(y3+y4))
            td1 = dcos(alpha)*dcos(alpha)
     1                            +poi*dsin(alpha)*dsin(alpha)
            td2 = dsin(alpha)*dsin(alpha)
     1                            +poi*dcos(alpha)*dcos(alpha)
            td3 = (1.0-poi)*dsin(2.0*alpha)
            as(i,j) = -(td1*phis*a1n+td2*a2n+td3*phi*a3n)
         endif
      elseif (icon.eq.2) then
         if ((testm.lt.0.0).and.(bms.gt.qlim)) then
             ex1 =  bm*dcos(emp)/(bms+(emp+enp)*(emp+enp))
             ex2 = -bm*dcos(emp)/(bms+(emp-enp)*(emp-enp))
             ex3 =  gm*dcos(emp)/(gms+(emp+enp)*(emp+enp))
             ex4 = -gm*dcos(emp)/(gms+(emp-enp)*(emp-enp))
             y1  =  dcos(emp)*(emp+enp)/((emp+enp)*(emp+enp)+bms)
             y2  = -dcos(emp)*(emp-enp)/((emp-enp)*(emp-enp)+bms)
             y3  =  dcos(emp)*(emp+enp)/((emp+enp)*(emp+enp)+gms)
             y4  = -dcos(emp)*(emp-enp)/((emp-enp)*(emp-enp)+gms)
         else
           ex1 = bm*dcos(emp)*dsinh(bm)/(bms+(emp+enp)*(emp+enp))
           ex2 =-bm*dcos(emp)*dsinh(bm)/(bms+(emp-enp)*(emp-enp))
           y1  =  dcos(emp)*(emp+enp)*dsinh(bm)
     1                 /((emp+enp)*(emp+enp)+bms)
```

```
                y2   = -dcos(emp)*(emp-enp)*dsinh(bm)
     1                      /((emp-enp)*(emp-enp)+bms)
                if (testm.lt.0.0) then
                  ex3 =  gm*dcos(emp)*dsinh(gm)
     1                     /(gms+(emp+enp)*(emp+enp))
                  ex4 = -gm*dcos(emp)*dsinh(gm)
     1                     /(gms+(emp-enp)*(emp-enp))
                  y3  =  dcos(emp)*(emp+enp)*dsinh(gm)
     1                        /((emp+enp)*(emp+enp)+gms)
                  y4  = -dcos(emp)*(emp-enp)*dsinh(gm)
     1                        /((emp-enp)*(emp-enp)+gms)
                else
                  ex3 = gm*dsin(gm+emp)
     1                   /(gms-(emp+enp)*(emp+enp))
                  ex4 = gm*dsin(gm+emp)
     1                   /((emp-enp)*(emp-enp)-gms)
                  y3  = (emp+enp)*dsin(gm-emp)
     1                     /((emp+enp)*(emp+enp)-gms)
                  y4  = (emp-enp)*dsin(gm+emp)
     1                     /(gms-(emp-enp)*(emp-enp))
                endif
              endif
              a1n = -tddm*(bm*(ex1+ex2)+tdm*gm*(ex3+ex4))
              a2n = tddm*emp*(y1+y2+tdm*(y3+y4))
              as(i,j) = a1n*dcos(alpha)+a2n*dsin(alpha)/phi
            endif
        endif
  200 continue
        return
        end




C*********************************************************************
      subroutine factor (as,icon,tphi)
C*********************************************************************
c     takes care of continuity between segments of the
c     parallelogram plates

      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
       dimension as(32,16)
       tphis  = tphi*tphi
       phirs = phir*phir
              do 39 mm = 1,ki
                 do 39 nn = 1,kj
                     as(mm,nn) = as(mm,nn)
                       if (icon.eq.2) then
                           as(mm,nn) = as(mm,nn)*phir/tphi
                       elseif (icon.eq.3) then
                           as(mm,nn)= as(mm,nn)*phirs/tphis
                       elseif (icon.eq.4) then
                       as(mm,nn)= as(mm,nn)*phirs*phir/(tphis*tphi)
                       endif
   39            continue
      return
      end
```

```
C*******************************************************************
      subroutine grid (ksx,ksy,xi,eta,nt,nr,incode)
C*******************************************************************
c     creates a grid of nodes for the properly proportioned
c     display of the mode shape
      implicit real*8 (a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension xi(126), eta(21), incode(21,126)
      integer nt,nr
      deltaxit = (2+phi/phir)/(ksx-1.0)
      deltaxir = (2*phir/phi + 1.0)/(ksx-1.0)
      deltaeta = 1.0/(ksy-1.0)
      do 10 i = 1,ksy
         eta(i) = (i-1)*deltaeta
c        write(6,*) i, eta(i)
   10 continue
c        write(6,*) 'xi:'
      do 20 j = 1,ksx
         if (j.le.nt) then
              xi(j) = (j-1)*deltaxit
         elseif (j.le.nr) then
              xi(j) = (j-1)*deltaxir - phir/phi
         else
              xi(j) = 1.0 - xi(ksx+1-j)
         endif
c        write(6,*) j, xi(j)
   20 continue
c        write(6,*) 'generating zero-matrix'
c            1 = inside, 2 = outside
      do 40 j = 1, ksx
        do 30 i = 2, ksy-1
           if (j.le.nt) then
              if (eta(i).gt.(1-xi(j))) then
                incode(i,j) = 1
              else
                incode(i,j) = 2
              endif
           elseif (j.le.nr) then
              incode(i,j) = 1
           else
              if (eta(i).lt.(1-xi(j))) then
                incode(i,j) = 1
              else
                incode(i,j) = 2
              endif
           endif
   30    continue
   40 continue
      do 50 j = 1, ksx
        incode(1,j) = 2
        incode(ksy,j) = 2
   50 continue
c      do 90 l = 1,ksy
c          write(6,999)(incode(l,m),m=1,ksx)
c  90 continue
c 999 format (' ',100i3)
      return
      end
```

```
c
C**********************************************************************
       subroutine shape12 (j1,j2,ksy,ka,e1m,e2m,xi,eta,incode,w)
C**********************************************************************
C
      implicit real*8(a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension xi(126), eta(21), incode(21,126)
      dimension w(21,126),e1m(16),e2m(16)
c       write(*,*) 'beginning shape12'
      phis   = phi*phi
      do 108 i=1,ksy
       do 107 j=j1,j2
          if (incode(i,j).eq.1) then
        do 104 m=1,ka
           emp     = m*pi
           emps    = emp*emp
           bms     = phis*(almds+emps)
           bm      = dsqrt (bms)
           testm   = almds-emps
           gms     = phis*testm
           if (gms.lt.0.0) gms = -gms
           gm      = dsqrt(gms)
             if (testm.lt.0.0) then
              td1m   = -(bms-poi*phis*emps)*dsinh(bm)
     1                  /((gms-poi*phis*emps)*dsinh(gm))
              tdd1m = 1.0/(dsinh(bm)+td1m*dsinh(gm))
              td2m   = -dsinh(bm)/dsinh(gm)
              tdd2m = -1.0/(bms*dsinh(bm)+td2m*gms*dsinh(gm))
              w1 = e1m(m)*tdd1m*(dsinh(bm*eta(i))
     1              +td1m*dsinh(gm*eta(i)))*dsin(emp*xi(j))
              w2 = e2m(m)*tdd2m*(dsinh(bm*eta(i))
     1              +td2m*dsinh(gm*eta(i)))*dsin(emp*xi(j))
          else
              td1m   = (bms-poi*phis*emps)*dsinh(bm)
     1                  /((gms+poi*phis*emps)*dsin(gm))
              tdd1m = 1.0/(dsinh(bm)+td1m*dsin(gm))
              td2m   = -sinh(bm)/sin(gm)
              tdd2m = -1.0/(bms*dsinh(bm)-td2m*gms*dsin(gm))
              w1 = e1m(m)*tdd1m*(dsinh(bm*eta(i))
     1              +td1m*dsin(gm*eta(i)))*dsin(emp*xi(j))
              w2 = e2m(m)*tdd2m*(dsinh(bm*eta(i))
     1              +td2m*dsin(gm*eta(i)))*dsin(emp*xi(j))
          endif
          w(i,j) = w(i,j)+w1+w2
 104     continue
         else
           w(i,j) = 0.0
         endif
 107   continue
 108 continue
c       write(*,*)'...done shape12'
      return
      end


C
C**********************************************************************
      subroutine shape34 (j1,j2,ksy,kb,e3m,e4m,xi,eta,incode,w)
C**********************************************************************
C
      implicit real*8(a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension xi(126), eta(21), incode(21,126)
      dimension w(21,126),e3m(16),e4m(16)
c       write(6,*) 'Beginning shape34...'
```

```
       phis    = phi*phi
       phins   = 1.0/phis
       do 108 i=1,ksy
        do 107 j=j1,j2
           if (incode(i,j).eq.1) then
            do 104 m=1,kb
              emp     = m*pi
              emps    = emp*emp
              bms     = phins*(phis*almds+emps)
              bm      = dsqrt (bms)
              testm   = phis*almds-emps
              gms     = phins*testm
              if (gms.lt.0.0) gms = -gms
              gm      = dsqrt(gms)
              if (testm.lt.0.0) then
               td3m   = -(bms-poi*phins*emps)*dsinh(bm)
     1                    /((gms-poi*phins*emps)*dsinh(gm))
               tdd3m = 1.0/(dsinh(bm)+td3m*dsinh(gm))
               td4m   = -dsinh(bm)/dsinh(gm)
               tdd4m = -1.0/(bms*dsinh(bm)+td4m*gms*dsinh(gm))
               w3 = e3m(m)*tdd3m*(dsinh(bm*xi(j))
     1                   +td3m*dsinh(gm*xi(j)))*dsin(emp*eta(i))
               w4 = e4m(m)*tdd4m*(dsinh(bm*xi(j))
     1                   +td4m*dsinh(gm*xi(j)))*dsin(emp*eta(i))
              else
               td3m   = (bms-poi*phins*emps)*dsinh(bm)
     1                    /((gms+poi*phins*emps)*dsin(gm))
               tdd3m = 1./(dsinh(bm)+td3m*dsin(gm))
               td4m   = -sinh(bm)/sin(gm)
               tdd4m = -1./(bms*dsinh(bm)-td4m*gms*dsin(gm))
               w3 = e3m(m)*tdd3m*(dsinh(bm*xi(j))
     1                +td3m*dsin(gm*xi(j)))*dsin(emp*eta(i))
               w4 = e4m(m)*tdd4m*(dsinh(bm*xi(j))
     1                +td4m*dsin(gm*xi(j)))*dsin(emp*eta(i))
              endif
              w(i,j) = w(i,j)+w3+w4
  104       continue
           else
             w(i,j) = 0.0
           endif
c         write(6,*) 'w(',i,',',j,')=',w(i,j),'incode=',incode(i,j)
  107   continue
  108 continue
c       write(6,*) '...done shape34'
      return
      end


C
C*********************************************************************
       subroutine shape56 (j1,j2,ksy,ka,e5m,e6m,xi,eta,incode,w)
C*********************************************************************
      implicit real*8(a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension xi(126), eta(21), incode(21,126)
      dimension w(21,126),e5m(16),e6m(16)
c       write(*,*) 'Beginning shape56...'
      phis    = phi*phi
      do 108 i=1,ksy
       do 107 j=j1,j2
          if (incode(i,j).eq.1) then
        do 104 m=1,ka
          emp     = m*pi
          emps    = emp*emp
          bms     = phis*(almds+emps)
          bm      = dsqrt (bms)
          testm   = almds-emps
```

```
          gms     = phis*testm
          if (gms.lt.0.0) gms = -gms
          gm      = dsqrt(gms)
            if (testm.lt.0.0) then
             td1m   = -(bms-poi*phis*emps)*dsinh(bm)
     1                    /((gms-poi*phis*emps)*dsinh(gm))
             tdd1m = 1.0/(dsinh(bm)+td1m*dsinh(gm))
             td2m   = -dsinh(bm)/dsinh(gm)
             tdd2m = -1.0/(bms*dsinh(bm)+td2m*gms*dsinh(gm))
             w5 = e5m(m)*tdd1m*(dsinh(bm*(1.0-eta(i)))
     1                  +td1m*dsinh(gm*(1.0-eta(i))))*dsin(emp*xi(j))
             w6 = e6m(m)*tdd2m*(dsinh(bm*(1.0-eta(i)))
     1                  +td2m*dsinh(gm*(1.0-eta(i))))*dsin(emp*xi(j))
          else
             td1m   = (bms-poi*phis*emps)*dsinh(bm)
     1                    /((gms+poi*phis*emps)*dsin(gm))
             tdd1m = 1.0/(dsinh(bm)+td1m*dsin(gm))
             td2m   = -sinh(bm)/sin(gm)
             tdd2m = -1.0/(bms*dsinh(bm)-td2m*gms*dsin(gm))
             w5 = e5m(m)*tdd1m*(dsinh(bm*(1.0-eta(i)))
     1                  +td1m*dsin(gm*(1.0-eta(i))))*dsin(emp*xi(j))
             w6 = e6m(m)*tdd2m*(dsinh(bm*(1.0-eta(i)))
     1                  +td2m*dsin(gm*(1.0-eta(i))))*dsin(emp*xi(j))
          endif
          w(i,j) = w(i,j)+w5+w6
  104    continue
         else
           w(i,j) = 0.0
         endif
  107  continue
  108 continue
c        write(*,*) '...done shape56'
      return
      end


C***********************************************************************
      subroutine shape78 (j1,j2,ksy,kb,e7m,e8m,xi,eta,incode,w)
C***********************************************************************
      implicit real*8(a-h,o-z)
      common k,ki,kj,poi,qlim,almds,phi,phir,pi,iedge,alpha
      dimension xi(126), eta(21), incode(21,126)
      dimension w(21,126),e7m(16),e8m(16)
      phis   = phi*phi
      phins  = 1.0/phis
      do 108 i = 1, ksy
       do 107 j = j1, j2
          if (incode(i,j).eq.1) then
           do 104 m=1,kb
             emp     = m*pi
             emps    = emp*emp
             bms     = phins*(phis*almds+emps)
             bm      = dsqrt (bms)
             testm   = phis*almds-emps
             gms     = phins*testm
             if (gms.lt.0.0) gms = -gms
             gm      = dsqrt(gms)
             if (testm.lt.0.0) then
              td3m   = -(bms-poi*phins*emps)*dsinh(bm)
     1                   /((gms-poi*phins*emps)*dsinh(gm))
              tdd3m = 1.0/(dsinh(bm)+td3m*dsinh(gm))
              td4m   = -dsinh(bm)/dsinh(gm)
              tdd4m = -1.0/(bms*dsinh(bm)+td4m*gms*dsinh(gm))
             w7 = e7m(m)*tdd3m*(dsinh(bm*(1.0-xi(j)))
     1            +td3m*dsinh(gm*(1.0-xi(j))))*dsin(emp*eta(i))
             w8 = e8m(m)*tdd4m*(dsinh(bm*(1.0-xi(j)))
     1            +td4m*dsinh(gm*(1.0-xi(j))))*dsin(emp*eta(i))
```

```
               else
                td3m   = (bms-poi*phins*emps)*dsinh(bm)
     1                       /((gms+poi*phins*emps)*dsin(gm))
                tdd3m  = 1.0/(dsinh(bm)+td3m*dsin(gm))
                td4m   = -sinh(bm)/sin(gm)
                tdd4m  = -1.0/(bms*dsinh(bm)-td4m*gms*dsin(gm))
                w7  = e7m(m)*tdd3m*(dsinh(bm*(1.0-xi(j)))
     1                   +td3m*dsin(gm*(1.0-xi(j))))*dsin(emp*eta(i))
                w8  = e8m(m)*tdd4m*(dsinh(bm*(1.0-xi(j)))
     1                   +td4m*dsin(gm*(1.0-xi(j))))*dsin(emp*eta(i))
               endif
               w(i,j)  = w(i,j)+w8 +w7
  104        continue
           else
             w(i,j)  = 0.0
           endif
  107    continue
  108 continue
      return
      end


c*********************************************************************
      subroutine determ (a,n,det)
c*********************************************************************
c      finds the determinant by pivotal reduction
c      courtesy of Dr. H.T. Saliba
      implicit real*8(a-h,o-z)
      dimension a(320,320)
      sign=1.
      last=n-1
c     start overall loop for(n-1) pivots
      do 200 i=1,last
c       find the largest remaining term in i-th column for pivot
        big=0.
        do 50 k=i,n
          term=dabs(a(k,i))
          if (term-big)50,50,30
   30     big=term
          l=k
   50   continue
c       check whether a non-zero term has been found
        if (big)80,60,80
c        l-th row has the biggest term----is i=l
   80   if (i-l)90,120,90
c        i is not equal to l, switch rows i and l
   90   sign=-sign
        do 100 j=1,n
          temp=a(i,j)
          a(i,j)=a(l,j)
  100   a(l,j)=temp
c       now start pivotal reduction
  120   pivot=a(i,i)
        nextr=i+1
c       for each of the rows after the i-th
        do 200 j=nextr,n
c         multiplying constant for the j-th row is
          const=a(j,i)/pivot
c         now reduce each term of the j-th row
          do 200 k=i,n
  200 a(j,k)=a(j,k)-const*a(i,k)
c     end of pivotal reduction----now compute determinant
      det=sign
      do 300 i=1,n
  300 det=det*a(i,i)*23
c                       ^ this factor can be adjusted to accomodate
c             various orders of magnitude of the trial eigenvalues
```

```
      go to 61
   60 det=0.
   61 return
      end


c**********************************************************************
      subroutine detsol  (a,n,x)
c**********************************************************************
c     solves for the eigenvector using pivotal reduction
c      and back substitution
c      courtesy of Dr. H.T. Saliba
      implicit real*8(a-h,o-z)
      dimension a(320,320),x(320)
      sign=1.
      m=n-1
      last=m-1
      do 10 i = 1,n
        x(i) = 0.0
   10 continue
c     start overall loop for(n-1) pivots
      do 200 i=1,last
c       find the largest remaining term in i-th column for pivot
        big=0.
        do 50 k=i,m
          term=dabs(a(k,i))
          if (term-big)50,50,30
   30       big=term
            l=k
   50   continue
c       check whether a non-zero term has been found
        if (big)80,60,80
c         l-th row has the biggest term------is i=l
   80     if (i-l)90,120,90
c           i is not equal to l, switch rows i and l
   90       do 100 j=1,n
              temp=a(i,j)
              a(i,j)=a(l,j)
  100       a(l,j)=temp
c         now start pivotal reduction
  120     pivot=a(i,i)
          nextr=i+1
c         for each of the rows after the i-th
          do 200 j=nextr,m
c           multiplying constant for the j-th row is:
            const=a(j,i)/pivot
c           now reduce each term of the j-th row
            do 200 k=i,n
  200 a(j,k)=a(j,k)-const*a(i,k)
c     end of pivotal reduction-- perform back substitution
      m=n-1
      do 500 i=1,m
c       irev is the backward index, going from m back to 1
        irev=m+1-i
c       get y(irev) in preparation
        y=a(irev,n)
        if (irev-m) 400,500,400
c         not working on last row, i is 2 or greater
  400     do 450 j=2,i
c           work backward for x(n),x(n-1)-----substituting previously
c           found values
            k=n+1-j
  450     y=y-a(irev,k)*x(k)
c         finally,compute x(irev)
  500 x(irev)=y/a(irev,irev)
   60 return
      end
```

IMAGE EVALUATION
TEST TARGET (QA-3)

1.0 2.8 2.5
3.2 2.2
3.6
4.0 2.0
1.1
1.8
1.25 1.4 1.6

150mm

6"

APPLIED IMAGE . Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989